On the Robustness of Deep Neural Models and Their Applications to Speech Science

Yossi Adi

Department of Computer Science

Ph.D. Thesis

Submitted to the Senate of Bar-Ilan University

Ramat-Gan, Israel **July 10, 2019**

This work was carried out under the supervision of
Prof. Joseph Keshet
Department of Computer Science,
Bar-Ilan University.

Table of contents

Abstract

Deep learning has redefined the landscape of machine intelligence by enabling several breakthroughs and reaching state-of-the-art performance in notoriously difficult problems such as image classification, speech recognition, human pose estimation and machine translation. In addition, these models were empirically proven to generalize well on inputs drawn from the same distribution as the ones used to train the model.

These observations raise two main research questions: (i) Can deep learning models be used for super-resolution, fine-grained measurements, so as to be used as a tool to automate and scale-up scientific research?; and (ii) As deep learning models are permeating nearly all segments of technology industry, from self-driving cars to automated dialog agents, are traditional metrics sufficient for deep neural model evaluation? How do we assess the reliability of these models?

In this study, we explore these two broad research questions from several points of view. In the first part of the thesis, we demonstrate how neural models can be applied to speech processing in the context of speech science. We develop highly accurate deep learning and structured prediction based tools for fine-grained duration measurement of phonetic properties such as voice-onset-time, vowel, pre-aspiration, and whole words. We empirically show that the proposed algorithms can be used to conduct linguistic studies with no manual annotation of the speech data.

In the second part of the study, we explore and analyze the robustness of deep learning models to adversarial examples and out-of-distribution examples in the domain of speech and image processing. We provide empirical and theoretical tools for uncertainty estimation, hence enabling the detection of such examples.

In the third part of the thesis, we empirically analyze the performance of commonly used deep learning architectures for sequential data processing, such as Recurrent Neural Networks, Sequence-to-Sequence, Convolutional Neural Networks using Multitask Learning or Adversarial Learning.

In the last part of the dissertation, we build on top of this line of research in order to implement and design secured and robust algorithms. We first show an innovative new way to watermark a general deep learning model, and analyze its performance both theoretically and empirically. Second, we demonstrate how neural networks can be applied to hide a given speech waveform within another given speech waveform.

Chapter 1

Introduction

With the growing availability of digital data such as texts, images and audio recordings, there is an increasing need to develop algorithms which can learn to organize and classify structured and unstructured data. Machine Learning is a sub-field of Artificial Intelligence (AI) that defines the theoretical bounds of which functions and structures can be learned by an artificial agent and how. Neural networks or, as they have been called more recently, *Deep Learning* is a sub-field of machine learning that is based on an extensive collection of connected neural units aimed to loosely model the way a biological brain solves problems. These models are composed of stacked layers of non-linear units which serve as neurons. Neurons can be connected to each other in different ways where each of them has a set of learned parameters.

This introduction gives an overview of the research conducted throughout my Ph.D studies. I will start by reviewing my works on deep neural models to measure fine-grained phonetic properties in order to scale-up and automate speech science. Then I will turn to review the works done on the robustness of deep neural models to adversarial examples and out-of-distribution examples. Moreover, I will show several ways to detect such cases, and how to use them in order to design secured systems.

I conclude the research review with an analysis of neural models for speech and language processing, followed by a description of two security-related applications, modeled by neural networks.

1.1 Speech science

A key component in studies of speech and language processing, and its disorders, is the measurement of fine-grained phonetic properties. This has been hampered by reliance on subjective, labor-intensive manual annotation, which might take over 3,000 person-hours for a single study (Goldrick et al., 2011). In this thesis I summarize my ongoing work on designing and developing machine learning algorithms for automatically measuring, with

high precision, phonetic properties of speech (at the level of human inter-transcriber reliability). The proposed methods rely on several advances over existing computational systems: (i) novel representations of the speech signal; (ii) new structured prediction and deep learning algorithms. These automatic methods allow for low-cost replication of phonetic studies (increasing the rigor of studies of speech) and expand the range of empirical and theoretical issues we can address.

In conjunction with colleagues, I proposed a series of algorithms that can be applied to accurately measure various phonetic properties, such as vowel, voice-onset-time, consonants, fundamental frequency, and so on (Adi and Keshet, 2016; Adi et al., 2015, 2016b,c, 2017b; Cohen et al., 2019; Naaman et al., 2017; Sheena et al., 2017). For the simplicity of the writing, I only describe here the methods for vowel measurement.

I summarize this section by reviewing my work, which demonstrates how a full-blown linguistic research study can be done using the above algorithms (Goldrick et al., 2018).

Frame-based classifiers. In many speech processing applications, we assume that the length of the input varies from one instance to the another, where we denote the input at each time step as *frame* or *input frame*. Our first straightforward approach is to train classifiers based on independent input frames. We train a classifier at the frame-level to detect whether the frame is a vowel or not. We used the state-of-the-art Deep Neural Network (DNN) as a classifier, comparing two DNN architectures: Deep Belief Network (DBN) and Convolutional Neural Network (CNN). Both architectures have produced good results in previous speech processing studies (Hinton et al., 2012; LeCun et al., 1995). Each architecture was trained on manually annotated data and their performance was compared. At inference time, the classifier predicts the probability of each frame of the input as being a vowel. The predictions are smoothed out to have a single chunk representing the vowel, and then vowel duration is computed (Adi et al., 2015).

The above approach assumes that the frames of each input instance are conditionally independent of each other. However, in sequential data, this is often not the case. Hence, in order to model the dependency between input frames, we explore the use of multi-class Recurrent Neural Network (RNN), followed by a Dynamic Programming (DP) algorithm to detect the best segmentation of the utterance based on the classifier predictions (Adi et al., 2016c).

We compare the accuracy of DBN, CNN, and RNN to an HMM-based force aligner on manually annotated data. Results suggest that CNN is better than DBN, and the CNN and HMM-based forced aligner are comparable to each other. RNN is superior to the previous methods, including current state-of-the-art systems (Adi et al., 2015, 2016c).

Structured prediction classifier. All of the above methods assume the output at every time-step is conditionally independent of other time-steps. However, we know this is not the case in phonetic segmentation. Moreover, these methods require an additional post-process

phase to find the final segmentation (e.g., smoothing, DP search, etc.). Hence, we next propose a model based on structured prediction techniques which have provided excellent results in analyzing other phonetic measures (Keshet et al., 2007; Sonderegger and Keshet, 2012). The algorithm was trained at the segment level on manually annotated data to extract the vowel start and end times; this provides a straightforward way to compute the duration of the vowel. Following the structure of the vast majority of laboratory studies of vowel duration, we assume the input signal contains a single vowel, proceeded and followed by a consonant – no additional detailed information about the phonetic transcription is required to process the speech.

We evaluated our method on data from three phonetic studies: one focusing on vowel duration (Heller and Goldrick, 2014), the second using vowel segmentation to determine points for formant analysis (Clopper and Tamati, 2014) automatically, and the third is a standard set of vowel production norms (Hillenbrand et al., 1995). We compared results from our model to state-of-the-art model in vowel duration measurement, HMM-based forced alignment (Rosenfelder et al., 2014). We also assessed whether inferential statistical models fit to data from our model and HMM-based forced alignment replicated the patterns obtained from manual data. The results suggest that the proposed algorithm is superior to the current gold standard at matching the manual measurements of vowel duration, both in terms of deviation and in replicating inferential statistical results (Adi et al., 2016b).

Jointly optimize frame-based and structured prediction classifiers. Although the structured prediction approach provides accurate measurements together with clean modeling of the problem, its main drawback is the use of a linear classifier, which is relatively weak compared to neural models.

Inspired by the work on combining DNNs and structured prediction models (Chen et al., 2015; Do et al., 2010; Kiperwasser and Goldberg, 2016; Lample et al., 2016; Zheng et al., 2015), together with our previous findings, we would like to further improve performance on speech sequence segmentation and propose a new efficient joint DNN and structure prediction model. Specifically, we jointly optimize RNN and structured loss parameters by using RNN outputs as feature functions for a structured prediction model. First, an RNN encodes the entire speech utterance and outputs new representation for each of the frames. Then, an efficient search is applied over all possible segments so that the most probable one can be selected. We show that the proposed approach outperforms previous methods on these several segmentation tasks (Adi et al., 2017b).

1.2 Robustness of neural networks to adversarial and out-of-distribution examples

DNNs have gained lots of success after enabling several breakthroughs in notably challenging problems such as image classification (He et al., 2016), speech recognition (Amodei

et al., 2016a) and machine translation (Bahdanau et al., 2014). These models are known to generalize well on inputs that are drawn from the same distribution as the examples that were used to train the model (Zhang et al., 2016b). In real-world scenarios, the input instances to the model can be drawn from different distributions. In these cases, DNNs tend to perform poorly. Nevertheless, it was observed that DNNs often produce high confidence predictions for unrecognizable inputs (Nguyen et al., 2015) or even for a random noise (Hendrycks and Gimpel, 2016). Moreover, recent works in the field of adversarial example generation show that due to small input perturbations, DNNs tend to produce high probabilities while being largely incorrect (Cisse et al., 2017b; Goodfellow et al., 2014; Kreuk et al., 2018).

As the most successful models are permeating nearly all of the segments of the technology industry, from self-driving cars to automated dialog agents, it becomes critical to revisit the evaluation protocol of deep learning models and design new ways to assess their reliability beyond the traditional metrics. When considering AI Safety, it is essential to train DNNs that are aware of the uncertainty in the predictions (Amodei et al., 2016b).

Adversarial examples. Adversarial examples are synthetic patterns carefully crafted by adding a peculiar noise to legitimate examples. They are indistinguishable from the legitimate examples by a human, yet they have demonstrated a strong ability to cause catastrophic failure of state-of-the-art systems (Goodfellow et al., 2015; Kurakin et al., 2016; Moosavi-Dezfooli et al., 2015).

The existence of adversarial examples highlights a potential security threat for machine learning systems at large (Papernot et al., 2016a). It has triggered an active line of research concerned with understanding the phenomenon (Fawzi et al., 2015, 2016) and making DNNs more robust (Cisse et al., 2017a,b; Kreuk et al., 2018; Papernot et al., 2016b).

To better understand this phenomena, we first generalized adversarial example generation beyond multi-class classification to structured prediction tasks such as Automatic Speech Recognition, Image Segmentation, and Pose Estimation.

We introduce *Houdini*, the first approach for fooling any gradient based learning machine by generating adversarial examples directly tailored for the task loss of interest, be it combinatorial, non-differentiable or both. We show the close relationship between Houdini and the task loss of the problem considered. We present the first successful attack on a deep Automatic Speech Recognition system, namely a *DeepSpeech-2* based ASR system (Amodei et al., 2015) by generating adversarial audio files not distinguishable from legitimate ones by the human ear (as validated by an ABX experiment). We also demonstrate the transferability of adversarial examples in speech recognition by fooling Google Voice in a black box attack. We present the first successful un-targeted and targeted attacks on a deep model for human pose estimation (Newell et al., 2016). Similarly, we validate the feasibility of untargeted and targeted attacks on a semantic segmentation system (Yu and Koltun, 2016) and show that we can make the system hallucinate an arbitrary segmentation of our choice for a given image.

Next, we extend *Houdini* to generate adversarial examples to more security oriented tasks, namely *speaker verification*. Automatic speaker verification is the task of verifying that a spoken utterance has been produced by a claimed speaker. It is one of the most mature technologies for biometric authentication deployed by banks and e-commerce as the primary means of authenticating customers online and over the phone. The vulnerability of such a system is a real threat to these applications.

The standard verification protocol comprises the following three steps: training, enrollment, and evaluation. In the training stage, one learns a suitable internal speaker representation from a set of utterances and builds a simple scoring function. In the enrollment stage, a speaker provides a few utterances which are used to estimate the speaker model. During the evaluation stage, the verification task is performed by scoring a new unknown utterance against the speaker model. If the resulted score is greater than a predefined threshold, the system predicts that the unknown utterance was produced by the claimed speaker. When the authentication is based on the voice of the speaker, irrespective of what the speaker said, the system is a text-independent speaker verification system. On the other hand, when the authentication is based on a specific set of words, we call it a text-dependent speaker verification system.

We investigate the generation of adversarial examples to attack an end-to-end neural based speaker verification model. We demonstrate the generation of adversarial examples to attack a text-dependent speaker verification system while using the architecture proposed by the authors in Heigold et al. (2016). To the best of our knowledge, this is the first time it has been proven possible to fool speaker verification systems using adversarial examples (Kreuk et al., 2018).

Out-of-distribution detection. Another set of examples which cause neural networks failures while producing highly confident predictions is out-of-distribution examples.

Several studies have proposed different approaches to handle this uncertainty (Hendrycks and Gimpel, 2016; Lakshminarayanan et al., 2017; Lee et al., 2017; Liang et al.). Hendrycks and Gimpel (2016) proposed a baseline method to detect out-of-distribution examples based on the models' output probabilities. Liang et al. extended the baseline method by using temperature scaling of the softmax function and adding small controlled perturbations to inputs (Hinton et al., 2015). Lee et al. (2017) suggested adding another term to the loss so as to minimize the Kullback-Leibler divergence between the models' output for out-of-distribution samples and the uniform distribution.

An ensemble of classifiers with optional adversarial training was proposed by Lakshminarayanan et al. (2017) for detecting out-of-distribution and adversarial examples. Despite their high detection rate, ensemble methods require the optimization of several models and therefore are resource intensive. Additionally, each of the classifiers which participated in the ensemble is trained independently and the representation is not shared among them.

We proposed a new measurement for uncertainty estimates based on the internal input representation norm. More specifically, we suggest replacing the traditional supervision

during training by using several word embeddings as the model's supervision, where each of the embeddings was trained on a different corpus or with a different architecture. This way we can create variance between the predictors.

We were inspired by several studies (Frome et al., 2013; Littwin and Wolf, 2016; Taigman et al., 2015). Littwin and Wolf (2016) presented a novel technique for robust transfer learning, where they proposed optimizing multiple orthogonal predictors while using a shared representation. Although being orthogonal to each other, according to their results, the predictors were likely to produce identical softmax probabilities. The idea of using word embeddings as a supervision was first proposed by Frome et al. (2013) for the task of *zero-shot learning*. Lastly, Taigman et al. (2015) found a link between the L2-norm of the input representation and the ability to discriminate in a target domain.

1.3 Analysis of neural networks for speech and language processing

While neural networks provide state-of-the-art results in various domains, little is known about the information that is captured by different neural models. To mitigate that, we propose a method for probing internal representations produced by neural models and analyze several models for speech and language processing. Namely, continuous-bag-of-words (CBOW) (Mikolov et al., 2013a,b); RNN (Elman, 1991); Sequence-to-Sequence (Sutskever et al., 2014) for sentence representation, and Fully Convolutional Models for Speech Recognition (Adi et al., 2019; Collobert et al., 2016).

Analyzing sentence embedding. We first perform a fine-grained comparison of different sentence embedding methods by probing them for three essential characteristics of every sequence: its length, the items within it, and their order (Adi et al., 2016a, 2017a).

In sentence embeddings, sentences, which are variable-length sequences of discrete symbols, are encoded into fixed length continuous vectors which are then used for further prediction tasks. A simple and common approach is producing word-level vectors using, e.g., word2vec (Mikolov et al., 2013a,b), and summing or averaging the vectors of the words participating in the sentence. The CBOW approach disregards the word order in the sentence.

The resulting sentence representations are opaque, and there is currently no good way of comparing different representations short of using them as input for different high-level semantic tasks (e.g. sentiment classification, entailment recognition, document retrieval, question answering, sentence similarity, etc.) and measuring how well they perform on these tasks. This method of comparing sentence embeddings leaves a lot to be desired: the comparison is at a very coarse-grained level, does not tell us much about the kind of information that is encoded in the representation, and does not help us form generalizable conclusions.

We take a first step towards opening the black box of sentence embeddings, while proposing a methodology that facilitates comparing embeddings on a much finer-grained level, and demonstrate its use by analyzing and comparing different embedding methods.

Our analysis reveals several interesting insights regarding the different sentence embedding methods: (i) Sentence representations based on averaged word vectors are surprisingly effective, and encode a non-trivial amount of information regarding sentence length. The information they contain can also be used to reconstruct a non-trivial amount of the original word order in a probabilistic manner (due to regularities in the natural language data); (ii) Long Short-Term Memory (LSTM) auto-encoders are very effective at encoding word order and word content; (iii) Increasing the number of dimensions benefits some tasks more than others; (iv) Adding more hidden units sometimes *degrades* the encoders' ability to encode word content; and (v) LSTM encoders trained as auto-encoders do not rely on ordering patterns in the training sentences when encoding novel sentences.

Analyzing acoustic models for speaker identity. In the context of speech processing, we follow the same approach and analyze how neural networks encode the identity of the speaker in an end-to-end acoustic modeling.

Traditionally, the identity of the speakers has been used to extract speaker representations and use it as an additional input to the acoustic model (Peddinti et al., 2015; Senior and Lopez-Moreno, 2014). This approach requires an additional feature extraction process of the input signal. Recently, two approaches have been proposed to leverage speaker information as another supervision to the acoustic model: Multi-Task Learning (MT) (Caruana, 1998; Collobert et al., 2011) and Adversarial Learning (AL) (Ganin et al., 2016).

In MT our goal is to jointly transcribe the speech signal together and to classify the speaker identity. Previous work has explored using auxiliary tasks such as gender or context (Pironkov et al., 2016a), as well as speaker classification (Pironkov et al., 2016b; Tang et al., 2016).

An opposite approach is to assume that good acoustic representations should be invariant to any task that is not the speech recognition task, in particular the speaker characteristics. A method for learning such invariances is AL: the branch of the speaker classification task is trained to reduce its classification error, however the main branch is trained to maximize the loss of this speaker classifier. By doing so, it learns invariance to speaker characteristics.

The following study arises from two observations. First, both the MT and AL approaches described above have been used with the same purpose despite being fundamentally opposed, and this raises the question of which one is the most appropriate choice to improve the performance of speech recognition systems. Secondly, they only differ by a simple computational step which consists of reversing the gradient at the fork between the speaker branch and the main branch. This allows for controlled experiments where both approaches can be compared with equivalent architectures and number of parameters.

A prerequisite for this study is the assessment of *to what extent the network encodes*

the speaker identity?. The motivation for answering this question is two-fold (i) to analyze the networks' behavior concerning the speaker information; and (ii) having a measurement of how much speaker information is encoded in the representation. To tackle this question, we follow the above approach, where we extract representations of the speech signal from different layers in the network and probe them on the speaker identity.

Our analysis reveals the following insights: (i) Deep models trained on big datasets, already develop invariant representations to speakers with neither AL nor MT; (ii) Neither AL nor MT have a clear impact on the acoustic model with respect to error rates; and (iii) Using additional, un-transcribed speaker labeled data (i.e. the speaker is known, but without the transcription) seems promising.

1.4 Security applications

Watermarking neural models. When considering the effectiveness of DNNs with the burden of the training and tuning stage, a new market of Machine Learning as a Service (MLaaS) has been opened. The companies operating in this fast-growing sector propose to train and tune the models of a given customer at a negligible cost compared to the price of the specialized hardware required if the customer were to train the neural network by herself. Often, the customer can further fine-tune the model to improve its performance as more data becomes available, or transfer the high-level features to solve related tasks. In addition to open source models, MLaaS allows the users to build more personalized systems without much overhead (Ribeiro et al., 2015).

Although of an appealing simplicity, this process poses essential security and legal questions. A service provider can be concerned that customers who buy a DNN might distribute it beyond the terms of the license agreement, or even sell the model to other customers, thus threatening its business. The challenge is to design a robust procedure for authenticating a DNN. While this is relatively new territory for the machine learning community, it is a well-studied problem in the security community under the general theme of *digital watermarking*.

Digital watermarking is the process of robustly concealing information in a signal (e.g., audio, video or image) in order to subsequently use it to verify either the authenticity or the origin of the signal. Watermarking has been extensively investigated in the context of digital media (see, e.g., (Boneh and Shaw, 1995; Katzenbeisser and Petitcolas, 2016; Petitcolas et al., 1999) and references within), and in the context of watermarking digital keys (e.g., in Naor et al. (2001)). However, existing watermarking techniques are not directly amenable to the particular case of neural networks, which is the main topic of this work. Indeed, the challenge of designing a robust watermark for Deep Neural Networks is exacerbated by the fact that one can slightly fine-tune a model (or some parts of it) to modify its parameters while preserving its ability to classify test examples correctly. Also, one will prefer a public watermarking algorithm that can be used to prove ownership

multiple times without the loss of credibility of the proofs. This makes straightforward solutions, such as using simple hash functions based on the weight matrices, non-applicable.

Instead, we propose to embed a watermark in neural network models by forcing the model to output specific outputs for carefully chosen inputs. Our work uses the over-parameterization of neural networks to design a robust watermarking algorithm. This over-parameterization has so far mainly been considered as a weakness (from a security perspective) because it makes backdooring possible (Cisse et al., 2017b; Goodfellow et al., 2014; Gu et al., 2017; Kreuk et al., 2018; Zhang et al., 2016b). We propose to use it in order to verify a neural model. We additionally present a cryptographic modeling of the tasks of watermarking and backdooring of DNNs, and show that the former can be constructed from the latter (using a cryptographic primitive functions called *commitments*) in a black-box way.

Speech steganography. Next, we address the topic of *speech* steganography – hiding secret spoken messages within public audio files. Although one can simply hide written text inside audio files and convey the same lexical content, concealing audio inside audio preserves additional features. For instance, the secret message may convey the speaker identity, the sentiment of the speaker, prosody, etc. These features can be used for later identification and authentication of the message.

Traditionally, steganographic functions have exploited actual or perceptual redundancies in the carrier signal. The most common approach is to encode the secret message in the least significant bits of individual sound samples (Jayaram et al., 2011). Other methods include concealing the secret message in the *phase* of the frequency components of the carrier (Dong et al., 2004) or in the form of the parameters of a miniscule echo that is introduced into the carrier signal (Bender et al., 1999). The redundancies to be exploited by any particular method must be explicitly selected, and are generally a feature of the steganography algorithm.

Recently, Baluja (2017); Zhu et al. (2018) proposed using DNNs as a stenographic function for hiding an image inside another image. In this line of work, the network *learns* to conceal a hidden message inside the carrier without manually specifying a particular redundancy to exploit.

Although these studies presented impressive results on image data, the applicability of such models for speech data was not explored. In this study, we show that steganography models proposed for vision are less suitable for speech. We build on the work by Baluja (2017); Zhu et al. (2018) and propose a new model that includes differentiable Fourier Transform layers within the network, thus imposing a vital constraint on the network output to fit speech data.

The proposed model is comprised of three parts. The first learns to encode a hidden message inside the carrier. The second component is differential Fourier Transform layers that simulate transformations between frequency and time domains. Lastly, the third component learns to decode a hidden message from a generated carrier. Additionally, we

demonstrated for the first time that the above scheme now permits us to hide *multiple* secret messages in a *single* carrier, each potentially with a different intended recipient who is the only person who can recover it.

Notice that generating steganographic instances (i.e., instances that contain a hidden message) is very similar to adversarial example generation. While the existence of adversarial examples is usually seen as a disadvantage of DNNs, it can be a desirable property for hiding secret information. Instead of injecting perturbations that lead to wrong classification, one can consider the possibility of encoding useful information via adding specific perturbations.

Chapter 2

Published Papers (Speech Science)

Vowel Duration Measurement using Deep Neural Networks

2015 IEEE INTERNATIONAL WORKSHOP ON MACHINE LEARNING FOR SIGNAL PROCESSING, SEPT. 17–20, 2015, BOSTON, USA

VOWEL DURATION MEASUREMENT USING DEEP NEURAL NETWORKS

Yossi Adi, Joseph Keshet Matthew Goldrick

Dept. of Computer Science
Bar-Ilan University, Ramat-Gan, Israel
adiyoss@cs.biu.ac.il, joseph.keshet@biu.ac.il

Dept. of Linguistics
Northwestern University, Evanston, IL, USA
matt-goldrick@northwestern.edu

ABSTRACT

Vowel durations are most often utilized in studies addressing specific issues in phonetics. Thus far this has been hampered by a reliance on subjective, labor-intensive manual annotation. Our goal is to build an algorithm for automatic accurate measurement of vowel duration, where the input to the algorithm is a speech segment contains one vowel preceded and followed by consonants (CVC). Our algorithm is based on a deep neural network trained at the frame level on manually annotated data from a phonetic study. Specifically, we try two deep-network architectures: convolutional neural network (CNN), and deep belief network (DBN), and compare their accuracy to an HMM-based forced aligner. Results suggest that CNN is better than DBN, and both CNN and HMM-based forced aligner are comparable in their results, but neither of them yielded the same predictions as models fit to manually annotated data.

Index Terms— vowel duration measurement, convolution neural networks, deep belief networks, hidden Markov models, forced alignment

1. INTRODUCTION

Vowel durations are often measured in studies addressing specific issues in phonetics [1, 2, 3]. These typically utilize vowel durations as a dependent measure, examining how duration changes across situations, e.g., when vowels are elicited in different contexts, produced by different speakers, etc.

To obtain accurate data most researchers have relied on manual annotation. This approach is clearly not ideal: it is highly resource intensive and fundamentally subjective. To address these issues, recent phonetic studies have used computational methods to measure acoustic properties of speech automatically, e.g., [2, 4, 5]. These methods greatly reduce the resources required as well as minimizing the role of subjective judgments.

In this paper we try to address the problem of automatic measurement of vowel duration. Most of the work related to vowel duration measurement was done using HMM-based forced alignment. However, those aligners require the phonetic transcription of the input signal, and should be carefully tuned [6]. Our work is focused on input of speech segments where a single vowel is preceded and followed by consonant (CVC), and the phonetic transcription is not needed.

Here we take a different route and train a classifier at the frame-level to detect whether the frame is a vowel or not. We used state-of-the-art deep neural network (DNN) as a classifier, comparing two DNN architectures: deep belief network (DBN) and convolutional neural network (CNN). Both architectures have produced good results in previous speech processing studies [7, 8]. Each architecture was trained on manually annotated data and their performance was compared. At inference time, the classifier predicts the probability of each frame of the input as being a vowel. The predictions are smoothed out to have a single chunk representing the vowel, and then vowel duration is computed.

We compare the accuracy of DBN, CNN and HMM-based force aligner on manually annotated data. The results show that CNN is better than DBN, and the CNN and HMM-based forced aligner are comparable. We further evaluated the performance of CNN and HMM on a phonetic study, which examines how placing words in a context that strongly emphasizes processing of sentence structure would influence speech articulation [3]. The results suggest that the CNN-based classifier is comparable to the HMM-based forced aligner in terms of deviation from the manual annotations. However, it seems that neither method is clearly superior in terms of matching the manual annotations.

The paper is organized as follows. In Section 2 we state the problem definition formally. In Section 3 we present the acoustic feature set and the architecture of the network used. In Section 4 we briefly describe the benchmark data and how it was recorded. We present a detailed experimental results and analysis in Section 5. We conclude the paper in Section 6.

2. PROBLEM SETTING

In the problem of *vowel duration measurement* we are provided with a speech signal which includes exactly one vowel preceded and followed by consonants, often denoted CVC. Our goal is to predict the vowel duration accurately. We de-

978-1-4673-7454-5/15/$31.00 ©2015 IEEE

note the domain of the acoustic feature vectors by $\mathcal{X} \subset \mathbb{R}^d$. The acoustic feature representation of a speech signal is therefore a sequence of vectors $\mathbf{x} = (x_1, x_2, \ldots, x_T)$ where $x_i \in \mathcal{X}$ for all $1 \leq i \leq T$. The length of the input signal varies from one signal to another, thus T is not fixed. We denote by $\mathcal{X}^*$ the set of all finite length sequences over $\mathcal{X}$. In addition, we denote by $t_b \in \mathcal{T}$ and $t_e \in \mathcal{T}$ the vowel onset and offset times, respectively, where $\mathcal{T} = \{1, ..., T\}$. For brevity we set $\boldsymbol{t} = (t_b, t_e)$.

Our goal is to learn a function, denoted f, which takes as input a speech signal $\mathbf{x}$ and returns the pair $\boldsymbol{t}$. The vowel duration can be computed from the output of this function. In other words, $f : \mathcal{X}^* \rightarrow \mathcal{T}^2$ is a function from the domain of all possible CVC speech segments to the domain of all possible onset and offset pairs.

In order to qualify the quality of the prediction we need to define a *measure of performance* or *evaluation metric* between the predicted and the target onset and offset pairs. We denote by $\gamma(\boldsymbol{t}, \hat{\boldsymbol{t}})$ the cost of predicting the pair $\hat{\boldsymbol{t}}$ while the target pair is $\boldsymbol{t}$. Formally, $\gamma : \mathcal{T}^2 \times \mathcal{T}^2 \rightarrow \mathbb{R}$ is a function that gets as input two ordered pairs, and returns a scalar. We assume that $\gamma(\hat{\boldsymbol{t}}, \boldsymbol{t}) \geq 0$ for any two pairs of time sequences, and $\gamma(\boldsymbol{t}, \boldsymbol{t}) = 0$. The cost function we use is:

$$\gamma(\hat{\boldsymbol{t}}, \boldsymbol{t}) = \left[|\hat{t}_b - t_b| - \epsilon_b\right]_+ + \left[|\hat{t}_e - t_e| - \epsilon_e\right]_+, \quad (1)$$

where $[\pi]_+ = \max\{0, \pi\}$, and ϵ_b, ϵ_e are pre-defined constants. The above function measures the absolute differences between the predicted and the manually annotated vowel onsets and offsets. Since the manual annotations are not exact, we allow a mistake of ϵ_b and ϵ_e frames at the vowel onset and offset respectively, and only panelize predictions that varies by more than ϵ_b or ϵ_e frames.

Our training algorithm is based on training set of m examples, $S = \{(\mathbf{x}_1, \boldsymbol{t}_1), \ldots, (\mathbf{x}_m, \boldsymbol{t}_m)\}$, which were manually labeled. See Section 4 for a detailed description of the data. Ideally, we would like to evaluate our predictions against the manually annotated predictions using a cost function that does take into account small variations in annotations. In the next section we show how we find a function f from this set.

3. THE ARCHITECTURE

In this section we describe the two network architectures that can be used as frame-level classifier for vowel detection. We start by presenting the acoustic features we used as the input for both architectures.

We extracted standard MFCC features (with energy, delta and delta-delta), and concatenate to each frame the previous and the next two frames. We also added the normalized pitch [9] as an additional feature. The features were extracted in frames of 10 msec and spanned a window of 25 msec. Overall we had d = 196 features (5×39 MFCC + 1 pitch).

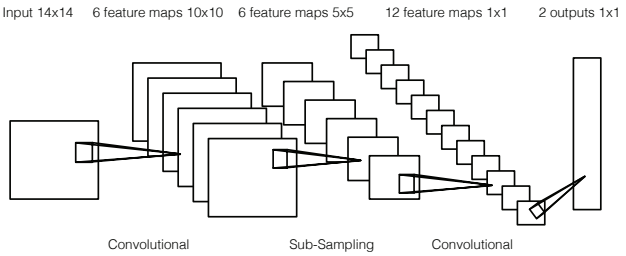


Fig. 1. *CNN architecture*

3.1. Convolutional neural network

Our first architecture is based on a variation of LeNet1 CNN [7]. The input of the network is a matrix $\mathbf{x} \in \mathbb{R}^{14\times14}$ (we reshape the input 196 features into a 14×14 matrix), and the output is a number $y \in [0, 1]$, indicating the probability that a given speech frame is a vowel. Our network is composed from four learned layers: two convolutional layers, a sub-sampling layer and an output layer.

The first layer is a convolutional layer which has 6 feature maps. Those are connected to the input layer using 6 kernels of size 5×5. The second layer is a sub-sampling layer, we use a 2×2 mean-polling-layer. The third layer has 12 feature maps which are fully connected to all 6 mean-pooling-layer using 72 kernels of size 5×5. Finally, we feed the output layer with the output of the third layer. The output layer consists two neurons corresponding to the occurrence of the vowel on this frame. A simplified description of the networks can be viewed in Figure 1.

The network is trained so as to minimize the mean square error using stochastic gradient descent with mini-batches of size m_b, namely,

$$\frac{1}{m_b}\sum_{i=1}^{m_b}(y_i - \hat{y}_i)^2, \quad (2)$$

where $y_i \in \{0, 1\}$ is 1 if the frame i is annotated as a vowel and 0 otherwise, and $\hat{y}_i \in (0, 1)$ is the network prediction. The network was trained on 36,000 frames, a mini-batch size of m_b=50, fixed learning rate equal to 1 and 100 epochs. Our implementation is based on a modified versions of the code in [10]. All parameters were chosen on a validation set.

3.2. Deep belief network

Our second architecture is based on DBN. The network is composed of five layers: an input layer, 3 hidden layers, and an output layer. The input layer is composed of 196 inputs. The first and second hidden layers are composed of 500 hidden units, the third hidden layer is composed of 2000 hidden units and the output layer was uses softmax activation function. A simplified description of the networks can be viewed in Figure 2.

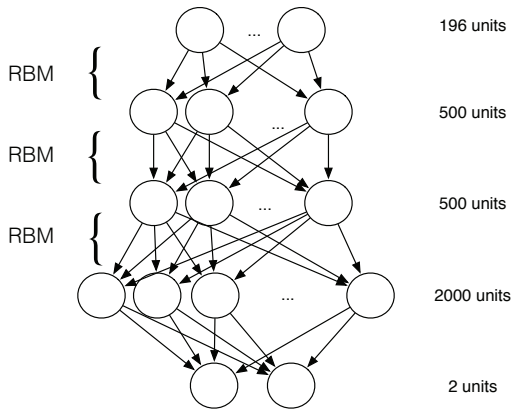


Fig. 2. *DBN architecture*

Due to the highly non-linear functions involved, DNNs are difficult to train directly by stochastic gradient descent. Hence, each layer is pre-trained in an unsupervised way to model the previous layer expectation. In this work, we use restricted Boltzmann machines (RBM) [11] to model the joint distribution of the previous layers units in a DBN [12]. The network is trained so as to minimize the cross entropy using stochastic gradient descent with a line search:

$$-\sum_{i=1}^{m_b} y_i \log \hat{y}_i, \tag{3}$$

where y_i and $\hat{y}_i$ are the i-th frame annotation and the network prediction, respectively. We pre-trained the system with 50 epochs per each layer, and 100 epochs for fine-tuning feed-forward training. The network was trained on 36,000 frames, with a mini-batch size of m_b=100, and a learning rate was 0.1. Our implementation is based on the code in published by R. Salakhutdinov and G. Hinton[1]. All parameters were chosen on a validation set.

3.3. Smoothing

We converted the network output from a vector of probabilities into a pair of vowel onset and offset. We rounded the probability of each frame to be either zero or one. Then we smoothed the resulted sequence to have a single chunk, representing a continues vowel.

Notice that the function each network is trained to minimize is different then in the desired cost function in (1). The difference is due to the fact the the network examine single frames at a time, and not build to work with variable length inputs. Nevertheless, it turns out that while minimizing (2) or (3) we get also good results when measuring the performance using (1).

[1] http://www.cs.toronto.edu/~hinton/MatlabForSciencePaper.html

4. DATASET

The dataset used in our experiments contains speech segments that were composed from 64 native English speakers (55 female) aged 18-34 with no history of speech or language deficits. To obtain the recordings, participants were asked to name aloud the noun depicted by a picture in two different ways: with the picture present alone and when the picture occurred at the end of a sentence. They were instructed to produce the name as quickly as possible. To avoid misunderstanding the participants were first familiarized with the pictures. Each recorded segment contains one English CVC noun, where the list of vowels are: /i, ɛ, ae, ɑ, o, u/. The total size of the data set is 2684 CVC speech segments.

Vowel duration was hand-measured in Praat [13]. Vowel onsets and offsets were marked using cues from the waveform and spectrogram [14]. A second coder marked 25 % of the data to assess measurement reliability. Measurements were well correlated, $r(627) = .84$, $p < .0001$. A detailed description of the dataset can be found in [3].

5. EXPERIMENTAL RESULTS

We compare the performance of both DNN architectures and an HMM-based forced aligner to manual benchmark data from picture naming of English monosyllabic words describe in the previous section.

Forced alignment is an algorithm to align a sequence of phonemes (or words) to the speech signal. The force alignment gets as input a speech signal and a sequence of phonemes (or words) uttered in the signal. The output of the aligner is the location of each phoneme in the speech. The forced alignment is based on an HMM trained as a phoneme recognizer. In our experiments we used Penn Aligner [15], which is HMM-based phoneme aligner trained with the HTK toolkit[2].

5.1. Baseline

First we compare the results of CNN and DBN to HMM on the cost defined in (1). The result are presented in Table 1. Each row in the table, corresponds to different values of ϵ_b and ϵ_e, and the results of the deviations in msec of the onset and the offset. For comparison, the inter-transcriber deviations are on average 3.5 msec for the onset and 20 msec for the offset (ϵ_b and ϵ_e are both 0).

It can be seen from the table that the performance of HMM is slightly better than the performance of CNN except for the onset in the higher values of ϵ_b and ϵ_e and much better then the results of the DBN. Nevertheless, the CNN and DBN does not need as input the sequence of uttered phonemes, while the HMM does.

[2] http://htk.eng.cam.ac.uk

Table 1. *Results of CNN, DBN and HMM relative to manual annotation. Average deviation of onset and offset for different values of ϵ_b and ϵ_e [in msec].*

		CNN		DBN		HMM	
ϵ_b	ϵ_e	onset	offset	onset	offset	onset	offset
0	0	21	36	62	106	15	25
10	15	20	33	61	102	12	20
20	40	7.3	22	60	98	9	12
30	50	3.5	19	55	97	6	10

Since both network architectures are not aimed to minimize (1), we give here the misclassification error rate as well. The CNN reaches misclassification rate of 4.57% while the DBN reaches 22%, both of these results were measured on the test set.

In the next subsections we perform a deeper analysis on the results of CNN comparing them manually labeled ones. We choose to preform the analysis on the CNN predictions due to the poor results of the DBN.

5.2. Measurement Deviation

Figure 3 provides a comparison of the vowel durations from the manual annotators (y-axis) vs. each algorithm (x-axis).

The mean squared error relative to the manual benchmark was slightly higher for CNN (2.4 msec^2) vs. HMM (1.8 msec^2). To assess the reliability of this difference, the distribution of differences was estimated using 1,000 bootstrap samples (created by resampling the observed differences in mean squared error across the methods). The mean difference of 0.6 msec^2 was reliable (95% confidence interval: [0.3, 0.9]).

5.3. Model-Based Comparison

As noted in the introduction, vowel durations are used in studies addressing specific phonetic issues. These typically utilize vowel durations as a dependent measure, examining how duration is modulated by properties of: the context in which vowels appear; the individuals producing the vowels; and the particular stimuli that were used to elicit the productions. Statistical models are used to assess the importance of these factors. To illustrate, the study that provides the benchmark data used here [3] examined how placing words in a context that strongly emphasizes processing of sentence structure would influence speech articulation. Speakers named a set of pictures both in isolation (where there is no need to process sentence structure) as well as following a sentence frame (strongly emphasizing the processing of sentence structure). In the speech field, hierarchical mixed-effects regressions [16] are the current state-of-the-art analytic technique for assessing the reliability of such effects. These allow estimation of the effect of the variable of interest (e.g., context of naming) while controlling for other properties. In this case, the analysis [17] revealed that context had a reliable effect (such that vowel durations were shorter when the picture was named following a sentence context vs. in isolation).

Given the critical role that such statistical models play in utilizing vowel duration data, our second evaluation method to compares the properties of statistical models fit to manually annotated data vs. data obtained from CNN or HMM.

5.3.1. Regression Model Structure

The source study here [3] manipulated two factors: the production context (picture naming in isolation vs. following a sentence) and the number of words phonologically similar to the target that share its grammatical category (*lexical density*). To account for these factors, the model included a centered density measure which interacted with a contrast-coded fixed effect reflecting production context (isolation vs. sentence). Additional contrast-coded factors controlled for block in which the picture was presented (first vs. second), as well as the target vowel identity.

To control for idiosyncratic contributions from the random sample of speakers (e.g., the specific individuals tested here vs. all English speakers) and stimuli (e.g., the specific words used here vs. all words of English), the model also included crossed random effects. These included random intercepts for participants and words, along with uncorrelated random slopes for context and density by participant and context by word.

To control for skew in the dependent measure, vowel durations were log transformed prior to analysis. To control for outliers, all models were refit after excluding observations with standardized residuals greater than 2.5 [18]. To assess whether a given factor made a significant contribution to variance in vowel duration, the likelihood ratio test was used to compare models with vs. without the factor [19].

5.3.2. Comparison of Model Parameters

Table 2 provides the parameter estimates for fixed effects. Several features of the manual annotation results are reflected in models fit by both annotation methods. Most parameters have the same sign. All methods show a significant shortening for naming in sentence contexts vs. isolation ($\chi^2(1)s > 7.4, ps < .01$).

There are also significant divergences. While all models show that durations are longer in the second block, only CNN found a (marginally) significant effect ($\chi^2(1)s > 3.81, ps < .051$). Similarly, while all models show shorter durations for vowels in words with more lexical neighbors, only the HMM found a significant effect ($\chi^2(1)s > 5.63, ps < .02$). The HMM also found significant effects of two factors related to vowel identity ($\chi^2(1)s > 4.15, ps < .05$). Thus, by this

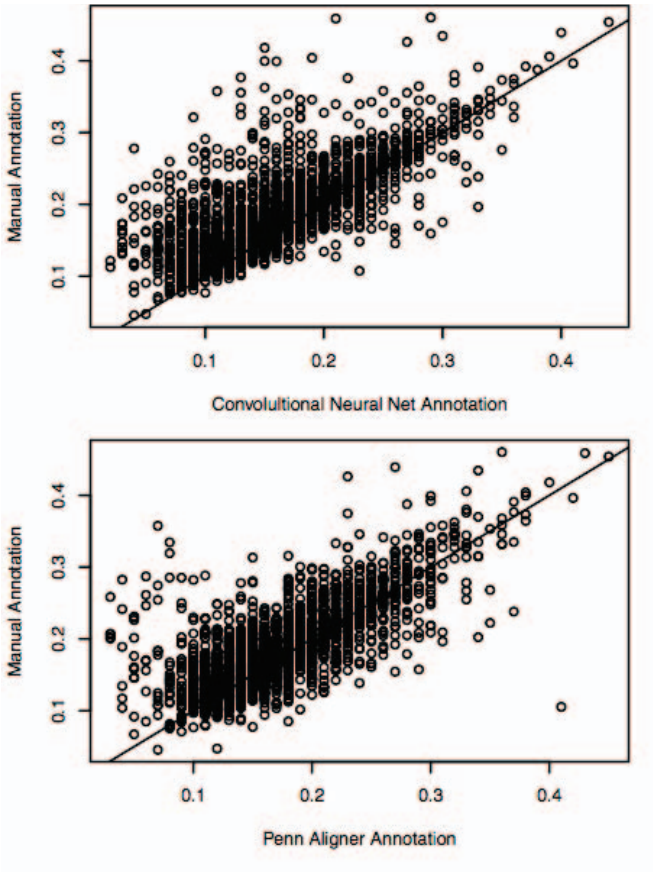


Fig. 3. *Scatterplot of algorithm vs. manual annotation vowel durations (in seconds)*

method of drawing inferences from statistical models, CNN lead to one false positive effect while HMM leads to three false positives (out of 10 possible effects).

5.3.3. Comparison of Model Predictions

While phonetic studies typically draw inferences based on assessments of individual predictors, an alternative means of assessing such models is examining their predictions on novel data. To generate such predictions, we performed 4-fold cross-validation. We maintained a roughly even balance of observations within each participant across high vs. low density items (using a median split) and production contexts.

Figure 4 provides a comparison of model predictions. The mean squared error, relative to the manual model, showed that models fit to the automated data showed substantial deviations in predictions. The model fit to CNN annotations showed a mean squared error of 334 log msec^2, significantly greater than that of the HMM model (65 log msec^2; bootstrapped 95% CI for difference [237, 302]). While both models show divergence from the manual model, the HMM model's predictions are more similar to than those of the CNN model.

Table 2. *Table 1. Estimates for the effect of each fixed parameter on log vowel duration (in seconds).* ***Bolded*** *parameters make significant contributions to variance.*

		Estimate		
	Parameter	Manual	CNN	HMM
	Intercept	-1.7046	-1.861	-1.7768
	Production Context	**-0.0561**	**-0.1414**	**-0.071**
	Lexical Density	-0.0117	-0.0075	**-0.0151**
	Density × Context	0.0002	-0.0001	-0.004
	Block	0.0109	**0.0379**	0.0096
Vowel	ɑ vs. æ	0.0664	0.1232	0.035
Vowel	ɛ vs. ɑ, æ	-0.0538	-0.0307	**-0.0584**
Vowel	i vs. ɛ,ɑ,æ	-0.0393	-0.0429	**-0.0346**
Vowel	o vs. i,ɛ,ɑ,æ	0.0025	-0.0117	0.0091
Vowel	u vs. all others	0.0005	-0.0052	-0.022

6. DISCUSSION AND FUTURE WORK

While both HMM and CNN annotations yield duration values that are, on average, quite similar to manual annotations, neither method yields the same conclusions as would be drawn from manually-annotated data. In our test set, both algorithms recovered an effect that was reliable in manually-annotated data; however, both also yielded false positives. Moreover, both automatic annotation methods yielded different predictions than models fit to manually annotated data. Neither method was clearly superior in terms of matching the manual annotations; CNN resulted in fewer incorrect inferences regarding significant effects, but the HMM model predictions were clearly more like those of models fit to manually annotated data.

We believe these results are promising since they show that neural networks, especially CNNs, can reach performance comparable to HMM-based models with no need for phonetic transcription. However, we would emphasize that these are initial results for the CNN and there is much room for exploring new architectures in the neural network field such as deeper networks with more layers or hidden units. The most promising direction would probably be focused on recurrent neural networks.

Acknowledgements: Funded in part by NIH-NICHD grant HD077140.

7. REFERENCES

[1] Benjamin Munson and Nancy Pearl Solomon, "The effect of phonological neighborhood density on vowel articulation," *Journal of speech, language, and hearing research*, vol. 47, no. 5, pp. 1048–1058, 2004.

[2] Susanne Gahl, Yao Yao, and Keith Johnson, "Why reduce? phonological neighborhood density and phonetic

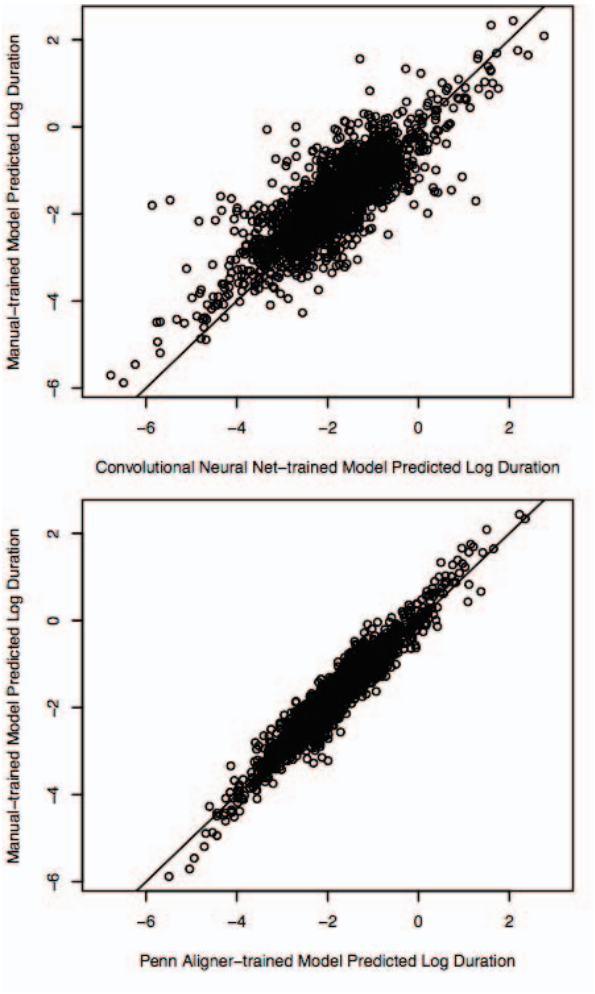


Fig. 4. *Scatterplot of predicted vowel durations (in log seconds) from regression models fit to algorithm vs. manual annotation*

reduction in spontaneous speech," *Journal of Memory and Language*, vol. 66, no. 4, pp. 789–806, 2012.

[3] Jordana R Heller and Matthew Goldrick, "Grammatical constraints on phonological encoding in speech production," *Psychonomic bulletin & review*, vol. 21, no. 6, pp. 1576–1582, 2014.

[4] William Labov, Ingrid Rosenfelder, and Josef Fruehwald, "One hundred years of sound change in philadelphia: Linear incrementation, reversal, and reanalysis," *Language*, vol. 89, no. 1, pp. 30–65, 2013.

[5] Jiahong Yuan and Mark Liberman, "F0 declination in english and mandarin broadcast news speech.," in *INTERSPEECH*. Citeseer, 2010, pp. 134–137.

[6] Keelan Evanini, *The permeability of dialect boundaries: A case study of the region surrounding Erie*, Ph.D. thesis, University of Pennsylvania, 2009.

[7] Yann LeCun, LD Jackel, L Bottou, A Brunot, C Cortes, JS Denker, H Drucker, I Guyon, UA Muller, E Sackinger, et al., "Comparison of learning algorithms for handwritten digit recognition," in *International conference on artificial neural networks*, 1995, vol. 60, pp. 53–60.

[8] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al., "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *Signal Processing Magazine, IEEE*, vol. 29, no. 6, pp. 82–97, 2012.

[9] David Talkin, "A robust algorithm for pitch tracking," *Speech coding and synthesis*, vol. 495, pp. 518, 1995.

[10] Rasmus Berg Palm, "Prediction as a candidate for learning deep hierarchical models of data," M.S. thesis, Technical University of Denmark, 2012.

[11] Paul Smolensky, "Information processing in dynamical systems: Foundations of harmony theory," 1986.

[12] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh, "A fast learning algorithm for deep belief nets," *Neural comp.*, vol. 18, no. 7, pp. 1527–1554, 2006.

[13] Paul Boersma and David Weenink, "Praat, a system for doing phonetics by computer," 2001.

[14] Gordon E Peterson and Ilse Lehiste, "Duration of syllable nuclei in english," *The Journal of the Acoustical Society of America*, vol. 32, no. 6, pp. 693–703, 1960.

[15] Jiahong Yuan and Mark Liberman, "Speaker identification on the scotus corpus," *Journal of the Acoustical Society of America*, vol. 123, no. 5, pp. 3878, 2008.

[16] R Harald Baayen, Douglas J Davidson, and Douglas M Bates, "Mixed-effects modeling with crossed random effects for subjects and items," *Journal of memory and language*, vol. 59, no. 4, pp. 390–412, 2008.

[17] Jordana R Heller and Matthew Goldrick, "Corrigendum to 'grammatical constraints on phonological encoding in speech production'," *Psychonomic bulletin & review*, 2014.

[18] Rolf Harald Baayen, *Analyzing linguistic data: A practical introduction to statistics using R*, Cambridge University Press, 2008.

[19] Dale J Barr, Roger Levy, Christoph Scheepers, and Harry J Tily, "Random effects structure for confirmatory hypothesis testing: Keep it maximal," *Journal of memory and language*, vol. 68, no. 3, pp. 255–278, 2013.

Automatic Measurement of Voice Onset Time and Prevoicing Using Recurrent Neural Networks

INTERSPEECH 2016
September 8–12, 2016, San Francisco, USA

Automatic Measurement of Voice Onset Time and Prevoicing using Recurrent Neural Networks

Yossi Adi[1]*, Joseph Keshet*[1]*, Olga Dmitrieva*[2]*, and Matt Goldrick*[3]

[1]Department of Computer Science, Bar-Ilan University, Ramat-Gan, Israel
[2]School of Languages and Cultures, Purdue University, West Lafayette, IN, USA
[3]Department of Linguistics, Northwestern University, Evanston, IL, USA

adiyoss@cs.biu.ac.il

Abstract

Voice onset time (VOT) is defined as the time difference between the onset of the burst and the onset of voicing. When voicing begins preceding the burst, the stop is called prevoiced, and the VOT is negative. When voicing begins following the burst the VOT is positive. While most of the work on automatic measurement of VOT has focused on positive VOT mostly evident in American English, in many languages the VOT can be negative. We propose an algorithm that estimates if the stop is prevoiced, and measures either positive or negative VOT, respectively. More specifically, the input to the algorithm is a speech segment of an arbitrary length containing a single stop consonant, and the output is the time of the burst onset, the duration of the burst, and the time of the prevoicing onset with a confidence. Manually labeled data is used to train a recurrent neural network that can model the dynamic temporal behavior of the input signal, and outputs the events' onset and duration. Results suggest that the proposed algorithm is superior to the current state-of-the-art both in terms of the VOT measurement and in terms of prevoicing detection.

Index Terms: voice onset time, prevoicing, recurrent neural networks

1. Introduction

Voice onset time (VOT), the time between the onset of a stop burst and the onset of voicing, is an important cue to stop voicing and place. It is widely measured in theoretical and clinical settings, for example to characterize how communication disorders affect speech [1] or how languages differ in the phonetic cues to stop contrasts [2, 3]; it is also increasingly used as a feature for automatic speech recognition (ASR) tasks such as stop consonant classification [4, 5, 6]. Automatic VOT measurement would be very beneficial for clinical and theoretical studies, where it is currently usually measured manually, and is essential for ASR applications.

Several recent studies have proposed VOT measurement algorithms [5, 6, 7, 8, 9, 10],[1] all making the assumption that VOT is always positive (burst onset precedes voicing onset). However, this assumption is well known to be false. VOT can in general also be negative (voicing onset precedes burst onset), in which case the stop is "prevoiced." In English, for example, voiceless stops (/p/, /t/, /k/) always have positive VOT, while voiced stops (/b/, /d/, /g/) can have positive or negative VOT [11]. In other languages (e.g., Dutch, French, Spanish), voiced stops usually have negative VOT, while voiceless stops have positive VOT [12, 11].

We are aware of only a single work [13] that handles both positive and negative VOTs by extending [8]. In that work two parallel classifiers were jointly trained: one for measuring positive VOTs and one for measuring negative VOTs. The classifiers operated on two sets of customized features based on spectro-temporal cues to the location of the burst and voicing onsets in the positive and negative VOT cases.

Current algorithms that focus on positive VOT solve two challenges in VOT measurement: detection of the onset of the burst and the onset of the voicing of the vowel. We extend these algorithms by addressing two additional challenges: determining whether or not prevoicing is present, and, when it is present, the onset of prevoicing. To simultaneously address all four challenges, we develop an algorithm that identifies up to four regions in each input utterance:

1. **Silence**: From utterance onset to prevoicing onset
2. **Prevoicing**: From prevoicing onset to burst onset
3. **Burst/Aspiration**: From burst onset to onset of voicing of vowel
4. **Vowel**: From onset of vowel voicing to end of utterance

We train a multiclass recurrent neural network to classify each frame of the input utterance as part of each region. We then use a dynamic programming algorithm to find the best segmentation of the utterance based on the classifier predictions, yielding the desired time points for calculating VOT.

Below, we outline our approach. We then assess its performance, first in the well-studied problem of positive VOT measurement and then in the less well studied case of measurement of prevoicing. We show that our algorithm out performs state-of-the-art alternatives in both cases, suggesting that it can provide a solution to the general problem of VOT measurement.

2. Problem definition

The input to our algorithm is a speech utterance containing a single stop consonant, and the output is the voice onset time (VOT), that is, the time difference between the onset of the burst and the onset of voicing. When voicing begins preceding the burst, the output is the time difference between the onset of the prevoicing and the onset of the burst. The input utterance can be of an arbitrary length, and its beginning need not be synchronized with the prevoicing (if exists), the burst onset, the

[1]This list is not exhaustive, due to space considerations.

Copyright © 2016 ISCA

http://dx.doi.org/10.21437/Interspeech.2016-893

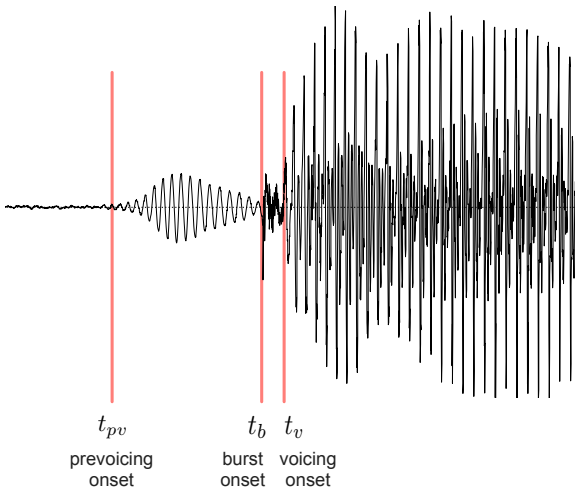


Figure 1: *Annotation example for prevoicing, burst and voicing onsets. The spoken word in this wav form is "dug."*

voicing onset, or the closure. It is required that the input utterance includes the burst, part of the vowel and the whole region of prevoicing (if exists).

Let $\bar{\mathbf{x}} = (\mathbf{x}_1, \ldots, \mathbf{x}_T)$ denotes the input speech utterance, represented as a sequence of acoustic feature vectors, where each $\mathbf{x}_t \in \mathbb{R}^D$ $(1 \leq t \leq T)$ is a D-dimensional vector. The length of the speech utterance, T, is not a fixed value since the input utterances can have different durations.

Each input utterance is associated with three elements: the prevoicing onset, $t_{pv} \in \mathcal{T}$, the onset of the burst, $t_b \in \mathcal{T}$, and the onset of the voicing of the vowel, $t_v \in \mathcal{T}$, where $\mathcal{T} = \{1, \ldots, T\}$, and $t_{pv} < t_b < t_v$. In the case of positive lag stops the prevoicing onset does not exist and t_{pv} is assigned to be -1, and the VOT is $t_v - t_b$, whereas in the case of negative lag (prevoiced) stops, all the three elements are defined and the VOT is $t_b - t_{pv}$. Our notation is depicted in Figure 1.

3. Learning apparatus

3.1. Features

Seven (D=7) acoustic features are extracted from the speech signal every 1 ms [8]. The first five features refer to an STFT taken with a 5 ms Hamming window: the total spectral energy (E_{total}), energy between 50–1000Hz (E_{low}), energy above 3000 Hz (E_{high}), Wiener entropy (H_{wiener}), and the number of zero crossings of the signal (ZC). Features 6–7 are the maximum of the FFT of the autocorrelation function of the signal from 6 ms before to 18 ms after the frame center (R_l), and a binary voicing detector based on the RAPT pitch tracker [14], smoothed with a 5 ms Hamming window (V).

In addition, we also use the cumulative mean, differences and max of these features similar to the feature functions used in [8] as another input to the classifier. These feature maps were chosen by empirical examination of the spectra and waveform of voiced stops with and without prevoicing. Overall we have 63 features per frame.

3.2. Recurrent neural network

One approach to determining the duration of a phonetic property is to predict at each time frame whether the property is present or absent; the predicted duration is then the smoothed, continuous set of frames where the property is likely to be present [15]. In this work we extend this method, generating predictions using a Recurrent Neural Network (RNN). This allows the prediction of whether a property is present to be sensitive to the relationship between frames.

We implement a network of two-layers of stacked LSTMs [16], which has shown considerable success in analyzing dynamic temporal behavior [17, 18]. We use an in-house implementation that is based on the Torch7 toolkit [19, 20]. Formally, the implementation is the following set of recursive equations, where the weights and the biases are denoted by $\mathbf{W}$ and $\mathbf{b}$, respectively, and σ is the sigmoid function:

$$\mathbf{i}_t = \sigma(\mathbf{W}_{xi}\mathbf{x}_t + \mathbf{W}_{hi}\mathbf{h}_{t-1} + \mathbf{W}_{ci}\mathbf{c}_{t-1} + \mathbf{b}_i) \quad (1)$$

$$\mathbf{f}_t = \sigma(\mathbf{W}_{xf}\mathbf{x}_t + \mathbf{W}_{hf}\mathbf{h}_{t-1} + \mathbf{W}_{cf}\mathbf{c}_{t-1} + \mathbf{b}_f) \quad (2)$$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tanh(\mathbf{W}_{xc}\mathbf{x}_t + \mathbf{W}_{hc}\mathbf{h}_{t-1} + \mathbf{b}_c) \quad (3)$$

$$\mathbf{o}_t = \sigma(\mathbf{W}_{xo}x_t + \mathbf{W}_{ho}\mathbf{h}_{t-1} + \mathbf{W}_{co}\mathbf{c}_t + \mathbf{b}_o) \quad (4)$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t) \quad (5)$$

The input to the RNN classifier is a sequence of T tuples, where each tuple is composed of the acoustic features $\mathbf{x}_t$ and a corresponding label y_t from the set $\mathcal{Y}$ ={silence, prevoicing, burst, vowel} for $1 \leq t \leq T$ as follows:

$$y_t = \begin{cases} \text{silence} & 1 \leq t < t_{pv} \\ \text{prevoicing} & t_{pv} \leq t < t_b \\ \text{burst} & t_b \leq t < t_v \\ \text{vowel} & t_v \leq t \leq T \end{cases} . \quad (6)$$

We trained a multiclass RNN to predict the label of each frame, and optimized the negative log-likelihood using Adagrad [21] with learning rate of 0.1 and batch size of 32 examples. We used two dropout layers after each LSTM with dropout rate of 0.8. We stoped training the network after 5 epochs with no loss improvement on the validation set.

3.3. Inference

The Multiclass RNN outputs a probability for each class. At inference time, we use these probabilities to predict the most likely segmentation of the utterance. Since the predictions can be noisy, and we require a smooth prediction, we use a dynamic programming algorithm to infer the best segmentation. This procedure is described in Figure 2. Denote by $\hat{P}(y_t|\mathbf{x}_t)$ the predicted probability of the network for the input $\mathbf{x}_t$ and class y_t, and denote by T_m the maximum allowable size of each segment. Given $y = \mathcal{Y}$ be the events in the utterance, and two time indices $t, t' \in \mathcal{T}$, denote by $D(y, t, t')$ the score for the prefix of the events sequence: silence,$\ldots, y$, assuming that their actual onsets are $1, t_{pv}, \ldots, t'$, and assuming that $y_{i+1} = t$. The best sequence of actual onsets is obtained from the algorithm by saving the intermediate values that maximize each expression in the recursion step.

4. Experiments

In order to have better understanding on the capabilities of the proposed model we divide the analyses into two sections. First,

Initialization:

for y =[silence]

$$D_{\text{neg}}(y,t,0) = \hat{P}(y|\mathbf{x}_t) \quad 1 \le t \le T_m$$

$$D_{\text{pos}}(y,t,0) = \hat{P}(y|\mathbf{x}_t) \quad 1 \le t \le T_m$$

Recursion:

for y =[prevoicing, burst, vowel]

$$D_{\text{neg}}(y,t',t'') = \max_{t'''} \sum_{t=t''}^{t'} \hat{P}(y|\mathbf{x}_t) + D_{\text{neg}}(y{-}1,t''',t'')$$

for y =[burst, vowel]

$$D_{\text{pos}}(y,t',t'') = \max_{t'''} \sum_{t=t''}^{t'} \hat{P}(y|\mathbf{x}_t) + D_{\text{pos}}(y{-}1,t''',t'')$$

Termination: for y =[vowel]

$$D^\star = \max_{t'} \left\{ D_{\text{neg}}(y,T,t'), D_{\text{pos}}(y,T,t') \right\}$$

Figure 2: *Dynamic programming algorithm for post-process inference.*

we trained the network to measure only positive VOT and compared it to the current state of the art algorithm. We then trained the network to measure positive and negative VOT jointly and compared it to the current state of the art algorithm.

4.1. Positive VOT

To evaluate the performance of our model in measuring positive VOT we used data from 9 speakers drawn from the Northwestern University community [22]. Participants read aloud tongue twisters consisting of alternating pairs of voiced and voiceless consonants (e.g., *pin bin bin pin*). Recordings were randomly assigned to two highly trained coders. VOT was coded via inspection of the waveform, from burst to onset of periodicity in the vowel. Reliability (n = 257 tokens from 5 participants) was very high (*r = 0.996*).

We trained the network on data from 4 speakers (7,654 acoustic segments), with 15% from the data for validation, and tested on data from the remaining 5 speakers (8,628 acoustic segments). Overall we used 504,790 frames for training, 89,080 frames for validation and 143,458 frames for test. The dataset is roughly balanced with respect to the number of VOT and none-VOT frames. We denote our system as *DeepVOT*. The same dataset with the same data split was used to train the algorithm in [8], denoted *AutoVOT*. Table 1 summarizes the distribution of automatic/manual differences over the test set.

Results suggests that our algorithm is superior to the AutoVOT algorithm; DeepVOT exhibits smaller deviations from manual measurements. This is a non-trivial improvement, especially when the tolerance value, t, is small, i.e. 2 or 5 msec.

To see if the system suffers a decline in results when using a model that was trained on one dataset but tested on a different one, we evaluated this trained DeepVOT system on a new data set. We examined positive-lag VOTs from 16 native English speakers at Purdue University who read aloud a list of printed words three times. Recordings were randomly assigned to four trained coders. The VOT intervals were coded via inspection of the waveform and the spectrogram of word- initial stops. Positive VOT was measured from the onset of burst until the onset

Table 1: *Proportion of differences between automatic and manual measures falling at or below a given tolerance value (in msec). For example, for* DeepVOT*, in 75.3% of examples in the test set the difference between automatic and manual measurements was 2 msec or less.*

Model		$t \le 2$	$t \le 5$	$t \le 10$	$t \le 15$	$t \le 25$	$t \le 50$
AutoVOT	mean	50.5	79.1	91.7	94.4	96.8	98.8
	std	4.5	4.7	2.6	1.9	1.2	0.6
DeepVOT	mean	75.3	91.9	95.9	97.1	98.2	99.1
	std	9.4	3.4	1.6	1.1	0.9	0.7

Table 2: *Performance when the system was trained on data from participants at Northwestern University and tested on a second dataset from Purdue University. Proportion of differences between automatic and manual measures falling at or below a given tolerance value (in msec).*

Type	$t \le 2$	$t \le 5$	$t \le 10$	$t \le 15$	$t \le 25$	$t \le 50$
Voiced	63.1	91.9	96.9	98.3	99.3	100
Voiceless	56.5	81.6	86.8	87.2	87.3	89.0

of periodicity in the vowel. All segmentations were inspected by a fifth, highly trained coder and corrected if needed. It can be seen from Table 2 that system performance was quite high even when testing on a novel dataset.

4.2. Negative VOT (prevoicing)

Next, we investigated the performance of our algorithm regarding negative VOT (prevoicing) measurement. We use the data set from a study of isolated word productions in picture naming and reading aloud by L1 English speakers and L1 Portuguese/L2 English bilinguals from the Northwestern University community [23]. All tokens were measured by one highly trained coder. Prevoicing, burst, and onset of periodicity in the vowel were coded via inspection of the waveform. Reliability was assessed by a second trained coder who measured 958 tokens; agreement was very high (*r = 0.972*).

We used a subset of this data consisting of 1446 word-initial voiced stops produced by 10 speakers (3 monolingual, 7 bilingual), evenly split between prevoiced and short-lag VOT. We used 1074 acoustic segments for a training set, with 15% of these used as validation set (146,254 frames for training set, 25,809 frames for validation set). The test set contained 372 acoustic segments (60,881 frames). Prevoiced and short-lag were evenly sampled in training, test and validation sets.

The network did extremely well at detecting prevoicing, with accuracy rate of 97.8%, precision rate of 95.9% and recall rate of 100%. To evaluate performance in measuring VOT, we report results of the percentage of test examples where automatic and manual VOT measurements differed by less than a series of time thresholds. For this analysis, in cases where the manual and network disagree in the presence of prevoicing, the duration of VOT was set by the following rule:

- If the network classifies the input as negative VOT, but the manual annotation was positive, we consider the pre-

Table 3: *Performance on dataset including prevoicing. Proportion of differences between automatic and manual measures falling at or below a given tolerance value (in msec), where* (c) *and* (a) *stand for* correct *and* all*, respectively.*

Model	Type	$t \leq 2$	$t \leq 5$	$t \leq 10$	$t \leq 15$	$t \leq 25$	$t \leq 50$
AutoVOT	neg (c)	53.9	77.1	92.7	96.0	98.8	100
	neg (a)	49.4	70.8	85.2	88.2	91.0	95.3
	pos (c)	53.2	84.4	97.2	98.3	98.7	99.0
	pos (a)	47.9	75.9	87.5	88.6	89.4	95.1
DeepVOT	neg (c)	63.5	78.1	91.0	95.0	98.9	100
	neg (a)	60.7	75.8	89.8	94.6	98.4	100
	pos (c)	80.1	95.7	98.4	98.9	100	100
	pos (a)	80.1	95.7	98.4	98.9	100	100

voicing duration as the VOT.

- If the network classifies the input as positive VOT, but the manual annotation was negative, we consider the burst duration as the VOT.

We compared our result to the state-of-the-art results on this dataset, reported in [13], provide a baseline for the DeepVOT algorithm's performance. The results are summarized in Table 3.

5. Discussion

We have presented a new system for detecting positive and negative VOTs. Our method is based on sequential deep learning, which allows us to use the same learning framework and the same set of feature set for measuring both positive and negative VOTs. For future work we would like to explore the option of optimizing the network end-to-end including the dynamic programming post-processing. Such optimization may further improve the accuracy of such networks.

This approach opens up the possibility of extending automatic analysis of VOT beyond prototypical English productions to cover the many languages that consistently utilize prevoicing. DeepVOT will be publicly available at `https://github.com/MLSpeech/DeepVOT`.

6. Acknowledgements

Research supported in part by NIH grant 1R21HD077140.

7. References

[1] P. Auzou, C. Ozsancak, R. Morris, M. Jan, F. Eustache, and D. Hannequin, "Voice onset time in aphasia, apraxia of speech and dysarthria: a review," *Clin. Linguist. Phonet.*, vol. 14, pp. 131–150, 2000.

[2] T. Cho and P. Ladefoged, "Variation and universals in VOT: evidence from 18 languages," *J. Phon.*, vol. 27, pp. 207–229, 1999.

[3] L. Lisker and A. Abramson, "A cross-language study of voicing in initial stops: acoustical measurements," *Word*, vol. 20, pp. 384–422, 1964.

[4] P. Niyogi and P. Ramesh, "The voicing feature for stop consonants: Recognition experiments with continuously spoken alphabets," *Speech Commun.*, vol. 41, pp. 349–367, 2003.

[5] V. Stouten and H. van Hamme, "Automatic voice onset time estimation from reassignment spectra," *Speech Commun.*, vol. 51, pp. 1194–1205, 2009.

[6] J. Hansen, S. Gray, and W. Kim, "Automatic voice onset time detection for unvoiced stops (/p/,/t/,/k/) with application to accent classification," *Speech Commun.*, vol. 52, pp. 777–789, 2010.

[7] M. Sonderegger and J. Keshet, "Automatic discriminative measurement of voice onset time," in *Proc. of Interspeech*, 2010, pp. 2961–2964.

[8] ——, "Automatic measurement of voice onset time using discriminative structured predictiona)," *The Journal of the Acoustical Society of America*, vol. 132, no. 6, pp. 3965–3979, 2012.

[9] C. Lin and H. Wang, "Automatic estimation of voice onset time for word-initial stops by applying random forest to onset detection," *J. Acoust. Soc. America*, vol. 130, pp. 514–525, 2011.

[10] A. Prathosh, A. Ramakrishnan, and T. Ananthapadmanabha, "Estimation of voice-onset time in continuous speech using temporal measures," *The Journal of the Acoustical Society of America*, vol. 136, no. 2, pp. EL122–EL128, 2014.

[11] O. Dmitrieva, F. Llanos, A. A. Shultz, and A. L. Francis, "Phonological status, not voice onset time, determines the acoustic realization of onset f0 as a secondary voicing cue in spanish and english," *Journal of Phonetics*, vol. 49, pp. 77–95, 2015.

[12] P. M. van Alphen and R. Smits, "Acoustical and perceptual analysis of the voicing distinction in Dutch initial plosives: The role of prevoicing," *J. Phonetics*, vol. 32, pp. 455–491, 2004.

[13] K. Henry, M. Sonderegger, and J. Keshet, "Automatic measurement of positive and negative voice onset time." in *INTERSPEECH*, 2012, pp. 871–874.

[14] D. Talkin, "A robust algorithm for pitch tracking (RAPT)," in *Speech coding and synthesis*, W. Kleijn and K. Paliwal, Eds. New York: Elsevier, 1995, pp. 495–518.

[15] Y. Adi, J. Keshet, and M. Goldrick, "Vowel duration measurement using deep neural networks," in *Machine Learning for Signal Processing (MLSP), 2015 IEEE 25th International Workshop on*. IEEE, 2015, pp. 1–6.

[16] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[17] A. Graves and N. Jaitly, "Towards end-to-end speech recognition with recurrent neural networks," in *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, 2014, pp. 1764–1772.

[18] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 2013, pp. 6645–6649.

[19] R. Collobert, K. Kavukcuoglu, and C. Farabet, "Torch7: A matlab-like environment for machine learning," in *BigLearn, NIPS Workshop*, no. EPFL-CONF-192376, 2011.

[20] N. Léonard, S. Waghmare, and Y. Wang, "rnn: Recurrent library for torch," *arXiv preprint arXiv:1511.07889*, 2015.

[21] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *The Journal of Machine Learning Research*, vol. 12, pp. 2121–2159, 2011.

[22] M. Goldrick, H. R. Baker, A. Murphy, and M. Baese-Berk, "Interaction and representational integration: Evidence from speech errors," *Cognition*, vol. 121, no. 1, pp. 58–72, 2011.

[23] N. Paterson, "Interactions in bilingual speech processing," Ph.D. dissertation, Northwestern University, 2011.

Automatic Measurement of Vowel Duration via Structured Prediction

Automatic measurement of vowel duration via structured prediction

Yossi Adi and Joseph Keshet
Department of Computer Science, Bar-Ilan University, Ramat-Gan, 52900, Israel

Emily Cibelli and Erin Gustafson
Department of Linguistics, Northwestern University, Evanston, Illinois 60208, USA

Cynthia Clopper
Department of Linguistics, Ohio State University, Columbus, Ohio 43210, USA

Matthew Goldrick[a)]
Department of Linguistics, Northwestern University, Evanston, Illinois 60208, USA

(Received 6 March 2016; revised 4 October 2016; accepted 6 December 2016; published online 27 December 2016)

A key barrier to making phonetic studies scalable and replicable is the need to rely on subjective, manual annotation. To help meet this challenge, a machine learning algorithm was developed for automatic measurement of a widely used phonetic measure: vowel duration. Manually-annotated data were used to train a model that takes as input an arbitrary length segment of the acoustic signal containing a single vowel that is preceded and followed by consonants and outputs the duration of the vowel. The model is based on the structured prediction framework. The input signal and a hypothesized set of a vowel's onset and offset are mapped to an abstract vector space by a set of acoustic feature functions. The learning algorithm is trained in this space to minimize the difference in expectations between predicted and manually-measured vowel durations. The trained model can then automatically estimate vowel durations without phonetic or orthographic transcription. Results comparing the model to three sets of manually annotated data suggest it outperformed the current gold standard for duration measurement, an hidden Markov model-based forced aligner (which requires orthographic or phonetic transcription as an input). © *2016 Acoustical Society of America*. [http://dx.doi.org/10.1121/1.4972527]

[MAHJ] Pages: 4517–4527

I. INTRODUCTION

Understanding the factors that modulate vowel duration has been a long-standing focus of laboratory research in acoustic phonetics (Peterson and Lehiste, 1960). The vast majority of such research—from the mid-20th century to the start of the 21st century—has relied on manual annotation to determine vowel duration. There are two key issues with such an approach. Given the labor intensive nature of such annotation, there are substantial limitations on the amount of data that can be practically analyzed; this limits the statistical power and types of issues that phonetic studies can address. Furthermore, given the fundamental reliance on subjective annotator judgments, analyses cannot be directly replicated by other researchers.

An alternative to this approach is to pursue algorithms for automatic alignment of vowel boundaries. The current standard approach for vowels is to utilize forced alignment algorithms (Yuan *et al.*, 2013). However, this approach suffers from two important limitations: it requires an orthographic transcription, and frequently requires substantial preprocessing of the data to insure adequate performance (Evanini, 2009).

In this paper we propose a method for automatic measurement of the duration of single vowels using structured prediction techniques which have provided excellent results in analyzing other phonetic measures (Keshet *et al.*, 2007; Sonderegger and Keshet, 2012). The algorithm was trained at the segment level on manually annotated data to extract the vowel start and end times; this provides a straightforward way to compute the duration of the vowel. Following the structure of the vast majority of laboratory studies of vowel duration, we assume the input signal contains a single vowel, proceeded and followed by a consonant (CVC)—no additional detailed information about the phonetic transcription is required to process the speech. This tool will allow laboratory studies to rapidly and reliably examine how the duration of vowels in monosyllabic words varies across participants and elicitation contexts.

We evaluated our method on data from three phonetic studies: one focusing on vowel duration (Heller and Goldrick, 2014; HG), the second using vowel segmentation to automatically determine points for formant analysis (Clopper and Tamati, 2014; CT), and the third a standard set of vowel production norms (Hillenbrand *et al.*, 1995; HGCW). We compared results with our model to the state-of-the-art in vowel duration measurement, HMM (hidden Markov model)-based forced alignment (Rosenfelder *et al.*, 2014). We also assessed whether inferential statistical

[a)]Electronic mail: matt-goldrick@northwestern.edu

models fit to data from our model and HMM-based forced alignment replicated the patterns obtained from manual data. The results suggest that our algorithm is superior to the current gold standard at matching the manual measurements of vowel duration, both in terms of deviation and in replicating inferential statistical results.

The paper is organized as follows. In Sec. II we state the problem definition formally. We then present the learning framework (Sec. III), algorithm (Sec. IV), and the acoustic features and feature functions (Sec. V). In Sec. VI we describe the datasets we use to train the models and evaluate the performance of the algorithm. In Sec. VII we detail the particular methods used to implement our algorithm here, along with the standard approach to vowel duration measurement. Experimental results are detailed in two sections: the first focusing on measurement deviation (Sec. VIII) and the second on the reproduction of results from inferential statistical models (Sec. IX). We conclude the paper with possible applications and extensions in Sec. X.

II. PROBLEM SETTING

In the context of a typical laboratory study of speech, the goal of automatic vowel duration measurement is to accurately predict the time difference between the vowel onset and offset, given a segment of the acoustic signal in which the vowel is preceded by a consonant and followed by a consonant. The acoustic sample can be of any length, but should include only one vowel. We assume there is a small portion of silence before and after the uttered speech, but do not require the beginning of the speech signal or the vowel onset to be synchronized with the onset or offset of the acoustic sample.

We turn to describing the problem formally. Throughout the paper we write scalars using lowercase Latin letters, e.g., x, and vectors using boldface letters, e.g., $\boldsymbol{x}$. A sequence of elements is denoted with a bar $\bar{x}$ and its length is written as $|\bar{x}|$. Similarly a sequence of vectors is denoted as $\bar{x}$ and its length by $|\bar{x}|$.

The acoustic sample is represented by a sequence of acoustic feature vectors denoted by $\bar{\boldsymbol{x}} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_T)$, where each $\boldsymbol{x}_t$ $(1 \le t \le T)$ is a d-dimensional vector that represents the acoustic content of the tth frame. The domain of the feature vectors is denoted as $\mathcal{X} \subset \mathbb{R}^d$. The input acoustic sample can be of an arbitrary length, thus T is not fixed. We denote by $\mathcal{X}^*$ the set of all finite length segments of acoustic signal over $\mathcal{X}$. In addition, we denote by $t_b \in \mathcal{T}$ and $t_e \in \mathcal{T}$ the vowel onset and offset times in frame units, respectively, where $\mathcal{T} = \{1, \ldots, T\}$, and the total duration of the input acoustic sample is T frames. For brevity we denote this pair by $\boldsymbol{t} = (t_b, t_e)$, and refer to it as an *onset–offset pair*. Practically there are constraints on t_b and t_e and they cannot take any value in $\mathcal{T}$, e.g., the vowel onset t_b cannot be T or $T-1$. Our notation is depicted in Fig. 1.

Our goal is to find a function f from the domain of segments of acoustic signal, $\mathcal{X}^*$, to the domain of all onset–offset pairs, $\mathcal{T}^2$. Given a segment of the acoustic signal $\bar{\boldsymbol{x}}$, let $\hat{t} = f(\bar{x})$ be the predicted onset–offset pair, where $\hat{\boldsymbol{t}} = (\hat{t}_b, \hat{t}_e)$. The quality of the prediction is assessed using a *loss function*, denote by $\gamma(\boldsymbol{t}, \hat{t})$, that measures the magnitude of the penalty when predicting the pair $\hat{\boldsymbol{t}}$ rather than the target pair $\boldsymbol{t}$. Formally, $\gamma : \mathcal{T}^2 \times \mathcal{T}^2 \to \mathbb{R}_+$ is a function that receives as input two ordered pairs, and returns a positive scalar. We assume that if both the predicted and the target pairs are the same, then the loss is zero, $\gamma(\boldsymbol{t}, \boldsymbol{t}) = 0$.

III. LEARNING FRAMEWORK

We assume that an input acoustic sample and a target onset–offset pair are drawn from a fixed but unknown distribution ρ over the domain of the segments of acoustic signal and the vowel onset–offset pairs, $\mathcal{X}^* \times \mathcal{T}^2$. We define the *risk* of f as the expected loss when using f to predict the onset–offset pair of the acoustic sample $\bar{\boldsymbol{x}}$, that is,

$$R(f) = \mathbb{E}_{(\bar{x},\boldsymbol{t})\sim\rho}[\gamma(\boldsymbol{t}, f(\bar{\boldsymbol{x}}))], \tag{1}$$

where the expectation is taken with respect to the input acoustic sample $\bar{\boldsymbol{x}}$ and the annotated onset–offset pair $\boldsymbol{t}$ drawn from ρ. Our goal is to find f that minimizes the risk. Unfortunately, this cannot be done directly since ρ is unknown. Instead, we use a training set of examples that served as a restricted window through which we can estimate the quality of the prediction function according to the distribution of unseen examples in the real world. The examples

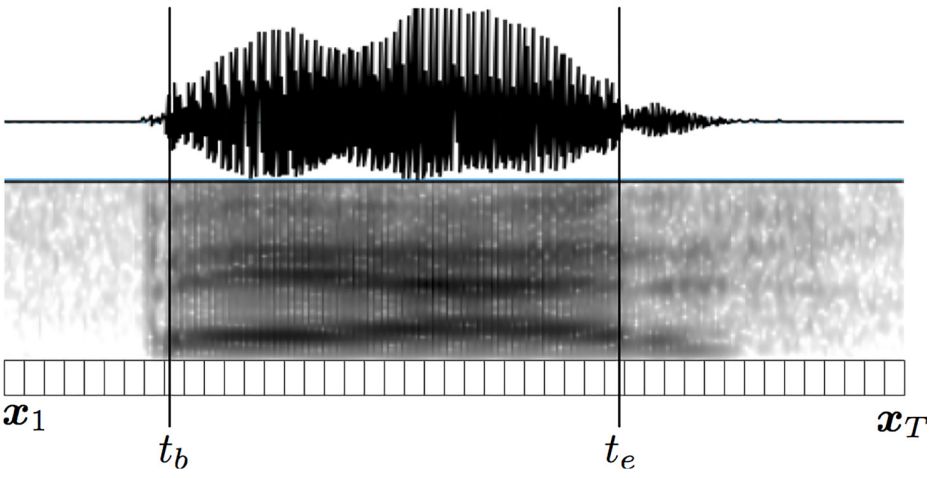


FIG. 1. (Color online) An example of our notation. The top panel presents the signal in the time domain, and the middle panel presents the spectrogram of the signal. The vertical solid lines present the annotated vowel's onset t_b and offset t_e. The speech signal is represented by a sequence of acoustic feature vectors as depicted in the lower panel. For the tth frame, the acoustic feature vector is denoted by $\boldsymbol{x}_t$.

are assumed to be identically and independently distributed according to the distribution ρ.

Each example in the training set is composed of a segment of the acoustic signal and a manually annotated vowel onset and offset pair. The manual annotations are not exact, and naturally depend both on human errors as well as objective difficulties in placing the vowel boundaries, e.g., between the vowel and a sonorant. Hence in this work we focus on a loss function that inherently takes into account the discrepancy in the annotations. That is,

$$\gamma(\boldsymbol{t},\hat{\boldsymbol{t}}) = [|\hat{t}_b - t_b| - \tau_b]_+ + [|\hat{t}_e - t_e| - \tau_e]_+, \tag{2}$$

where $[\pi]_+ = \max\{0,\pi\}$, and τ_b, τ_e are pre-defined parameters. The above function measures the absolute differences between the predicted and target vowel onsets and offsets. It allows a mistake of τ_b and τ_e frames at the onset and offset of the vowel, respectively, and only penalizes predictions that are greater than τ_b or τ_e frames.

Our learning model belongs to the structured prediction framework. In this framework it is assumed that the output prediction is complex and has some internal structure. In our case, the vowel onset and vowel offset times are related and dependent, e.g., the vowel has a typical duration that depends on the vowel onset and offset.

In the structured prediction model, the function f is based on a fixed mapping $\phi : \mathcal{X}^* \times \mathcal{T}^2 \rightarrow \mathbb{R}^n$ from the set of segments of acoustic signal and target onset–offset pairs to a real vector of length n; we call the elements of this mapping *feature functions* or *feature maps*. Intuitively, the feature functions represent our knowledge regarding good locations of the onset or offset of the vowel within the acoustic signal. For example, consider Fig. 1. It can be seen that the spectral change is high at the areas of t_b and t_e. Based on this, we can adopt a feature function based on the distance between the spectrum a frame before and a frame after the presumed t_b. This function is going to be high if the presumed t_b is in the vicinity of the actual target vowel onset and is going to be low at a random place in the acoustic signal.

Our prediction function is a linear decoder with a vector of parameters $\boldsymbol{w} \in \mathbb{R}^n$ that is defined as follows:

$$f_w(\bar{\boldsymbol{x}}) = \arg\max_{\hat{t}\in\mathcal{T}^2} \boldsymbol{w}^\top \phi(\bar{\boldsymbol{x}},\hat{t}). \tag{3}$$

The subscript $\boldsymbol{w}$ is added to the function f to stress that it depends on the weight vector $\boldsymbol{w}$.

Ideally, we would like our learning algorithm to find $\boldsymbol{w}$ such that the prediction minimizes the loss on unseen data. Recall, we assume there exists some unknown probability distribution ρ over pairs $(\bar{\boldsymbol{x}},\boldsymbol{t})$. We would like to set $\boldsymbol{w}$ so as to minimize the expected loss, or the *risk*, for predicting $f_w(\bar{\boldsymbol{x}})$,

$$\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} \mathbb{E}_{(\bar{\boldsymbol{x}},\boldsymbol{t})\sim\rho}[\gamma(\boldsymbol{t},f_w(\bar{\boldsymbol{x}}))]. \tag{4}$$

It is hard to directly minimize this objective function since ρ is unknown and the loss γ is often a combinatorial non-convex function (Keshet, 2014). In Sec. IV we describe the learning algorithm that aims at minimizing the risk, and then we describe the set of feature functions in Sec. V.

IV. DIRECT LOSS MINIMIZATION (DLM) ALGORITHM

Recall that our goal is to directly optimize the objective in Eq. (4). Unfortunately, if the output space is discrete we cannot use direct gradient descent since the loss $\gamma(\boldsymbol{t},f_w(\bar{x}))$ is not a differentiable function of $\boldsymbol{w}$ (Keshet, 2014). McAllester *et al.* (2010) showed that if the input space $\mathcal{X}^*$ is continuous, we can compute the gradient of the expected loss, i.e., the risk, in Eq. (4) even when the output space is discrete in terms of the feature functions. Specifically the gradient can be expressed in a closed form solution as follows:

$$\nabla_w \mathbb{E}\big[\gamma(\boldsymbol{t},f_w(\bar{x})\big] = \lim_{\epsilon\rightarrow 0} \frac{\mathbb{E}\Big[\phi\big(\bar{\boldsymbol{x}},f_w^\epsilon(\bar{\boldsymbol{x}})\big) - \phi\big(\bar{\boldsymbol{x}},f_w(\bar{\boldsymbol{x}})\big)\Big]}{\epsilon}, \tag{5}$$

where the expectation on both sides is with respect to the tuple $(\bar{\boldsymbol{x}},\boldsymbol{t})$ drawn from ρ, and f_w^ϵ is defined as follows:

$$f_w^\epsilon(\bar{\boldsymbol{x}}) = \arg\max_{\hat{t}\in\mathcal{T}^2} \boldsymbol{w}^\top \phi(\bar{\mathbf{x}},\hat{\mathbf{t}}) + \epsilon\,\gamma(\boldsymbol{t},\hat{\mathbf{t}}). \tag{6}$$

Using stochastic gradient descent we get the following update rule:

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + \frac{\eta_t}{\epsilon}\Big(\phi\big(\bar{\boldsymbol{x}},f_w(\bar{\boldsymbol{x}})\big) - \phi\big(\bar{\boldsymbol{x}},f_w^\epsilon(\bar{\boldsymbol{x}})\big)\Big), \tag{7}$$

where η_t is the learning rate. At training time, we set $\eta_t = \eta_0/\sqrt{t}$, where η_0 is a parameter and t is the iteration number, and we set ϵ as a fixed small parameter, which is selected from a held-out development set. The actual values of the parameters are detailed in Sec. VIII.

Since the gradient in Eq. (5) is not a convex function in the model parameters $\boldsymbol{w}$, the gradient descent optimization method is not guaranteed to find the optimal parameter settings; it may converge to a local minimum. We initialize the model parameters with a weight vector of parameters that was pre-trained using the structured prediction passive–aggressive (PA) algorithm (Crammer *et al.*, 2006). The vector of parameters that was obtained from the PA training is denoted by $\boldsymbol{w}_{\mathrm{PA}}$. A pseudo code of the training algorithm is given in Fig. 2.

V. FEATURES AND FEATURE FUNCTIONS

In this section we describe the acoustic features $\boldsymbol{x} \in \mathcal{X}$ and the feature functions $\phi(\bar{\boldsymbol{x}},\boldsymbol{t})$, which were designed specifically for the problem of vowel duration measurement. These features are primarily motivated by the desire to reflect the workflow of the annotators (Peterson and Lehiste, 1960). For example, in Peterson and Lehiste (1960) the beginning of the final voiced plosives was determined by comparing narrowband and broadband spectrograms in order to determine the point at which the energy in the higher harmonics is greatly diminished. In a similar fashion, we have

INPUT: training set $S = \{(\bar{x}_i, t_i)\}_{i=1}^{m}$; parameters η_0, ϵ

INITIALIZATION: $w_1 = w_{\mathrm{PA}}$

FOR $t = 1, 2, \ldots,$ T

Pick example $(\bar{x}_i, t_i)$ from S uniformly at random

Predict $\hat{t} = \arg\max_{\hat{t}} \; w_t^{\top} \phi(\bar{x}_i, \hat{t})$

Predict $\hat{t}^{\epsilon} = \arg\max_{\hat{t}} \; w_t^{\top} \phi(\bar{x}_i, \hat{t}) + \epsilon \gamma(t_i, \hat{t})$

Set $w_{t+1} = w_t + \frac{\eta_0}{\epsilon\sqrt{t}} (\phi(\bar{x}_i, \hat{t}) - \phi(\bar{x}_j, \hat{t}^{\epsilon}))$

OUTPUT: the average weight vector $w^{\star} = \frac{1}{T} \sum_t w_t$

FIG. 2. DLM training procedure.

acoustic features of narrowband and broadband energies, and a set of feature functions that allow the machine learning algorithm to compare and weight them, respectively. After an empirical evaluation of performance of these features, and analyzing the source of various errors, we added features which related to the predictions of the phoneme classifiers (e.g., vowel, nasal, etc.).

A. Acoustic features

The main function of the acoustic features representation is to preserve the crucial information-bearing elements of the speech signal and to suppress irrelevant details. We extracted $d = 16$ acoustic features every 5 ms, in a similar way to the feature set used in Sonderegger and Keshet (2012), but with different time spans. The first 5 features are based on the short-time Fourier transform taken with a 25 ms Hamming window. The features are short-term energy, $E_{\mathrm{short-term}}$; the log of the total spectral energy, E_{total}; the log of the energy between 20 and 300 Hz, E_{low}; the log of the energy above 3000 Hz, E_{high}; and the Wiener entropy, H_{Wiener}, a measure of spectral flatness (Sonderegger and Keshet, 2012). The sixth feature, S_{max}, is the maximum of the power spectrum calculated in a region from 6 ms before to 18 ms after the frame center. The seventh feature, $\hat{F}_0$, is the normalized fundamental frequency estimator, extracted using the algorithm of Sha *et al.* (2004) every 5 ms, and smoothed with a Hamming window. The eighth feature is the binary output of a voicing detector based on the RAPT pitch tracker (Talkin, 1995), smoothed with a Hamming window—denoted as V_{RAPT}. The ninth feature is the number of zero crossings in a 5 ms window—denoted as N_{ZC}.

The next set of acoustic features are based on a pretrained phoneme classifier's predictions and scores as in Dekel *et al.* (2004). This phoneme classifier was trained on the TIMIT dataset (Garofolo *et al.*, 1993), using the standard Mel-frequency cepstral coefficient (MFCC) features with a passive aggressive classifier (Crammer *et al.*, 2006). The tenth acoustic feature is an estimated probability of whether a vowel is uttered at the input frame. The feature is a smoothed version of an indicator function that states if the phoneme predicted by the classifier at the current frame is a vowel: $\mathbf{1}[\hat{y}_t \in \mathbf{vowels}]$, where $\hat{y}_t$ is the phoneme predicted by the phoneme classifier at time t and **vowels** are the set of all vowels. The 11th feature is defined to be the same as the 10th feature, but is for nasal phonemes. These features are denoted as G_{vowel} and G_{nasal}, respectively. The 12th feature is the likelihood of a vowel at the current time frame, L_{vowel}. The likelihood is computed as the Gibbs measure of the phoneme classifier's scores for all the vowel phonemes normalized by the total score for all phonemes.

The last four features are based on the spectral changes between adjacent frames, using MFCCs to represent the spectral properties of the frames. Define by $D_j = d(\boldsymbol{a}_{t-j}, \boldsymbol{a}_{t+j})$ the Euclidean distance between the MFCC feature vectors $\boldsymbol{a}_{t-j}$ and $\boldsymbol{a}_{t+j}$, where $\boldsymbol{a}_t \in \mathbb{R}^{39}$ for $1 \le t \le T$. The features are denoted by D_j, for $j \in \{1, 2, 3, 4\}$.

Figure 3 shows the trajectories of the features for a typical vowel in a CVC context (the word "got").

B. Feature functions

We turn now to describe the feature functions. Recall that the feature functions are designed to be correlated with a good positioning of the onset–offset pair, $\boldsymbol{t}$, in the acoustic signal, $\bar{x}$. While generally each feature function $\phi_i(\bar{x}, \boldsymbol{t})$, for $1 \le i \le n$, gets as input a sequence of acoustic features, $\bar{x}$, and a presumed onset–offset pair $\boldsymbol{t}$, they can practically use only a subset of the acoustic features (e.g., only a sequence over the sixth feature, as opposed to a sequence over any features). Some of the feature functions are based on the average of an acoustic feature x from frame t_1 to frame t_2 defined as

$$\mu(\bar{x}, t_1, t_2) = \frac{1}{t_2 - t_1 + 1} \sum_{t=t_1}^{t_2} x_t. \tag{8}$$

The functions that we identified as being correlated with a good positioning of the onset–offset pair can be divided into four groups based on the structure of the functions; each group was implemented over a set of the above acoustic features (n.b.: a given acoustic feature might be associated with multiple feature functions):

Type 1: This type of feature function gets as input a sequence of one of the features, $\bar{x}$, and a time t. The time t can represent the onset or offset of the vowel. Formally, the features of this type are of the form

$$\phi(\bar{x}, t) = x_t. \tag{9}$$

Feature functions from this type are computed for the acoustic features E_{total}, E_{low}, E_{high}, and S_{max} at the presumed vowel onset time t_b. It is also computed for the acoustic feature D_j, $j = 1, 2, 3, 4$ for both vowel onset t_b and offset times t_e. It can be seen from Fig. 3 that these acoustic features have a high value exactly at t_b or t_e or both, respectively.

Type 2: The second set of feature functions is a coarse estimation of the derivative around a time frame of interest.

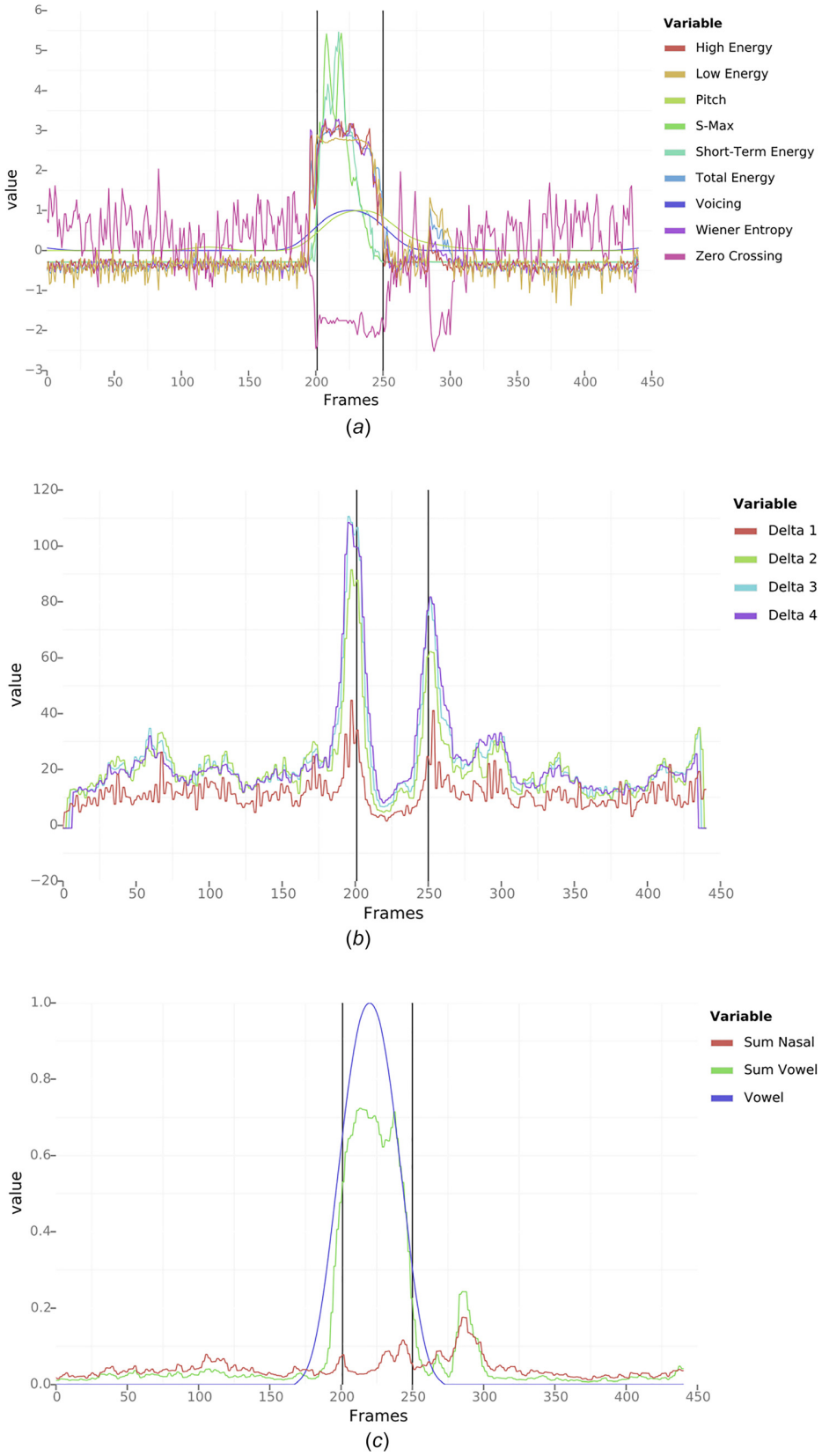


FIG. 3. (Color online) Values of acoustic features for an example acoustic sample (the word "got"). The vertical dashed lines indicate the annotated onset and offset of the vowel.

Given a sequence of acoustic feature values $\bar{x}$ and a specific time frame t, the feature functions of this type compute the difference in the mean of Δ frames before t and the mean of Δ frames after t. That is,

$$\phi(\bar{x}, t, \Delta) = \mu(\bar{x}, t - \Delta, t - 1) - \mu(\bar{x}, t, t + \Delta - 1). \quad (10)$$

Table I describes all the feature functions of this type, for the values of the number of frames processed, Δ, and the acoustic

TABLE I. The feature functions of type 2, $\phi(\bar{x}, t_1, \Delta)$. Values in the table are the time frame t_1 used for each acoustic feature (columns) and for each window length (rows). Window lengths 8 and 10 are repeated to indicate cases where the point of interest is offset (see text for details).

Δ	$E_{\text{short–term}}$	E_{low}	E_{high}	E_{total}	H_{Wiener}	$S_{\max}$	$\hat{F}_0$	V_{RAPT}	N_{ZC}	G_{vowel}	G_{nasal}	L_{vowel}	D_1	D_2	D_3	D_4
1						t_b										
2						t_b										
3	t_b, t_e					t_b							t_b, t_e	t_b, t_e	t_b, t_e	t_b, t_e
4	t_b, t_e					t_b										
5	t_b					t_b										
6		t_b								t_b, t_e		t_b, t_e				
8		t_b, t_e	t_b, t_e	t_b, t_e	t_b, t_e		t_b, t_e	t_b, t_e	t_b, t_e	t_b, t_e	t_e	t_b, t_e				
8		$t_b - 2$	$t_b - 2$	$t_b - 2$												
10		t_b, t_e	t_b, t_e	t_b, t_e	t_b, t_e		t_b, t_e	t_b, t_e	t_b, t_e	t_b, t_e	t_e	t_b, t_e	t_b, t_e	t_b, t_e	t_b, t_e	t_b, t_e
10		$t_b - 4$	$t_b - 4$	$t_b - 4$												

features. For some acoustic features we wanted to take into account the possibility that they might not be coincident with vowel boundary, but occur at an adjacent point in the acoustic signal. We did that by considering the point of interest to be an offset version of the presumed onset or offset. For example, $t_b - 2$ means the feature computes averages before and after the time frame t_b with an offset of 2 frames. It can be seen from Fig. 3 that indeed the acoustic features described in Table I have abrupt changes at the specified time frame t.

Type 3: The third type of feature function is an extension of the second type. Rather than considering the average of an acoustic feature before and after a single time frame, we refer to the average between the presumed onset and offset times. Feature functions of this type return two values: (i) the average between the presumed onset and offset times minus the average of Δ frames before t_b; and (ii) the average between the presumed onset and offset times minus the average of Δ frames after the offset time t_e. Those two values correspond to two elements in the vector

$$\phi(\bar{x}, t_b, t_e, \Delta) = \begin{bmatrix} \mu(\boldsymbol{x}, t_b, t_e) - \mu(\boldsymbol{x}, t_b - \Delta, t_b - 1) \\ \mu(\boldsymbol{x}, t_b, t_e) - \mu(\boldsymbol{x}, t_e + 1, t_e + \Delta) \end{bmatrix}. \tag{11}$$

This type of feature function is computed for the acoustic features $E_{\text{short–term}}$, E_{low}, E_{high}, E_{total}, V_{RAPT}, N_{ZC}, and L_{vowel}. Again, it can be see from Fig. 3 that these features have high (or low) mean values between the time frames of interest.

Type 4: The fourth type of feature function is oblivious to the acoustic signal and returns a probability score for the presumed vowel duration, $t_e - t_b$. We have two feature functions of this type. The first one is based on the Normal distribution with parameters $\hat{\mu}$, $\hat{\sigma}^2$ that are estimated from the training data

$$\phi(t_b, t_e) = \mathcal{N}(t_e - t_b; \hat{\mu}, \hat{\sigma}^2). \tag{12}$$

Similarly, the second feature function of this form is based on the Gamma distribution with parameters $\hat{k}$ and $\hat{\theta}$, that are estimated from the training set

$$\phi(t_b, t_e) = \Gamma(t_e - t_b; \hat{k}, \hat{\theta}). \tag{13}$$

While the Gamma distribution is an appropriate distribution to describe the vowel duration alone, we found empirically that adding the Normal distribution improves performance.

VI. DATASETS

In order to get reliable results we used three different datasets to evaluate the performance of our system. In this section we will give a short description of each dataset.

A. HG

This corpus (HG) is drawn from a study investigating how grammatical class constraints influence the activation of phonological neighbors. The influence of neighbors on phonetic processing was indexed by vowel durations. It contains segments of acoustic signal from 64 native English speakers (55 female) aged 18–34 with no history of speech or language deficits. Participants were first familiarized to a set of pictures along with their intended labels. They were then asked to name aloud the noun depicted by a picture in two contexts: when the picture was presented alone, and when the picture occurred at the end of a non-predictive sentence. Participants were instructed to produce the name as quickly and accurately as possible. Trials with errors or disfluencies were excluded, along with two items with high error rates. In addition, data from three subjects reported in the original paper were excluded from the present analysis, as they did not consent to public use of their data. The remaining 2395 recorded segments of acoustic signal contain one English CVC noun with vowels /i, ɛ, ae, ɑ, oʊ, u/.

B. CT

The second dataset (CT) contains segments of acoustic signal from 20 female native English speakers aged 18–22 with no history of speech or language deficits. The participants were evenly split between two American English dialects (Northern and Midland). As part of a larger study (Clopper *et al.*, 2002), participants read aloud a list of 991 CVC words. This study focused on 39 target words (777 tokens) which did or did not have a lexical contrast between either /ɛ/ vs /ae/ (e.g., dead-dad vs deaf-*daff) or /ɑ/ vs /ɔ/ (e.g., cot-caught vs dock-*dawk). Words with a lexical

contrast are referred to as *competitor* items and those without are referred to as *no competitor* items.

C. HGCW

The third dataset consists of data from a laboratory study conducted by HGCW. It contains segments of acoustic signal from 45 men, 48 women, and 46 10 - to 12-yr-old children (27 boys and 19 girls). The participants (87%) were raised in Michigan, primarily in the southeastern and southwestern parts of the state. The audio recordings contain 12 different vowels (/i, ɪ, ɛ, æ, ɑ, ɔ, ʊ, u, ʌ, ɝ, e, o/) from the words: heed, hid, head, had, hod, hawed, hood, who would, hud, heard, hayed, hoed.

VII. METHODS

Code implementing this algorithm is publicly available at https://github.com/adiyoss/AutoVowelDuration. Segments of acoustic signal for the HG dataset along with algorithmic and manual annotations are available in the Online Speech/Corpora Archive and Analysis Resource (https://oscaar.ci.northwestern.edu/; dataset "Within Category Neighborhood Density"). For the HGCW dataset, segments of acoustic signal and manual annotations are available at http://homepages.wmich.edu/~hillenbr/voweldata.html.

A. DLM

The DLM algorithm described above was trained and tested on both HG and CT datasets. For the CT corpus, the training set was a set of 4049 tokens from the original corpus (Clopper *et al.*, 2002), and the test set was the same 777 tokens reported in CT. The model parameters were tuned on a dedicated development set (∼10% of the training set). The parameters that yielded the lowest error rate were $\eta = 0.1$ and $\epsilon = -1.36$. We used $\tau_b = 1$ frame and $\tau_e = 2$ frames for the loss during training. The initial weight vector was set to be the averaged weight vector from the PA algorithm (Crammer *et al.*, 2006) with $C = 0.5$ and 100 epochs. When trained and tested on the HG corpus, we used tenfold cross validation (the reported results are the average error over all tenfolds) with the same settings and parameters as described above.

In order to better comprehend the influence of the phoneme classifier, the only language dependent feature in our system, we also trained and tested the direct loss algorithm without the acoustic features related to the phoneme classifier G_{vowel}, G_{nasal}, or L_{vowel}, and their corresponding feature functions. In the case when the phoneme classifier was used, we trained multiclass PA as described in Dekel *et al.* (2004) on the TIMIT corpus of read speech.

B. HMM

To validate the effectiveness of the proposed approach, we compared it to the most common approach currently used in automatic phonetic measurement of speech: forced alignment. This is an algorithm which, given a speech utterance and its phonetic content, finds the start time of each phoneme in the speech utterance. Often the orthographic content of the speech utterance is given, and it is converted to its phonetic content using a lexicon. This procedure may not be accurate as often the surface pronunciation uttered in spontaneous speech is not the same as the canonical pronunciation that appears in the lexicon.

Conventionally, the forced alignment is implemented by forcing the decoder of an HMM phoneme recognizer to pass through the states corresponding to the given phoneme sequence. So far, automatic vowel extraction has been done using such forced alignment procedures (Reddy and Stanford, 2015). While there are several open source implementations of HMMs, all the publicly available forced aligner packages (Goldman, 2011; Gorman *et al.*, 2011; Rosenfelder *et al.*, 2014; Yuan and Liberman, 2008) are based on the HTK toolkit (Young and Young, 1994). Since this is the case, we chose to compare our results with the most recent one, namely, the *FAVE aligner* (Rosenfelder *et al.*, 2014), based on the *Penn Phonetics Lab Forced Aligner* (Yuan and Liberman, 2008). In our analyses below, we refer to this system as the HMM aligner. Note that in contrast to the DLM the forced aligner requires a phonetic or orthographic transcription as input.

Since neither CT nor HG has enough data to train an HMM forced alignment system, in this work we use a pretrained model which was trained on different corpus [see details in Rosenfelder *et al.* (2014)]. In order to make a fair comparison between the proposed model and the HMM system, we evaluate the models on two different settings: (i) a dataset which was not used during training any of the models (HGCW), and (ii) on mismatched training and test datasets.

VIII. RESULTS: MEASUREMENT DEVIATION

We first examine results on the HG and CT datasets when the DLM algorithm is trained and tested on the same corpus. The difference between the automatic and manual measurements of vowel duration are given in Table II, where the evaluation metric is the loss in Eq. (2) with both τ_b and τ_e equal to 0, and in Table III, where the error is given in terms of the percentage of predictions that do not fall within the boundaries of 20 ms from the manual onset and 50 ms from the manual offset. (We allow for greater deviations in vowel offset, where manual annotators often show a greater disagreement.) These reveal that when trained and tested on the same corpus, the DLM generally outperforms the standard HMM approach, particularly when the phoneme classifier is incorporated into the algorithm. An exception is seen

TABLE II. Results of DLM (with and without phoneme classifier, trained and tested on the same corpus) and HMM relative to manual annotation. Average deviation of onset and offset (in msec). Bold indicates minimum deviation for each dataset within each context.

	DLM		DLM (no classifier)		HMM	
	onset	offset	onset	offset	onset	offset
HG	**5.21**	**22.80**	5.84	28.98	16.67	24.72
CT	9.42	**16.76**	**9.24**	23.18	35.90	30.61

TABLE III. The percentage of predictions that do not fall within the boundaries of 20 ms at the onset and 50 ms at the offset from the manual annotation when the DLM is trained and tested on the same corpus. Bold indicates minimum deviation for each dataset within each context.

	DLM		DLM (no classifier)		HMM	
	onset	offset	onset	offset	onset	offset
HG	**6.15%**	13.15%	8.05%	18.44%	31.90%	**10.94%**
CT	**9.46%**	**8.31%**	9.59%	9.34%	41.50%	13.14%

in the HG offset data, where the percentage of predictions outside of 50 ms is smallest for the HMM.

It is possible that the accuracy of the algorithms may differ by consonantal context. To assess this, we classified the consonant preceding or following the vowel of each word as belonging to one of four categories: voiceless stops, voiced stops, fricatives/affricates, and sonorants. We then calculated the overall measurement error for each algorithm within each category, as reported in Table IV.

The context-specific HG data shows that the general advantage for the DLM with the phoneme classifier holds, but there are some exceptions. The performance of the HMM is comparable to the DLM with classifier when a vowel is followed by a voiceless stop, and appears to outperform the DLM when voiceless stops or fricatives precede the vowel. In the CT data, there is a clear DLM advantage across all contexts.

A. Mismatched training and test datasets

The comparison above may be biased in favor of the DLM, as the algorithm's parameters may be tuned to specific features of the annotators that worked with the HG and CT corpora. To provide a more even comparison, the DLM was trained and tested on different corpora—training on HG and testing on CT and vice versa. We also present the result of our algorithm trained on CT (the manual dataset with the highest manual inter-annotator reliability; see below) and tested on the third dataset, HGCW.

As before, the difference between the automatic and manual measurements of vowel duration are given in Table V, and the error in terms of the percentage of predictions that do not fall within the boundaries of 20 ms from the manual onset and 50 ms from the manual offset is given in Table VI. These results again show that for the onset data, the DLM outperforms the HMM, even when the training and testing sets for the algorithm are mismatched. In vowel offsets, the HMM meets or outperforms the DLM for the HG and CT data, but the DLM advantage is maintained for the HGCW offset data.

TABLE IV. Results of DLM (with and without phoneme classifier, trained and tested on the same corpus) and HMM relative to manual annotation, broken down by consonant context. Onset consonant context is shown in the first, followed by coda consonant context. Average deviation of full vowel duration [in msec]. Fricatives are all voiceless, with the exception of [dʒ] in CT coda consonants. There were no sonorants in CT codas. Bold indicates minimum deviation within each context for each dataset.

	DLM		DLM (no classifier)		HMM	
Onset	HG	CT	HG	CT	HG	CT
Voiceless stops	25.67	17.07	30.65	**16.96**	**21.95**	97.45
Voiced stops	**27.81**	**18.62**	31.23	20.27	31.66	53.90
Fricatives/affricates	25.03	16.74	32.40	**16.28**	**18.01**	32.86
Sonorants	**24.12**	**26.97**	36.32	27.95	42.35	40.52
	DLM		DLM (no classifier)		HMM	
Coda	HG	CT	HG	CT	HG	CT
Voiceless stops	**25.72**	18.06	28.76	**17.78**	25.91	43.22
Voiced stops	**16.16**	**23.08**	20.10	27.22	19.04	45.19
Fricatives/affricates	**15.00**	20.42	27.25	**20.41**	27.92	72.82
Sonorants	**32.51**	—	39.04	—	35.36	—

B. Measurement correlation

Above, we examined the degree to which each algorithm agreed with manual annotators based on the amount of deviation between measures. Another measure of agreement conventionally reported in phonetic studies is the correlation between measurements of vowel durations. This is typically done by assigning a random subset of the data to a second annotator. For the HG dataset, the second annotator's correlation with the original annotator was $r(627) = 0.84$. We performed a similar analysis, treating the algorithm as a second annotator relative to the manual labels. For the same subset of the data (with the DLM trained and tested on the same corpus), the correlations between each algorithm and the original manual annotator are DLM = 0.79, DLM (no classifier) = 0.64, and HMM = 0.73. While using the models that were trained on mismatched data the correlations are DLM = 0.67, DLM (no classifier) = 0.55. Similarly, for the CT dataset, the second annotator's correlation with the original annotator was $r(398) = 0.95$. For the same subset of the data, each algorithm's correlations are: DLM = 0.93, DLM (no classifier) = 0.93, and HMM = 0.57. However, when using the models that were trained on mismatched data the correlations are DLM = 0.54, DLM (no classifier) = 0.53. Thus, when training the models on mismatched datasets, correlations of the HMM model and DLM model with human annotators are equivalent; the DLM only outperforms HMM when using the same training and testing set.

C. Discussion

Analysis of measurement deviation suggests our model generally outperforms the HMM-forced alignment algorithm, even when training and testing sets are mismatched.

TABLE V. Results of DLM (with and without phoneme classifier) and HMM under mismatched training and test datasets. Average deviation of onset and offset [in msec]. Bold indicates minimum deviation for each dataset within each context.

	DLM		DLM (no classifier)		HMM	
	onset	offset	onset	offset	onset	offset
HG (*CT* model)	**14.44**	38.66	14.92	43.95	16.67	**24.72**
CT (*HG* model)	**9.84**	30.85	10.17	**29.94**	35.90	30.61
HGCW (*CT* model)	**12.91**	**9.28**	15.86	23.89	19.95	27.30

TABLE VI. The percentage of predictions that do not fall within the boundaries of 20 ms at the onset and 50 ms at the offset from the manual annotation (with mismatched training and test datasets for the DLM). Bold indicates minimum deviation for each dataset within each context.

	DLM		DLM (no classifier)		HMM	
	onset	offset	onset	offset	onset	offset
HG (*CT* model)	**11.55%**	23.99%	12.25%	28.80%	31.90%	**10.94%**
CT (*HG* model)	**8.36%**	14.92%	**8.36%**	**10.94%**	41.50%	13.14%
HGCW (*CT* model)	**19.18%**	**2.16%**	27.58%	6.54%	28.54%	11.69%

Incorporating the phoneme classifier improved performance, but even without the classifier the DLM typically outperforms the HMM. However, with respect to measurement correlation, there was no clear advantage; DLM and HMM exhibited equivalent performance when trained on mismatched training and test sets.

IX. RESULTS: REPRODUCING REGRESSION MODEL FINDINGS

Empirical studies of speech and language processing use acoustic properties such as vowel duration as behavioral measures of the effects of various types of variables that influence speech and language. Canonical examples of variables of study include the phonetic context of vowels (e.g., preceding voiced vs voiceless stops), properties of speakers who produce those vowels (e.g., native vs non-native speakers), and the linguistic (lexical, syntactic, etc.) context in which the vowels appear (e.g., predictable vs unpredictable). The effects of these variables on acoustic properties are typically examined using inferential statistical models.

The second evaluation metric of our algorithms therefore examined the similarity of inferential statistical model fits based on measurements generated by algorithms vs manual measurements. Out of the three datasets, only HG constructed models based on duration data; our analysis therefore focused on this dataset.[1]

HG examined whether processes involved in sentence planning influence the processing of sound. Speakers named pictures in a context that strongly emphasized sentence planning (following a sentence fragment) as well as a context that did not require substantial planning (producing the picture name in isolation). The order of these contexts was counterbalanced across speakers. To index effects of sound structure processing, this study also manipulated the number of words phonologically similar to the target that share its grammatical category (category-specific lexical density).

To analyze the effect of this variable, HG used linear mixed effects regression models (Baayen *et al.*, 2008), an approach that has become dominant in the analysis of speech and psycholinguistic data. These regression models predict dependent measures based on a linear combination of predictors, including both fixed and randomly distributed predictors capturing variation in effects by both participants and items. This allows researchers to examine the effects of interest while controlling for other properties of the words and speakers.

Analysis of the full data set showed that lexical density and vowel category had no effect on vowel durations (Heller and Goldrick, 2014, 2015); however, these effects were not stable when the subset of data used here was examined. Therefore, the models used here were simplified from those reported in the original paper, including two contrast-coded fixed effect factors: production context (isolation vs sentence) and block (first vs second). To control for contributions from the random sample of participants and items used in this experiment (compared to all English speakers and all words of English), we included two sets of random effects. Random intercepts for both speakers and words were included, along with uncorrelated slopes for context by both speaker and word.

A. Significance of fixed effects in models of the HG dataset

We first examine the primary interest of many phonetic studies—the binary distinction between significant vs insignificant effects of fixed-effect predictors (e.g., the effect of experimental condition). To control for skew, vowel durations were log-transformed prior to analysis. Given that outliers can have an outsized influence on parameter estimates, all regressions models were refit after excluding observations with standardized residuals exceeding 2.5 (Baayen, 2008). The significance of fixed-effects predictors was assessed by using the likelihood ratio test to compare models with and without the predictor of interest (Barr *et al.*, 2013). Table VII compares the estimates for the two fixed effects parameters for the manual model and each algorithmic model. Although there was some variation in the model estimates of the fixed effects for each algorithm compared to the manual annotations, each algorithm recovered the overall pattern of a significant effect of context but not block.

1. Comparison of predictions of models of the HG dataset

An alternative, more global, assessment of model similarity is to compare model predictions. This was assessed by leave-one-out validation. For each observation, we excluded it from the dataset and re-fit the regression to the remaining observations. After residual-based outlier trimming (and re-fitting), we examined the predictions of this re-fitted model for the excluded observation. Figure 4 shows distributions of the deviation of each algorithm's predicted fit to the predicted model fit to the manual data.

The DLM algorithm outperformed the other algorithms. The mean squared error relative to the predictions of the

TABLE VII. Estimates and *t*-values for fixed effects in regression model, HG dataset (standard error of estimate in parentheses). Significant effects ($p <= 0.05$, as assessed by likelihood ratio tests) are bolded.

	Context	*t*	Block	*t*
Manual	**−0.057 (0.017)**	**−3.29**	0.012 (0.016)	0.72
DLM	**−0.072 (0.019)**	**−3.82**	0.016 (0.018)	0.91
DLM (no classifier)	**−0.059 (0.018)**	**−3.34**	0.015 (0.017)	0.90
HMM	**−0.058 (0.018)**	**−3.13**	0.008 (0.014)	0.55

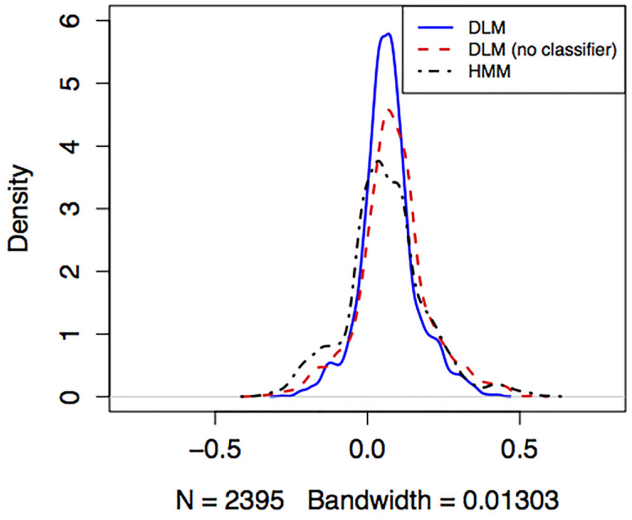


FIG. 4. (Color online) Comparisons of leave-one-out model predictions of vowel durations for the HG dataset. Deviance of each algorithmic method as compared to the manual predictions (manual-algorithmic) are plotted as individual lines.

model fit to the manually annotated data was lowest for the DLM algorithm (0.479 msec^2), higher for HMM (0.835 msec^2), and larger still for DLM (no classifier) (0.935 msec^2). Bootstrap confidence intervals (CIs) of the differences across algorithms showed that DLM outperformed the other two algorithms (95% CI of differences from HMM: [0.00039, 0.00043] msec^2; DLM (no classifier): [0.00037, 0.00055] msec^2). The HMM algorithm did not significantly differ from DLM (no classifier) (95% CI [-0.000200, 0.000002] msec^2).

Parallel to the analysis of measurement deviation, these analyses suggest our model outperforms the HMM-forced aligner. Incorporating the phoneme classifier improved performance, but even without the classifier the performance of the DLM matches that of the HMM.

X. GENERAL DISCUSSION

We presented an algorithm for automatically estimating the durations of vowels based on the structured prediction framework, relying on a set of acoustic features and feature functions finely tuned to reflect the properties of vowel acoustics relevant to vowel segmentation. The DLM algorithm, when including a phoneme classifier, clearly succeeds at matching manual measurements of vowel duration. With respect to both measurement deviation and reproduction of regression model results, the DLM algorithm outperforms or matches a commonly-used forced alignment system while not requiring a phonetic transcription. This approach can allow laboratory experiments to address much larger samples of data in a way that is replicable and reliable.

Having achieved some success with monosyllabic, laboratory stimuli, future development of this algorithm should extend this approach to more naturalistic production. In current work, we are extending this approach to vowel durations in multisyllabic words. We believe that moving outside the laboratory will be facilitated by the structure of our approach; in contrast with existing systems, our algorithm has the additional benefit of not requiring a transcript of the desired speech prior to analysis. Extending the capability of the system to analyze untranscribed speech will support the analysis of more naturalistic, connected speech, including speech styles or dialects that may be difficult to robustly sample in a laboratory context (Labov, 1972; Rischel, 1992).

With respect to the algorithms utilized in future work, we aim to better determine the acoustic features that allow the most precise estimation of vowel duration. While our current work relies on features determined by expert annotators (Peterson and Lehiste, 1960), in ongoing work we aim to have the algorithm discover these (as well as perhaps novel features not used by experts). We are currently pursuing this by applying new advances in deep learning to automatic estimation of vowel duration. In preliminary work, we used this approach to train a binary classifier to detect for each frame whether it contains a vowel or not (Adi *et al.*, 2015). However, since this approach does not take into account the relationships between adjacent frames, it requires an additional algorithm to yield durations. In future work, we plan to focus on combining new advances in sequence deep learning (Elman, 1990; Graves *et al.*, 2013; Graves and Jaitly, 2014) with our current structured prediction scheme.

ACKNOWLEDGMENTS

Research supported by NIH Grant No. 1R21HD077140 and NSF Grant No. BCS1056409.

[1]See supplementary material at http://dx.doi.org/10.1121/1.4972527 for regression analysis of the CT dataset, where the vowel duration measurements of our algorithm as well as the HMM are provided as inputs to a formant estimator.

Adi, Y., Keshet, J., and Goldrick, M. (**2015**). "Vowel duration measurement using deep neural networks," in *Proceedings of the IEEE International Workshop on Machine Learning for Signal Processing*, pp. 1–6.

Baayen, R. H. (**2008**). *Analyzing Linguistic Data: A Practical Introduction to Statistics using R* (Cambridge University Press, Cambridge).

Baayen, R. H., Davidson, D. J., and Bates, D. M. (**2008**). "Mixed-effects modeling with crossed random effects for subjects and items," J. Memory Lang. **59**(4), 390–412.

Barr, D. J., Levy, R., Scheepers, C., and Tily, H. J. (**2013**). "Random effects structure for confirmatory hypothesis testing: Keep it maximal," J. Memory Lang. **68**(3), 255–278.

Clopper, C. G., Carter, A. K., Dillon, C. M., Hernandez, L. R., Pisoni, D. B., Clarke, C. M., Harnsberger, J. D., and Herman, R. (**2002**). "The Indiana speech project: An overview of the development of a multi-talker multi-dialect speech corpus," Indiana University Speech Research Laboratory Research on Speech Perception Progress Report, Vol. 25, pp. 367–380.

Clopper, C. G., and Tamati, T. N. (**2014**). "Effects of local lexical competition and regional dialect on vowel production," J. Acoust. Soc. Am. **136**(1), 1–4.

Crammer, K., Dekel, O., Keshet, J., Shalev-Shwartz, S., and Singer, Y. (**2006**). "Online passive aggressive algorithms," J. Machine Learn. Res. **7**, 551–585.

Dekel, O., Keshet, J., and Singer, Y. (**2004**). "An online algorithm for hierarchical phoneme classification," in *Workshop on Multimodal Interaction and Related Machine Learning Algorithms; Lecture Notes in Computer Science*, Vol. 3361/2005 (Springer-Verlag, Berlin), pp. 146–159.

Elman, J. L. (**1990**). "Finding structure in time," Cognitive Sci. **14**(2), 179–211.

Evanini, K. (**2009**). "The permeability of dialect boundaries: A case study of the region surrounding Erie," Ph.D. thesis, University of Pennsylvania.
Garofolo, J. S., Lamel, L. F., Fisher, W. M., Fiscus, J. G., Pallett, D. S., Dahlgren, N. L., and Zue, V. (**1993**). "TIMIT acoustic-phonetic continuous speech corpus," in *Linguistic Data Consortium*, Philadelphia, PA, 33 pp.
Goldman, J.-P. (**2011**). "Easyalign: An automatic phonetic alignment tool under Praat," in *Proceedings of Interspeech*.
Gorman, K., Howell, J., and Wagner, M. (**2011**). "Prosodylab-aligner: A tool for forced alignment of laboratory speech," Canadian Acoust. **39**(3), 192–193.
Graves, A., and Jaitly, N. (**2014**). "Towards end-to-end speech recognition with recurrent neural networks," in *Proceedings of the 31st International Conference on Machine Learning* (*ICML-14*), pp. 1764–1772.
Graves, A., Mohamed, A.-R., and Hinton, G. (**2013**). "Speech recognition with deep recurrent neural networks," in *Proceedings of 2013 IEEE International Conference on Acoustics, Speech and Signal Processing* (*ICASSP*), pp. 6645–6649.
Heller, J. R., and Goldrick, M. (**2014**). "Grammatical constraints on phonological encoding in speech production," Psychonomic Bull. Rev. **21**(6), 1576–1582.
Heller, J. R., and Goldrick, M. (**2015**). "Erratum to: 'Grammatical constraints on phonological encoding in speech production,' " Psychonomic Bull. Rev. **22**(5), 1475.
Hillenbrand, J., Getty, L. A., Clark, M. J., and Wheeler, K. (**1995**). "Acoustic characteristics of American English vowels," J. Acoust. Soc. Am. **97**(5), 3099–3111.
Keshet, J. (**2014**). "Optimizing the measure of performance in structured prediction," in *Advanced Structured Prediction*, edited by S. Nowozin, P. V. Gehler, J. Jancsary, and C. H. Lampert (MIT Press, Cambridge, MA).
Keshet, J., Shalev-Shwartz, S., Singer, Y., and Chazan, D. (**2007**). "A large margin algorithm for speech-to-phoneme and music-to-score alignment," IEEE Trans. Audio, Speech Lang. Process **15**, 2373–2382.
Labov, W. (**1972**). *Sociolinguistic Patterns* (University of Pennsylvania Press, Philadelphia, PA).
McAllester, D., Hazan, T., and Keshet, J. (**2010**). "Direct loss minimization for structured prediction," in *Proceedings of the 23rd International Conference on Neural Information Processing Systems*, Vancouver, British Columbia, Canada (December 6–9, 2010).
Peterson, G. E., and Lehiste, I. (**1960**). "Duration of syllable nuclei in English," J. Acoust. Soc. Am. **32**(6), 693–703.
Reddy, S., and Stanford, J. N. (**2015**). "Toward completely automated vowel extraction: Introducing DARLA," Linguistics Vanguard **1**, 15–28.
Rischel, J. (**1992**). "Formal linguistics and real speech," Speech Commun. **11**(4), 379–392.
Rosenfelder, I., Fruehwald, J., Evanini, K., Seyfarth, S., Gorman, K., Prichard, H., and Yuan, J. (**2014**). "Fave (forced alignment and vowel extraction)," Program suite v1.2.2 10.5281/zenodo. 22281.
Sha, F., Burgoyne, J. A., and Saul, L. K. (**2004**). "Multiband statistical learning for f0 estimation in speech," in *Proceeding of IEEE International Conference on Acoustics, Speech, and Signal Processing* (*ICASSP*), Vol. 5, pp. 661–664.
Sonderegger, M., and Keshet, J. (**2012**). "Automatic discriminative measurement of voice onset time," J. Acoust. Soc. Am. **132**(6), 3965–3979.
Talkin, D. (**1995**). "A robust algorithm for pitch tracking," in *Speech Coding and Synthesis*, edited by W. B. Kleijn and K. K. Paliwel (Elsevier, New York), pp. 495–518.
Young, S., and Young, S. (**1994**). "The HTK hidden Markov model toolkit: Design and philosophy," Entropic Cambridge Res. Lab., Ltd. **2**, 2–44.
Yuan, J., and Liberman, M. (**2008**). "Speaker identification on the SCOTUS corpus," J. Acoust. Soc. Am. **123**(5), 3878.
Yuan, J., Ryant, N., Liberman, M., Stolcke, A., Mitra, V., and Wang, W. (**2013**). "Automatic phonetic segmentation using boundary models," in *Proceedings of Interspeech*, pp. 2306–2310.

Sequence Segmentation using Joint RNN and Structured Prediction Models

SEQUENCE SEGMENTATION USING JOINT RNN AND STRUCTURED PREDICTION MODELS

Yossi Adi[1]*, Joseph Keshet*[1]*, Emily Cibelli*[2]*, Matthew Goldrick*[2]

[1]Department of Computer Science, Bar-Ilan University, Ramat-Gan, Israel
[2]Department of Linguistics, Northwestern University, Evanston, IL, USA

ABSTRACT

We describe and analyze a simple and effective algorithm for sequence segmentation applied to speech processing tasks. We propose a neural architecture that is composed of two modules trained jointly: a recurrent neural network (RNN) module and a structured prediction model. The RNN outputs are considered as feature functions to the structured model. The overall model is trained with a structured loss function which can be designed to the given segmentation task. We demonstrate the effectiveness of our method by applying it to two simple tasks commonly used in phonetic studies: word segmentation and voice onset time segmentation. Results suggest the proposed model is superior to previous methods, obtaining state-of-the-art results on the tested datasets.

Index Terms— Sequence segmentation, recurrent neural networks (RNNs), structured prediction, word segmentation, voice onset time

1. INTRODUCTION

Sequence segmentation is an important task for many speech and audio applications such as speaker diarization, laboratory phonology research, speech synthesis, and automatic speech recognition (ASR). Segmentation models can be used as a pre-process step to clean the data (e.g., removing non-speech regions such as music or noise to reduce ASR error [1, 2]). They can also be used as tools in clinically- or theoretically-focused phonetic studies that utilize acoustic properties as a dependent measure. For example, voice onset time, a key feature distinguishing voiced and voiceless consonants across languages [3], is important both in ASR [4], clinical [5], and theoretical studies [6].

Previous work on speech sequence segmentation focuses on generative models such as hidden Markov models (see for example [7] and the references therein); on discriminative methods [2, 8, 9]; or on deep learning [10, 11].

Inspired by the recent work on combined deep network and structured prediction models [12, 13, 14, 15, 16], we would like to further improve performance on speech sequence segmentation and propose a new efficient joint deep network and structure prediction model. Specifically, we jointly optimize RNN and structured loss parameters by using RNN outputs as feature functions for a structured prediction model. First, an RNN encodes the entire speech utterance and outputs new representation for each of the frames. Then, an efficient search is applied over all possible segments so that the most probable one can be selected. We evaluate this approach using two tasks: word segmentation and voice onset time segmentation. In both tasks the input is a speech segment and the goal is to determine the boundaries of the defined event. We show that the proposed approach outperforms previous methods on these two segmentation tasks.

Supported in part by NIH grant 1R21HD077140.

2. PROBLEM SETTING

In the problem of speech segmentation we are provided with a speech utterance, denoted as $\bar{\mathbf{x}} = (\mathbf{x}_1, \ldots, \mathbf{x}_T)$, represented as a sequence of acoustic feature vectors, where each $\mathbf{x}_t \in \mathbb{R}^D$ $(1 \leq t \leq T)$ is a D-dimensional vector. The length of the speech utterance, T, is not a fixed value, since the input utterances can have different durations.

Each input utterance is associated with a timing sequence, denoted by $\bar{\mathbf{y}} = (y_1, \ldots, y_p)$, where p can vary across different inputs. Each element $y_i \in \mathcal{Y}$, where $\mathcal{Y} = \{1, \ldots, T\}$ indicates the start time of a new event in the speech signal. We annotate all the possible timing sequence of size p by $\mathcal{Y}^p$

For example, in *word segmentation* the goal is to segment a word from silence and noise in the signal. In this case the size of $\bar{\mathbf{y}}$ is 2, namely word onset and offset. However, in phoneme segmentation the goal is to segment every phoneme in a spoken word. In this case the size of $\bar{\mathbf{y}}$ is different for each input sequence.

Generally, our method is suitable for different sequence size $|\bar{\mathbf{y}}|$. In this paper we focused on $|\bar{\mathbf{y}}| = 2$, and leave the problem of $|\bar{\mathbf{y}}| > 2$ to future work.

3. MODEL DESCRIPTION

We now describe our model in greater detail. First, we present the structured prediction framework and then discuss how it is combined with an RNN.

3.1. Structured Prediction

We consider the following prediction rule with $\boldsymbol{w} \in \mathbb{R}^d$, such that $\bar{\mathbf{y}}'_{\boldsymbol{w}}$ is a good approximation to the true label of $\bar{\mathbf{x}}$, as follows:

$$\bar{\mathbf{y}}'_{\boldsymbol{w}}(\bar{\mathbf{x}}) = \underset{\bar{\mathbf{y}} \in \mathcal{Y}}{\operatorname{argmax}} \ \boldsymbol{w}^\top \boldsymbol{\phi}(\bar{\mathbf{x}}, \bar{\mathbf{y}}) \tag{1}$$

Following the structured prediction framework, we assume there exists some unknown probability distribution ρ over pairs $(\bar{\mathbf{x}}, \bar{\mathbf{y}})$ where $\bar{\mathbf{y}}$ is the desired output (or reference output) for input $\bar{\mathbf{x}}$. Both $\bar{\mathbf{x}}$ and $\bar{\mathbf{y}}$ are usually structured objects such as sequences, trees, etc. Our goal is to set $\boldsymbol{w}$ so as to minimize the expected cost, or the *risk*,

$$\boldsymbol{w}^* = \underset{\boldsymbol{w}}{\operatorname{argmin}} \ \mathbb{E}_{(\bar{\mathbf{x}}, \bar{\mathbf{y}}) \sim \rho}[\ell(\bar{\mathbf{y}}, \bar{\mathbf{y}}'_{\boldsymbol{w}}(\bar{\mathbf{x}}))]. \tag{2}$$

This objective function is hard to minimize directly since the distribution ρ is unknown. We use a training set $\mathcal{S} = \{(\bar{\mathbf{x}}_1, \bar{\mathbf{y}}_1), \ldots, (\bar{\mathbf{x}}_m, \bar{\mathbf{y}}_m)\}$ of m examples that are drawn i.i.d. from ρ, and replace the expectation in (2) with a mean over the training set.

The cost is often a combinatorial non-convex quantity, which is hard to minimize. Hence, instead of minimizing the cost directly, we minimize a slightly different function called a *surrogate loss*, denoted $\bar{\ell}(\boldsymbol{w}, \bar{\mathbf{x}}, \bar{\mathbf{y}})$, and closely related to the cost. Overall, the objective function in (2) transforms into the following objective function, denoted as F:

$$F(\boldsymbol{w}, \bar{\mathbf{x}}, \bar{\mathbf{y}}) = \frac{1}{m} \sum_{i=1}^{m} \bar{\ell}(\boldsymbol{w}, \bar{\mathbf{x}}, \bar{\mathbf{y}}) \tag{3}$$

In this work the surrogate loss function is the structural hinge loss [17] defined as

$$\bar{\ell}(\boldsymbol{w}, \bar{\mathbf{x}}, \bar{\mathbf{y}}) = \max_{\bar{\mathbf{y}}' \in \mathcal{Y}} \ \left[\ell(\bar{\mathbf{y}}, \bar{\mathbf{y}}') - \boldsymbol{w}^\top \boldsymbol{\phi}(\bar{\mathbf{x}}, \bar{\mathbf{y}}) + \boldsymbol{w}^\top \boldsymbol{\phi}(\bar{\mathbf{x}}, \bar{\mathbf{y}}')\right]$$

Usually, $\boldsymbol{\phi}(\bar{\mathbf{x}}, \bar{\mathbf{y}})$ is manually chosen using data analysis techniques and involves manipulation on local and global features. In the next subsection we describe how to use an RNN as feature functions.

3.2. Recurrent Neural Networks as Feature Functions

RNN is a deep network architecture that can model the behavior of dynamic temporal sequences using an internal state which can be thought of as memory [18, 19]. RNN provides the ability to predict the current frame label based on the previous frames. Bidirectional RNN is a model composed of two RNNs: the first is a standard RNN while the second reads the input backwards. Such a model can predict the current frame based on both past and future frames. By using the RNN outputs we can jointly train the structured and network models.

Recall our prediction rule in Eq. (1): notice that $\boldsymbol{\phi}(\bar{\mathbf{x}}, \bar{\mathbf{y}})$ can be viewed as $\sum_{i=1}^{p} \boldsymbol{\phi}'(\bar{\mathbf{x}}, y_i)$ where each $\boldsymbol{\phi}$ can be extracted using different techniques, e.g., hand-crafted, feed-forward neural network, RNNs, etc. We can formulate the prediction rule as follows:

$$\begin{aligned} \bar{\mathbf{y}}'_{\boldsymbol{w}}(\bar{\mathbf{x}}) &= \underset{\bar{\mathbf{y}} \in \mathcal{Y}^p}{\operatorname{argmax}} \ \boldsymbol{w}^\top \boldsymbol{\phi}(\bar{\mathbf{x}}, \bar{\mathbf{y}}) \\ &= \underset{\bar{\mathbf{y}} \in \mathcal{Y}^p}{\operatorname{argmax}} \ \boldsymbol{w}^\top \sum_{i=1}^{p} \boldsymbol{\phi}'(\bar{\mathbf{x}}, y_i) \\ &= \underset{\bar{\mathbf{y}} \in \mathcal{Y}^p}{\operatorname{argmax}} \ \boldsymbol{w}^\top \sum_{i=1}^{p} \mathrm{RNN}(\bar{\mathbf{x}}, y_i), \end{aligned} \tag{4}$$

where the RNN can be of any type and architecture. For example, we can use bidirectional RNN and consider $\boldsymbol{\phi}$ as the concatenation of both outputs BI-RNN$_{\text{forward}} \oplus$BI-RNN$_{\text{backward}}$. This is depicted in Figure 1. We call our model *DeepSegmentor* .

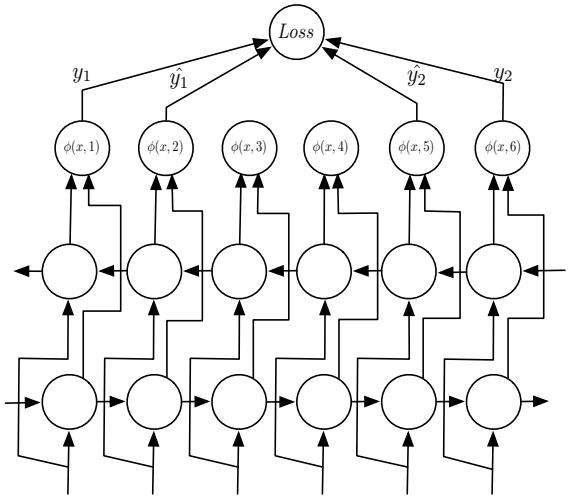


Fig. 1. An illustration for using BI-RNN as feature functions. We search through all possible locations and predict the one with the highest score. In this example the target timing sequence is $(1, 6)$ and the predicted timing sequence is $(2, 5)$.

Our goal is to find the model parameters so as to minimize the risk as in Eq. (2). Recall, we use the structural hinge loss function, and since both the loss function and the RNN are differentiable we can optimize them using gradient based methods such as stochastic gradient descent (SGD). In order to optimize the network parameters using the back-propagation algorithm [20], we must find the outer derivative of each layer with respect to the model parameters and inputs.

The derivative of the loss layer with respect to the layer parameters $\boldsymbol{w}$ for the training example $(\bar{\mathbf{x}}, \bar{\mathbf{y}})$ is

$$\frac{\partial F}{\partial \boldsymbol{w}} = \boldsymbol{\phi}(\bar{\mathbf{x}}, \bar{\mathbf{y}}^\ell_{\boldsymbol{w}}) - \boldsymbol{\phi}(\bar{\mathbf{x}}, \bar{\mathbf{y}}),$$

where

$$\bar{\mathbf{y}}^\ell_{\boldsymbol{w}} = \underset{\bar{\mathbf{y}}' \in \mathcal{Y}^p}{\operatorname{argmax}} \ \boldsymbol{w}^\top \boldsymbol{\phi}(\bar{\mathbf{x}}, \bar{\mathbf{y}}') + \ell(\bar{\mathbf{y}}, \bar{\mathbf{y}}'). \tag{5}$$

Similarly, the derivatives with respect to the layer's inputs are

$$\frac{\partial F}{\partial \boldsymbol{\phi}(\bar{\mathbf{x}}, \bar{\mathbf{y}})} = -\boldsymbol{w} \qquad \frac{\partial F}{\partial \boldsymbol{\phi}(\bar{\mathbf{x}}, \bar{\mathbf{y}}')} = \boldsymbol{w}.$$

The derivatives of the rest of the layers are the same as an RNN model.

4. EXPERIMENTAL RESULTS

We investigate two segmentation problems; word segmentation and voice onset time segmentation. We describe each of them in details in the following subsections.[1]

4.1. Word Segmentation

In the problem of word segmentation we are provided with a speech utterance which contains a single word; our goal is to predict its start and end times. The ability to determine these timings is crucial to phonetic studies that measure speaker properties (e.g. response time [23]) or as a preprocessing step for other phonetic analysis tools [11, 10, 9, 8, 24].

4.1.1. Dataset

Our dataset comes from a laboratory study by Fink and Goldrick [23]. Native English speakers were shown a set of 90 pictures. Some participants produced the name of the picture (e.g., saying "cat", "chair") while others performed a semantic classification task (e.g., saying "natural", "man-made"). Productions other than the intended response or disfluencies were excluded. Recordings were randomly assigned to two transcribers who annotated the onset and offset of each word. We analyze a subset of the recordings, including data from 60 participants, evenly distributed across tasks.

4.1.2. Results

We compare our model to an RNN that was trained using the Negative-Log-Liklihood (NLL). The NLL model makes a binary decision in every frame to predict whether there is voice activity or not. Recall, our goal is to find the start and end times of the word; in this task, the RNN leaves us with a distribution over all possible onsets. To account for this, we apply a smoothing algorithm and find the most probable pair of timings.

We trained the DeepSegmentor model using the structured loss function as in (6), denoted as Combined Duration (CD) loss. The motivation for using this function is due to disparities in the manual annotations, which are common and depend both on human errors and objective difficulties in placing the boundaries. Hence we chose a loss function that takes into account the variations in the annotations.

$$\gamma(\bar{\mathbf{y}}, \bar{\mathbf{y}}') = [|y_1 - y_1'| - \tau]_+ + [|y_2 - y_2'| - \tau]_+, \qquad (6)$$

where $[\pi]_+ = \max\{0, \pi\}$, and τ is a user defined tolerance parameter.

We use two layers of bidirectional LSTMs for the DeepSegmentor model with dropout [25] after each recurrent layer. We extracted the 13 Mel-Frequency Cepstrum Coefficients (MFCCs), without the deltas, every 10 ms, and use them as inputs to the network. We optimize the networks using AdaGrad [26]. All parameters were tuned on a dedicated development set for both of the models. As for the NLL models, we trained 4 different models; LSTM with one and two layers, and bidirectional LSTM with one and two layers, denoted as RNN, 2RNN, BI-RNN and BI-2-RNN, respectively. Table 1 summarizes the results for both models.

Table 1. *The mean loss for NLL models and DeepSegmentor for the word segmentation task. Results are reported for the onset, offset and overall CD separately. The loss function was measured using (6) (with τ=0) in frames of 10ms.*

	RNN	2-RNN	BI-RNN	BI-2-RNN	DeepSeg.
Onset	6.0	5.84	2.88	3.48	**2.02**
Offset	9.43	8.92	4.46	**3.75**	3.96
CD	15.42	14.76	7.35	7.24	**5.98**

Besides being efficient and more elegant, DeepSegmentor is superior to the NLL models when measuring (6), with the exception of BI-2-RNN, which was slightly better for the offset measurement.

4.2. VOT Segmentation

Voice onset time (VOT) is the time between the onset of a stop burst and the onset of voicing. As noted in the introduction, it is widely used in theoretical and clinical studies as well as ASR tasks. In this problem the input is a speech utterance containing a single stop consonant, and the output is the VOT onset and offset times.

We compared our model to two other methods for VOT measurement. First is the *AutoVOT* algorithm [9]. This algorithm follows the structured prediction approach of linear classifier with hand-crafted features and feature-functions. The second algorithm is the *DeepVOT* algorithm [11]. This algorithm uses RNNs with NLL as loss function. Hence, it predicts for each frame whether it is related to the VOT or not. Using the RNN predictions, a dynamic programming algorithm is applied to find the best onset and offset times. Our approach combines both of these methods while jointly training RNN with structured loss function.

[1] All models were implemented using Torch7 toolkit [21, 22]

4.2.1. Datasets

We use two different datasets. The first one, PGWORDS, is from a laboratory study by Paterson and Goldrick [6]. American English monolinguals and Brazilian Portuguese (L1)-English bilinguals (24 participants each) named a set of 144 pictures. Productions other than the intended label as well as those with code-switching or disfluencies were excluded. VOT of remaining words was annotated by one transcriber.

The second dataset, BB, consists of spontaneous speech from the 2008 season of Big Brother UK, a British reality television show [27, 9]. The speech comes from 4 speakers recorded in the "diary room," an acoustically clean environment. VOTs were manually measured by two transcribers.

4.2.2. Results — PGWORDS

For the PGWORDS dataset we use two layers of bidirectional LSTMs with dropout after each recurrent layer. We use (6) as our loss function. The input features are the same as in [9, 11]; overall we have 63 features per frame. We optimize the networks using AdaGrad optimization. All parameters were tuned on a dedicated development set. Table 2 summarizes the results using the same loss function as in [9]. Results suggests that DeepSegmentor outperforms the AutoVOT model over all tolerance values. However, when comparing to DeepVOT, the picture is mixed. In the lower tolerance values DeepSegmentor is superior to the DeepVOT while for higher values DeepVOT performs better. We believe these results are due to the DeepVOT being less delicate and solving a much coarser problem than the DeepSegmentor ; hence, it performs better when considering high tolerance values. We believe the integration between these two systems, (using DeepVOT as pre-training for the DeepSegmentor), will yield more accurate and robust results. We leave this investigation for future work.

Table 2. *Proportion of differences between automatic and manual measures falling at or below a given tolerance value (in msec). For example, for* DeepVOT*, the difference between automatic and manual measurements in the test set was 2 msec or less in 53.8% of examples. These results are for the* PGWORDS *dataset.*

Model	$t \leq 2$	$t \leq 5$	$t \leq 10$	$t \leq 15$	$t \leq 25$	$t \leq 50$
AutoVOT	49.1	81.3	93.9	96.0	97.2	98.1
DeepVOT	53.8	91.6	**97.6**	**98.7**	**99.6**	**100**
DeepSeg.	**78.2**	**94.1**	97.1	98.6	99.1	99.4

4.2.3. Results — BB

For the BB dataset we use two layers of LSTMs with dropout after each recurrent layer. We have experiences with bidirectional LSTMs as well but only forward LSTM performs better on this dataset. We use (6) as our loss function. We use the same features as in [9, 11], overall we have 51 features per frame. We optimize the networks using AdaGrad optimization. All parameters were tuned on a dedicated development set. Table 3 summarize the results using the loss function as in [9]. It is worth notice that we see the same behavior on this dataset as well, regarding the DeepVOT preforms better then the DeepSegmentor in hight tolerance values.

Table 3. *Proportion of differences between automatic and manual measures falling at or below a given tolerance value (in msec). These results are for the* BB *dataset.*

Model	$t \leq 2$	$t \leq 5$	$t \leq 10$	$t \leq 15$	$t \leq 25$	$t \leq 50$
AutoVOT	59.1	80.5	89.9	94.3	96.8	98.1
DeepVOT	60.3	84.2	**94.3**	94.9	**98.1**	**98.7**
DeepSeg.	**64.8**	**85.5**	**94.3**	**95.0**	96.2	97.5

5. FUTURE WORK

Future work will explore timing sequence of length greater than 2 - for instance, in phoneme segmentation, where the sequence varies across training examples. The model's robustness to noise and length as well as its ability to generalize are also key areas of future development. We would therefore like to explore training the model in two stages: first as a multi-class version and then fine-tuning using structured loss. With respect to machine learning, future directions include the effect of network size, depth, and loss function on model performance.

6. CONCLUSION

In this paper we present a new algorithm for speech segmentation and evaluate its performance to two different tasks. The proposed algorithm combines structured loss function with recurrent neural networks and outperforms current state-of-the-art methods.

7. REFERENCES

[1] Francis Kubala, Tasos Anastasakos, Hubert Jin, Long Nguyen, and Richard Schwartz, "Transcribing radio news," in *ICSLP*, 1996, vol. 2, pp. 598–601.

[2] David Rybach, Christian Gollan, Ralf Schluter, and Hermann Ney, "Audio segmentation for speech recognition

using segment features," in *ICASSP*, 2009, pp. 4197–4200.

[3] L. Lisker and A. Abramson, "A cross-language study of voicing in initial stops: acoustical measurements," *Word*, vol. 20, pp. 384–422, 1964.

[4] J.H.L. Hansen, S.S. Gray, and W. Kim, "Automatic voice onset time detection for unvoiced stops (/p/,/t/,/k/) with application to accent classification," *Speech Commun.*, vol. 52, pp. 777–789, 2010.

[5] P. Auzou, C. Ozsancak, R.J. Morris, M. Jan, F. Eustache, and D. Hannequin, "Voice onset time in aphasia, apraxia of speech and dysarthria: a review," *Clin. Linguist. Phonet.*, vol. 14, pp. 131–150, 2000.

[6] Nattalia Paterson, *Interactions in Bilingual Speech Processing*, Ph.D. thesis, Northwestern University, 2011.

[7] Doroteo Torre Toledano, Luis A Hernández Gómez, and Luis Villarrubia Grande, "Automatic phonetic segmentation," *IEEE transactions on speech and audio processing*, vol. 11, no. 6, pp. 617–625, 2003.

[8] Joseph Keshet, Shai Shalev-Shwartz, Yoram Singer, and Dan Chazan, "A large margin algorithm for speech-to-phoneme and music-to-score alignment," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 15, no. 8, pp. 2373–2382, 2007.

[9] Morgan Sonderegger and Joseph Keshet, "Automatic measurement of voice onset time using discriminative structured predictiona)," *JASA*, vol. 132, no. 6, pp. 3965–3979, 2012.

[10] Yossi Adi, Joseph Keshet, and Matthew Goldrick, "Vowel duration measurement using deep neural networks," in *MLSP*, 2015, pp. 1–6.

[11] Yossi Adi, Joseph Keshet, Olga Dmitrieva, and Matt Goldrick, "Automatic measurement of voice onset time and prevoicing using recurrent neural networks," .

[12] Trinh Do, Thierry Arti, et al., "Neural conditional random fields," in *AISTATS*, 2010, pp. 177–184.

[13] Shuai Zheng, Sadeep Jayasumana, Bernardino Romera-Paredes, Vibhav Vineet, Zhizhong Su, Dalong Du, Chang Huang, and Philip HS Torr, "Conditional random fields as recurrent neural networks," in *ICCV*, 2015, pp. 1529–1537.

[14] Liang-Chieh Chen, Alexander G Schwing, Alan L Yuille, and Raquel Urtasun, "Learning deep structured models," in *ICML*, 2015.

[15] Eliyahu Kiperwasser and Yoav Goldberg, "Simple and accurate dependency parsing using bidirectional lstm feature representations," *arXiv preprint*, 2016.

[16] Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer, "Neural architectures for named entity recognition," *arXiv preprint*, 2016.

[17] Ioannis Tsochantaridis, Thorsten Joachims, Thomas Hofmann, and Yasemin Altun, "Large margin methods for structured and interdependent output variables," in *JMLR*, 2005, pp. 1453–1484.

[18] Jeffrey L. Elman, "Distributed representations, simple recurrent networks, and grammatical structure," *Machine learning*, vol. 7, no. 2-3, pp. 195–225, 1991.

[19] Alan Graves, Abdel-rahman Mohamed, and Geoffrey Hinton, "Speech recognition with deep recurrent neural networks," in *ICASSP*, 2013, pp. 6645–6649.

[20] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams, "Learning representations by back-propagating errors," *Cognitive modeling*, vol. 5, no. 3, pp. 1, 1988.

[21] Ronan Collobert, Koray Kavukcuoglu, and Clément Farabet, "Torch7: A matlab-like environment for machine learning," in *BigLearn, NIPS Workshop*, 2011, number EPFL-CONF-192376.

[22] Nicholas Léonard, Sagar Waghmare, and Yang Wang, "rnn: Recurrent library for torch," *arXiv preprint*, 2015.

[23] Angela Fink, *The Role of Domain-General Executive Functions, Conceptualization, and Articulation during Spoken Word Production*, Ph.D. thesis, Northwestern University, 2016.

[24] Ingrid Rosenfelder, Josef Fruehwald, Keelan Evanini, Scott Seyfarth, Kyle Gorman, Hilary Prichard, and Jiahong Yuan, "Fave (forced alignment and vowel extraction)," Program suite v1.2.2 10.5281/zenodo.22281, 2014.

[25] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *CoRR*, 2012.

[26] John Duchi, Elad Hazan, and Yoram Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *JMLR*, vol. 12, pp. 2121–2159, 2011.

[27] Max Bane, Peter Graff, and Morgan Sonderegger, "Longitudinal phonetic variation in a closed system," *Proc. CLS*, vol. 46, pp. 43–58, 2010.

Automatic Measurement of Pre-Aspiration

INTERSPEECH 2017
August 20–24, 2017, Stockholm, Sweden

Automatic Measurement of Pre-aspiration

Yaniv Sheena[1], *Míša Hejná*[2], *Yossi Adi*[1], *Joseph Keshet*[1]

[1]Department of Computer Science, Bar-Ilan University, Ramat-Gan, Israel
[2]Department of English, Aarhus University, Denmark

yaniv.sheena@live.biu.ac.il, misa.hejna@cc.au.dk, jkeshet@cs.biu.ac.il

Abstract

Pre-aspiration is defined as the period of glottal friction occurring in sequences of vocalic/consonantal sonorants and phonetically voiceless obstruents. We propose two machine learning methods for automatic measurement of pre-aspiration duration: a feedforward neural network, which works at the frame level; and a structured prediction model, which relies on manually designed feature functions, and works at the segment level. The input for both algorithms is a speech signal of an arbitrary length containing a single obstruent, and the output is a pair of times which constitutes the pre-aspiration boundaries. We train both models on a set of manually annotated examples. Results suggest that the structured model is superior to the frame-based model as it yields higher accuracy in predicting the boundaries and generalizes to new speakers and new languages. Finally, we demonstrate the applicability of our structured prediction algorithm by replicating linguistic analysis of pre-aspiration in Aberystwyth English with high correlation.

Index Terms: pre-aspiration, feedforward neural network, structured prediction, laboratory phonology

1. Introduction

Pre-aspiration is a period of (primarily) glottal friction, which is found in the sequences of sonorants and phonetically voiceless obstruents prior to the release of the consonant, e.g., in Welsh English *kit* /kʰɪʰtˢ/, *miss* /mɪʰs/, *milk* /mɪl̥kʰ/, etc. Although pre-aspiration was previously considered rare [1,2], with the advances in recording technology it has been recently reported to occur in increasingly more languages and varieties thereof [3–15]. Recent studies of pre-aspiration have revealed that the phenomenon is very individual [1, 7, 16], sensitive to sex/gender [17], and can be affected by age [5, 7] and the first language of the speaker [17].

Most of the work on pre-aspiration has been based on subjective, labor-intensive manual annotation. The vast majority of researchers have coded pre-aspiration manually by identifying its boundaries, usually based on a manual segmentation of the signal into segmental and subsegmental intervals [1, 4, 7, 9, 10, 12, 13, 17–21].

In this paper we tackle the problem of automatic measurement of pre-aspiration duration. We propose two algorithms for this purpose. The first one is a feedforward neural network, which works at the frame-level, and the second algorithm is based on structured prediction techniques which rely on highly tuned feature maps which were specifically designed for the task, and which works at the segment level.

As far as we know this is the first attempt to automatically detect pre-aspiration. This work is based on our ongoing work on designing and developing machine learning algorithms for automatically measuring with high precision, phonetic properties of speech at the level of human inter-transcriber reliability [22–26]. Our methods rely on several advances over existing computational systems: novel representations of the speech signal and new structured prediction and deep learning algorithms. These automatic methods allow for low-cost replication of phonetic studies and expand the range of empirical and theoretical issues we can address.

The code is available to used by the research community. It can be downloaded from `https://github.com/MLSpeech/AutoPreaspiration`.

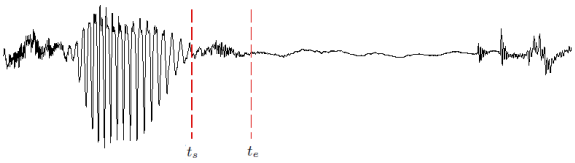


Figure 1: *Annotation example of pre-aspiration. The signal consists of a vowel followed by a plosive of the word "cook".*

2. Problem Setting

In the context of a typical phonological study, given a segment of an acoustic signal of an arbitrary length, the goal of an automatic pre-aspiration measurement algorithm is to predict the onset and offset times of pre-aspiration as accurate as possible. In this work we assume that the acoustic signal contains exactly one coded obstruent within a word and the signal starts before the expected pre-aspiration, i.e., during the previous phoneme.

We turn to describing the problem formally. Throughout the paper, scalars are denoted using lower case Latin letters, e.g., x, and vectors using bold face letters, e.g., $\mathbf{x}$. A sequence of elements is denoted with a bar $\bar{\mathbf{x}}$ and its length is written as $|\bar{\mathbf{x}}|$.

The acoustic input is represented as a sequence of feature vectors, $\bar{\mathbf{x}} = (\mathbf{x}_1, \ldots, \mathbf{x}_T)$, where each vector, $\mathbf{x}_t$, is D-dimensional vector which represents the acoustic content of the t-frame ($1 \leq t \leq T$). The domain of the feature vectors is denoted as $\mathcal{X} \subset \mathbb{R}^D$. The acoustic input is of an arbitrary length, hence the number of frames, T, is not fixed. We denote by $\mathcal{X}^*$ the set of all finite-length sequences over $\mathcal{X}$.

Each acoustic input is associated with a timing pair: the *onset* of the pre-aspiration, $t_s \in \mathcal{T}$, and the *offset* of the pre-aspiration, $t_e \in \mathcal{T}$, where $\mathcal{T} = \{1, \ldots, T\}$. To sum up, given the speech interval $\bar{\mathbf{x}}$, our goal is to learn a function f from the domain of all acoustic inputs $\mathcal{X}^*$ to the domain of all possible onset offset pairs $\mathcal{T}^2$. Our notation is depicted in Figure 1.

3. Learning Apparatus

In this section, we describe how we learn the prediction function from a training set of examples. We denote the training set of

Copyright © 2017 ISCA

http://dx.doi.org/10.21437/Interspeech.2017-870

m examples by $S = \{(\bar{\mathbf{x}}^i, t_s^i, t_e^i)\}_{i=1}^m$, where the i-th example is composed of a sequence of acoustic features $\bar{\mathbf{x}}^i$ labeled with an onset and offset pair, (t_s^i, t_e^i).

Let us denote the predicted onset and offset pair by $(\hat{t}_s, \hat{t}_e)$, namely $(\hat{t}_s, \hat{t}_e) = f_\theta(\bar{\mathbf{x}})$, where θ denotes the set of parameters of the prediction function f. In order to assess the quality of the prediction we use a *task loss* function, $\gamma\left((t_e, t_s), (\hat{t}_e, \hat{t}_s)\right)$, which returns a real positive number that measures how the prediction pair $(\hat{t}_s, \hat{t}_e)$ is close to the manually annotated pair (t_e, t_s). Our goal in learning is to find the set of parameters θ so as to minimize the expected task loss.

We start by describing the acoustic features, and then two different machine learning algorithms that learn the parameters of function f. The first algorithm works at the frame level and hence cannot aim at minimizing a global task loss function, while the second algorithm is aimed directly at minimizing the expected task loss. These differences are reflected in the accuracy level of the different algorithms.

3.1. Acoustic features

For both machine learning models, we extract the same set of features. Consider the acoustic input $\bar{\mathbf{x}} = (\mathbf{x}_1, \ldots, \mathbf{x}_T)$ consisting of T frames, where each acoustic feature vector $\mathbf{x}_t$ consists of D features. Similar to [22], we extract eight (D=8) acoustic features from the speech signal every 1 ms. The first four features refer to the total spectral energy (E_{total}), energy between 50-1000 Hz (E_{low}), energy above 3000 Hz (E_{high}), and Wiener entropy (H_{wiener}) — all are based on the short-time Fourier transform (STFT) taken every 1 ms with a 5 ms Hamming window. The fifth feature, P_{max}, is the maximum of the power spectrum calculated in a region from 6 ms before to 18 ms after the frame center. The sixth feature (R_l) is the pitch of the signal using a real-time pitch detector [27]. The seventh feature is the $0/1$ output of a voicing detector based on the RAPT pitch tracker [28], smoothed with a 5 ms Hamming window (V). The last feature is the number of zero crossings (ZC) in a 5 ms window around the frame center.

3.2. Frame-based model

Our first model works at the frame level. We train a binary classifier such that given an input speech frame $\mathbf{x}_t$ predicts whether or not it is associated with a pre-aspiration event. In order to take advantage of the local temporal context of each time-frame, the input of this binary classifier is based on five concatenated feature vectors $(\mathbf{x}_{t-2}, \mathbf{x}_{t-1}, \mathbf{x}_t, \mathbf{x}_{t+1}, \mathbf{x}_{t+2})$ rather than a single frame. We use a feedforward neural network. We tried several network architectures, and chose the one that performed best on a validation set. The network consists of an input layer of 40 units (recall that we extract $D = 8$ feature per frame, and have 5 consecutive frames), a hidden layer of 40 units with ReLU activation function and a dropout rate of 0.3, and one output unit followed by a sigmoid. The network was trained to minimize the binary cross entropy loss using the gradient descent optimization algorithm.

At inference time, we use the trained model sequentially over the input frames to produce a sequence of predictions. Since this sequence can be noisy, we smooth it out using a moving average, followed by a binary mapping that uses a threshold for each such average. In the final step, we search the longest subsequence of frames that were predicted as associated with pre-aspiration, and output its boundaries as the onset and offset of the pre-aspiration.

3.3. Structured model

The second algorithm takes advantage of the input as a whole segment, hence we can introduce feature maps such as typical pre-aspiration duration, mean energy during the presumed pre-aspiration compared to the mean energy before or after the pre-aspiration. Similar to previous work in structured prediction [29, 30], the function f is constructed from a predefined set of N feature maps $\{\phi_i\}_{i=1}^N$, each of the form $\phi_i : \mathcal{X} \times \mathcal{T} \times \mathcal{T} \times \rightarrow \mathbb{R}^N$, and a weight vector $\mathbf{w} \in \mathbb{R}^N$. The function is a linear predictor of the following form

$$(\hat{t}_s, \hat{t}_e) = \underset{(t_s, t_e)}{\arg\max}\, \mathbf{w} \cdot \boldsymbol{\phi}(\bar{\mathbf{x}}, t_s, t_e), \tag{1}$$

where we have used vector notation for the feature maps $\boldsymbol{\phi} = (\phi_1, \ldots, \phi_N)^\top$. This vector-valued function is used to map the variable length of input speech along with a presumed onset-offset pair to an abstract vector space in $\mathbb{R}^N$ (described in 3.4).

The algorithm presented here aims to find the weights $\mathbf{w}$ that minimize the expected task loss function γ which measures the distance between predicted and manually coded labels. In a similar manner to [22], we define the task loss function as follows:

$$\gamma\left((t_e, t_s), (\hat{t}_e, \hat{t}_s)\right) = \max\{|(\hat{t}_e - \hat{t}_s) - (t_e - t_s)| - \epsilon, 0\}. \tag{2}$$

That is, only the differences between the predicted pre-aspiration and the manually labeled pre-aspiration that are greater than a threshold ϵ (in milliseconds), are penalized. This task loss function takes into account that manual measurements are not usually exact, and ϵ can be adjusted according to the level of measurement uncertainty in a dataset.

The weight vector $\mathbf{w}$ is learned using an iterative algorithm based on the *Passive-Aggressive* family of algorithms for structured prediction [31]. Let $\mathbf{w}_t$ be the weight vector after the t-th iteration, and let $\mathbf{w}_0 = \mathbf{0}$. At each iteration a single example $(\bar{\mathbf{x}}^i, t_s^i, t_e^i)$ is considered, and the current weight vector $\mathbf{w}_t$ is updated by finding the solution to the following optimization problem:

$$\begin{aligned} \mathbf{w}_{t+1} = \underset{\mathbf{w}, \xi \geq 0}{\arg\min}\ & \frac{1}{2}\|\mathbf{w} - \mathbf{w}_t\|^2 + C\xi \\ \text{s.t. } & \mathbf{w} \cdot \boldsymbol{\phi}(\bar{\mathbf{x}}^i, t_s^i, t_e^i) - \mathbf{w} \cdot \boldsymbol{\phi}(\bar{\mathbf{x}}^i, \tilde{t_s}, \tilde{t_e}) \geq \gamma_i - \xi\ , \end{aligned} \tag{3}$$

where $\gamma^i = \gamma\left((t_s^i, t_e^i), (\tilde{t_s}, \tilde{t_e})\right)$, C serves as a trade-off parameter between loss and regularization minimization, and ξ is a non-negative slack variable.

The optimization problem tries to keep the new weight vector $\mathbf{w}$ close to the previous weight vector $\mathbf{w}_t$ while satisfying the constraint that the score of the manually annotated onset-offset pair $\mathbf{w} \cdot \boldsymbol{\phi}(\bar{\mathbf{x}}^i, t_s^i, t_e^i)$ will be higher than a different set of onset-offset, $\mathbf{w} \cdot \boldsymbol{\phi}(\bar{\mathbf{x}}^i, \tilde{t_s}, \tilde{t_e})$, where the onset-offset pair $(\tilde{t_s}, \tilde{t_e})$ in the constraint is the most violated pair [30], namely:

$$(\tilde{t_s}, \tilde{t_e}) = \underset{(\tilde{t_s}, \tilde{t_e})}{\arg\max}\, \mathbf{w} \cdot \boldsymbol{\phi}(\bar{\mathbf{x}}^i, \tilde{t_s}, \tilde{t_e}) + \gamma^i. \tag{4}$$

3.4. Feature maps for the structured model

The feature maps are built from local differences, cumulative mean and max over subsets features with respect to the pre-aspiration boundaries. They were chosen based on manual inspection of the values and the trends of the features in intervals

Table 1: *Summary of the feature maps. The rows represent the type of calculations that should be performed. The columns are the eight acoustic features. F in row i and column j indicates that there is a feature map of type i for feature x_j; Δ indicates that there are three feature maps of type i for the local difference of feature x_j, evaluated at $s = 5, 10, 15$ [22]. For example, the F in the last row, column E_{high} denotes a feature-map that gets the mean of the value of E_{high} over 50 frames starting from t_e.*

Feature map type	E_{total}	E_{low}	E_{high}	H_{wiener}	P_{max}	R_l	V	ZC
Value at t_s	F, Δ	Δ	F, Δ	F, Δ	Δ	F, Δ	Δ	Δ
Value at t_e	F, Δ	F, Δ	F, Δ	F, Δ	F, Δ	F, Δ		
Mean & max over (t_s, t_e)					F, Δ_5, Δ_{10}			
Mean over (t_s, t_e) - mean over $(t_e, t_e + 50)$			F	F				F
Mean over $(t_e, t_e + 50)$			F	F		F	F	
Max over $(t_e, t_e + 50)$			F	F				

near t_s and t_e, and they reflect the knowledge about the problem of pre-aspiration measurement, which in our case is based on [7].

As an example for such a feature map we considered the observation that when pre-aspiration is immediately followed by a silent interval in the context of plosives, we expect that high frequencies (above 3000 Hz) will decrease compared to the interval of pre-aspiration, which has high energy presence in these frequencies. In order to express this observation, we define feature maps that compute the differences of the means of E_{high} and H_{wiener} over a post pre-aspiration interval $(t_e, t_e + 50)$ and the pre-aspiration interval (t_s, t_e).

Another key observation is that t_s usually comes right after a formant structure which becomes less distinct and usually indicates that voicing is ending. Hence a set of feature maps are based on the local differences of 5, 10 and 15 milliseconds over the acoustic features P_{max} and V in order to allow our algorithm to capture these changes. A full specification of ϕ is shown in Table 1.

4. Data Set

The data is composed of 5,297 examples of 16 speakers of English from the town of Aberystwyth within mid Wales. All of the speakers were born and raised in the town and are L1 Welsh speakers. The speakers presented are 10 females and 6 males, and their age range spans from 22–91 to allow for preliminary generational comparisons. The data itself consisted of a list of words that the participants read; each word contained a sequence of a vowel and a post-tonic plosive. Each word was read once in isolation and twice in a frame sentence. For more details on the segmental and prosodic characteristics of the tokens related to aspects such as vowel height, place and manner of articulation of the plosive, stress, and foot-position see [7]. The recordings were obtained with an H4 Zoom Handy Recorder (sampled at 44.1kHz) in conjunction with the head-mounted AKG C520 Microphone, which was attached to the speaker's head and ensured a constant distance from the mouth, irrespective of the speaker's movements.

5. Experiments

We evaluated the performance of our algorithms using three main methods. First, we compared the predictions of the frame-based algorithm and of the structured algorithm to the manual annotations. Then we tested how the structured prediction algorithm, which was superior to the frame-based algorithm, generalizes on new set of speakers and languages. Finally, we replicated a linguistic study on the model predictions and compared it to the results obtained from the manually annotated data. All of the experiments in this section are based on models which were trained using search windows of 50 ms before the labeled left boundary (t_s) and 60 ms after the right boundary (t_e), where we tried several window-sizes and the effects were insignificant.

5.1. Models performance

We evaluated both algorithms on the corpus described in the previous section using 5-fold cross-validation. For each fold, 15% of the data served as a validation set. We regularized both algorithms using early stopping. For the frame-based algorithm, we balanced the data by randomly dropping negative examples, as the amount of such examples was significantly greater than the amount of positive examples. For the structured model we used $C = 50$, which was chosen on a validation set, and ϵ in the task loss was set to 2 ms.

We compared the models by reporting the percentage of examples in the test set with automatic/manual differences which were less than a time tolerance, where we used tolerances of 5, 10, 15, and 20 ms. We also report the average error in the prediction of the onset and the offset. The results are given in Table 2. Results should read as follows: The percentage of correctly predicted pre-aspirations within a threshold of 2 ms was 43.3% for the frame-based algorithm and 56.2% for the structured algorithm. The mean difference between the predicted and manually-annotated onset was 4.4 ms for the frame-based algorithm and 2.6 ms for the structured algorithm.

Table 2: *Results for the two algorithms. The first 4 columns are the percentage of accurately predicted pre-aspirations within a given threshold (thresholds in ms, values are percentages). The remaining columns are the mean distance between the boundaries of manual/automatic pre-aspirations in ms.*

Algorithm	$\leq$5	$\leq$10	$\leq$15	$\leq$20	Δt_s	Δt_e
Frame-based	43.3	74.3	88.9	95.3	4.4	5.3
Structured	56.2	84.8	93.1	96.3	2.6	4.9

In Table 3 we report the mean and the standard deviation of the manually annotated pre-aspiration duration and the predicted pre-aspiration duration for both algorithms.

Table 3: *Mean and standard deviation of pre-aspiration duration. Manually coded versus models' predictions.*

Type	Mean (ms)	Standard deviation (ms)
Manual	37.6	20.8
Structured	37.2	19.9
Frame-based	42.0	17.8

Results suggest that the structured algorithm is superior to the framed-based algorithm. This demonstrates the power of the structured algorithm, which uses a set of dedicated feature-maps each of which measures properties and trends of intervals surrounding the boundaries of the pre-aspiration, contrary to the frame-based method that is based on local windows. Henceforth, we conducted the rest of the experiments using only the structured algorithm.

5.2. Generalization

In order to test our algorithm generalization on data produced by different sources, we conducted several experiments. First, we performed leave-one-speaker-out (LOSO) using the 16 speakers of Aberystwyth English, i.e., the utterances of each speaker were tested individually with a structured model trained on the rest of the speakers. The averaged results are summarized in the first row of Table 4. Note that the results are very similar to the results shown in the second row of Table 2, where the data was randomly partitioned using 5-fold cross-validation.

Next, we evaluated our structured algorithm on a data set consisting of 607 speech intervals of the Welsh language (16 speakers from Bethesda, North Wales). The performance of the algorithm on the Welsh corpus using 4-fold cross-validation results are summarized in the second row of Table 4. Our goal was to compare the performance of the structured algorithm that was trained on one language on a different language. The last two rows in Table 4 outline the performance of the algorithm that trained on Aberystwyth English but was tested on Welsh and vice versa.

It is interesting to note that there are different performance drops when languages are mismatched. When training on the Aberystwyth data and testing on Welsh data results drop by 7.2% with a tolerance level of 10 ms. However, when tested on Aberystwyth data and training on the Welsh data we suffered only a 4.3% decrease in results, again with a 10 ms tolerance level.

Table 4: *Mismatched training and test sets. LOSO stands for leave-one-speaker-out. Thresholds and time differences in ms, values are percentages.*

Train/Test	$\leq$5	$\leq$10	$\leq$15	$\leq$20	Δt_s	Δt_e
Aber./Aber. LOSO	54.0	81.9	90.5	94.3	2.9	5.6
Welsh/Welsh	52.7	76.7	87.1	91.7	5.3	4.5
Aber./Welsh	45.8	69.5	84.4	88.8	6.6	6.5
Welsh/Aber.	54.6	80.5	90.5	94.7	3.2	5.2

5.3. Linguistic analysis

As the purpose of the algorithm is to reliably measure pre-aspiration duration for linguistic and language-related analyses, we also carried out an analysis of what language-internal and language-external factors affect pre-aspiration duration measured manually as opposed to pre-aspiration duration measured automatically with the algorithm. The outputs of four Linear Mixed Effects Models were compared. These models differed only in the dependent variable, which was (i) manually coded raw pre-aspiration duration, (ii) automatically coded raw pre-aspiration duration, (iii) manually coded normalized pre-aspiration duration, and (iv) automatically coded normalized pre-aspiration duration. The normalization method expressed the raw measurements as a percentage of the overall word duration.

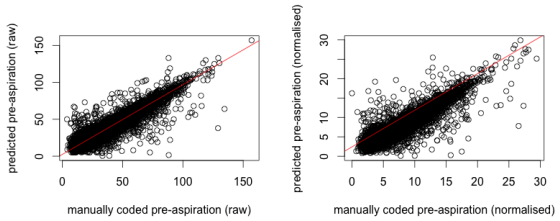


Figure 2: *Correlations between the manually coded and the predicted durations of pre-aspiration; raw (left) and normalized (right) values.*

The models shared the following independent variables: Age (continuous variable), Sex (with two levels: *female* and *male*), Vowel type (with eight levels: /a/ contrasted with /e/, /ɪ/, /ɒ, /ʊ/, /ʌ/, /ɑː/, and /oː/), Place of articulation of the plosive (with three levels: /p/, /t/, /k/), and Word position (with two levels: *word-medial* and *word-final*); Speaker and Word were selected as random effects. Forward difference coding was applied to the place of articulation.

The tests show the same results regarding language-internal variables, irrespective of whether normalized or raw data are used. In all cases, pre-aspiration intervals are longest with /a/ as opposed to the other vowels ($p < 0.0001$), with the exception of /ɒ/, which does not differ from /a/ in the durations of pre-aspiration with which it is associated. Furthermore, the duration of pre-aspiration increases as we move further back in the oral cavity (/p/ $<$ /t/ $<$ /k/; $p < 0.0001$). In addition, pre-aspiration is longer foot-finally than medially ($p < 0.0001$). The models also yield the same results concerning the external variable of sex in that no effect is found (p = 0.06-0.39). The only difference across the models is related to the variable of age, which comes out as significant when the dependent variable is manually coded normalized pre-aspiration (< 0.05; as opposed to p = 0.06-1). Moreover, there is a very strong positive correlation between the predicted and the manually coded values (raw measurements: r = 0.87; normalized measured: r = 0.89; $p < 0.0001$; Pearson's product moment correlation) as shown in Figure 2.

6. Discussion

We have presented a new trainable algorithm for automatic measurement of pre-aspiration. To our knowledge, this is the first attempt to develop an automatic tool for measuring pre-aspiration. We have shown that the structured algorithm outperforms the frame-based neural network algorithm. Furthermore, we reproduced the results of a linguistic analysis based on pre-aspiration, solely using our structured algorithm's predictions. Future work can involve extending the use of the algorithm to fricative context, automatically detecting whether there is pre-aspiration in a given interval, and applying the presented methods to lower quality data.

7. Acknowledgements

We wish to thank Jon Morris for letting us use his Welsh data.

8. References

[1] P. Helgason, "Preaspiration in the Nordic Languages. Synchronic and Diachronic Aspects," Ph.D. dissertation, Department of Linguistics, Stockholm University, 2002.

[2] D. Silverman, "On the rarity of pre-aspirated stops," *Journal of Linguistics*, vol. 39, pp. 575–598, 2003.

[3] M. Clayards and T. Knowles, "Prominence enhances voicelessness and not place distinction in English voiceless sibilants," in *Proceedings of the 18th International Congress of Phonetic Sciences (ICPhS), Glasgow*, 2015.

[4] C. di Canio, "The phonetics of fortis and lenis consonants in Itunyoso Trique," *International Journal of American Linguistics*, vol. 78, no. 2, pp. 239–272, 2012.

[5] G. Docherty and P. Foulkes, "Derby and Newcastle: instrumental phonetics and variationist studies," in *Urban Voices: Accent Studies in the British Isles*, P. Foulkes and G. Docherty, Eds. Routledge, 1999, pp. 47–71.

[6] O. Gordeeva and J. Scobbie, "Preaspiration as a correlate of word-final voice in Scottish English fricatives," in *Turbulent Sounds: an Interdisciplinary Guide*, 2010, pp. 167–207.

[7] M. Hejná, "Preaspiration in Welsh English: A Case Study of Aberystwyth," Ph.D. dissertation, Department of Linguistics, University of Manchester, 2015.

[8] M. Hejná and J. Scanlon, "New laryngeal allophony in Manchester English," in *Proceedings of the 18th International Congress of Phonetic Sciences (ICPhS), Glasgow*, 2015.

[9] J. J. Jones and C. Llamas, "Fricated pre-aspirated /t/ in Middlesbrough English: an acoustic study," in *Proceedings of the 15th International Congress of Phonetic Sciences (ICPhS), Barcelona*, 2003, pp. 655–658.

[10] T. Kettig, "The BAD-LAD Split: a Phonetic Investigation," Ph.D. dissertation, Department of Theoretical and Applied Linguistics, University of Cambridge, 2015.

[11] M. Roos, "Preaspiration in Western Yugur monosyllables," *Turgologica*, vol. 32, pp. 28–41, 1998.

[12] M. Stevens, "How widespread is preaspiration in Italy? a preliminary acoustic phonetic overview," *Lund University Centre for Languages and Literature Phonetics Working Papers*, vol. 54, pp. 97–102, 2010.

[13] M. Stevens and J. Hajek, "How pervasive is preaspiration? investigating sonorant devoicing in Sienese Italian," *Proceedings of the 10th Australian International Conference on Speech Science and Technology*, pp. 334–339, 2004.

[14] K. Watson, "The Phonetics and Phonology of Plosive Lenition in Liverpool English," Ph.D. dissertation, University of Lancaster, Edge Hill College, 2007.

[15] A. Dwyer, "Consonantalization and obfuscation," *Turgologica*, vol. 46, pp. 423–432, 2000.

[16] C. Ringen and W. A. van Dommelen, "Quantity and laryngeal contrasts in Norwegian," *Journal of Phonetics*, vol. 41, pp. 479–490, 2013.

[17] C. Nance and J. Stuart-Smith, "Pre-aspiration and post-aspiration in Scottish Gaelic stop consonants," *Journal of the International Phonetic Association*, vol. 43, no. 2, pp. 129–152, 2013.

[18] M. Hejná, "Multiplicity of the acoustic correlates of the fortis-lenis contrast: plosives in Aberystwyth English," in *Proceeding of Interspeech*, 2016, pp. 3147–3151.

[19] P. Helgason and C. Ringen, "Voicing and aspiration in Swedish stops," *Journal of Phonetics*, vol. 36, pp. 607–628, 2008.

[20] P. Helgason, "The perception of medial stop contrasts in Central Standard Swedish: a pilot study," *Proceeding of FONETIK 2004, Stockholm*, pp. 92–95, 2004.

[21] W. A. van Dommelen, "Production and perception of preaspiration in Norwegian," in *Proceeding of FONETIK 1998, Stockholm*, 1998.

[22] M. Sonderegger and J. Keshet, "Automatic measurement of voice onset time using discriminative structured prediction)," *The Journal of the Acoustical Society of America*, vol. 132, no. 6, pp. 3965–3979, 2012.

[23] Y. Adi, J. Keshet, E. Cibelli, E. Gustafson, C. Clopper, and M. Goldrick, "Automatic measurement of vowel duration via structured prediction," *The Journal of the Acoustical Society of America*, vol. 140, no. 6, pp. 4517–4527, 2016.

[24] Y. Adi, J. Keshet, O. Dmitrieva, and M. Goldrick, "Automatic measurement of voice onset time and prevoicing using recurrent neural networks," in *Proceeding of the 17th Annual Conference of the International Speech Communication Association (Interspeech)*, 2016.

[25] Y. Dissen and J. Keshet, "Formant estimation and tracking using deep learning," in *Proceeings of the 17th Annual Conference of the International Speech Communication Association (Interspeech)*, 2016.

[26] Y. Adi, J. Keshet, E. Cibelli, and M. Goldrick, "Sequence segmentation using joint RNN and structured prediction models," in *Proceeding of the 42st IEEE International Conference in Acoustic, Speech and Signal Processing (ICASSP)*, 2017.

[27] F. Sha and L. K. Saul, "Real-time pitch determination of one or more voices by nonnegative matrix factorization," in *Proceeding of NIPS*, 2004, pp. 1233–1240.

[28] D. Talkin, "A robust algorithm for pitch tracking (RAPT)," in *Speech coding and synthesis*, W. Kleijn and K. Paliwal, Eds. New York: Elsevier, 1995, pp. 495–518.

[29] B. Taskar, C. Guestrin, and D. Koller, "Max-margin Markov networks," in *Proceeding of NIPS*, 2003.

[30] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun, "Support vector machine learning for interdependent and structured output spaces," in *Proceeding of the 21st International Conference on Machine Learning (ICML)*, 2004.

[31] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer, "Online passive-aggressive algorithms," *Journal of Machine Learning Resaerch*, vol. 7, pp. 551–585, 2006.

Learning Similarity Functions for Pronunciation Variations

INTERSPEECH 2017
August 20–24, 2017, Stockholm, Sweden

Learning Similarity Functions for Pronunciation Variations

Einat Naaman, Yossi Adi, and Joseph Keshet

Department of Computer Science, Bar-Ilan University, Ramat-Gan, Israel
jkeshet@cs.biu.ac.il

Abstract

A significant source of errors in Automatic Speech Recognition (ASR) systems is due to pronunciation variations which occur in spontaneous and conversational speech. Usually ASR systems use a finite lexicon that provides one or more pronunciations for each word. In this paper, we focus on learning a similarity function between two pronunciations. The pronunciations can be the canonical and the surface pronunciations of the same word or they can be two surface pronunciations of different words. This task generalizes problems such as lexical access (the problem of learning the mapping between words and their possible pronunciations), and defining word neighborhoods. It can also be used to dynamically increase the size of the pronunciation lexicon, or in predicting ASR errors. We propose two methods, which are based on recurrent neural networks, to learn the similarity function. The first is based on binary classification, and the second is based on learning the ranking of the pronunciations. We demonstrate the efficiency of our approach on the task of lexical access using a subset of the Switchboard conversational speech corpus. Results suggest that on this task our methods are superior to previous methods which are based on graphical Bayesian methods.

1. Introduction

Spontaneous and conversational speech are significantly different both acoustically and linguistically from read speech. One of the key differences is the vast pronunciation variations in spontaneous and conversational speech, as the speaking rate is accelerated and consequently the pronunciation becomes reduced or coarticulated. Other factors, such as the neighboring words and the speaker's style, also influence the way words are produced.

We distinguish between two types of pronunciations of a word. The typical pronunciation found in a dictionary is called *canonical* pronunciation, whereas the actual way in which speakers produce the word is called *surface* pronunciation. Spontaneous speech often includes pronunciations that differ from the one found in the dictionary. For example, pronunciations of the word "probably" in the Switchboard conversational speech corpus include [p r aa b iy], [p r aa l iy], [p r ay], etc. [1]. Fewer than half of the word productions are pronounced canonically in the phonetically transcribed portion of Switchboard [2].

In this work we propose to learn a similarity function between two pronunciation variations. The function should score the similarity between any type of pronunciations. The input can be, for example, canonical and surface pronunciations or it can be two surface pronunciations. We expect such a function to output a high number if the inputs are canonical and surface pronunciations of the same word or if the input consists of two surface pronunciations of the same word.

Such a similarity measure can be utilized in many tasks, including lexical access, word neighborhood and pronunciation scoring. In the task of lexical access, the goal is to predict which word in the dictionary was uttered given its pronunciation in terms of sub-word units [3, 4, 5]. The problem of lexical access can be handled using a pronunciation similarity function as follows: the input surface pronunciation is compared to all of the (canonical) pronunciations in the dictionary using the similarity function, and the one with the maximal similarity is predicted as the articulated word.

In the word neighborhood task the goal is to find the acoustic neighborhood density of a given word [6]. It has been suggested as an explanatory variable for quantifying errors in ASR, but is also used in linguistic studies. In most works in the psycholinguistics literature, word neighborhood is defined to be the set of words which differ by a single phone from the given word. Here we propose a different definition which is closer to the definition suggested in [6]. The word neighborhood can be defined as all the words in the dictionary in proximity to the given word, where proximity is measured using the similarity function between pronunciations.

ASR systems are based on a finite dictionary which provides each word with one or more pronunciations. The variance of pronunciations in spontaneous and conversational speech leads to a high rate of errors in ASR systems [7, 8]. The standard approach to this problem is to expand the dictionary, either by adding alternate pronunciations with probabilities, or with phonetic substitution, insertion and deletion rules derived from linguistic knowledge and learned from data. A similarity function can be used to add pronunciation variations to each word either statically or dynamically during the dictionary lookup.

Mapping between words and their possible pronunciations in terms of sub-word units was explored in light of the lexical access task [9, 4, 5]. Note that this problem is different from the grapheme-to-phoneme problem, in which pronunciations are predicted from a word's spelling, whereas in lexical access we assume a dictionary of canonical pronunciations like the one used in speech recognition. Learning word neighborhood automatically was proposed by [6].

Our work is restricted to proposing algorithms for learning the pronunciation similarity score. Specifically we propose two different deep network architectures to tackle this problem. The first is based on binary classification of two recurrent neural networks (RNNs), while the second is based on triplet networks designed to rank the pronunciations.

The paper is organized as follows. Section 2 introduces our notation and the problem definition. In Section 3 we propose two deep network architectures to learn similarity between two pronunciations. In Section 4 we present a set of experiments on the Switchboard conversational speech corpus, and in Section 5 we analyze the experimental results. We conclude the paper in Section 6.

Copyright © 2017 ISCA

http://dx.doi.org/10.21437/Interspeech.2017-1117

2. Problem settings

We denote a word pronunciation by a sequence of phones, $\boldsymbol{p} = (p_1, \ldots, p_N)$ where $p_n \in \mathcal{P}$ for all $1 \leq n \leq N$ and $\mathcal{P}$ is the set of all sub-word units (all phones). Naturally, N is not fixed since the number of phonemes varies across different words. We denote the set of all finite-length phone streams as $\mathcal{P}^*$.

Given two pronunciation sequences, our goal is to find a similarity function between these two sequences. Formally, our goal is to learn a function $f : \mathcal{P}^* \times \mathcal{P}^* \rightarrow \mathbb{R}$ which gets as input two pronunciation sequences and returns the similarity between the two. That is if two pronunciations $\boldsymbol{p}_1, \boldsymbol{p}_2 \in \mathcal{P}^*$ are similar, then the function $f(\boldsymbol{p}_1, \boldsymbol{p}_2)$ will be high. Otherwise it will be low.

This type of function can be utilized in various tasks. For example, in the lexical access task [5] the goal is to predict which word w from a finite dictionary $\mathcal{V}$ is associated with a given surface pronunciation $\boldsymbol{p}^s$. Assume that the dictionary $\mathcal{V}$ is a list of pairs, where each pair is composed of a word w and its canonical pronunciation $\boldsymbol{p}^c$, namely $(w, \boldsymbol{p}^c) \in \mathcal{V}$. Define $sort^k$ as a function that gets a set of unordered scores and returns a vector of the top k maximal ordered scores. Similarly define $arg\, sort^k$ to be the function the returns the indices of the ordered scores. Then the best k-words can be found to be those whose canonical pronunciations get the highest similarity to the input surface pronunciation:

$$\boldsymbol{w} = \underset{(w,\boldsymbol{p}^c)\in\mathcal{V}}{\arg\mathrm{sort}^{\mathrm{k}}} f(\boldsymbol{p}^s, \boldsymbol{p}^c), \tag{1}$$

where $\boldsymbol{w}$ is a list of k-best words in the dictionary. Either the word with the highest unigram probability can be predicted or it can be combined to generated a prediction based on the n-gram probability.

In the word neighborhood task the goal is to find the acoustic neighborhood density of a given word [6]. This can be achieved by comparing the similarity of a canonical pronunciation of a given word to the set of all canonical pronunciations in the dictionary. Formally we propose to define word neighborhood $\mathcal{N}(w)$ for a given word w as the set of all words u in the dictionary $\mathcal{V}$ for which the similarity function is less than some threshold θ:

$$\mathcal{N}(w) = \{u \mid \forall (u, \boldsymbol{p}^u) \in \mathcal{V},\ f(\boldsymbol{p}^w, \boldsymbol{p}^u) < \theta\}, \tag{2}$$

where $\boldsymbol{p}^w, \boldsymbol{p}^u \in \mathcal{P}^*$ are the canonical pronunciations of the word w and u respectively.

3. Network architecture

Is this section we suggest two network architectures to learn the similarity function. Both architectures are based on Recurrent Neural Networks (RNNs) [10]. The first one is based on Siamese RNNs which were designed as a binary classifier and were trained to minimize the negative log likelihood loss function. The second architecture is based on three identical RNNs, which are combined together to create a ranking architecture and optimized with a ranking loss function. We also tried several sequence-to-sequence architectures [11, 12], but since all of them led to poor results in terms of Word Error Rate (WER), we will not discuss them in the paper.

3.1. Binary loss function

The first architecture is a network that is designed to learn a mapping between two pronunciations using a binary classifier, which is trained to predict whether two pronunciations are of the same word. Each example in the training set of m examples, $S = \{(\boldsymbol{p}_i^s, \boldsymbol{p}_i^c, y_i)\}_{i=1}^m$, is composed of a surface pronunciation $\boldsymbol{p}^s \in \mathcal{P}^*$, a canonical pronunciation $\boldsymbol{p}^c \in \mathcal{P}^*$ and a binary label $y \in \{-1, +1\}$, which indicates if both pronunciations are of the same word or not.

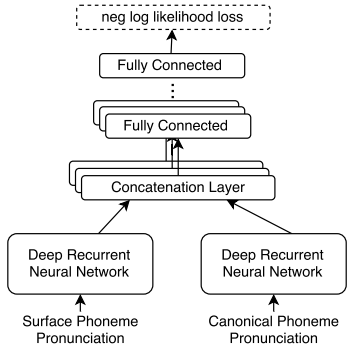


Figure 1: *Network architecture of the binary loss network. The two RNNs share the same parameters.*

We would like to train a neural network to learn this binary mapping. In order to do so, we encode both pronunciations using two RNNs with a shared set of parameters. The input to each of the RNNs is a sequence of phones, which represents either surface or canonical pronunciations, and the output is a real vector. Denote by $\boldsymbol{v}^c$ and $\boldsymbol{v}^s$ the output of the RNNs for the canonical and the surface representations, respectively. Then, both vectors are concatenated $[\boldsymbol{v}^c, \boldsymbol{v}^s]$ and are fed to three fully connected layers with shared parameters over time. The output is a series of vectors, one for each time step. These are all concatenated[1] and are fed to a fully-connected layer followed by a softmax layer. The similarity function f is the output of the softmax layer, and it can be interpreted as a probability function. The whole network is trained so as to minimize the negative log likelihood loss function. The network architecture is depicted schematically in Figure 1.

3.2. Ranking loss function

A different approach to learning the similarity function is to score the similarity between a given pronunciation and other pronunciations according to their ranking. In order to do so we use a slightly different training set than the one used to train the binary network. Each example in the new training set is composed of a triplet: a surface pronunciation $\boldsymbol{p}^s \in \mathcal{P}^*$, a positive canonical pronunciation $\boldsymbol{p}^+ \in \mathcal{P}^*$, and a negative canonical pronunciation $\boldsymbol{p}^- \in \mathcal{P}^*$. The positive canonical pronunciation is the canonical pronunciation associated with the surface pronunciation $\boldsymbol{p}^s$, and the negative canonical pronunciation is a canonical pronunciation of a different word. Overall the training set of m examples is denoted $S = \{(\boldsymbol{p}_i^s, \boldsymbol{p}_i^+, \boldsymbol{p}_i^-)\}_{i=1}^m$. As in the previous model, we represent each of the three pronunciations using an RNN. The output of the RNN is fed into two fully-connected layers and is considered to be the *pronunciation embedding*. Pronunciation embedding is a function g that maps a pronunciation $\boldsymbol{p} \in \mathcal{P}^*$ to a fixed size vector, $\boldsymbol{u} \in \mathbb{R}^n$, where $\boldsymbol{u} = g(\boldsymbol{p})$. We measure the closeness between the two embeddings, $\boldsymbol{u} \in \mathbb{R}^n$ and $\boldsymbol{v} \in \mathbb{R}^n$, using the cosine distance

[1] We apply sequence padding when needed.

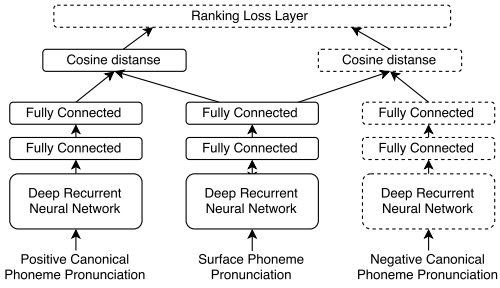


Figure 2: *Network architecture of the ranking model. The three RNNs have share parameters and output an embedding vector.*

score [13]:

$$d_{\cos}(\boldsymbol{u}, \boldsymbol{v}) = \frac{1}{2}\left(1 - \frac{\boldsymbol{u} \cdot \boldsymbol{v}}{\|\boldsymbol{u}\|\|\boldsymbol{v}\|}\right).$$

The cosine distance score between two embeddings is close to 0 if the vectors $\boldsymbol{u}$ and $\boldsymbol{v}$ are close, and close to 1 if they are far apart. Formally, the similarity function between two pronunciations $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ is defined as 1 minus the cosine distance

$$f(\boldsymbol{p}_1, \boldsymbol{p}_2) = 1 - d_{\cos}(g(\boldsymbol{p}_1), g(\boldsymbol{p}_2)). \quad (3)$$

During training we minimize the hinge loss over the cosine distance so that the score of the related canonical-surface pronunciations is higher than the score of the unrelated canonical-surface pronunciation by a margin of at least γ, where $\gamma \in \mathbb{R}_+$ is a positive scalar. Formally, the loss function is defined as [14, 13]:

$$\ell(\boldsymbol{p}^s, \boldsymbol{p}^+, \boldsymbol{p}^-) = \max\{0, \gamma - f(\boldsymbol{p}^s, \boldsymbol{p}^+) + f(\boldsymbol{p}^s, \boldsymbol{p}^-)\}, \quad (4)$$

where f is the similarity function defined in Eq. (3), and γ is the margin parameter. The network architecture is depicted schematically in Figure 2.

This architecture has three advantages over the binary architecture: (i) the ranking optimization criterion is closer to the notion of similarity function: related pronunciations get higher score than unrelated ones; (ii) as a by-product we introduce the *pronunciation embedding*, which maps a sequence of phones to a fixed size vector; and (iii) we use many negative surface pronunciations for each positive surface pronunciation and increase our training data. We found that it has a great impact on the model's performance, and we analyze it in Section 5.

4. Experiments

We evaluated the proposed architectures on the lexical access task, where we would like to predict the word in the dictionary that is associated with a given surface pronunciation. All experiments are conducted on a subset of Switchboard conversational speech corpus that has been labeled at a fine phonetic level [1]; these phonetic transcriptions are the input to our similarity models. The data subsets, phone set $\mathcal{P}$, and dictionary $\mathcal{V}$ are the same as those previously used in [9, 4, 5]. The dictionary contains 5,117 words, consisting of the most frequent words in Switchboard. The base-form uses a similar, slightly smaller phone set (lacking, e.g., nasalization). We used the same partition of the corpus as in [9, 4, 5] into 2,942 words in the training set, 165 words in the development set, and 236 words in the test set.

Results are presented in terms of the word error rate when the top k predictions are considered. This is denoted by WER@k. Table 1 summarizes the results for all our architectures. For each architecture we tested three types of RNNs: LSTM with one layer, LSTM with two layers (2-LSTM), and bidirectional LSTM with two layers (BI-2-LSTM). We optimize all our models using Adagrad [15] with learning rate value of 0.01. We use ReLU [16] as an activation function after each fully connected layer. For the ranking models we use a margin value of $\gamma = 0.3$. All hyper-parameters were tuned on a validation set.

Table 1: *WER for the binary-loss model and the ranking-loss model on the lexical access problem.*

	Models	Test	
		WER@1	**WER@2**
	dictionary lookup [2]	59.3%	-
	dictionary + Levenshtein dist.	41.8%	-
	Jyothi et al., 2011 [4]	29.1%	-
	Hao et al., 2012 [5]	15.2%	-
Binary	LSTM	20.8%	18.2%
Binary	2-LSTM	24.6%	22.5%
Binary	BI-LSTM	21.6%	19.5%
Rank	LSTM	25.9%	16.5%
Rank	2-LSTM	22.9%	14.8%
Rank	BI-LSTM	23.3%	15.3%

For comparison we added to Table 1 the word error rate of other algorithms for lexical access: a dictionary lookup with and without Levenshtein distance [9], a dynamic Bayesian network (Jyothi *et al.*, 2011 [4]), and discriminative structured prediction model (Hao *et al.*, 2012 [5]). It can be seen from the table that both of our models outperform the dictionary lookup approaches and the model which is based on dynamic Bayesian networks [4]. However the discriminative structured prediction model [5] performs much better than any of our models. The discriminative model was trained specifically for the lexical access task with a unique set of feature maps, but it cannot be used as a similarity score between two pronunciations.

The performance of the binary loss model and the ranking model are almost the same, with the binary loss performing slightly better than the rank loss. The reason is likely that it was trained to maximize the probability that two pronunciations are related and not to match a similarity score.

5. Analysis

In this section we analyze the performance of the ranking model. We investigate the effect of the number of negative examples, the effect of the embedding size and present some of the model's outputs and the way it errs.

5.1. The effect of the number of negative samples

Recall that the ranking loss model was trained on triplet made of a canonical pronunciation of a word, a surface pronunciation of the word (positive sample), and a surface pronunciation which is not associated with the word (negative sample). In our experiments the negative surface pronunciation was a surface pronunciation of a random word.

Deep neural networks require a lot of training data in order to converge to a good local minimum. We expanded the training set by using many different negative samples for each positive samples. In order to examine if this approach leads to a better performance, we trained the network several times with

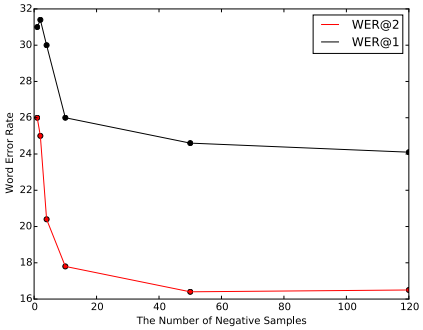


Figure 3: *WER@1 and WER@2 of the test set as a function of the number of samples per example.*

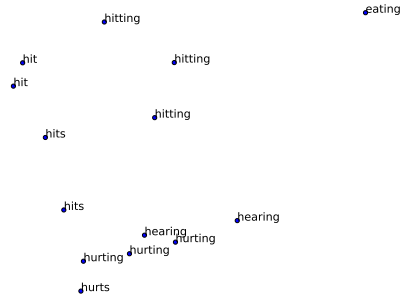


Figure 4: *Two-dimensional t-SNE projection of the representation vectors (zoom to a specific area.)*

a different number of negative samples per positive sample and evaluated the performance on the test set.

Figure 3 shows WER@1 and WER@2 of the lexical access task, as a function of the number of negative samples per positive one. Notice that when adding more examples the error rate keeps decreasing until the limit of 50 negative samples, from this point the error rate stays roughly the same.

5.2. The effect of the embedding size

Next, we investigated the effect of the pronunciation embedding size n, i.e., the size of the output of the last layer after the RNN. We tried different embeddings sizes, and evaluated their performance on the validation and test sets. Table 2 summarizes the results. From the table we see that embedding of size 40 is too small, but the performance of embeddings of size 80 and above are all good. We found the embedding of size 120 to yield the best performance.

Table 2: *Performance (WER) for different embedding sizes.*

Dim.	Test		Validation	
	WER@1	WER@2	WER@1	WER@2
150	24.2%	16.9%	18.8%	12.7%
120	25.9%	16.5%	18.2%	11.5%
80	25.0%	17.4%	21.2%	14.6%
40	27.1%	18.7%	24.9%	15.8%

5.3. Visualization

Lastly, we performed a few visualizations in order to get a sense of what the model learned. In Figure 4 we visualized a subset from the embedding space of the canonical pronunciations in the dictionary, using t-SNE [17] for dimensionality reduction. The words which have a similar pronunciation appear to be close in the embeddings space.

In Table 3 we present the 4 most similar words according to the learned similarity function for the words *sense, write, die, male*, and *their*. Again, we can see that the most similar words in the embedding space are the words which contain a similar phone sequence.

Table 3: *Four most similar words from the dictionary computed in the embedding space.*

word	neighborhood
sense	cents, since, sent, sentence
write	right, ride, righty, writing
die	diet, died, dying, idea
male	mail, meal, may, makes
their	there, they're, there'd, there're

In Table 4 we illustrate the predictions of the model for the lexical access task for hard cases of surface pronunciations. The table shows the first three predictions of the model, ordered (left to right) from most similar to least similar. The table is separated into two panels: the upper panel shows the correct predictions made by the model and the lower panel shows incorrect predictions. It can be seen that both the correct and the incorrect prediction are hard to classify, even for a human. In the lower panel, it seems that there might be an error in the transcription (e.g., $[s, tcl, t, ao, r]$ is more similar to the predicted word *store* rather than to labeled word *start*), or that we have reached the limit of the possible discrimination in this task (e.g., $[w_n, ah_n, n]$ can equally be predicted as either *want* or *won*.

Table 4: *Prediction of the model for hard cases of surface pronunciations.*

surface pronunciation	desired word	predicted words
$[bcl, b, ao]$	bought	bought, bob, ball
$[m, ey1, ey2, dcl, jh, er]$	major	major, mayor, made
$[f, ay1, ay2, n, ih_n, ng]$	finding	finding, fighting, flying
$[pcl, p, r, aa, er]$	proper	proper, prior, property
$[n, pcl, p, eh, er, z]$	peppers	peppers, persons, present
$[s, tcl, t, ao, r]$	start	store, star, sent
$[eh, r, uw, ay1, ay2]$	everybody	iraq, dry, era
$[w_n, ah_n, n]$	want	won, one, want
$[pcl, p, uw, el]$	people	pool, pull, people

6. Conclusion

We presented two very different architectures to learn similarity between two phone streams. The first architecture learns the alignment between phone streams that represent close pronunciations. The second architecture is designed to learn a mapping of a phone stream to a vector space, such that close pronunciations will have close representation in the output vector space.

Future work will explore similarity between other sub-word units. Specifically we would like to propose a similarity function between articulatory features, and analyze it in the light of articulatory phonology [18].

7. Acknowledgment

We would like to thank Karen Livescu, Preethi Jyothi and the anonymous reviewers for their helpful comments.

8. References

[1] S. Greenberg, J. Hollenback, and D. Ellis, "Insights into spoken language gleaned from phonetic transcription of the Switchboard corpus," in *Proceeding of the 4th International Conference on Spoken Language Processing (ICSLP)*, 1996.

[2] K. Livescu, "Feature-based pronunciation modeling for automatic speech recognition," Ph.D. dissertation, Massachusetts Institute of Technology, 2005.

[3] L. Fissore, P. Laface, G. Micca, and R. Pieraccini, "Lexical access to large vocabularies for speech recognition," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 37, no. 8, pp. 1197–1213, 1989.

[4] P. Jyothi, K. Livescu, and E. Fosler-Lussier, "Lexical access experiments with context-dependent articulatory feature-based models," in *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2011.

[5] H. Tang, J. Keshet, and K. Livescu, "Discriminative pronunciation modeling: A large-margin, feature-rich approach," in *The 50th Annual Meeting of the Association of Computational Linguistics (ACL)*, 2012.

[6] P. Jyothi and K. Livescu, "Revisiting word neighborhoods for speech recognition," in *ACL MORPHFSM Workshop*, 2014.

[7] D. Jurafsky, W. Ward, Z. Jianping, K. Herold, Y. Xiuyang, and Z. Sen, "What kind of pronunciation variation is hard for triphones to model?" in *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2001.

[8] M. Saraçlar and S. Khudanpur, "Pronunciation change in conversational speech and its implications for automatic speech recognition," *Computer Speech and Language*, vol. 18, no. 4, 2004.

[9] K. Livescu and J. R. Glass, "Feature-based pronunciation modeling with trainable asynchrony probabilities." in *Proceeding of the 8th International Conference on Spoken Language Processing (ICSLP)*, 2004.

[10] J. L. Elman, "Finding structure in time," *Cognitive science*, vol. 14, no. 2, pp. 179–211, 1990.

[11] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, 2014, pp. 3104–3112.

[12] K. Yao and G. Zweig, "Sequence-to-sequence neural net models for grapheme-to-phoneme conversion," *arXiv preprint arXiv:1506.00196*, 2015.

[13] H. Kamper, W. Wang, and K. Livescu, "Deep convolutional acoustic word embeddings using word-pair side information," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016, pp. 4950–4954.

[14] J. Keshet, D. Grangier, and S. Bengio, "Discriminative keyword spotting," *Speech Communication*, vol. 51, no. 4, pp. 317–329, 2009.

[15] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research*, vol. 12, no. Jul, pp. 2121–2159, 2011.

[16] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th international conference on machine learning (ICML)*, 2010, pp. 807–814.

[17] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," *Journal of Machine Learning Research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[18] C. P. Browman and L. Goldstein, "Articulatory phonology: An overview," *Phonetica*, vol. 49, no. 3-4, pp. 155–180, 1992.

The Influence of Lexical Selection Disruptions on Articulation

American Psychological Association

Journal of Experimental Psychology:
Learning, Memory, and Cognition

© 2018 American Psychological Association
0278-7393/18/$12.00

http://dx.doi.org/10.1037/xlm0000633

The Influence of Lexical Selection Disruptions on Articulation

Matthew Goldrick
Northwestern University

Rhonda McClain
The Pennsylvania State University

Emily Cibelli
Northwestern University

Yossi Adi
Bar Ilan University

Erin Gustafson
Northwestern University

Cornelia Moers
Max Planck Institute for Psycholinguistics

Joseph Keshet
Bar Ilan University

Interactive models of language production predict that it should be possible to observe long-distance interactions; effects that arise at one level of processing influence multiple subsequent stages of representation and processing. We examine the hypothesis that disruptions arising in nonform-based levels of planning—specifically, lexical selection—should modulate articulatory processing. A novel automatic phonetic analysis method was used to examine productions in a paradigm yielding both general disruptions to formulation processes and, more specifically, overt errors during lexical selection. This analysis method allowed us to examine articulatory disruptions at multiple levels of analysis, from whole words to individual segments. Baseline performance by young adults was contrasted with young speakers' performance under time pressure (which previous work has argued increases interaction between planning and articulation) and performance by older adults (who may have difficulties inhibiting nontarget representations, leading to heightened interactive effects). The results revealed the presence of interactive effects. Our new analysis techniques revealed these effects were strongest in initial portions of responses, suggesting that speech is initiated as soon as the first segment has been planned. Interactive effects did not increase under response pressure, suggesting interaction between planning and articulation is relatively fixed. Unexpectedly, lexical selection disruptions appeared to yield some degree of facilitation in articulatory processing (possibly reflecting semantic facilitation of target retrieval) and older adults showed weaker, not stronger interactive effects (possibly reflecting weakened connections between lexical and form-level representations).

Keywords: speech production, interaction, articulation, automatic acoustic analysis

This document is copyrighted by the American Psychological Association or one of its allied publishers.
This article is intended solely for the personal use of the individual user and is not to be disseminated broadly.

To produce a single word, a speaker must map an intended message to a lexical representation and select detailed representations regarding the word's sound structure (e.g., Garrett, 1975; Schriefers, Meyer, & Levelt, 1990; van Turennout, Hagoort, & Brown, 1997; see Levelt, Roelofs, & Meyer, 1999, for an overview). This is typically assumed to rely on several distinct processing stages collectively referred to as *formulation*. Speech production begins with the selection of a concept to verbalize the message, and then the speaker activates the concept's relevant semantic features. During lexical selection, these meaning-based representations are used to select an appropriate lexical representation. Phonological encoding associates this lexical representation with a form-based planning representation. Phonetic encoding or articulatory processing then implements this plan as a set of movements of the articulators.

Most contemporary perspectives on speech planning agree that speaking involves interaction among stages of formulation. At each stage of processing, multiple representations are coactivated, and subsequently influence the following stage of processing. For example, many studies have shown that the process of lexical selection results in the coactivation of multiple semantically related words (e.g., Peterson & Savoy, 1998). These semantic cohort members influence subsequent phonological encoding. Other work

Matthew Goldrick, Department of Linguistics, Northwestern University; Rhonda McClain, Department of Spanish, Italian, and Portuguese, The Pennsylvania State University; Emily Cibelli, Department of Linguistics, Northwestern University; Yossi Adi, Department of Computer Science, Bar Ilan University; Erin Gustafson, Department of Linguistics, Northwestern University; Cornelia Moers, Max Planck Institute for Psycholinguistics; Joseph Keshet, Department of Computer Science, Bar Ilan University.

This work was supported by National Institute of Health (NIH NICHD) Grant 1R21HD077140 to Matthew Goldrick and by a scholarship for Cornelia Moers from the Max Planck International Research Network on Aging (MaxNetAging). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NIH or MaxNetAging.

Correspondence concerning this article should be addressed to Matthew Goldrick, Department of Linguistics, Northwestern University, 2016 Sheridan Road, Evanston, IL 60208. E-mail: matt-goldrick@northwestern.edu

has shown that disruptions originating in phonological planning extend to phonetic processing, altering the phonetic properties of speech (e.g., Goldrick & Blumstein, 2006; McMillan, Corley, & Lickley, 2009; Pouplier, 2007). However, evidence for long-distance interactions—effects of disruptions to conceptual processes and lexical selection that influence articulatory processing—have been inconsistent. The current work provides new evidence on such interactive effects, examining the influence of semantic competitors on articulation during picture naming.

Evidence for Interactions Between Adjacent Levels of Formulation

Abundant evidence supports the idea that lexical selection processes interact with phonological planning. Specifically, semantically related competitors activated during lexical selection activate their corresponding phonological representations (for reviews, see Goldrick, 2006; Melinger, Branigan, & Pickering, 2014). For example, in the picture-word interference paradigm (Schriefers et al., 1990), picture naming is disrupted by the presentation of an auditory or visual distractor word. Distractors that are phonological relatives of a semantic competitor show evidence of priming, suggesting their phonological representations have been activated during target processing (e.g., during processing of target *couch*, the semantic competitor *sofa* primes *soda*; Cutting & Ferreira, 1999; Peterson & Savoy, 1998; Taylor & Burke, 2002). Further support for the semantically driven activation of phonological representations comes from studies of speech errors showing that mixed errors (sharing both semantic and phonological structure with the target) occur at a higher rate than predicted by the rate of pure semantic or phonological errors (Dell & Reich, 1981; Rapp & Goldrick, 2000).

Interactive effects are also found between phonological planning and articulatory processing. Speech errors reflect a blend of articulatory/acoustic properties of the target and error outcomes (e.g., when producing target *big* as "pig," the production of /p/ reflects a blend of the intended /b/ and error outcome /p/; Frisch & Wright, 2002; Goldrick, Baker, Murphy, & Baese-Berk, 2011; Goldrick & Blumstein, 2006; Goldrick, Keshet, Gustafson, Heller, & Needle, 2016; Goldstein, Pouplier, Chen, Saltzman, & Byrd, 2007; McMillan & Corley, 2010; McMillan et al., 2009; Pouplier, 2007, 2008). Such effects can be attributed, in part, to the partial activation of the target representation during phonological planning (see Goldrick et al., 2016, for review and discussion). Similar effects are found when phonological competitors are primed not within production processes but by comprehension processes. Yuen, Davis, Brysbaert, and Rastle (2010) examined articulatory processing during reading aloud of a target while participants listened to a matching (i.e., identical) syllable or a phonologically related (rhyming) competitor. Articulatory processing of the target sound was distorted when a competitor was presented, such that articulation reflected a blend of the target and the initial sound of the spoken competitor.

A key study with respect to the work reported here is Drake and Corley (2015), who examined picture naming when phonologically related competitors were primed by sentence preambles. Participants heard sentences like "Jimmy used a washer to fix the drip from the old leaky . . ." (priming a mismatching word, *tap*) and "On his head he wore the school . . ." (priming the target word, *cap*). In both cases, participants then named a picture of a cap. Articulations in these two conditions were compared to a baseline: picture naming with no sentence preamble. Productions following unrelated primes showed greater difference from baseline than those following target primes, suggesting that the primed competitor disrupted articulatory processing of the target.

Evidence for Long-Distance Interactions: The Influence of Disruptions to Lexical Selection on Phonetic Processing

Given the evidence that lexical selection interacts with phonological processing, which in turn interacts with articulatory processing, one would expect long-distance interactions between lexical selection and articulation. In this work, we focus on how disruptions to lexical selection influence phonetic processing. We review evidence from paradigms using conditions that slow reaction times (RTs) and/or increase errors relative to baselines, deferring discussion of facilitatory effects until the following section.

Kello, Plaut, and MacWhinney (2000) adapted the Stroop task to examine how lexical selection disruptions modulate phonetic processing. In the typical color-word version of the Stroop task, written words specifying color concepts are presented. The word is printed in color font and only the color of the font is to be named aloud. In some conditions, the color is congruent with the conceptual representation of the written word (e.g., say *green* to the word "GREEN" presented in a green colored font) or neutral (e.g., say *green* to XXXX presented in a green colored font). The Stroop interference effect refers to the fact that relative to the two conditions above responses are initiated more slowly when the color of the font is incongruent with the meaning of the written word (e.g., say *green* to the word "RED" presented in a green colored font). Kello et al. used this paradigm to examine long distance interactions from lexical selection to articulation. Stroop interference lengthened RTs as well as spoken word durations—but only when speakers were pressured to respond quickly. To account for their findings, Kello et al. proposed a dynamic interaction hypothesis. This claims that interactive effects extending from lexical selection to phonetic processing will be strongest under processing circumstances that allow insufficient time for speakers to resolve disruptions during lexical selection. Time pressure increases temporal overlap between processes, increasing interaction.

However, other work has found no evidence supporting such effects. Damian (2003) failed to replicate Kello and colleagues' (2000) Stroop task results; even under time pressure, there was no increase in duration of words subject to Stroop interference. Furthermore, he failed to show articulatory effects in two additional tasks. Using the picture-word interference task, participants named pictures aloud in the presence of semantically related auditory distractors (Schriefers et al., 1990). Although the presence of distractors increased RTs, Damian found no effect of distractors on spoken durations, even when speakers were pressured to respond quickly. Above and beyond the empirical uncertainty regarding effects in durations, there is debate over whether these two paradigms actually tap lexical selection processes (Roelofs, 2014) or arise in processes external to lexical selection (e.g., Dhooge, Baene, & Hartsuiker, 2013; Finkbeiner & Caramazza, 2006a, 2006b). That is, it is unclear whether these findings speak to the question of interaction between lexical selection and articulation or

This document is copyrighted by the American Psychological Association or one of its allied publishers.
This article is intended solely for the personal use of the individual user and is not to be disseminated broadly.

This document is copyrighted by the American Psychological Association or one of its allied publishers.
This article is intended solely for the personal use of the individual user and is not to be disseminated broadly.

rather to articulatory interactions with other aspects of formulation and more general cognitive processing.

The semantic blocking paradigm (Belke, Meyer, & Damian, 2005; see also Damian, Vigliocco, & Levelt, 2001) induces inhibitory effects that are more widely assumed to arise within lexical selection (Oppenheim, Dell, & Schwartz, 2010). (Note the paradigm can also give rise to facilitatory effects; Belke, 2017.) Pictures are presented in homogeneous blocks (including only pictures from the same semantic category) or heterogeneous blocks (including pictures from a mixture of categories). The so-called blocking effect reflects increased response latencies in homogeneous blocks relative to heterogeneous blocks. Damian (2003) found a very robust blocking effect observed in measures of response latency, but no significant effects in response duration, even under time pressure. However, using a much larger sample of participants ($n = 96$ compared to 24 in each condition of Damian's 2003 study), Fink, Oppenheim, and Goldrick (2018) found effects of semantic blocking on word durations (with longer durations in homogeneous vs. heterogeneous blocks). Interestingly, this effect was detected only when individual differences in susceptibility to the blocking effect were taken into account: individuals that showed large blocking effects in their RTs exhibited semantic effects in their word durations; those participants with small blocking effects in RTs showed no duration effects. Fink et al. (2018) also found evidence in favor of long-distance interactions between lexical selection and articulation using another paradigm, continuous picture naming, which also induces semantic interference effects during lexical selection (Howard, Nickels, Coltheart, & Cole-Virtue, 2006). In this paradigm, participants name pictures from intermixed sets of semantic categories. Within each category, response latencies increase with each successive member of the category. Consistent with long-distance interactions, Fink et al. found that increases were also found in word durations.

Mixed results have also been found in the manual articulatory domain in studies of typing. Logan and Zbrodoff (1998) found that Stroop interference effects impacted typing latencies, but not durations. Parallel to Kello et al. (2000) and Damian (2003); Damian and Freeman (2008) examined Stroop effects under response pressure; similar to Damian (2003), Damian and Freeman found no effects on typewritten response durations either with or without time pressure. However, in a regression-based analysis using a large sample of participants ($n = 86$) and a diverse array of pictures ($n = 260$), Scaltritti, Arfé, Torrance, and Peressotti (2016) found that variables influencing lexical selection (word frequency and name agreement) influence both response latencies and typing durations in written picture naming.

Note that all of the studies reviewed above (in both spoken and manual modalities) have focused on duration of the entire response. However, this might obscure interactive effects if they are present only in certain portions of the word. For example, suppose response initiation occurs as soon as the first element (e.g., initial segment, letter) is planned, but planning continues while it is being articulated (i.e., response planning for different components of a word occurs in parallel). During articulation of the initial element, continued planning of subsequent elements in the word might allow such element to overcome the effects of any delays or disruptions. Effects may therefore be limited to initial portions of articulation and dissipate at later positions (Kawamoto, Kello, Higareda, & Vu, 1999). There is some evidence from typing studies consistent with this possibility. Scaltritti, Pinet, Longcamp, and Alario (2017) found that while semantic priming did not significantly influence the whole-word duration of typewritten responses, there was an influence on duration of the initial interkeystroke-intervals. However, this effect was not reliable in some of the subset analyses they performed, and the study of Stroop effects by Damian and Freeman (2008) found no effects in initial position or whole word durations. Furthermore, using electroencephalographic measures, Scaltritti et al. failed to find evidence that semantic priming interacted with motor response preparation.

Another important limitation of the work reviewed above is that focuses on young, monolingual participants, whose formulation abilities are likely operating at peak efficiency. In particular, young adults may possess strong selection processes, which serve to enhance the activation of a single lexical representation relative to its coactivated competitors (via boosting target activation and/or inhibiting nontarget activation). The reduced strength of competitors relative to targets will significantly reduce the strength of interactive effects (see Dell & O'Seaghdha, 1992 and Rapp & Goldrick, 2000, for discussion and simulation data). Consistent with this claim, Jescheniak, Hahne, Hoffmann, and Wagner (2006) found that children (~7 years of age) were more susceptible to interactive effects than young adults (~24 years old). Interactive processing accounts predict that cascading activation from the semantic representation of a target (e.g., *cat*) will activation category coordinates (*dog*) which will in turn activate phonologically related words (*doll*). Critically, the strength of such effects will depend on the relative activation of target versus nontarget representations. Jescheniak et al. found that children showed significant effects, whereas young adults did not; this is consistent with the claim that children have weaker selection processes than young adults. The focus of the literature on long-distance interactions on young adults may therefore have reduced our ability to detect effects.

Facilitation of Formulation and Its Impact on Phonetic Processing

Although the above discussion has focused on disruptions, other work has focused on how phonetic processing may be facilitated by ease of formulation. A large number of studies have shown that words that are more predictable with respect to the linguistic and/or discourse context[1] in which they appear are retrieved more quickly and accurately (suggesting facilitated processing) and produced with reduced phonetic forms (e.g., shorter acoustic duration, de-accented variants, centralization of vowels, etc.; see Arnold, 2016, for a recent review). Some accounts have claimed that such effects are due, in part, to long-distance interactive mechanisms; context-driven facilitation of the retrieval of representations at the lexical or conceptual level modulates the activation of word form and phonetic representations, producing reduction (e.g., Balota, Boland, & Shields, 1989). However, other accounts have empha-

[1] In contrast to the context-specific effects of disruption examined here, other work has examined how acoustic properties are related to context-independent features of words (e.g., word frequency: Gahl, 2008; phonological neighborhood density: Gahl & Strand, 2016; informativity: Seyfarth, 2014). Note that these effects may be due to mechanisms that overlap with those driving context-specific predictability effects (e.g., Bell, Brenier, Gregory, Girand, & Jurafsky, 2009).

This document is copyrighted by the American Psychological Association or one of its allied publishers.
This article is intended solely for the personal use of the individual user and is not to be disseminated broadly.

sized more local interactions, presenting evidence that facilitation of phonological retrieval, over and above facilitation to lexical or conceptual processing, is the key mechanism within the production system that results in phonetic reduction. Jacobs, Yiu, Watson, and Dell (2015) examined reductions in the time to initiate production of a target word as well the word duration when the target was repeated. They contrasted conditions where the first utterance of a word was fully articulated versus read silently. Critically, although target word initiation was facilitated to the same degree in both conditions, reduction in target duration only occurred when the word was read aloud. Jacobs et al. concluded that repetition of any kind was sufficient to facilitate lexical and conceptual processing (reducing the time required to initiate production). In contrast, facilitation of phonetic processing (i.e., reduction) requires facilitation specifically of word form processing.

These findings suggest that independent of the presence or absence of effects of disruption, there are widespread situations where phonetic processing is facilitated by the context in which a word is produced. However, in situations where processing at multiple levels of representation are enhanced, it is unclear whether phonetic effects arise due to long-distance or more local interactive mechanisms.

The Current Study

In the current study, we used a sentence completion paradigm (Ferreira & Griffin, 2003) to examine long-distance interactive effects. Ferreira and Griffin used visually presented sentence preambles to prime competitors during picture naming. Participants in their study read sentences like "The woman went to the convent to become a . . ." (priming *nun*) and then attempted to name a picture of a priest. These primes disrupted processing, resulting in the overt production of semantic errors. Semantically related primed words were produced significantly more often than control trials where the sentence primes a semantically unrelated word (e.g., "He lit the candle with just one . . ." priming *match*; here, participants had less difficulty naming *priest*). Interestingly, errors were also produced at a rate greater than control trials when the sentence primes a homophone of a semantic competitor *("I thought that there would be some cookies left, but there were . . .*" priming *none*), but there was no increase in errors when the sentence primed a purely phonologically related word (*present*; Ferreira & Griffin, 2003; Li & Slevc, 2017; Severens, Ratinckx, Ferreira, & Hartsuiker, 2008). The fact that homophones also induced substitutions suggests that the processing disruption that leads to semantic errors in this task arose specifically at a postsemantic, lexical level of processing (where homophones share representations) but prior to phonological processes manipulating sublexical units of form (accounting for the absence of phonological errors). Thus, in contrast to paradigms such as Stroop and picture-word interference, there is clear evidence that this paradigm can induce disruption specifically within lexical selection processes.

As discussed above, using a similar sentence prime paradigm, Drake and Corley (2015) found articulatory interference during picture naming after priming by a sentence stem predicting a semantically unrelated word (n.b. interference was relative to an unprimed baseline). This finding suggests that sentence primes serve to activate representations inconsistent with the target at semantic, lexical, and form-based levels; the activation of these competing representations produces articulatory disruptions. Note that while matching primes could also have facilitated processing, Drake and Corley (2015) found no difference between the articulation of matching primes and an unprimed baseline. This suggests that in this type of paradigm, interference effects dominate processing.

Here, we examine the impact of semantically related primes. Although such primes activate semantic representations that overlap with the target, the pattern of errors reviewed above suggest that they produce enhanced conflict (relative to unrelated primes) specifically at the lexical level. Comparing articulation of targets following semantically related versus unrelated primes will therefore provide an index of long-distance interactive effects: how enhanced disruption of lexical selection impacts articulatory processing.

Our design also took into account three other factors that may have contributed to the mixed results observed in previous work. In Experiment 1, we examined healthy, young adult monolinguals' patterns of response time and articulation in this paradigm. In this (and subsequent) experiments, we extended previous work by examining articulatory properties of the whole word as well as properties specific to the initial segments of the word (where, as noted above, interactive effects might be strongest). In Experiment 2, we examined whether interactive effects were modulated by pressure to respond (which, as noted above, has been suggested to increase temporal overlap and interaction between processes).

Finally, in Experiment 3, we examined these effects in healthy, monolingual older adults (parallel to Experiment 1, under no explicit time pressure). As noted above, there is evidence that across the life span there are changes in the strength of lexical selection, such that young adults show weaker interactive effects than children (Jescheniak et al., 2006). There is some evidence that formulation processes undergo declines as a consequence of normal aging, as indexed by a higher rate of tip-of-the-tongue (TOTs) retrieval failures (see Gollan & Brown, 2006, for a review) and higher rates of speech errors (Gollan & Goldrick, 2018; but see Ramscar, Hendrix, Shaoul, Milin, & Baayen, 2014). One account of effects such as these is the inhibitory deficit hypothesis, which claims that aging leads to difficulty in suppressing inappropriate responses (e.g., Zacks & Hasher, 1994). If these domain-general inhibitory mechanisms are used during lexical selection, their age-related decline would allow for greater activation of nontarget lexical representations in older versus younger adults. Cascade from these representations would be predicted to strengthen interactive effects in the older adults. However, it should be noted that there are other accounts of aging deficits that predict decreased interactive effects. Specifically, the transmission deficit hypothesis (e.g., Burke, MacKay, Worthley, & Wade, 1991) proposes that language production difficulties in older adults arise due to reduced flow of activation between lexical and phonological representations. This account therefore predicts that there should be less activation of nontarget representations at the phonological level, and therefore less disruption of articulatory processing.

This design did not eliminate all potential issues. To facilitate group comparisons, we recruited the same number of participants across groups. We based the sample size for all groups ($N = 18$) on that used in previous studies with this paradigm (Severens et al., 2008). This sample size was achievable given practical limitations

on our recruitment of older adults. It's possible this does not provide sufficient power for detecting long-distance interactive effects; replicating our findings with larger groups is an important area for future work.

To summarize, our study includes three experiments examining how disruptions to lexical selection modulate articulatory processing. Experiment 1 examines effects in younger adults, examining effects on whole word durations as well as specific properties of initial segments. Experiment 2 aimed to increase interactive effects by increasing response pressure. Experiment 3 aimed to increase effects by testing older adults; difficulties these individuals may have in inhibiting nontarget representations would increase interactive effects.

Acoustic Analysis Methods

Key to our study is the measurement of phonetic properties of productions. Using a combination of algorithms (all available at https://github.com/MLSpeech), we limited manual processing in the analysis pipeline. First, participants' speech was recorded on one channel of a stereo recording and the second channel simply recorded when pictures were presented for naming. These clicks were used to automatically segment the original audio stream into separate files containing the signal from each individual trial. After segmentation of each trial's data, several algorithms were combined to extract the phonetic variables of interest. We first used two algorithms to estimate several key time points in the signal:

- **Word onset and offset**. We developed a novel algorithm (DeepWDM, short for deep word duration measurement, described below) that, given a signal consisting of speech preceded and followed by silence (minimally noisy nonspeech signals), would automatically determine the onset and offset of the word.
- **Vowel onset and offset**. Given speech consisting of a vowel with one or more flanking consonants on each side, the AutoVowelDuration algorithm (Adi et al., 2016) outputs the onset and offset of the vowel. In monosyllabic words, this can operate without any additional processing. In disyllabic words, the AutoAligner forced aligner (Keshet, Shalev-Shwartz, Singer, & Chazan, 2007; McAllester, Hazan, & Keshet, 2010) was used to determine the location of the initial syllable (always the location of the stressed syllable in this dataset), and then AutoVowelDuration was used to determine the precise location of the vowel onset and offset.

Once these time points had been determined, several duration measures could be extracted: RT (the duration between trial onset and word onset); word duration (time between word onset and offset); duration of initial consonant or consonants (time between word onset and vowel onset, for consonant-initial words only); and vowel duration (time between vowel onset and offset). Examination of whole word, initial consonant, and vowel durations allows us to examine both overall effects of articulatory disruption as well as effects that may specifically target the initial segments of the word. Finally, the DeepFormants algorithm (Dissen & Keshet, 2016) was used to estimate first (F1) and second (F2) formant values within the window identified by AutoVowelDuration. Measuring these spectral qualities gives us another index (beyond duration) of vowel articulation. Disruption of processing was indexed by vowel dispersion (calculated as the euclidean distance from the overall F1/F2 midpoint of the vowel space, within-subject; Löfqvist, Sahlén, & Ibertsson, 2010). Based on previous work, we predict that disruptions to formulation will lead to greater vowel dispersion (i.e., lower distance from the overall midpoint; see Munson, 2007, for discussion).

In the remainder of this section, we describe in detail the structure of the novel DeepWDM algorithm; detailed characterization of the other speech processing algorithms can be found in the publications cited above.

Problem Setting

The input to our algorithm is an acoustic signal containing one dominant speech portion (i.e., the uttered word) which can be surrounded by noisy nonspeech signals. (Such nonspeech noise is a persistent challenge to voice key systems that simply rely on signal intensity to determine speech onset.) The output is the onset and offset times of the speech portion. The acoustic signal can be of an arbitrary length, and its beginning does not need to be synchronized with speech onset.

Let $\boldsymbol{x} = (x_1, \ldots, x_T)$ denote the input acoustic signal, represented as a sequence of feature vectors, where each $x_T \in \mathbb{R}^D$ $(1 \leq t \leq T)$ is a D-dimensional vector. The length of the speech portion, T, is not a fixed value because the acoustic signals and target words can have different durations.

Each acoustic signal is associated with a timing pair, denoted by $\boldsymbol{t} = (\boldsymbol{t}_b, \boldsymbol{t}_e)$ where $\boldsymbol{t}_b$ and $\boldsymbol{t}_e$ are the onset and offset of the speech portion respectively (see Figure 1). Our goal is to predict the onset and offset times of the speech portion as accurately as possible.

Model

One approach to determining the duration of a phonetic property is to predict at each time frame whether the property is present or absent; the predicted duration is then the smoothed, continuous set of frames where the property is likely to be present (Adi, Keshet, Dmitrieva, & Goldrick, 2016; Adi, Keshet, & Goldrick, 2015). In this work, we follow this method with generating predictions using a Recurrent Neural Network (RNN).

Learning model. To predict the voice activity's onset and offset (i.e., speech onset and offset) we trained a RNN (Elman, 1991) as a speech detection system. The input to the network is a sequence of T tuples, where each tuple is composed of the feature vector x_t and a corresponding label y_t, from the set of $\{1, -1\}$, for $1 \leq t \leq T$ as follows:

$$y_t = \begin{cases} -1 & 1 \leq t \leq t_b \\ 1 & t_b \leq t \leq t_e \\ -1 & t_e \leq t \leq T \end{cases}$$

We label every frame that is placed inside the boundaries of the speech portion of the acoustic signal as positive and every frame that is outside of those boundaries as negative.

Our RNN model is composed of two stacked layers of bidirectional long-short term memory (LSTM) units (Hochreiter & Schmidhuber, 1997), which have shown remarkable results in modeling speech sequences (Graves & Jaitly, 2014; Graves, Mohamed, & Hinton, 2013). The inputs to the network were 39 mel-frequency cepstrum coefficients (MFCCs), including delta

This document is copyrighted by the American Psychological Association or one of its allied publishers.
This article is intended solely for the personal use of the individual user and is not to be disseminated broadly.

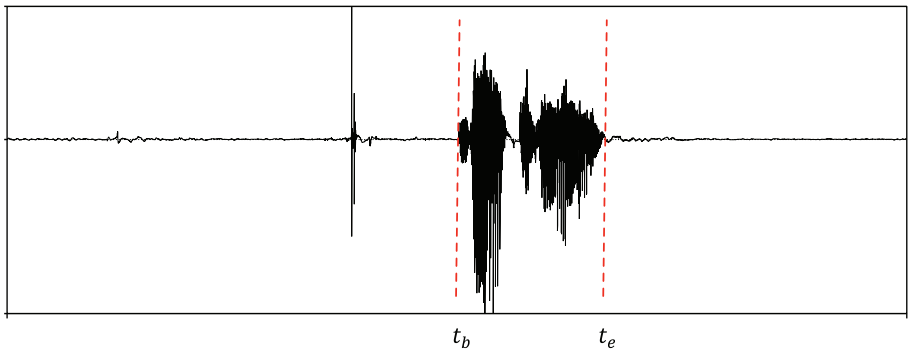


Figure 1. Example acoustic signal with annotations marking onset t_b and offset t_e of speech. Note prior to speech onset there is a high intensity nonspeech signal (lip smack) that the DeepWDM algorithm can learn to ignore. See the online article for the color version of this figure.

and delta–delta, extracted every 10 ms. To avoid overfitting, a dropout layer (Hinton, Srivastava, Krizhevsky, Sutskever, & Salakhutdinov, 2012) is used after each recurrent layer, with rate of 0.5.

Training. The training data consisted of 2,369 hand-annotated productions of single words (drawn from Fink, 2016). Participants named a set of 90 pictures ("carrot," "violin," etc.) in sequence. Ten percent of the data was used as a validation set for tuning model parameters (including hyper-parameters). We optimized the negative log-likelihood loss function using Adagrad (Duchi, Hazan, & Singer, 2011) with a learning rate of 0.01. Training was stopped after 5 epochs with no loss improvement on the validation set.

Inference. The RNN outputs a probability for each class (speech vs. nonspeech) every frame, which can be used to characterize a probability distribution over all possible sequences. To extract the onset and offset times from the RNN outputs we perform three steps. First, we predict the class with the largest probability in every frame. Second, we remove noisy predictions by smoothing the predictions using a window of 10 frames. Finally, because we know that in every sequence there is one major voice activity which we are interested in, we output the timing pair with the longest duration.

Validation. To assess performance of the DeepWDM algorithm, novel data (not used in model training) from Fink (2016) was used to compare manual and algorithmic measurements of word duration on 6641 tokens. The correlation between manual and algorithmic measures was 0.72; the mean absolute deviation was 56 ms (SD = 73 ms). This level of performance is well within that of the vowel duration algorithm, which our previous work has shown faithfully reproduces results from behavioral data (Adi, Keshet, Cibelli, & Goldrick, 2016).

Overview of the Experiments

As discussed above, in this paradigm each trial consists of a visually presented prime sentences followed by picture; participants are asked to orally produce the picture name. Table 1 summarizes our design, which limits nontarget primes to semantically related and semantically/phonologically unrelated items. The key findings are summarized in Table 2.

Experiment 1: Young Adults Naming

Methods

This experiment along with the following two were approved by the Northwestern University Institutional Review Board.

Participants. We recruited 18 participants at Northwestern University using the Linguistics Department subject pool. Participants received course credit. They reported learning no language other than English before age 5 and no history of color blindness or language impairment. Age ranged from 18 to 24 years (M = 19.6, SD = 1.5).

Table 1
Example Illustrating the Design of the Study

Prime condition	Cloze sentence	Primed response	Picture target
Match	The fairy tale princess lived in a majestic . . .	"Castle"	
Semantically related (mismatch)	Every halloween, they turned their home into a haunted . . .	"House"	
Unrelated (mismatch)	The joint connecting the thigh and shin is the . . .	"Knee"	

Note. Images are from the bank of standardized stimuli (BOSS; Brodeur, Guérard, & Bouras, 2014) and are authorised for redistribution according to the creative commons attribution (Share Alike 3.0 License, https://creativecommons.org/licenses/by-Sa/3.0/).

This document is copyrighted by the American Psychological Association or one of its allied publishers.
This article is intended solely for the personal use of the individual user and is not to be disseminated broadly.

Table 2
Key Predictions and Results Across the Three Experiments

Key prediction	Results
Interaction of formulation and articulation	**Confirmed**
Interactive theories of speech production predict that disruptions to formulation processes (lexical selection and/or phonological encoding) disrupt articulation via cascading activation.	*Articulation is disrupted following mismatching vs. matching primes.*
Long-distance interaction	**Not confirmed**
Interactive theories that allow for long-distance interactions predict that disruptions to lexical selection will disrupt phonological encoding and, in turn, articulation processes.	*Semantically related primes, which yield overt speech errors during lexical selection, show no more disruption than unrelated primes, which lead to fewer errors.*
Dynamic interaction	**Not confirmed**
Theories incorporating dynamic interaction predict that insufficient time for selection process will increase the overlap between formulation and articulation processes; this greater overlap will increase interactive effects relative to conditions which allow for greater processing time.	*Articulatory effects do not interact with trial-level reaction time, nor do they increase with response pressure.*
Inhibitory deficit	**Not confirmed**
The inhibitory deficit hypothesis for cognitive aging predicts that older adults will be less able to inhibit non-target representations; in the context of interactive theories of production, older adults are predicted to show stronger interactive effects than younger adults.	*Articulatory effects in older adults are of comparable magnitude to younger adults.*
Initial segment speech initiation	**Confirmed**
If speech is initiated prior to completion of planning for remainder of the word, interactive theories predict that interactive effects will be stronger in the initial vs. later part of the word (where additional planning time mitigates the effects of disruptions).	*Proportional effects of disruption are numerically larger on duration of initial consonants vs. duration of following vowels.*

This document is copyrighted by the American Psychological Association or one of its allied publishers.
This article is intended solely for the personal use of the individual user and is not to be disseminated broadly.

Materials. Details of norming procedures for picture and sentence stimuli can be found in Appendix A. One hundred eighty colored photographs were selected from a larger pool of photographs retrieved from the BOSS database (Brodeur, Dionne-Dostie, Montreuil, & Lepage, 2010; Brodeur, Guérard, & Bouras, 2014) and Google images. Each picture had a name agreement of at least 75%. The selected pictures had an average word frequency of 32.2 words per million (from the SUBTLEX-US corpus, Brysbaert & New, 2009, extracted from the CLEARPOND database; Marian, Bartolotti, Chabal, & Shook, 2012). All pictures were 300 × 300 pixels. In addition, 180 unique sentence fragments were created for the experiment. For each picture, we constructed a cloze probability sentence. Sentences were normed with both younger adults and older adults (see Appendix B for details); only sentences with at least 45% cloze agreement were selected. For younger adults, the average cloze probability for all sentences was 90.7% (SD = 11.9%). Sentences ranged in length from 5 to 15 words (M = 8.7, SD = 2.0). Sentences and target picture names can be found in Appendix B. A small percentage of sentence contexts (1.7%) had an indefinite article that mismatched with the onset of the picture completion. Visual inspection of response times for these trials indicated that they were not different from trials where no mismatch was present and excluding these trials from analysis did not appear to qualitatively impact the results.

Design and procedure. Participants were presented high probability cloze sentences, one word at a time. Each sentence was followed by a picture that was to be named aloud. As shown in Table 1, pictures were paired with one of three sentences: match (where the sentence primed the picture name), or one of two mismatching sentences—competitor (where the sentence primed a semantically related word) and unrelated (where the sentence primed a phonologically and semantically unrelated word).

Each participant named 60 pictures three times, for a total of 180 trials. Across the blocks of picture naming, each appearance of a given picture was paired with a different prime sentence (reflecting the three conditions). Note that sentences were not repeated across blocks so as to minimize experiment-specific expectancy effects. Within a given block, the number of trials was evenly divided between the three conditions. The order of conditions for each picture was counterbalanced across lists.

Participants were tested individually in a sound-proof room. They first provided informed consent and completed a background questionnaire. Speech during the experiment was recorded using a head-mounted microphone. After the experimental task was completed, participants completed a measure of receptive vocabulary (the Shipley-2 Institute of Living Vocabulary Subscale; Shipley, Gruber, Martin, & Klein, 2009) and a separate measure of productive vocabulary (the Multilingual Naming Test; Gollan, Weissberger, Runnqvist, Montoya, & Cera, 2012). These helped us control for any effects of differences in vocabulary knowledge on lexical processing (e.g., Mainz, Shao, Brysbaert, & Meyer, 2017).

Sentence prime with picture naming task. Each trial began with a fixation cross (+) presented in the center of the screen for 500 ms. The fixation was followed by the first word of the sentence fragment. Subsequently, the remaining words of the sentence were presented one at a time at the center of the screen in

This document is copyrighted by the American Psychological Association or one of its allied publishers.
This article is intended solely for the personal use of the individual user and is not to be disseminated broadly.

standard rapid serial visual presentation fashion. Each word remained on the screen for 275 ms. After the final word of the sentence fragment was presented, a picture appeared and remained on the screen for 600 ms. Participants were instructed to read the words within the sentence silently for comprehension and to name the picture aloud before it disappeared. If the participant did not respond within 600 ms, their response could be registered for an additional 300 ms, during which a blank screen was displayed. An interstimulus interval of 1500 ms occurred between trials.

Multilingual Naming Test. Immediately following the picture naming in context task, participants completed the Multilingual Naming Test to measure individual differences in native language vocabulary knowledge. Participants were shown a set of 68 black and white line drawing images and instructed to name each image aloud as quickly as possible. Participants were given two different kinds of prompts if they gave an incorrect response. A semantic cue was provided, in the form of a brief definition of the object. If participants did not retrieve the correct word after receiving the semantic cue, they were also provided a phonological cue, the first letter of the response. If participants still could not respond with the correct name, the response was marked as incorrect and they were instructed to move on. Pictures were presented in an order of ascending difficulty. Score on the test is number of pictures correct.

Shipley-2 Institute of Living Vocabulary Subscale. The Shipley-2 Test comprises 40 stimulus words, presented in generally ascending order of difficulty. Participants selected the word that was the closest synonym to the stimulus word from among four presented options. Score on this test is a standardized score (with 100 indicating average performance on demographically matched sample).

Data on these vocabulary measures is shown in Table 3. Two sample heteroscedastic *t* tests (using the Welch-Satterthwaite correction) were used to compare scores from Experiments 2 and 3 to the Experiment 1 baseline. No differences were found in Multilingual Naming Test scores (ts <1.5, ps $> .15$). Young adults in Experiment 2 had higher Shipley-2 scores than participants in Experiment 1, $t(31.013 = 3.291$, $p < .005$, and older adults in Experiment 3 showed a similar trend relative to the Experiment 1 baseline, $t(29.370) = 1.786$, $p < .09$. We therefore included Shipley-2 scores as a covariate in our analyses.

Results

Errors. Responses were categorized as correct or one of four types of errors: (a) name agreement errors (production of names that differed from those designated by the experimenter); (b) verbal disfluencies (stuttering, utterance repairs, and production of nonverbal sounds); (c) omissions; and (d) completion errors (where the sentence prime completion was produced instead of the picture name). We assessed interrater reliability on error classification by taking data from a random selection of four of the 54 participants across the three experiments. Two raters agreed on response classification for 98.5% ($n = 720$).

Participants were quite accurate, with a mean of 8.6% of trials ($SE = 0.9\%$) eliciting errors. Given the fairly high rate of agreement errors (mean 5.7% of trials, $SE = 0.7\%$) we excluded from analysis pictures that elicited 60% or more name agreement errors across Experiments 1 and 2 (*beaver, canoe, cheetah, dropper, jeep, leg, peeler, raccoon, sheep*). With these items removed, the average cloze probability of the target sentences (as assessed by younger adults) increased from 90.7% to 91.1%.

After removal of items with low name agreement, 17 completion errors remained in the data set (0.5% of trials); of these, 15 were in the semantic competitor condition and two in the unrelated condition. This was not sufficient to fit a regression model; however, this pattern indicates that participants were more likely to make a completion error when a semantic competitor was primed, as compared to an unrelated prime.

Acoustic properties: Data analysis methods. All items excluded in accuracy analyses were also excluded from the articulation models. Six additional items were also excluded (*cookie, mop, leg, olive, tire,* and *foot*) because participants frequently named a competitor for these items. Once these data were excluded, outlier removal on each dependent variable was conducted, by removing trials with measurements 3 standard deviations above and below each participant's mean. We first removed response time and word durations outliers from the entire data set, and fit models to these dependent variables using this set of data (98.2–98.5% of data retained within each experiment). At this point, the data set contained 7,462 tokens (2,168 in Experiment 1, 2,045 in Experiment 2, and 2,136 in Experiment 3). Then, in conducting more exploratory analyses of subword components, we separately removed outliers for these three dependent variables: first consonant duration (98.5–98.9% of remaining data retained), initial consonant duration (99.1–99.6% retained), and vowel distance (99.1–99.4% retained).[2] For the initial consonant duration model, tokens which were word-initial were also removed; after this, 85.1% of the data was retained (6,349 data points across all three experiments).

Models were fit in R using the lme4 package (Bates, Mächler, Bolker, & Walker, 2014). Fixed effects of each models are described in the sections for each dependent variable that follow. Selection of random effects followed Bates, Kliegl, Vasishth, and Baayen (2015); models were initially fit with the maximal random effects structure (Barr, Levy, Scheepers, & Tily, 2013), and PCA was used to identify components of the structure that did not contribute variance to the model fit. After model selection, each model was refit excluding data points with extreme residuals (>2.5 *SD*, following Baayen, 2008). Likelihood ratio tests were used to assess the significance of fixed effects, as they are less anticonservative than *t*-as-*Z* tests (Barr et al., 2013).

Results: Acoustic Properties

Below, we discuss key predictions of the theoretical accounts discussed above (repeated below; see Table 2). The full results of the linear mixed effects regressions can be found in Appendix C. In assessing these effects, we included a series of control variables in our regression model:

[2] We have also conducted models using more restrictive criteria, in which we restrict the data pool to only items where a response is available for all three conditions (match, unrelated, competitor) for a single participant. Although this more conservative approach better controls the distribution of data across conditions, it resulted in the removal of nearly 30% of the data. We found the effects in this more conservative analysis to be broadly consistent with the analyses reported here, so we have chosen to present this analysis, which is a more complete data set.

Table 3
Vocabulary Measures for Each Experiment

	Experiment 1		Experiment 2		Experiment 3	
	Young adults		Young adults		Older adults	
	No time pressure		Time pressure		No time pressure	
Measure	*M*	*SE*	*M*	*SE*	*M*	*SE*
MINT (productive vocabulary)	64.556	.623	62.222	.712	64.500	.526
Shipley-2 (receptive vocabulary)	113.444	1.190	118.278*	.863	117.333	1.811

Note. MINT = Multilingual Naming Test.
* Score significantly different from experiment 1 baseline.

This document is copyrighted by the American Psychological Association or one of its allied publishers.
This article is intended solely for the personal use of the individual user and is not to be disseminated broadly.

- Block, along with interactions of block with match status and semantic relatedness: controls for any effects due to the repetition of picture targets across blocks (e.g., repetition reduction; Baker & Bradlow, 2009).
- Shipley-2 score: controls for effects of overall differences in vocabulary on lexical processing (as noted above).
- Trial-level response time; overall response speed of participant: if articulation is ballistic, such that all effects of formulation are fixed at the moment of response initiation, planning measures (i.e., RT) should be positively correlated with acoustic measures (e.g., word duration), and there should be no remaining independent effects of formulation variables on the acoustic measures (see Buz & Jaeger, 2016; Fink et al., 2018, for discussion; see Strijkers & Costa, 2016, for additional discussion). We control for this in two ways: response time for each individual trial and a measure of the overall response speed of an individual. The latter is estimated by the best linear unbiased predictors (BLUPS; Baayen, 2008) of by-participant intercepts in a mixed effects model of RTs (detailed in Appendix C).

With these factors under statistical control, we examine the key predictions tested in our study. Table 4 provides descriptive statistics for our principal dependent measure, word duration, in each condition.

Interaction of formulation and articulation. As predicted, mismatch trials had longer word durations than match trials (β = 18.783), $\chi^2(1) = 30.09$, $p < .001$, suggesting that formulation disruptions lead to articulatory disruptions.

Long-distance interaction. There was no significant difference between word durations following unrelated versus semantically related primes ($\beta = -4.268$), $\chi^2(1) = 2.33$, $p = .127$, suggesting that disruptions to lexical selection do not yield enhanced articulatory disruptions.

Table 4
Mean Word Durations With Standard Errors Across Participants for Each Condition, Experiment 1

Condition	Mean word duration	*SE*
Match	370.838	(3.479)
Semantically related	376.807	(3.762)
Unrelated	380.109	(3.801)

Dynamic interaction. The match versus mismatching primes effect did not significantly interact with RT ($\beta = -19.718$), $\chi^2(1) = 1.56$, $p = .212$, suggesting that overlap between formulation and articulation did not increase when responses were more speeded. There was a significant positive interaction of semantic relatedness and RT ($\beta = 50.667$), $\chi 2\ (1) = 9.84$, $p = .002$. This was driven by a greater relationship between RT and word duration for semantically related ($r = .104$) versus unrelated primes ($r = .0597$). Critically, unrelated trials were particularly longer than semantically related trials at fast RTs (below the median RT, difference in word durations = 3.3 ms) as compared to slow RTs (above the median RT, mean difference = 2.1 ms; note that in the model RT was entered as a continuous factor). This suggests that across the range of RTs semantically related primes always have less of an effect on articulation than unrelated primes—contra the predictions of long-distance interaction.

Initial segment speech initiation. A series of regression models (including the control variables above) examined whether these condition effects could be found in the duration of initial consonants and following vowel, as well as in spectral properties of the vowel. Full model specifications and condition means for each measure can be found in Appendices C and D, respectively.

The effect of match status was significant for initial consonants ($\beta = 10.852$), $\chi^2(1) = 16.90$, $p < .001$, but not for vowel durations, $\chi^2(1) < 1$, $p > .80$. Semantic relatedness did not significantly affect either duration measure, χ^2s$(1) < 2$, ps > 0.15. Figure 2 provides a visualization of the relative effect sizes on the three duration measures. This suggests that, as predicted by an account where speech is initiated prior to completing planning, the effects of formulation disruptions on articulation are largest for initial consonants.

There was mixed evidence as to whether these duration effects increased in faster responses. For initial consonants, there was a significant negative interaction of match status and RT ($\beta = -31.232$), $\chi^2(1) = 9.87$, $p = .002$, such that disruptions caused by mismatch effects increased at shorter RTs (as predicted by dynamic interaction accounts). For vowel durations, there was no significant interaction of match status and RT, χ^2s$(1) < 1$, ps > 0.40. However, there was a significant interaction of response time and semantic relatedness (as is found in the whole word analysis; $\beta = 20.607$, $\chi^2(1) = 6.99$, $p = .008$).

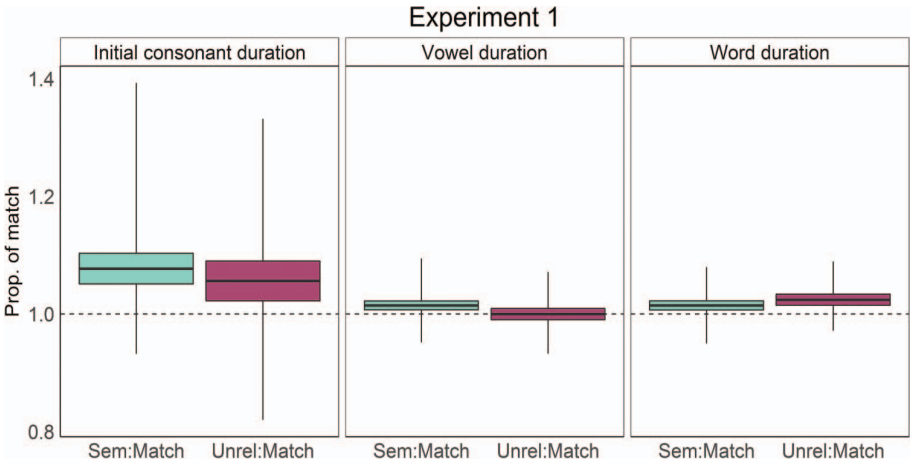


Figure 2. Condition means of each dependent variable for the no-match conditions (sem = semantically related competitor, unrel = unrelated) as a proportion of match condition. An average value within each condition was generated for each participant. The boxplots show the mean (central line) and standard error across participants (width of box). Wings show range. A value of 1 indicates that the match condition and no-match condition had the same average measurement; values above 1 indicate that the no-match condition had a longer or larger measurement than the match condition. See the online article for the color version of this figure.

This document is copyrighted by the American Psychological Association or one of its allied publishers.
This article is intended solely for the personal use of the individual user and is not to be disseminated broadly.

Finally, with respect to spectral properties of the vowel, there was no significant effect of match status or semantic relatedness, χ^2s(1) < 1, *p*s > 0.10.

Discussion

Experiment 1 provided evidence consistent with interactions between formulation and articulation, but no evidence for long-distance interaction effects. As predicted by accounts where speech is initiated as soon as the initial segment is planned, these formulation-articulation interactions appeared strongest for initial segments.

There was no evidence that these interactive effects were dynamic. This may be due to our reliance on natural, planned variation in response time. In Experiment 2, we followed the design of previous work argued to support dynamic interaction (e.g., Kello et al., 2000) and tested young adults naming pictures with explicit time pressure. This stronger manipulation should increase the chances that the degree of overlap between planning and articulation will increase.

Experiment 2: Younger Adults Naming Under Explicit Time Pressure

Methods

Participants. We recruited 18 younger adult participants (nine male, nine female) at Northwestern University using the Linguistics Department subject pool and ads recruiting participants on campus for monetary payment. Participants received either course credit or were paid $10/hr. They reported learning no language other than English before age 5. Age ranged from 18 to 22 years ($M = 19.72$, $SD = 0.89$).

Materials. The sentence completion materials were identical to Experiment 1.

Design and procedure. The design and procedure was nearly identical to Experiment 1, with the following exceptions. Following Severens et al. (2008, Experiment 3), a deadline was imposed for initiating the naming response. Pictures appeared for 600 ms. Participants were instructed to name the picture before it disappeared. After the picture disappeared, a blank screen appeared and responses could still be registered for an additional 900 ms. When a software voice-key (with amplitude threshold adjusted for each participant) detected a response or when the 900 ms period ended, there was a blank screen for 1,500 ms. If participants did not initiate their response within 600 ms (as measured by the software voice key), a written message appeared on the right portion of the screen at the end of the trial to indicate that the response was too slow.

Results

Errors. Analysis indicated that participants were much less accurate in Experiment 2, with 20.8% of trials eliciting errors (compared to 8.6% in Experiment 1). Most of the errors were categorized as name agreement errors, with an error rate of 12.1%. However, participants made many more completion errors with time pressure. 4.6% of responses were completion errors. We identified 2.9% of the responses as verbal disfluencies and only 1.2% as omissions. After removing low name agreement pictures, 97.8% of the dataset was retained (2,636 trials).

We analyzed the rate of completion errors (relative to correct trials) for semantic versus unrelated primes. The average rate of completion errors for competitor trials was 9.716% ($SE = 2.025\%$) versus 4.039% ($SE = 1.303\%$) for unrelated trials. We fit a logistic mixed effects regression model to this data, with semantic relatedness and block as predictors (random effects included correlated by-subject slopes for semantic relatedness, block, and their interactions; random intercept for items). There was a significant main effect of semantic relatedness ($\beta = 5.770$), $\chi^2(1) = 9.25$, $p = .002$,

This document is copyrighted by the American Psychological Association or one of its allied publishers.
This article is intended solely for the personal use of the individual user and is not to be disseminated broadly.

indicating that there were more completion errors from semantic versus unrelated primes. There was a marginal main effect of block ($\beta = -1.308$), $\chi^2(1) = 3.83$, $p = .051$, indicating that errors decreased over the course of the experiment. The interaction of block and semantic relatedness was also significant ($\beta = -1.143$), $\chi^2(1) = 6.25$, $p = .0124$. The rate of errors following unrelated primes was relatively constant across blocks (Block 1: 4.18%; Block 2: 3.79%; Block 3: 3.61%), whereas the rate of errors on semantically related primes decreased over repeated presentations (Block 1: 14.4%; Block 2: 7.84%; Block 3: 7.26%). Because the error rate is relatively low, we confirmed this pattern by running a logistic regression of rare events model (Choirat et al., 2018; King & Zeng, 2001). This approach confirmed the critical significant effect of semantic relatedness ($\beta = 0.937$, $z = 4.404$, $p < .001$) and the marginal effect of block ($\beta = -0.242$, $z = -1.950$, $p = .064$); the interaction did not reach significance ($\beta = -0.330$, $z = -1.262$, $p = .207$).

Results: Acoustic Properties

Analysis methods followed Experiment 1. We first note that our experimental manipulation not only increased error rates relative to Experiment 1, but also successfully produced decreased RTs relative to Experiment 1 (see Appendix D). Table 5 provides descriptive statistics for our principal dependent measure, word duration, in each condition. (Comparison with Table 4 will reveal that word durations are overall shorter in Experiment 2 vs. 1.)

Interaction of formulation and articulation. Mismatch trials had longer word durations than match trials ($\beta = 12.717$), $\chi^2(1) = 10.12$, $p = .002$, suggesting that the conflicting representations activated by mismatched primes disrupted target articulation.

Long-distance interaction. There was a nonsignificant (marginal) *decrease* in word durations following semantically related versus unrelated primes ($\beta = -5.298$), $\chi^2(1) = 3.66$, $p = .056$. As can be seen in Table 5, semantically related primes had essentially the same duration as match trials. Note this occurred in spite of an increase in the production of completion errors under speeded responding. This pattern is opposite that predicted by long-distance interaction; the disruption that leads to overt speech errors should lead to distortions in articulation.

Dynamic interaction. There was a nonsignificant (marginal) positive interaction of match versus mismatching primes with RT ($\beta = 27.527$), $\chi^2(1) = 3.45$, $p = .063$; if anything, effects of mismatching primes increase with longer RTs. In contrast, dynamic interaction accounts predict that interactions should increase with shorter RTs (where there is greater overlap in processing). There was no significant interaction of semantic relatedness and RT ($\beta = 12.328$), $\chi^2(1) = 0.76$, $p = .382$.

Table 5
Mean Word Durations With Standard Errors Across Participants for Each Condition, Experiment 2

Condition	Mean word duration	*SE*
Match	324.491	(3.295)
Semantically related	323.857	(3.663)
Unrelated	331.083	(3.532)

Experiment 2 also allows a stronger test of the dynamic interaction hypothesis; a cross-experiment comparison between Experiment 1 (without explicit time pressure, yielding longer RTs) and the current experiment (with explicit time pressure, yielding overall shorter RTs). We examined this via a separate regression over data from both experiments. This was structured similarly to the overall model of word durations, with the addition of fixed effects for experiment and interactions of experiment and all other effects (full results can be found in Appendix C). Critically, the interaction of experiment with the effect of matching primes was not significant, nor was the interaction with semantic relatedness of the prime, χ^2s(1) < 0.10, ps > 0.80.[3] This suggests that the main influence of explicit time pressure was to simply speed responses, not increase interactive effects.

Initial segment speech initiation. The effect of match status was significant for initial consonants ($\beta = 8.621$), $\chi^2(1) = 15.36$, $p < .001$, but not significant (marginal) for vowel durations ($\beta = 3.395$), $\chi^2(1) = 3.85$, $p = .05$. Figure 3 provides a visualization of the relative effect sizes on the three duration measures. Similar to Experiment 1, this suggests that the relative strength of condition effects is largest on initial consonants (although note that the range of effect sizes exhibits considerable overlap across measures).

This conclusion is tempered by a significant effect of match status on spectral properties of vowels ($\beta = 12.291$), $\chi^2(1) = 6.01$, $p = .014$; mismatch trials had greater vowel distances than match trials, consistent with disruption to vowel articulation. Thus, although duration effects are larger on initial consonants versus vowels, effects of formulation disruption persist into the vowel. There was no evidence that any of these by-position effects interacted with RT (see Appendix C for full results).

With respect to long-distance interactions in the context of initial segment speech initiation, the effect of semantic relatedness was significant for initial consonants (such that semantic competitors resulted in less interference than unrelated primes; $\beta = -3.759$, $\chi^2(1) = 6.44$, $p = .011$) but not vowel durations ($\beta = -0.284$, $\chi^2(1) = 0.04$, $p = .841$). Again, the direction of this effect is opposite that predicted by long-distance interactions. With respect to vowel distance, there was no significant effect of semantic relatedness ($\beta = 1.172$, $\chi^2(1) = 0.07$, $p = .790$). In addition, there was no evidence that any of these by-position effects interacted with production speed (see Appendix C for full results).

Finally, to examine whether dynamic interaction effects would appear when comparing experiments with versus without explicit time pressure, models compared each of these measures of vowel and consonant articulation across Experiment 1 versus Experiment 2. There were no significant interactions of interactive effects with RT and experiment (see footnote 3 and Appendix C for full results).

[3] There was a significant three-way interaction of experiment, match status, and RT; this reflected the non-significant negative interaction of match status and RT in Experiment 1 vs. the non-significant positive interaction in the current experiment. This does not provide evidence in favor of increased effects with decreased RTs. (Note a similar interaction was found for initial consonant duration as well.)

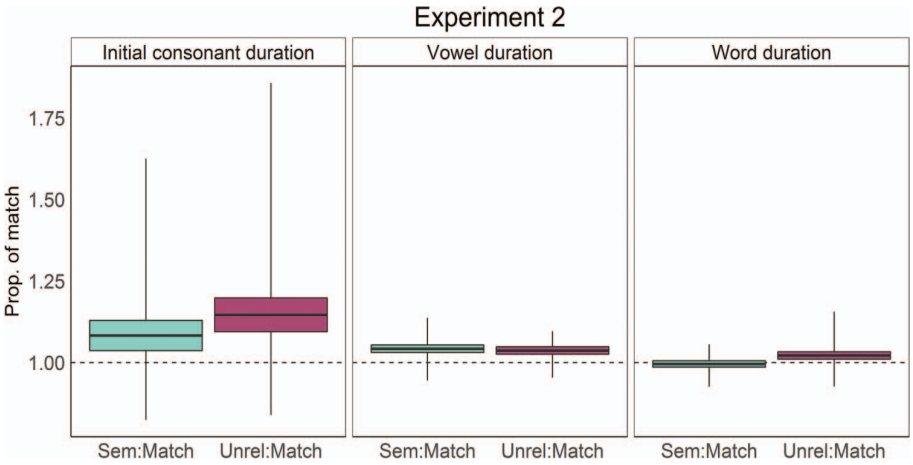


Figure 3. Condition means of each dependent variable for the no-match conditions (sem = semantically related competitor, unrel = unrelated) as a proportion of match condition. An average value within each condition was generated for each participant. The boxplots show the mean (central line) and standard error across participants (width of box). Wings show range. A value of 1 indicates that the match condition and no-match condition had the same average measurement; values above 1 indicate that the no-match condition had a longer or larger measurement than the match condition. See the online article for the color version of this figure.

Discussion

Parallel to previous work (Severens et al., 2008), imposing a deadline for responding produced greater error rates. The phonetic effects were largely parallel to Experiment 1. We saw clear evidence for interactions between formulation and articulation. Speech appeared to be initiated by planning of the initial segment, as disruption was particularly strong for initial consonants (but also extending into vowels). However, there was no clear evidence of long-distance interactions; furthermore, in spite of the inclusion of explicit time pressure, there was no evidence for dynamic interaction effects (parallel to Damian, 2003, but in contrast to the effects observed by Kello et al., 2000).

The finding that increased disruptions to lexical selection (due to time pressure) did not yield enhanced articulatory effects may reflect the overall stability of lexical selection in younger adults. We examine this in Experiment 3 by testing older adults. If, following the inhibitory deficit hypothesis, the older adults are less effective at suppressing nontarget representations, interactive effects are predicted to be stronger than in younger adults.

Experiment 3: Older Adults Naming

Methods

Participants. We recruited 18 older adult participants (3 male, 15 female) from communities in Chicago and Evanston, IL. Participants were paid $10/hr. They reported learning no language other than English before age 5 and no history of color blindness or language/cognitive impairment. Age ranged from 60 to 77 years (M = 68.38, SD = 5.88).

Materials and design. The sentence completion materials used here were identical to Experiment 1. We separately normed cloze probability in older adults; the average probability was 86.7% (*SD* 17.9%)

Procedure. The procedure was identical to Experiment 1.

Results

Errors. Analysis indicated that 14.9% of trials elicited errors. As in Experiments 1 and 2, most of the errors were categorized as name agreement errors (error rate = 11.4%). There were few completion errors (error rate = 0.5%). We identified 2.0% of the responses as verbal disfluencies and only 1.0% as omissions. We excluded 17 pictures from analysis that elicited a large number of agreement errors within this group of participants (*beaver, cheetah, couch, dollar, dolphin, fly, headphones, ipod, jeans, juice, laptop, nachos, speaker, vase, lime, peeler, mop, olive, ring, wheel*). This allowed for retention of 90.3% of the dataset (2614 trials). Excluding these items, the average cloze probability of the target sentences for older adults remained relatively stable, from 86.7% to 86.8%.

After removal of items with low name agreement, 14 completion errors remained in the data set (0.5% of trials); of these, all were in the semantic competitor condition. As such, it was not possible to fit a model to the error data in Experiment 3. The pattern matches those in Experiment 1 and indicates that participants were more likely to make a completion error when a semantic competitor was primed.

Results: Acoustic Properties

Analysis methods followed Experiment 1. Four additional items were removed from articulation models because of high rates where participants named the competitor (*mop, olive, ring, wheel*). Table 6 provides descriptive statistics for our principal dependent measure, word duration, in each condition. (Comparison with Table 4 will reveal that word durations are overall

This document is copyrighted by the American Psychological Association or one of its allied publishers.
This article is intended solely for the personal use of the individual user and is not to be disseminated broadly.

Table 6
Mean Word Durations With Standard Errors Across Participants for Each Condition, Experiment 3

Condition	Mean word duration	*SE*
Match	393.569	(3.815)
Semantically related	397.651	(4.105)
Unrelated	421.518	(4.501)

longer in Experiment 3 vs. 1, consistent with slower speech in older adults.)

Interaction of formulation and articulation; long-distance interaction. In contrast to the experiments with young adults, collapsing across semantically unrelated and semantically related, mismatch trials did not have longer word durations than match trials ($\beta = 5.105$, $\chi^2(1) = 1.31$, $p = .253$).[4] However, there was a significant *decrease* in word durations following semantically related versus unrelated primes ($\beta = -11.304$, $\chi^2(1) = 8.86$, $p = .003$). As can be seen in Table 6, semantically related primes had essentially the same duration as match trials (with unrelated trials showing longer durations). So, although the result with unrelated primes is consistent with interactions between formulation and articulation, the pattern with semantically related primes is opposite that predicted by long-distance interaction; the disruption that leads to overt speech errors should lead to distortions in articulation (certainly relative to the match condition).

Dynamic interaction. Neither match status ($\beta = -20.459$), $\chi^2(1) = 1.75$, $p = .186$, nor semantic relatedness interacted with RT ($\beta = 7.360$), $\chi^2(1) = 0.19$, $p = .667$, failing to provide support for the predictions of accounts with dynamic interaction between cognitive processes.

Inhibitory deficit. The inhibitory deficit hypothesis predicts that older adults (with less ability to suppress competitors) should show larger interactive effects than young adults. We assessed this prediction using a regression model; this was structured similarly to the overall model of word durations, with the addition of fixed effects for experiment (Experiment 3 vs. Experiment 1) and interactions of experiment and all other effects (full results can be found in Appendix C). The interaction of experiment with the effect of matching primes was not significant.[5] ($\beta = -2.320$, $\chi^2(1) = 0.14$, $p = .704$). However, the effect of semantic relatedness did interact with experiment ($\beta = -10.309$, $\chi^2(1) = 4.41$, $p = .038$); underscoring the unexpected finding above, there was a larger (more negative) effect was found with older adults. This fails to support an increased effect of lexical selection disruptions on processing.

Initial segment speech initiation. In contrast to the previous two experiments, there was no clear effect on initial consonants, nor any effect on vowels. The effect of match status was nonsignificant for initial consonants ($\beta = 3.381$), $\chi^2(1) = 2.72$, $p = .099$, and nonsignificant for vowel durations ($\beta = 2.743$), $\chi^2(1) = 1.90$, $p = .168$. Similarly, the effect of semantic relatedness was not significant for initial consonants ($\beta = -2.486$), $\chi^2(1) = 2.39$, $p = .122$, as well as vowel durations ($\beta = -0.729$), $\chi^2(1) = 0.22$, $p = .636$. Figure 4 provides a visualization of the relative effect sizes on the three duration measures. Spectral properties of the vowel also failed to show significant effects—match status: $\beta = 1.295$, $\chi^2(1) = 0.04$, $p = .843$; semantic relatedness: $\beta = -2.664$, $\chi^2(1) = 0.36$, $p = .547$. There were no significant modulation of these effects by RT (see Appendix C for full models.).

There was also little evidence of dynamic long-distance interactions in the context of initial speech initiation effects. Experiment 3 RTs did not interact with semantic relatedness for initial consonants ($\beta = -5.706$), $\chi^2(1) = 0.67$, $p = .413$; vowel durations ($\beta = -0.173$), $\chi^2(1) = 0$, $p = .975$; or vowel spectral properties ($\beta = -19.634$), $\chi^2(1) = 1.09$, $p = .297$.

Finally, the cross-experiment models examining consonant and vowel articulation measures failed to find support for inhibitory deficits (see Appendix C for full results). There was, however, a significant effect of age; vowel distances were overall larger for older speakers ($\beta = 35.565$), $\chi^2(1) = 4.650$, $p = .031$.

Discussion

Although older adults produced more errors (14.1% vs. 8.6% in Experiment 1) and had overall longer word durations (consistent with slowed speech associated with aging), they did not show heightened sensitivity to differences across experimental conditions. Their rate of completion errors was numerically lower than that of younger speakers, and the majority of the effects on articulatory measures were weaker (e.g., lack of significant condition effects on initial consonants). We discuss the implications of this finding in the following section.

General Discussion

Previous research has yielded mixed results on the extent and scope of interactions between formulation and articulatory processes. This study aimed to provide novel evidence on these issues using a paradigm that disrupted formulation. Participants named a picture following a sentence priming a matching or nonmatching picture label (Ferreira & Griffin, 2003). A novel automatic phonetic analysis tool allowed us to examine word durations; in concert with other analysis tools, we used this to automatically measure more fine-grained aspects of word articulation associated with specific segments. Consistent with a number of previous studies, when priming a nontarget word disrupts formulation, articulation is disrupted as well. This effect appears to be stronger in the initial position of the words, suggesting that speech is initiated as soon as these segments are planned; subsequent segments benefit from additional planning, suppressing the temporary effects of disruption. Some of the mixed evidence in the literature may therefore reflect the use of coarse-grained measures of articulatory processing. However, two other possible confounds in previous work did not appear to influence the current results. Disruptions to formulation did not exert a stronger influence with explicit time pressure or faster RTs, suggesting overlap between formulation and articulation is relatively fixed. Older participants showed, if anything, weaker interactive effects than younger

[4] There was a significant interaction of block and match status, such that overall reduction of word durations across blocks was strongest for mismatch trials.

[5] There was a significant three-way interaction of experiment, match status, and block; this reflected the non-significant interaction of match status and block in Experiment 1 vs. the significant interaction in the current experiment.

This document is copyrighted by the American Psychological Association or one of its allied publishers.
This article is intended solely for the personal use of the individual user and is not to be disseminated broadly.

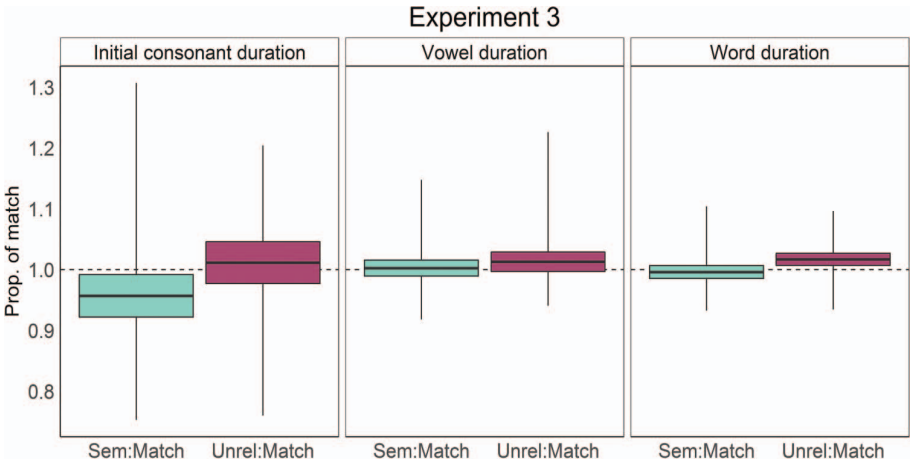


Figure 4. Condition means of each dependent variable for the no-match conditions (sem = semantically related competitor, unrel = unrelated) as a proportion of match condition. An average value within each condition was generated for each participant. The boxplots show the mean (central line) and standard error across participants (width of box). Wings show range. A value of 1 indicates that the match condition and no-match condition had the same average measurement; values above 1 indicate that the no-match condition had a longer or larger measurement than the match condition. See the online article for the color version of this figure.

This document is copyrighted by the American Psychological Association or one of its allied publishers.
This article is intended solely for the personal use of the individual user and is not to be disseminated broadly.

adults, suggesting that aging does not decrease the ability to suppress nontarget representations during speech planning.

A central goal of this work was to provide a clearer picture of long-distance interactive effects: specifically, the impact of disruptions to lexical selection (as opposed to other aspects of formulation) on articulatory processing. The priming paradigm successfully disrupted lexical selection; semantically related primes yielded overt, whole-word picture naming errors at a higher rate than unrelated primes. However, this enhanced disruption of lexical selection was associated with weaker disruptions to articulation than the unrelated condition.

Challenges Raised by Results From Semantic Primes

As discussed in the introduction, a large body of work has demonstrated that there are many situations where articulatory processing is facilitated when formulation processes are facilitated. Several theoretical proposals in the language production literature assume that conceptual processes preceding lexical selection are facilitated by semantic relationships (for reviews and discussion see Abdel Rahman & Melinger, 2009; Scaltritti, Peressotti, & Navarrete, 2017); under some contexts, this can facilitate subsequent processing. In the context of a task using a sentence prime, previous studies have measured the difficulty of reading a word following a high-cloze probability sentence; in such tasks, there is less disruption when the unexpected word is semantically related to the cloze completion than when it is semantically unrelated to the target (Federmeier & Kutas, 1999). Furthermore, results from a wide array of studies have suggested that older adults show larger semantic priming effects (e.g., Laver & Burke, 1993); here, older adults show larger facilitatory effects of semantic primes than younger adults. These results make it plausible that semantic primes could facilitate target retrieval, and as a consequence, articulatory processing. The challenge for such an account is to explain the error data. Whatever facilitation the semantic primes provide the target during formulation, it is clearly insufficient to suppress the occurrence of semantic errors. If articulatory effects arise as a result of cascading activation, why does the heightened activation of semantic competitors fail to disrupt articulatory processing?

One possibility (suggested by an anonymous reviewer) is that these effects represent a kind of speed/accuracy tradeoff; participants capitalize on the facilitatory effects of semantic primes to speed formulation and articulation at the cost of allowing more errors to be produced. Consistent with this, there is a trend toward facilitatory effects of semantic primes in RTs (see Appendixes C and D for analyses and statistical models). However, a key prediction of this account is not confirmed by our data. Experiment 2 differs from Experiment 1 in that speed is explicitly emphasized. This results in a clear increase in errors and a decrease in RTs (illustrating a speed/accuracy tradeoff). However, it is not accompanied by a significant increase in semantic effects on articulatory processing. Although the effect of semantic primes is significant in Experiment 2 but not in Experiment 1, regression models explicitly comparing the two experiments show no interaction of semantic primes with experiment. This possibility deserves further investigation; it is possible that our study did not have sufficient power to detect such interactions.

Another alternative is to allow for multiple processes or information sources to directly influence articulatory processing. Rather than articulation being modulated solely by the relative activation of target and competitor representations (i.e., the output of formulation processes), it may be that target activation has a privileged effect on articulation, irrespective of the activation of competitors. For example, Baker and Bradlow (2009) presented analyses showing that the facilitatory effects of contextual predictability cannot be reduced to the prosodic structure of an utterance (with predictable items occurring in less prominent positions). They therefore propose that predictability exerts a direct influence on articulation.

Such a direct influence could serve to reduce articulatory disruptions for semantically related primes.

A natural question is then whether predictability provides a sufficient account of the data, obviating the need to appeal to interaction to account for the data reported here. Other results in the literature on interactive effects suggest a pure predictability account is insufficient (see Arnold, 2016, for a review). As reviewed in the introduction, Jacobs et al. (2015) shows that facilitation of articulation requires the prior facilitation of word form processing. Findings such as these suggest that predictability and interactive effects codetermine articulatory prominence.

Initiation of Speech by Initial Segment

The enhanced effects on initial positions are consistent with previous findings from reading aloud (Kawamoto et al., 1999) and typing (Scaltritti et al., 2017; but see Damian & Freeman, 2008). Such results are predicted if planning continues following articulation onset. While the initial portions of the word are being articulated, planning may continue for later portions. This extra time may allow the production system to resolve planning conflicts before having to articulate, reducing effects at later portions of the word. Future research should be focused on examining more fine-grained measures of speech articulation to capture such transient effects.

Dynamic Interaction

The failure to find effects of naturally occurring variation in response speed is consistent with the absence of such effects in Fink et al.'s (2018) study of semantic interference effects. The current study provides converging evidence from the absence of effects from explicit response pressure, consistent with Damian (2003) and Damian and Freeman (2008). This suggests that the coordination of formulation processes and articulation processes is not as flexible as suggested by Kello et al. (2000). Future work, perhaps including high-powered replications of Kello et al. (2000), might allow us to determine if the original results were spurious. The automated phonetic analysis developed here could facilitate such work.

Lack of Inhibitory Deficits in Aging

Although older adults produced longer word durations (suggesting a reduction in articulatory rate), and somewhat lower accuracy overall, aging did not enhance the effect of lexical selection disruptions on articulation. In fact, older adults showed enhanced facilitatory effects of semantic primes. This is consistent with previous work suggesting that various predictions of the inhibitory deficit hypothesis are not confirmed by empirical studies of cognitive aging (see Burke & College, 1997, for a review). The transmission deficit hypothesis (Burke et al., 1991; MacKay, 1987) might provide a framework for understanding such effects. According to this theory, connections between representational units are weakened with increasing age. Specifically, weak connections between phonological and lexical levels prevent form-level representations from becoming adequately activated, yielding difficulty in retrieval (MacKay, 1987; Taylor & Burke, 2002). Furthermore, because of the degraded links from semantics to phonology, the amount of activation that cascades to subsequent phonetic/articulatory processes should be significantly reduced. This would reduce the influence of weakly activated competitor representations, decreasing interactive effects.

The Relationship Between Formulation and Articulation

More generally, this set of findings suggests that interactions between formulation and articulation do not simply cause articulation to mirror all aspects of formulation processing. Although disruptions to formulation can yield articulatory disruptions, the extent and nature of these disruptions is not a straightforward consequence of the outcome of formulation processes. Articulatory effects may be sensitive to multiple aspects of processing (e.g., lexical selection difficulties as well as positive effects of contextual predictability). A more nuanced model of planning-articulation interactions may better account for this complex relationship and yield predictions about contexts that facilitate versus inhibit observing long-distance interactive effects.

References

Abdel Rahman, R., & Melinger, A. (2009). Semantic context effects in language production: A swinging lexical network proposal and a review. *Language and Cognitive Processes, 24,* 713–734. http://dx.doi.org/10.1080/01690960802597250

Adi, Y., Keshet, J., Cibelli, E., & Goldrick, M. (2017). Sequence segmentation using joint RNN and structured prediction models. In *Proceedings of the 42nd IEEE international conference on acoustics, speech, and signal processing (ICASSP),* (pp. 2422–2426). New Orleans, LA: Institute of Electrical and Electronics Engineers. http://dx.doi.org/10.1109/ICASSP.2017.7952591

Adi, Y., Keshet, J., Cibelli, E., Gustafson, E., Clopper, C., & Goldrick, M. (2016). Automatic measurement of vowel duration via structured prediction. *The Journal of the Acoustical Society of America, 140,* 4517–4527. http://dx.doi.org/10.1121/1.4972527

Adi, Y., Keshet, J., Dmitrieva, O., & Goldrick, M. (2016). Automatic measurement of voice onset time and prevoicing using recurrent neural networks. *Proceedings of Interspeech, 2016,* 3152–3155. http://dx.doi.org/10.21437/Interspeech.2016-893

Adi, Y., Keshet, J., & Goldrick, M. (2015). Vowel duration measurement using deep neural networks. In *Proceedings of the IEEE International Workshop on Machine Learning for Signal Processing*. Boston, MA: IEEE. http://dx.doi.org/10.1109/MLSP.2015.7324331

Arnold, J. E. (2016). Explicit and emergent mechanisms of information status. *Topics in Cognitive Science, 8,* 737–760. http://dx.doi.org/10.1111/tops.12220

Baayen, R. H. (2008). *Analyzing linguistic data: A practical introduction to statistics using R*. Cambridge, UK: Cambridge University Press. http://dx.doi.org/10.1017/CBO9780511801686

Baker, R. E., & Bradlow, A. R. (2009). Variability in word duration as a function of probability, speech style, and prosody. *Language and Speech, 52,* 391–413. http://dx.doi.org/10.1177/0023830909336575

Balota, D. A., Boland, J. E., & Shields, L. W. (1989). Priming in pronunciation: Beyond pattern recognition and onset latency. *Journal of Memory and Language, 28,* 14–36. http://dx.doi.org/10.1016/0749-596X(89)90026-0

Barr, D. J., Levy, R., Scheepers, C., & Tily, H. J. (2013). Random effects structure for confirmatory hypothesis testing: Keep it maximal. *Journal of Memory and Language, 68,* 255–278. http://dx.doi.org/10.1016/j.jml.2012.11.001

This document is copyrighted by the American Psychological Association or one of its allied publishers.
This article is intended solely for the personal use of the individual user and is not to be disseminated broadly.

Bates, D., Kliegl, R., Vasishth, S., & Baayen, H. (2015). Parsimonious mixed models. *arXiv preprint:1506.04967*.

Bates, D., Mächler, M., Bolker, B., & Walker, S. (2014). Fitting linear mixed-effects models using lme4. *arXiv preprint arXiv:1406.5823*.

Belke, E. (2017). The role of task-specific response strategies in blocked-cyclic naming. *Frontiers in Psychology, 7,* 1955. http://dx.doi.org/10.3389/fpsyg.2016.01955

Belke, E., Meyer, A. S., & Damian, M. F. (2005). Refractory effects in picture naming as assessed in a semantic blocking paradigm. *The Quarterly Journal of Experimental Psychology A: Human Experimental Psychology, 58,* 667–692. http://dx.doi.org/10.1080/02724980443000142

Bell, A., Brenier, J. M., Gregory, M., Girand, C., & Jurafsky, D. (2009). Predictability effects on durations of content and function words in conversational English. *Journal of Memory and Language, 60,* 92–111. http://dx.doi.org/10.1016/j.jml.2008.06.003

Biegler, K. A., Crowther, J. E., & Martin, R. C. (2008). Consequences of an inhibition deficit for word production and comprehension: Evidence from the semantic blocking paradigm. *Cognitive Neuropsychology, 25,* 493–527. http://dx.doi.org/10.1080/02643290701862316

Brodeur, M. B., Dionne-Dostie, E., Montreuil, T., & Lepage, M. (2010). The Bank of Standardized Stimuli (BOSS), a new set of 480 normative photos of objects to be used as visual stimuli in cognitive research. *PLoS ONE, 5,* e10773. http://dx.doi.org/10.1371/journal.pone.0010773

Brodeur, M. B., Guérard, K., & Bouras, M. (2014). Bank of Standardized Stimuli (BOSS) phase II: 930 new normative photos. *PLoS ONE, 9,* e106953. http://dx.doi.org/10.1371/journal.pone.0106953

Brysbaert, M., & New, B. (2009). Moving beyond Kučera and Francis: A critical evaluation of current word frequency norms and the introduction of a new and improved word frequency measure for American English. *Behavior Research Methods, Instruments & Computers, 41,* 977–990. http://dx.doi.org/10.3758/BRM.41.4.977

Burke, D. M., & College, P. (1997). Language, aging, and inhibitory deficits: Evaluation of a theory. *The Journals of Gerontology: Series B: Psychological Sciences and Social Sciences, 52,* 254–264. http://dx.doi.org/10.1093/geronb/52B.6.P254

Burke, D. M., MacKay, D. G., Worthley, J. S., & Wade, E. (1991). On the tip of the tongue: What causes word finding failures in young and older adults? *Journal of Memory and Language, 30,* 542–579. http://dx.doi.org/10.1016/0749-596X(91)90026-G

Buz, E., & Jaeger, T. F. (2016). The (in)dependence of articulation and lexical planning during isolated word production. *Language, Cognition and Neuroscience, 31,* 404–424. http://dx.doi.org/10.1080/23273798.2015.1105984

Choirat, C., Gandrud, C., Honaker, J., Imai, K., King, G., & Lau, O. (2018). relogit: Rare events logistic regression for dichotomous dependent variables. In C. Choirat, C. Gandrud, J. Honaker, K. Imai, G. King, and O. Lau (Eds.), *Zelig: Everyone's statistical software*. Retrieved from http://zeligproject.org/

Cutting, J. C., & Ferreira, V. S. (1999). Semantic and phonological information flow in the production lexicon. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 25,* 318–344. http://dx.doi.org/10.1037/0278-7393.25.2.318

Damian, M. F. (2003). Articulatory duration in single-word speech production. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 29,* 416–431. http://dx.doi.org/10.1037/0278-7393.29.3.416

Damian, M. F., & Freeman, N. H. (2008). Flexible and inflexible response components: A Stroop study with typewritten output. *Acta Psychologica, 128,* 91–101. http://dx.doi.org/10.1016/j.actpsy.2007.10.002

Damian, M. F., Vigliocco, G., & Levelt, W. J. (2001). Effects of semantic context in the naming of pictures and words. *Cognition, 81,* B77–B86. http://dx.doi.org/10.1016/S0010-0277(01)00135-4

Dell, G. S., & O'Seaghdha, P. G. (1992). Stages of lexical access in language production. *Cognition, 42,* 287–314. http://dx.doi.org/10.1016/0010-0277(92)90046-K

Dell, G. S., & Reich, P. A. (1981). Stages in sentence production: An analysis of speech error data. *Journal of Verbal Learning & Verbal Behavior, 20,* 611–629. http://dx.doi.org/10.1016/S0022-5371(81)90202-4

Dhooge, E., Baene, W. D., & Hartsuiker, R. J. (2013). A late locus of the distractor frequency effect in picture-word interference: Evidence from event-related potentials. *Brain and Language, 124,* 232–237. http://dx.doi.org/10.1016/j.bandl.2012.12.005

Dissen, Y., & Keshet, J. (2016). Formant estimation and tracking using deep learning. Paper presented at the 17th Annual Conference of the International Speech Communication Association, San Francisco, CA.

Drake, E., & Corley, M. (2015). Effects in production of word pre-activation during listening: Are listener-generated predictions specified at a speech-sound level? *Memory & Cognition, 43,* 111–120. http://dx.doi.org/10.3758/s13421-014-0451-9

Duchi, J., Hazan, E., & Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research, 12,* 2121–2159.

Elman, J. L. (1991). Distributed representations, simple recurrent networks, and grammatical structure. *Machine Learning, 7,* 195–225. http://dx.doi.org/10.1007/BF00114844

Federmeier, K. D., & Kutas, M. (1999). A rose by any other name: Long-term memory structure and sentence processing. *Journal of Memory and Language, 41,* 469–495. http://dx.doi.org/10.1006/jmla.1999.2660

Ferreira, V. S., & Griffin, Z. M. (2003). Phonological influences on lexical (mis)selection. *Psychological Science, 14,* 86–90. http://dx.doi.org/10.1111/1467-9280.01424

Fink, A. (2016). *The role of domain-general executive functions, articulation, and conceptualization in spoken word production* (Doctoral dissertation). Northwestern University.

Fink, A., Oppenheim, G. M., & Goldrick, M. (2018). Interactions between lexical access and articulation. *Language, Cognition and Neuroscience, 33,* 12–24. http://dx.doi.org/10.1080/23273798.2017.1348529

Finkbeiner, M., & Caramazza, A. (2006a). Now you see it, now you don't: On turning semantic interference into facilitation in a Stroop-like task. *Cortex: A Journal Devoted to the Study of the Nervous System and Behavior, 42,* 790–796. http://dx.doi.org/10.1016/S0010-9452(08)70419-2

Finkbeiner, M., & Caramazza, A. (2006b). Lexical selection is not a competitive process: A reply to La Heij et al. *Cortex: A Journal Devoted to the Study of the Nervous System and Behavior, 42:*1032–1036, 2006. http://dx.doi.org/10.1016/S0010-9452(08)70210-7

Frisch, S. A., & Wright, R. (2002). The phonetics of phonological speech errors: An acoustic analysis of slips of the tongue. *Journal of Phonetics, 30,* 139–162. http://dx.doi.org/10.1006/jpho.2002.0176

Gahl, S. (2008). "Thyme" and "time" are not homophones. The effect of lemma frequency on word durations in spontaneous speech. *Language, 84,* 474–496. http://dx.doi.org/10.1353/lan.0.0035

Gahl, S., & Strand, J. F. (2016). Many neighborhoods: Phonological and perceptual neighborhood density in lexical production and perception. *Journal of Memory and Language, 89,* 162–178. http://dx.doi.org/10.1016/j.jml.2015.12.006

Garrett, M. F. (1975). The analysis of sentence production. *Psychology of Learning and Motivation, 9,* 133–177. http://dx.doi.org/10.1016/S0079-7421(08)60270-4

Goldrick, M. (2006). Limited interaction in speech production: Chronometric, speech error, and neuropsychological evidence. *Language and Cognitive Processes, 21,* 817–855. http://dx.doi.org/10.1080/01690960600824112

Goldrick, M., Baker, H. R., Murphy, A., & Baese-Berk, M. (2011). Interaction and representational integration: Evidence from speech errors. *Cognition, 121,* 58–72. http://dx.doi.org/10.1016/j.cognition.2011.05.006

This document is copyrighted by the American Psychological Association or one of its allied publishers.
This article is intended solely for the personal use of the individual user and is not to be disseminated broadly.

Goldrick, M., & Blumstein, S. E. (2006). Cascading activation from phonological planning to articulatory processes: Evidence from tongue twisters. *Language and Cognitive Processes, 21,* 649–683. http://dx.doi.org/10.1080/01690960500181332

Goldrick, M., Keshet, J., Gustafson, E., Heller, J., & Needle, J. (2016). Automatic analysis of slips of the tongue: Insights into the cognitive architecture of speech production. *Cognition, 149,* 31–39. http://dx.doi.org/10.1016/j.cognition.2016.01.002

Goldstein, L., Pouplier, M., Chen, L., Saltzman, E., & Byrd, D. (2007). Dynamic action units slip in speech production errors. *Cognition, 103,* 386–412. http://dx.doi.org/10.1016/j.cognition.2006.05.010

Gollan, T. H., & Brown, A. S. (2006). From tip-of-the-tongue (TOT) data to theoretical implications in two steps: When more TOTs means better retrieval. *Journal of Experimental Psychology: General, 135,* 462–483. http://dx.doi.org/10.1037/0096-3445.135.3.462

Gollan, T. H., & Goldrick, M. (2018). Aging deficits in speech production revealed through reading aloud. Manuscript submitted for publication.

Gollan, T. H., Weissberger, G. H., Runnqvist, E., Montoya, R. I., & Cera, C. M. (2012). Self-ratings of spoken language dominance: A multilingual naming test (MINT) and preliminary norms for young and aging Spanish-English bilinguals. *Bilingualism: Language and Cognition, 15,* 594–615. http://dx.doi.org/10.1017/S1366728911000332

Graves, A., & Jaitly, N. (2014). Towards end-to-end speech recognition with recurrent neural networks. In *Proceedings of the 31st International Conference on Machine Learning* (Vol. 14, pp. 1764–1772). Beijing, China: International Conference on Machine Learning.

Graves, A., Mohamed, A. R., & Hinton, G. (2013). Speech recognition with deep recurrent neural networks. In *Acoustics, speech and signal processing (ICASSP), 2013 IEEE international conference on* (pp. 6645–6649). Vancouver, British Columbia, Canada: IEEE. http://dx.doi.org/10.1109/ICASSP.2013.6638947

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation, 9,* 1735–1780. http://dx.doi.org/10.1162/neco.1997.9.8.1735

Howard, D., Nickels, L., Coltheart, M., & Cole-Virtue, J. (2006). Cumulative semantic inhibition in picture naming: Experimental and computational studies. *Cognition, 100,* 464–482. http://dx.doi.org/10.1016/j.cognition.2005.02.006

Jacobs, C. L., Yiu, L. K., Watson, D. G., & Dell, G. S. (2015). Why are repeated words produced with reduced durations? Evidence from inner speech and homophone production. *Journal of Memory and Language, 84,* 37–48. http://dx.doi.org/10.1016/j.jml.2015.05.004

Jescheniak, J. D., Hahne, A., Hoffmann, S., & Wagner, V. (2006). Phonological activation of category coordinates during speech planning is observable in children but not in adults: Evidence for cascaded processing. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 32,* 373–386. http://dx.doi.org/10.1037/0278-7393.32.3.373

Kawamoto, A. H., Kello, C. T., Higareda, I., & Vu, J. V. (1999). Parallel processing and initial phoneme criterion in naming words: Evidence from frequency effects on onset and rime duration. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 25,* 362–381.

Kello, C. T., Plaut, D. C., & MacWhinney, B. (2000). The task dependence of staged versus cascaded processing: An empirical and computational study of Stroop interference in speech production. *Journal of Experimental Psychology: General, 129,* 340–360. http://dx.doi.org/10.1037/0096-3445.129.3.340

Keshet, J., Shalev-Shwartz, S., Singer, Y., & Chazan, D. (2007). A large margin algorithm for speech-to-phoneme and music-to-score alignment. *IEEE Transactions on Audio, Speech, and Language Processing, 15,* 2373–2382. http://dx.doi.org/10.1109/TASL.2007.903928

King, G., & Zeng, L. (2001). Logistic regression in rare events data. *Political Analysis, 9,* 137–163. http://dx.doi.org/10.1093/oxfordjournals.pan.a004868

Laver, G. D., & Burke, D. M. (1993). Why do semantic priming effects increase in old age? A meta-analysis. *Psychology and Aging, 8,* 34–43. http://dx.doi.org/10.1037/0882-7974.8.1.34

Levelt, W. J., Roelofs, A., & Meyer, A. S. (1999). A theory of lexical access in speech production. *Behavioral and Brain Sciences, 22,* 1–38. http://dx.doi.org/10.1017/S0140525X99001776

Li, L., & Slevc, L. R. (2017). Of papers and pens: Polysemes and homophones in lexical (mis)Selection. *Cognitive Science, 41*(Suppl 6), 1532–1548. http://dx.doi.org/10.1111/cogs.12402

Löfqvist, A., Sahlén, B., & Ibertsson, T. (2010). Vowel spaces in Swedish adolescents with cochlear implants. *The Journal of the Acoustical Society of America, 128,* 3064–3069.

Logan, G. D., & Zbrodoff, N. J. (1998). Stroop-type interference: Congruity effects in color naming with typewritten responses. *Journal of Experimental Psychology: Human Perception and Performance, 24,* 978–992. http://dx.doi.org/10.1037/0096-1523.24.3.978

MacKay, D. G. (1987). *The organization of perception and action: A theory for language and other cognitive skills*. Berlin, Germany: Springer.

Mainz, N., Shao, Z., Brysbaert, M., & Meyer, A. S. (2017). Vocabulary knowledge predicts lexical processing: Evidence from a group of participants with diverse educational backgrounds. *Frontiers in Psychology, 8,* 1164. http://dx.doi.org/10.3389/fpsyg.2017.01164

Marian, V., Bartolotti, J., Chabal, S., & Shook, A. (2012). CLEARPOND: Cross-linguistic easy-access resource for phonological and orthographic neighborhood densities. *PLoS ONE, 7,* e43230. http://dx.doi.org/10.1371/journal.pone.0043230

McAllester, D., Hazan, T., & Keshet, J. (2010). Direct loss minimization for structured prediction. In J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, & A. Culotta (Eds.), *Advances in neural information processing systems* (Vol. 23, pp. 1594–1602). Vancouver, British Columbia, Canada: NIPS.

McMillan, C. T., & Corley, M. (2010). Cascading influences on the production of speech: Evidence from articulation. *Cognition, 117,* 243–260. http://dx.doi.org/10.1016/j.cognition.2010.08.019

McMillan, C. T., Corley, M., & Lickley, R. J. (2009). Articulatory evidence for feedback and competition in speech production. *Language and Cognitive Processes, 24,* 44–66. http://dx.doi.org/10.1080/01690960801998236

Melinger, A., Branigan, H. P., & Pickering, M. J. (2014). Parallel processing in language production. *Language, Cognition and Neuroscience, 29,* 663–683. http://dx.doi.org/10.1080/23273798.2014.906635

Munson, B. (2007). Lexical access, lexical representation, and vowel production. In J. Cole & J. I. Hualde (Eds.), *Papers in laboratory phonology 9* (pp. 201–228). Berlin, Germany: Mouton de Gruyter.

Oppenheim, G. M., Dell, G. S., & Schwartz, M. F. (2010). The dark side of incremental learning: A model of cumulative semantic interference during lexical access in speech production. *Cognition, 114,* 227–252. http://dx.doi.org/10.1016/j.cognition.2009.09.007

Peterson, R. R., & Savoy, P. (1998). Lexical selection and phonological encoding during language production: Evidence for cascaded processing. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 24,* 539–557. http://dx.doi.org/10.1037/0278-7393.24.3.539

Pouplier, M. (2007). Tongue kinematics during utterances elicited with the SLIP technique. *Language and Speech, 50,* 311–341. http://dx.doi.org/10.1177/00238309070500030201

Pouplier, M. (2008). The role of a coda consonant as error trigger in repetition tasks. *Journal of Phonetics, 36,* 114–140. http://dx.doi.org/10.1016/j.wocn.2007.01.002

Ramscar, M., Hendrix, P., Shaoul, C., Milin, P., & Baayen, H. (2014). The myth of cognitive decline: Non-linear dynamics of lifelong learning.

This document is copyrighted by the American Psychological Association or one of its allied publishers.
This article is intended solely for the personal use of the individual user and is not to be disseminated broadly.

Topics in Cognitive Science, 6, 5–42. http://dx.doi.org/10.1111/tops.12078

Rapp, B., & Goldrick, M. (2000). Discreteness and interactivity in spoken word production. *Psychological Review, 107,* 460–499. http://dx.doi.org/10.1037/0033-295X.107.3.460

Roelofs, A. (2014). Integrating psycholinguistic and motor control approaches to speech production: Where do they meet? *Language, Cognition and Neuroscience, 29,* 35–37. http://dx.doi.org/10.1080/01690965.2013.852687

Scaltritti, M., Arfé, B., Torrance, M., & Peressotti, F. (2016). Typing pictures: Linguistic processing cascades into finger movements. *Cognition, 156,* 16–29. http://dx.doi.org/10.1016/j.cognition.2016.07.006

Scaltritti, M., Peressotti, F., & Navarrete, E. (2017). A joint investigation of semantic facilitation and semantic interference in continuous naming. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 43,* 818–823. http://dx.doi.org/10.1037/xlm0000354

Scaltritti, M., Pinet, S., Longcamp, M., & Alario, F. X. (2017). On the functional relationship between language and motor processing in typewriting: An EEG study. *Language, Cognition and Neuroscience, 32,* 1086–1101. http://dx.doi.org/10.1080/23273798.2017.1283427

Schriefers, H., Meyer, A. S., & Levelt, W. J. (1990). Exploring the time course of lexical access in language production: Picture-word interference studies. *Journal of Memory and Language, 29,* 86–102. http://dx.doi.org/10.1016/0749-596X(90)90011-N

Severens, E., Ratinckx, E., Ferreira, V. S., & Hartsuiker, R. J. (2008). Are phonological influences on lexical (mis)selection the result of a monitoring bias? *The Quarterly Journal of Experimental Psychology, 61,* 1687–1709. http://dx.doi.org/10.1080/17470210701647422

Seyfarth, S. (2014). Word informativity influences acoustic duration: Effects of contextual predictability on lexical representation. *Cognition, 133,* 140–155. http://dx.doi.org/10.1016/j.cognition.2014.06.013

Shipley, W. C., Gruber, C. P., Martin, T. A., & Klein, A. M. (2009). *Shipley-2 manual*. Los Angeles, CA: Western Psychological Services.

Strijkers, K., & Costa, A. (2016). On words and brains: Linking psycholinguistics with neural dynamics in speech production. *Language, Cognition and Neuroscience, 31,* 524–535. http://dx.doi.org/10.1080/23273798.2016.1158845

Taylor, J. K., & Burke, D. M. (2002). Asymmetric aging effects on semantic and phonological processes: Naming in the picture-word interference task. *Psychology and Aging, 17,* 662–676. http://dx.doi.org/10.1037/0882-7974.17.4.662

van Turennout, M., Hagoort, P., & Brown, C. M. (1997). Electrophysiological evidence on the time course of semantic and phonological processes in speech production. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 23,* 787–806. http://dx.doi.org/10.1037/0278-7393.23.4.787

Yuen, I., Davis, M. H., Brysbaert, M., & Rastle, K. (2010). Activation of articulatory information in speech perception. *Proceedings of the National Academy of Sciences of the United States of America, 107,* 592–597. http://dx.doi.org/10.1073/pnas.0904774107

Zacks, R. T., & Hasher, L. (1994). Directed ignoring: Inhibitory regulation of working memory. In D. Dagenbach & T. H. Carr (Eds.), *Inhibitory processes in attention, memory, and language* (pp. 241–264). San Diego, CA: Academic Press.

This document is copyrighted by the American Psychological Association or one of its allied publishers.
This article is intended solely for the personal use of the individual user and is not to be disseminated broadly.

Appendix A

Stimulus Norming Procedures

Separate groups of participants were recruited to norm the picture and sentence materials. To ensure high name agreement among the selected pictures, a large pool of images were first normed for name agreement using Mechanical Turk (mTurk). mTurk workers (n = 20) were native English speakers between the ages of 19 and 60. During norming, participants were shown a picture one at a time and asked to provide a single word that best described the picture. Another group of mTurk workers normed sentences for cloze probability. mTurk workers (n = 19) were native English speakers between the age of 19 and 60. During sentence norming, participants were shown pictures and asked to provide a single word that was the best completion for the sentence. Sentence norming data were used in a similar manner to the data trimming procedure described by Li and Slevc (2017). Cloze norms were inspected prior to the analysis of data from the main task to identify items which over 45% of participants responded with an incorrect response. In the current study, we attempted to exclude incorrect responses in which the participant provided the competitor as a completion to the sentence. These responses would introduce a confound. Our inspection revealed that no items met these criteria and no sentences were removed.

(*Appendices continue*)

Appendix B

Items

Table B1 presents items used in the experiment. Items consist of a picture and a sentence context. There are 180 unique pictures across the items; each picture is part of a semantically related pair (e.g., *anchor* and *rope*). These items were split into six lists of 180 items, with a subset of 60 unique pictures in each list. Pictures were evenly distributed across these lists, and sentence contexts were not repeated within a list.

Within a list, each picture was repeated three times, once in each condition. Match and semantic competitor sentence contexts for each picture were designed to prime that specific picture or its semantic pair, respectively. Unrelated contexts within a list were selected from match sentences for pictures not presented in that list. Across lists, a picture could be paired with one of two unrelated sentence contexts; in the table below, only one of the two appears.

Each participant saw one list of 180 items split across three blocks, with each picture occurring once per block. The order of conditions across blocks was counterbalanced. The assignment of lists was distributed such that picture/sentence pairings were evenly split across participants.

This document is copyrighted by the American Psychological Association or one of its allied publishers.
This article is intended solely for the personal use of the individual user and is not to be disseminated broadly.

Table B1
List of Items

Picture	Condition	Sentence	Completion	Cloze (younger)	Cloze (older)
anchor	match	The sailor stopped the ship and dropped the . . .	anchor	.842	.667
anchor	semantic	They tied the ship to the dock with an . . .	rope	.842	.667
anchor	unrelated	Her favorite dish is macaroni and . . .	cheese	.842	.667
ant	match	The colony worked together to build a hill for the queen . . .	ant	.789	.917
ant	semantic	Honey is produced by an insect called a . . .	bee	.789	.917
ant	unrelated	Her favorite dish is macaroni and . . .	cheese	.789	.917
apple	match	Snow White was poisoned when she bit into an . . .	apple	1.000	1.000
apple	semantic	Her favorite dish is macaroni and . . .	cheese	1.000	1.000
apple	unrelated	At the ballpark, the boys enjoyed a bag of salty . . .	peanuts	1.000	1.000
axe	match	He chopped down the tall pine with an . . .	axe	1.000	.583
axe	semantic	He put the bookshelf together not with a screwdriver, but with a . . .	hammer	1.000	.583
axe	unrelated	The colony worked together to build a hill for the queen . . .	ant	1.000	.583
baby	match	The parents bought a stroller for their newborn . . .	baby	.947	.833
baby	semantic	The newborn sleeps peacefully in her . . .	crib	.947	.833
baby	unrelated	He kicked the ball with his left . . .	foot	.947	.833
backpack	match	The student took her books home in her . . .	backpack	.632	.750
backpack	semantic	After home room, she dropped some books off in her . . .	locker	.632	.750
backpack	unrelated	The salesman was as sly as a . . .	fox	.632	.750
bagel	match	Angela likes cream cheese on a . . .	bagel	.947	.917
bagel	semantic	John spread butter and grape jelly on his morning . . .	toast	.947	.917
bagel	unrelated	Sandy watered her garden using a rubber . . .	hose	.947	.917
balloon	match	The clown blew up a big green . . .	balloon	1.000	1.000
balloon	semantic	It was windy enough to fly a . . .	kite	1.000	1.000
balloon	unrelated	The man happily sat down in the comfortable . . .	chair	1.000	1.000
basket	match	He collected Easter eggs in his . . .	basket	1.000	1.000
basket	semantic	The shipment arrived in a large cardboard . . .	box	1.000	1.000
basket	unrelated	When I eat in Maine, I always order a big red . . .	lobster	1.000	1.000
bat	match	The blind, flying rodent that lives in caves is the . . .	bat	.895	.833
bat	semantic	Matt grabbed the swatter to kill the . . .	fly	.895	.833
bat	unrelated	The stagecoach wasn't moving because of the broken wagon . . .	wheel	.895	.833
beach	match	Mary went to Hawaii to tan on a sandy . . .	beach	1.000	1.000
beach	semantic	She likes to swim laps at the . . .	pool	1.000	1.000
beach	unrelated	At the petting zoo Suzie's snack was stolen by a pesky billy . . .	goat	1.000	1.000
beans	match	Mexican food comes with a side of rice and . . .	beans	.895	1.000
beans	semantic	Caesar salad is made with romaine . . .	lettuce	.895	1.000
beans	unrelated	The message was broadcast to the students over a loud . . .	speaker	.895	1.000
bear	match	The campers were frightened by a large grizzly . . .	bear	.947	.750
bear	semantic	The nocturnal animal that looks like a masked robber is a . . .	raccoon	.947	.750

(*Appendices continue*)

Table B1 (*continued*)

Picture	Condition	Sentence	Completion	Cloze (younger)	Cloze (older)
bear	unrelated	The man brushed his hair using a fine-toothed . . .	comb	.947	.750
beaver	match	The rodent famous for building dams in rivers is called a . . .	beaver	1.000	.917
beaver	semantic	He laid a trap with cheese to catch the . . .	mouse	1.000	.917
beaver	unrelated	Others won't hear your music when you listen to it wearing . . .	headphones	1.000	.917
bed	match	Bob sleeps in a king-sized . . .	bed	1.000	.580
bed	semantic	He lies his head down to sleep on his . . .	pillow	1.000	.580
bed	unrelated	To garnish the margarita, the bartender sliced a wedge from a sour green . . .	lime	1.000	.580
bee	match	Honey is produced by an insect called a . . .	bee	1.000	.250
bee	semantic	The colony worked together to build a hill for the queen . . .	ant	1.000	.250
bee	unrelated	A Vespa is a type of motorized . . .	scooter	1.000	.250
bench	match	The homeless man slept on a park . . .	bench	1.000	.250
bench	semantic	The family sat in their living room on a big comfy . . .	couch	1.000	.250
bench	unrelated	His mouth puckered when he ate the sour yellow . . .	lemon	1.000	.250
boat	match	He grabbed the oars and got in the row . . .	boat	.842	.417
boat	semantic	They paddled down the river in a wooden . . .	canoe	.842	.417
boat	unrelated	She needed new laces for just one . . .	shoe	.842	.417
bomb	match	The explosion came from a homemade pipe . . .	bomb	.895	.947
bomb	semantic	The soldiers were protected by the armored fighting vehicle called a . . .	tank	.895	.947
bomb	unrelated	Alice ended up in Wonderland when she followed the . . .	rabbit	.895	.947
boot	match	Her foot was cold even though she wore a sock and a winter . . .	boot	.684	.833
boot	semantic	Each year his grandmother knit him an ugly Christmas . . .	sweater	.684	.833
boot	unrelated	For his birthday, his mom baked him a chocolate . . .	cake	.684	.833
box	match	The shipment arrived in a large cardboard . . .	box	1.000	1.000
box	semantic	He collected Easter eggs in his . . .	basket	1.000	1.000
box	unrelated	She felt hot in the office and plugged in the . . .	fan	1.000	1.000
bra	match	Under their shirts, most women wear a supportive . . .	bra	1.000	.500
bra	semantic	Between his jacket and shirt, the usher had buttoned his . . .	vest	1.000	.500
bra	unrelated	The chef chopped the vegetables with a . . .	knife	1.000	.500
broccoli	match	You can eat the green stalk and flowering head of . . .	broccoli	.526	1.000
broccoli	semantic	Bugs Bunny chewed on a . . .	carrot	.526	1.000
broccoli	unrelated	Because it was out of fluid, there was no flame but only sparks from the . . .	lighter	.526	1.000
broom	match	The boy swept up his mess with the . . .	broom	.947	.833
broom	semantic	Mary cleaned the floor using a bucket and . . .	mop	.947	.833
broom	unrelated	The activists scolded the poacher for clubbing a baby . . .	seal	.947	.833
cactus	match	In the desert, he pricked his finger on the spine of a . . .	cactus	1.000	1.000
cactus	semantic	The bird built a nest high up in an elm . . .	tree	1.000	1.000
cactus	unrelated	The joint connecting the thigh and shin is the . . .	knee	1.000	1.000
cake	match	For his birthday, his mom baked him a chocolate . . .	cake	1.000	.917
cake	semantic	My favorite treat is a chocolate chip . . .	cookie	1.000	.917
cake	unrelated	She needed new laces for just one . . .	shoe	1.000	.917
candle	match	When the power went out, they lit a . . .	candle	.842	1.000
candle	semantic	She placed the flowers in a glass . . .	vase	.842	1.000
candle	unrelated	The shipment arrived in a large cardboard . . .	box	.842	1.000
candy	match	On Halloween kids trick-or-treat to collect . . .	candy	.947	.750
candy	semantic	At the ballpark, the boys enjoyed a bag of salty . . .	peanuts	.947	.750
candy	unrelated	The sailor stopped the ship and dropped the . . .	anchor	.947	.750
canoe	match	They paddled down the river in a wooden . . .	canoe	.737	1.000
canoe	semantic	He grabbed the oars and got in the row . . .	boat	.737	1.000
canoe	unrelated	The footwear typically worn in the summer is a . . .	sandal	.737	1.000
carrot	match	Bugs Bunny chewed on a . . .	carrot	1.000	.500
carrot	semantic	You can eat the green stalk and flowering head of . . .	broccoli	1.000	.500
carrot	unrelated	The shipment arrived in a large cardboard . . .	box	1.000	.500
castle	match	The fairy tale princess lived in a majestic . . .	castle	1.000	.917
castle	semantic	Every Halloween, they turned their home into a haunted . . .	house	1.000	.917
castle	unrelated	Chocolate glazed with sprinkles is his favorite kind of . . .	donut	1.000	.917
cat	match	Susan hates dogs, but loves Garfield, her fat tabby . . .	cat	1.000	1.000
cat	semantic	The animal bacon and ham come from is the . . .	pig	1.000	1.000
cat	unrelated	He chopped down the tall pine with an . . .	axe	1.000	1.000
chair	match	The man happily sat down in the comfortable . . .	chair	.895	.833

(*Appendices continue*)

This document is copyrighted by the American Psychological Association or one of its allied publishers.
This article is intended solely for the personal use of the individual user and is not to be disseminated broadly.

Table B1 (*continued*)

Picture	Condition	Sentence	Completion	Cloze (younger)	Cloze (older)
chair	semantic	I like to sit at the bar on a tall wooden . . .	stool	.895	.833
chair	unrelated	The clown blew up a big green . . .	balloon	.895	.833
cheese	match	Her favorite dish is macaroni and . . .	cheese	1.000	.917
cheese	semantic	Snow White was poisoned when she bit into an . . .	apple	1.000	.917
cheese	unrelated	During their bike tour Mike had to fix a flat . . .	tire	1.000	.917
cheetah	match	The wild cat that runs the fastest is the . . .	cheetah	.895	1.000
cheetah	semantic	The king of the jungle is the . . .	lion	.895	1.000
cheetah	unrelated	Levi makes high quality denim blue . . .	jeans	.895	1.000
chips	match	At the party, we had some salsa and tortilla . . .	chips	.947	1.000
chips	semantic	Before the movie started everyone bought some buttery . . .	popcorn	.947	1.000
chips	unrelated	Nick sneezed and blew his . . .	nose	.947	1.000
clock	match	He keeps track of time by looking to the wall at the . . .	clock	.947	.833
clock	semantic	The ornate shade covered the bulb of the . . .	lamp	.947	.833
clock	unrelated	The smallest bank note is one . . .	dollar	.947	.833
comb	match	The man brushed his hair using a fine-toothed . . .	comb	.947	.417
comb	semantic	Proper dental hygiene includes cleaning each day with a bristled . . .	toothbrush	.947	.417
comb	unrelated	The campers were frightened by a large grizzly . . .	bear	.947	.417
compass	match	The navigation device that points north is a . . .	compass	.947	.917
compass	semantic	The lead broke so she sharpened the . . .	pencil	.947	.917
compass	unrelated	She placed the flowers in a glass . . .	vase	.947	.917
cookie	match	My favorite treat is a chocolate chip . . .	cookie	.947	.917
cookie	semantic	For his birthday, his mom baked him a chocolate . . .	cake	.947	.917
cookie	unrelated	They paddled down the river in a wooden . . .	canoe	.947	.917
corn	match	The vegetable that comes on a cob is . . .	corn	.947	.917
corn	semantic	The vegetables that come in pods are . . .	peas	.947	.917
corn	unrelated	The bike was protected from theft by an expensive . . .	lock	.947	.917
couch	match	The family sat in their living room on a big comfy . . .	couch	.579	1.000
couch	semantic	The homeless man slept on a park . . .	bench	.579	1.000
couch	unrelated	To garnish the margarita, the bartender sliced a wedge from a sour green . . .	lime	.579	1.000
cow	match	Every day the farmer goes to milk his only . . .	cow	1.000	.917
cow	semantic	The farmer woke up to the cock-a-doodle-doo of the . . .	rooster	1.000	.917
cow	unrelated	For his interview Mr. Jones needed a new . . .	suit	1.000	.917
crib	match	The newborn sleeps peacefully in her . . .	crib	.895	.833
crib	semantic	The parents bought a stroller for their newborn . . .	baby	.895	.833
crib	unrelated	He wished the actor luck by saying break a . . .	leg	.895	.833
desk	match	During work Danny sits all day long at his . . .	desk	.842	.947
desk	semantic	For dinner, the family gathers at the dining room . . .	table	.842	.947
desk	unrelated	She likes to swim laps at the . . .	pool	.842	.947
dollar	match	The smallest bank note is one . . .	dollar	.895	1.000
dollar	semantic	A one cent coin is called a . . .	penny	.895	1.000
dollar	unrelated	He keeps track of time by looking to the wall at the . . .	clock	.895	1.000
dolphin	match	The fisherman came upon a pod with a baby bottlenose . . .	dolphin	.684	.750
dolphin	semantic	The activists scolded the poacher for clubbing a baby . . .	seal	.684	.750
dolphin	unrelated	Mary cleaned the floor using a bucket and . . .	mop	.684	.750
donut	match	Chocolate glazed with sprinkles is his favorite kind of . . .	donut	.684	.917
donut	semantic	At brunch, Maggie either eats pancakes or a Belgian . . .	waffle	.684	.917
donut	unrelated	She likes to swim laps at the . . .	pool	.684	.917
dropper	match	He applied the medicine to his eye using a . . .	dropper	.632	.833
dropper	semantic	Andy removed a splinter with some . . .	tweezers	.632	.833
dropper	unrelated	The Canadian flag features a maple . . .	leaf	.632	.833
eagle	match	The national bird of the United States is the . . .	eagle	.947	.917
eagle	semantic	Every ugly duckling eventually becomes a beautiful . . .	swan	.947	.917
eagle	unrelated	He melted cheese over tortilla chips to make . . .	nachos	.947	.917
elbow	match	The joint connecting the forearm to the bicep is the . . .	elbow	.947	.833
elbow	semantic	The joint connecting the thigh and shin is the . . .	knee	.947	.833
elbow	unrelated	He collected Easter eggs in his . . .	basket	.947	.833
fan	match	She felt hot in the office and plugged in the . . .	fan	.947	1.000
fan	semantic	To let some cool air in the apartment they opened a . . .	window	.947	1.000
fan	unrelated	He chopped down the tall pine with an . . .	axe	.947	1.000
feather	match	The sense of relief made him feel as light as a . . .	feather	1.000	1.000

(*Appendices continue*)

This document is copyrighted by the American Psychological Association or one of its allied publishers.
This article is intended solely for the personal use of the individual user and is not to be disseminated broadly.

Table B1 (*continued*)

Picture	Condition	Sentence	Completion	Cloze (younger)	Cloze (older)
feather	semantic	She went to the salon to color her . . .	hair	1.000	1.000
feather	unrelated	The boy took a pole to the lake to catch a . . .	fish	1.000	1.000
fish	match	The boy took a pole to the lake to catch a . . .	fish	1.000	.917
fish	semantic	When I eat in Maine, I always order a big red . . .	lobster	1.000	.917
fish	unrelated	To fix his torn paper he needs some . . .	tape	1.000	.917
floss	match	After brushing his teeth, . . .also uses dental . . .	floss	.947	.833
floss	semantic	Every morning the man shaves using a . . .	razor	.947	.833
floss	unrelated	The animal with a long neck and long legs is a . . .	giraffe	.947	.833
fly	match	Matt grabbed the swatter to kill the . . .	fly	1.000	1.000
fly	semantic	The blinding rodent that lives in caves is the . . .	bat	1.000	1.000
fly	unrelated	On Halloween kids trick-or-treat to collect . . .	candy	1.000	1.000
foot	match	He kicked the ball with his left . . .	foot	.947	1.000
foot	semantic	He wished the actor luck by saying break a . . .	leg	.947	1.000
foot	unrelated	He found a pot of gold at the end of the . . .	rainbow	.947	1.000
fox	match	The salesman was as sly as a . . .	fox	1.000	.583
fox	semantic	Alice ended up in Wonderland when she followed the . . .	rabbit	1.000	.583
fox	unrelated	The student took her books home in her . . .	backpack	1.000	.583
giraffe	match	The animal with a long neck and long legs is a . . .	giraffe	1.000	1.000
giraffe	semantic	The horse-like animal with black and white stripes is the . . .	zebra	1.000	1.000
giraffe	unrelated	The newborn sleeps peacefully in her . . .	crib	1.000	1.000
glasses	match	She is as blind as a bat without her . . .	glasses	.947	1.000
glasses	semantic	Swimmers protect their eyes by wearing . . .	goggles	.947	1.000
glasses	unrelated	You can eat the green stalk and flowering head of . . .	broccoli	.947	1.000
globe	match	The spherical object showing the entire world is a . . .	globe	.947	1.000
globe	semantic	The directions did not match any roads on the . . .	map	.947	1.000
globe	unrelated	In the desert, he pricked his finger on the spine of a . . .	cactus	.947	1.000
glue	match	Emily fixed the broken mug with some . . .	glue	.947	.750
glue	semantic	She cut the paper using . . .	scissors	.947	.750
glue	unrelated	Peter serves the soup out of the pot with a . . .	ladle	.947	.750
goat	match	at the petting zoo Suzie's snack was stolen by a pesky billy . . .	goat	.895	.917
goat	semantic	The farmer shaved the wool off of the . . .	sheep	.895	.917
goat	unrelated	Mary went to Hawaii to tan on a sandy . . .	beach	.895	.917
goggles	match	Swimmers protect their eyes by wearing. . . .	goggles	1.000	1.000
goggles	semantic	She is as blind as a bat without her. . . .	glasses	1.000	1.000
goggles	unrelated	Bugs Bunny chewed on. . . .	carrot	1.000	1.000
goose	match	The children loved to play Duck-Duck . . .	goose	.947	.833
goose	semantic	The bird that looks like it's wearing a tuxedo is a . . .	penguin	.947	.833
goose	unrelated	after brushing his teeth, Mike also uses dental . . .	floss	.947	.833
hair	match	She went to the salon to color her . . .	hair	1.000	.833
hair	semantic	The sense of relief made him feel as light as a . . .	feather	1.000	.833
hair	unrelated	Sam measured the length of the paper using a . . .	ruler	1.000	.833
hammer	match	He put the bookshelf together not with a screwdriver, but with a . . .	hammer	.684	.917
hammer	semantic	He chopped down the tall pine with an . . .	axe	.684	.917
hammer	unrelated	Honey is produced by an insect called a . . .	bee	.684	.917
hat	match	To protect his head from sunburn, the bald man wore a wide-brimmed . . .	hat	1.000	1.000
hat	semantic	A + B224fter doing laundry Derek noticed he was missing just one . . .	sock	1.000	1.000
hat	unrelated	Andy removed a splinter with some . . .	tweezers	1.000	1.000
headphones	match	Others won't hear your music when you listen to it wearing . . .	headphones	.947	.750
headphones	semantic	The message was broadcast to the students over a loud . . .	speaker	.947	.750
headphones	unrelated	The rodent famous for building dams in rivers is called a . . .	beaver	.947	.750
hose	match	Sandy watered her garden using a rubber . . .	hose	.895	1.000
hose	semantic	The gardener collects the leaves in a pile using a . . .	rake	.895	1.000
hose	unrelated	Angela likes cream cheese on a . . .	bagel	.895	1.000
house	match	Every Halloween, they turned their home into a haunted . . .	house	1.000	.917
house	semantic	The fairy tale princess lived in a majestic . . .	castle	1.000	.917
house	unrelated	She wrote the list on a piece of . . .	paper	1.000	.917
ipod	match	Apple's mp3 player is called an . . .	ipod	.895	.833
ipod	semantic	Before the CD existed, music was listened to off of a black vinyl . . .	record	.895	.833
ipod	unrelated	The old girlfriends chatted over a bottle of red . . .	wine	.895	.833
jeans	match	Levi makes high quality denim blue . . .	jeans	1.000	.917

This document is copyrighted by the American Psychological Association or one of its allied publishers.
This article is intended solely for the personal use of the individual user and is not to be disseminated broadly.

(*Appendices continue*)

Table B1 (*continued*)

Picture	Condition	Sentence	Completion	Cloze (younger)	Cloze (older)
jeans	semantic	He wore a suit with a Windsor knot in his . . .	tie	1.000	.917
jeans	unrelated	The wild cat that runs the fastest is the . . .	cheetah	1.000	.917
jeep	match	Many people like the Grand Cherokee, but the Wrangler is my favorite kind of . . .	jeep	.684	.894
jeep	semantic	We couldn't get a truck, but we managed to pack everything in a moving . . .	van	.684	.894
jeep	unrelated	Before the movie started everyone bought some buttery . . .	popcorn	.684	.894
juice	match	With breakfast Julie always drinks some orange . . .	juice	1.000	1.000
juice	semantic	The old girlfriends chatted over a bottle of red . . .	wine	1.000	1.000
juice	unrelated	During work Danny sits all day long at his . . .	desk	1.000	1.000
key	match	To start a car, you need the . . .	key	.947	.833
key	semantic	The bike was protected from theft by an expensive . . .	lock	.947	.833
key	unrelated	In the desert, he pricked his finger on the spine of a . . .	cactus	.947	.833
kite	match	It was windy enough to fly a . . .	kite	1.000	.917
kite	semantic	The clown blew up a big green . . .	balloon	1.000	.917
kite	unrelated	The animal with antlers that is much larger than a deer is a . . .	moose	1.000	.917
knee	match	The joint connecting the thigh and shin is the . . .	knee	.789	.917
knee	semantic	The joint connecting the forearm to the bicep is the . . .	elbow	.789	.917
knee	unrelated	The fairy tale princess lived in a majestic . . .	castle	.789	.917
knife	match	The chef chopped the vegetables with a . . .	knife	1.000	.833
knife	semantic	She heated the stew in a large metal . . .	pot	1.000	.833
knife	unrelated	Under their shirts, most women wear a supportive . . .	bra	1.000	.833
ladle	match	Peter serves the soup out of the pot with a . . .	ladle	.789	.583
ladle	semantic	Cooks remove skins from vegetables using a . . .	peeler	.789	.583
ladle	unrelated	He garnished the martini with a green . . .	olive	.789	.583
lamp	match	The ornate shade covered the bulb of the . . .	lamp	.737	1.000
lamp	semantic	He keeps track of time by looking to the wall at the . . .	clock	.737	1.000
lamp	unrelated	A portable computer is called a . . .	laptop	.737	1.000
laptop	match	A portable computer is called a . . .	laptop	1.000	.750
laptop	semantic	He hooked up his computer and discovered the ink had run dry in the . . .	printer	1.000	.750
laptop	unrelated	The ornate shade covered the bulb of the . . .	lamp	1.000	.750
leaf	match	The Canadian flag features a maple . . .	leaf	1.000	1.000
leaf	semantic	This frozen turkey is as hard as a . . .	rock	1.000	1.000
leaf	unrelated	He applied the medicine to his eye using a . . .	dropper	1.000	1.000
leg	match	He wished the actor luck by saying break a . . .	leg	1.000	1.000
leg	semantic	He kicked the ball with his left . . .	foot	1.000	1.000
leg	unrelated	The blind, flying rodent that lives in caves is the . . .	bat	1.000	1.000
lemon	match	His mouth puckered when he ate the sour yellow . . .	lemon	.737	.830
lemon	semantic	To garnish the margarita, the bartender sliced a wedge from a sour green . . .	lime	.737	.830
lemon	unrelated	The homeless man slept on a park . . .	bench	.737	.830
lettuce	match	Caesar salad is made with romaine . . .	lettuce	1.000	.917
lettuce	semantic	Mexican food comes with a side of rice and . . .	beans	1.000	.917
lettuce	unrelated	The clown blew up a big green . . .	balloon	1.000	.917
lighter	match	Because it was out of fluid, there was no flame but only sparks from the . . .	lighter	.526	.940
lighter	semantic	She opened the fireplace and lit the kindling with a wooden . . .	match	.526	.940
lighter	unrelated	You can eat the green stalk and flowering head of . . .	broccoli	.526	.940
lightning	match	Kids are often frightened by thunder and . . .	lightning	.947	.917
lightning	semantic	He found a pot of gold at the end of the . . .	rainbow	.947	.917
lightning	unrelated	The newborn sleeps peacefully in her . . .	crib	.947	.917
lime	match	To garnish the margarita, the bartender sliced a wedge from a sour green . . .	lime	.947	.833
lime	semantic	His mouth puckered when he ate the sour yellow . . .	lemon	.947	.833
lime	unrelated	Bob sleeps in a king-sized . . .	bed	.947	.833
lion	match	The king of the jungle is the . . .	lion	.737	.417
lion	semantic	The wild cat that runs the fastest is the . . .	cheetah	.737	.417
lion	unrelated	He dried his wet hands with a . . .	towel	.737	.417
lips	match	In the winter, she uses lots of chapstick on her . . .	lips	1.000	1.000
lips	semantic	Nick sneezed and blew his . . .	nose	1.000	1.000
lips	unrelated	Many people like the Grand Cherokee, but the Wrangler is my favorite kind of . . .	jeep	1.000	1.000
lobster	match	When I eat in Maine, I always order a big red . . .	lobster	.789	1.000
lobster	semantic	The boy took a pole to the lake to catch a . . .	fish	.789	1.000
lobster	unrelated	In the Olympic games, she won a gold . . .	medal	.789	1.000
lock	match	The bike was protected from theft by an expensive . . .	lock	.737	.833

(*Appendices continue*)

This document is copyrighted by the American Psychological Association or one of its allied publishers.
This article is intended solely for the personal use of the individual user and is not to be disseminated broadly.

Table B1 (*continued*)

Picture	Condition	Sentence	Completion	Cloze (younger)	Cloze (older)
lock	semantic	To start a car, you need the . . .	key	.737	.833
lock	unrelated	The vegetable that comes on a cob is . . .	corn	.737	.833
locker	match	After home room, she dropped some books off in her . . .	locker	.947	.667
locker	semantic	The student took her books home in her . . .	backpack	.947	.667
locker	unrelated	The explosion came from a homemade pipe . . .	bomb	.947	.667
map	match	The directions did not match any roads on the . . .	map	.947	.833
map	semantic	The spherical object showing the entire world is a . . .	globe	.947	.833
map	unrelated	Every Halloween, they turned their home into a haunted . . .	house	.947	.833
match	match	She opened the fireplace and lit the kindling with a wooden . . .	match	.947	.667
match	semantic	Because it was out of fluid, there was no flame but only sparks from the . . .	lighter	.947	.667
match	unrelated	Bugs Bunny chewed on a . . .	carrot	.947	.667
medal	match	In the Olympic games, she won a gold . . .	medal	1.000	.667
medal	semantic	The team that won the tournament took home a . . .	trophy	1.000	.667
medal	unrelated	After brushing his teeth, Mike also uses dental . . .	floss	1.000	.667
moon	match	Neil Armstrong was the first man to walk on the . . .	moon	1.000	1.000
moon	semantic	The hopeful girl wished upon a . . .	star	1.000	1.000
moon	unrelated	Every ugly duckling eventually becomes a beautiful . . .	swan	1.000	1.000
moose	match	The animal with antlers that is much larger than a deer is a . . .	moose	.579	1.000
moose	semantic	In the desert, he got bitten by a rattle . . .	snake	.579	1.000
moose	unrelated	Raymond needed a belt to hold up his . . .	pants	.579	1.000
mop	match	Mary cleaned the floor using a bucket and . . .	mop	.947	.917
mop	semantic	The boy swept up his mess with the . . .	broom	.947	.917
mop	unrelated	Neil Armstrong was the first man to walk on the . . .	moon	.947	.917
mouse	match	He laid a trap with cheese to catch the . . .	mouse	.947	1.000
mouse	semantic	The rodent famous for building dams in rivers is called a . . .	beaver	.947	1.000
mouse	unrelated	Mexican food comes with a side of rice and . . .	beans	.947	1.000
nachos	match	He melted cheese over tortilla chips to make . . .	nachos	1.000	1.000
nachos	semantic	She snacks on a peanut butter and jelly . . .	sandwich	1.000	1.000
nachos	unrelated	The national bird of the United States is the . . .	eagle	1.000	1.000
nose	match	Nick sneezed and blew his . . .	nose	1.000	1.000
nose	semantic	In the winter, she uses lots of chapstick on her . . .	lips	1.000	1.000
nose	unrelated	The animal bacon and ham come from is the . . .	pig	1.000	1.000
notebook	match	A binder of ruled pages used by students is a . . .	notebook	.842	1.000
notebook	semantic	She wrote the list on a piece of . . .	paper	.842	1.000
notebook	unrelated	Chocolate glazed with sprinkles is his favorite kind of . . .	donut	.842	1.000
olive	match	He garnished the martini with a green . . .	olive	1.000	1.000
olive	semantic	The burger came with a side of chips and a dill . . .	pickle	1.000	1.000
olive	unrelated	At the party, we had some salsa and tortilla . . .	chips	1.000	1.000
owl	match	The bird that says “hoo” is the . . .	owl	.947	1.000
owl	semantic	The bird whose tail feathers make a colorful fan is a . . .	peacock	.947	1.000
owl	unrelated	My favorite treat is a chocolate chip . . .	cookie	.947	1.000
pants	match	Raymond needed a belt to hold up his . . .	pants	1.000	.750
pants	semantic	For his interview Mr. Jones needed a new . . .	suit	1.000	.750
pants	unrelated	Susan hates dogs, but loves Garfield, her fat tabby . . .	cat	1.000	.750
paper	match	She wrote the list on a piece of . . .	paper	1.000	1.000
paper	semantic	A binder of ruled pages used by students is a . . .	notebook	1.000	1.000
paper	unrelated	Every Halloween, they turned their home into a haunted . . .	house	1.000	1.000
pasta	match	Spaghetti and penne are types of . . .	pasta	1.000	.917
pasta	semantic	Nothing helps a cold like a bowl of chicken noodle . . .	soup	1.000	.917
pasta	unrelated	The boy swept up his mess with the . . .	broom	1.000	.917
peacock	match	The bird whose tail feathers make a colorful fan is a . . .	peacock	1.000	1.000
peacock	semantic	The bird that says “hoo” is the . . .	owl	1.000	1.000
peacock	unrelated	He laid a trap with cheese to catch the . . .	mouse	1.000	1.000
peanuts	match	At the ballpark, the boys enjoyed a bag of salty . . .	peanuts	.737	1.000
peanuts	semantic	On Halloween kids trick-or-treat to collect . . .	candy	.737	1.000
peanuts	unrelated	The animal with a long neck and long legs is a . . .	giraffe	.737	1.000
peas	match	The vegetables that come in pods are . . .	peas	.842	1.000
peas	semantic	The vegetable that comes on a cob is . . .	corn	.842	1.000
peas	unrelated	To start a car, you need the . . .	key	.842	1.000
peeler	match	Cooks remove skins from vegetables using a . . .	peeler	.789	.667

(*Appendices continue*)

This document is copyrighted by the American Psychological Association or one of its allied publishers.
This article is intended solely for the personal use of the individual user and is not to be disseminated broadly.

Table B1 (*continued*)

Picture	Condition	Sentence	Completion	Cloze (younger)	Cloze (older)
peeler	semantic	Peter serves the soup out of the pot with a . . .	ladle	.789	.667
peeler	unrelated	Matt grabbed the swatter to kill the . . .	fly	.789	.667
pen	match	The ink ran out in my ballpoint . . .	pen	1.000	1.000
pen	semantic	John joined the pieces of paper together by pushing down hard on the . . .	stapler	1.000	1.000
pen	unrelated	The dog ran in circles chasing his own . . .	tail	1.000	1.000
pencil	match	The lead broke so she sharpened the . . .	pencil	.947	.833
pencil	semantic	The navigation device that points north is a . . .	compass	.947	.833
pencil	unrelated	With breakfast Julie always drinks some orange . . .	juice	.947	.833
penguin	match	The bird that looks like it's wearing a tuxedo is a . . .	penguin	.842	.947
penguin	semantic	The children loved to play Duck-Duck . . .	goose	.842	.947
penguin	unrelated	He kicked the ball with his left . . .	foot	.842	.947
penny	match	A one cent coin is called a . . .	penny	1.000	1.000
penny	semantic	The smallest bank note is one . . .	dollar	1.000	1.000
penny	unrelated	The ornate shade covered the bulb of the . . .	lamp	1.000	1.000
pickle	match	The burger came with a side of chips and a dill . . .	pickle	.947	.833
pickle	semantic	He garnished the martini with a green . . .	olive	.947	.833
pickle	unrelated	Before the movie started everyone bought some buttery . . .	popcorn	.947	.833
pig	match	The animal bacon and ham come from is the . . .	pig	1.000	1.000
pig	semantic	Susan hates dogs, but loves Garfield, her fat tabby . . .	cat	1.000	1.000
pig	unrelated	Honey is produced by an insect called a . . .	bee	1.000	1.000
pillow	match	He lies his head down to sleep on his . . .	pillow	.579	1.000
pillow	semantic	Bob sleeps in a king-sized . . .	bed	.579	1.000
pillow	unrelated	His mouth puckered when he ate the sour yellow . . .	lemon	.579	1.000
pizza	match	Chicago is famous for deep dish . . .	pizza	1.000	.667
pizza	semantic	At the Mexican restaurant, he ordered one hard and one soft . . .	taco	1.000	.667
pizza	unrelated	The joint connecting the thigh and shin is the . . .	knee	1.000	.667
plate	match	She put her salad on a large . . .	plate	.737	1.000
plate	semantic	He ate his cereal in a bowl with a . . .	spoon	.737	1.000
plate	unrelated	The nocturnal animal that looks like a masked robber is a . . .	raccoon	.737	1.000
pool	match	She likes to swim laps at the . . .	pool	.947	1.000
pool	semantic	Mary went to Hawaii to tan on a sandy . . .	beach	.947	1.000
pool	unrelated	The farmer shaved the wool off of the . . .	sheep	.947	1.000
popcorn	match	Before the movie started everyone bought some buttery . . .	popcorn	1.000	1.000
popcorn	semantic	At the party, we had some salsa and tortilla . . .	chips	1.000	1.000
popcorn	unrelated	Many people like the Grand Cherokee, but the Wrangler is my favorite kind of . . .	jeep	1.000	1.000
pot	match	She heated the stew in a large metal . . .	pot	.947	1.000
pot	semantic	The chef chopped the vegetables with a . . .	knife	.947	1.000
pot	unrelated	The lead broke so she sharpened the . . .	pencil	.947	1.000
printer	match	He hooked up his computer and discovered the ink had run dry in the . . .	printer	.842	1.000
printer	semantic	A portable computer is called a . . .	laptop	.842	1.000
printer	unrelated	He keeps track of time by looking to the wall at the . . .	clock	.842	1.000
rabbit	match	Alice ended up in Wonderland when she followed the . . .	rabbit	.842	1.000
rabbit	semantic	The salesman was as sly as a . . .	fox	.842	1.000
rabbit	unrelated	The explosion came from a homemade pipe . . .	bomb	.842	1.000
raccoon	match	The nocturnal animal that looks like a masked robber is a . . .	raccoon	.842	1.000
raccoon	semantic	The campers were frightened by a large grizzly . . .	bear	.842	1.000
raccoon	unrelated	She put her salad on a large . . .	plate	.842	1.000
rainbow	match	He found a pot of gold at the end of the . . .	rainbow	.947	.667
rainbow	semantic	Kids are often frightened by thunder and . . .	lightning	.947	.667
rainbow	unrelated	The parents bought a stroller for their newborn . . .	baby	.947	.667
rake	match	The gardener collects the leaves in a pile using a . . .	rake	.947	1.000
rake	semantic	Sandy watered her garden using a rubber . . .	hose	.947	1.000
rake	unrelated	The directions did not match any roads on the . . .	map	.947	1.000
razor	match	Every morning the man shaves using a . . .	razor	.947	1.000
razor	semantic	After brushing his teeth, Mike also uses dental . . .	floss	.947	1.000
razor	unrelated	The bird that looks like it's wearing a tuxedo is a . . .	penguin	.947	1.000
record	match	Before the CD existed, music was listened to off of a black vinyl . . .	record	.789	.667
record	semantic	Apple's mp3 player is called an . . .	ipod	.789	.667
record	unrelated	During work Danny sits all day long at his . . .	desk	.789	.667

This document is copyrighted by the American Psychological Association or one of its allied publishers.
This article is intended solely for the personal use of the individual user and is not to be disseminated broadly.

(*Appendices continue*)

Table B1 (*continued*)

Picture	Condition	Sentence	Completion	Cloze (younger)	Cloze (older)
ring	match	The man gave his fiancé an engagement . . .	ring	1.000	.750
ring	semantic	A Rolex is an expensive type of . . .	watch	1.000	.750
ring	unrelated	The bike was protected from theft by an expensive . . .	lock	1.000	.750
rock	match	This frozen turkey is as hard as a . . .	rock	.947	1.000
rock	semantic	The Canadian flag features a maple . . .	leaf	.947	1.000
rock	unrelated	To protect his head from sunburn, the bald man wore a wide-brimmed . . .	hat	.947	1.000
rooster	match	The farmer woke up to the cock-a-doodle-doo of the . . .	rooster	.947	1.000
rooster	semantic	Every day the farmer goes to milk his only . . .	cow	.947	1.000
rooster	unrelated	Her foot was cold even though she wore a sock and a winter . . .	boot	.947	1.000
rope	match	They tied the ship to the dock with a . . .	rope	.895	1.000
rope	semantic	The sailor stopped the ship and dropped the . . .	anchor	.895	1.000
rope	unrelated	Snow White was poisoned when she bit into an . . .	apple	.895	1.000
ruler	match	Sam measured the length of the paper using a . . .	ruler	.947	.750
ruler	semantic	To fix his torn paper he needs some . . .	tape	.947	.750
ruler	unrelated	Peter serves the soup out of the pot with a . . .	ladle	.947	.750
sandal	match	The footwear typically worn in the summer is a . . .	sandal	.789	.917
sandal	semantic	She needed new laces for just one . . .	shoe	.789	.917
sandal	unrelated	They paddled down the river in a wooden . . .	canoe	.789	.917
sandwich	match	She snacks on a peanut butter and jelly . . .	sandwich	.947	.417
sandwich	semantic	He melted cheese over tortilla chips to make . . .	nachos	.947	.417
sandwich	unrelated	Swimmers protect their eyes by wearing . . .	goggles	.947	.417
scissors	match	She cut the paper using . . .	scissors	1.000	1.000
scissors	semantic	Emily fixed the broken mug with some . . .	glue	1.000	1.000
scissors	unrelated	It was windy enough to fly a . . .	kite	1.000	1.000
scooter	match	A Vespa is a type of motorized . . .	scooter	.526	.417
scooter	semantic	The boy went to the half pipe and practiced tricks on his . . .	skateboard	.526	.417
scooter	unrelated	John joined the pieces of paper together by pushing down hard on the . . .	stapler	.526	.417
seal	match	The activists scolded the poacher for clubbing a baby . . .	seal	.842	.583
seal	semantic	The fisherman came upon a pod with a baby bottlenose . . .	dolphin	.842	.583
seal	unrelated	The boy swept up his mess with the . . .	broom	.842	.583
sheep	match	The farmer shaved the wool off of the . . .	sheep	.947	1.000
sheep	semantic	At the petting zoo Suzie's snack was stolen by a pesky billy . . .	goat	.947	1.000
sheep	unrelated	The bird whose tail feathers make a colorful fan is a . . .	peacock	.947	1.000
shoe	match	She needed new laces for just one . . .	shoe	.895	.833
shoe	semantic	The footwear typically worn in the summer is a . . .	sandal	.895	.833
shoe	unrelated	For his birthday, his mom baked him a chocolate . . .	cake	.895	.833
skateboard	match	The boy went to the half pipe and practiced tricks on his . . .	skateboard	.947	.250
skateboard	semantic	A Vespa is a type of motorized . . .	scooter	.947	.250
skateboard	unrelated	The ink ran out in my ballpoint . . .	pen	.947	.250
snake	match	In the desert, he got bitten by a rattle . . .	snake	1.000	.750
snake	semantic	The animal with antlers that is much larger than a deer is a . . .	moose	1.000	.750
snake	unrelated	The man happily sat down in the comfortable . . .	chair	1.000	.750
soap	match	In the shower, he washed with a bar of . . .	soap	1.000	1.000
soap	semantic	He dried his wet hands with a . . .	towel	1.000	1.000
soap	unrelated	Levi makes high quality denim blue . . .	jeans	1.000	1.000
sock	match	After doing laundry Derek noticed he was missing just one . . .	sock	.947	1.000
sock	semantic	To protect his head from sunburn, the bald man wore a wide-brimmed . . .	hat	.947	1.000
sock	unrelated	Susan hates dogs, but loves Garfield, her fat tabby . . .	cat	.947	1.000
soup	match	Nothing helps a cold like a bowl of chicken noodle . . .	soup	1.000	1.000
soup	semantic	Spaghetti and penne are types of . . .	pasta	1.000	1.000
soup	unrelated	Mary cleaned the floor using a bucket and . . .	mop	1.000	1.000
speaker	match	The message was broadcast to the students over a loud . . .	speaker	.947	.250
speaker	semantic	Others won't hear your music when you listen to it wearing . . .	headphones	.947	.250
speaker	unrelated	Mexican food comes with a side of rice and . . .	beans	.947	.250
spoon	match	He ate his cereal in a bowl with a . . .	spoon	1.000	.833
spoon	semantic	She put her salad on a large . . .	plate	1.000	.833
spoon	unrelated	The campers were frightened by a large grizzly . . .	bear	1.000	.833

This document is copyrighted by the American Psychological Association or one of its allied publishers.
This article is intended solely for the personal use of the individual user and is not to be disseminated broadly.

(*Appendices continue*)

Table B1 (*continued*)

Picture	Condition	Sentence	Completion	Cloze (younger)	Cloze (older)
stapler	match	John joined the pieces of paper together by pushing down hard on the . . .	stapler	.684	1.000
stapler	semantic	The ink ran out in my ballpoint . . .	pen	.684	1.000
stapler	unrelated	A Vespa is a type of motorized . . .	scooter	.684	1.000
star	match	The hopeful girl wished upon a . . .	star	1.000	1.000
star	semantic	Neil Armstrong was the first man to walk on the . . .	moon	1.000	1.000
star	unrelated	The national bird of the United States is the . . .	eagle	1.000	1.000
stool	match	I like to sit at the bar on a tall wooden . . .	stool	.842	.917
stool	semantic	The man happily sat down in the comfortable . . .	chair	.842	.917
stool	unrelated	It was windy enough to fly a . . .	kite	.842	.917
suit	match	For his interview Mr. Jones needed a new . . .	suit	.632	.750
suit	semantic	Raymond needed a belt to hold up his . . .	pants	.632	.750
suit	unrelated	The wild cat that runs the fastest is the . . .	cheetah	.632	.750
swan	match	Every ugly duckling eventually becomes a beautiful . . .	swan	.895	.917
swan	semantic	The national bird of the United States is the . . .	eagle	.895	.917
swan	unrelated	Neil Armstrong was the first man to walk on the . . .	moon	.895	.917
sweater	match	Each year his grandmother knit him an ugly Christmas . . .	sweater	.947	.917
sweater	semantic	Her foot was cold even though she wore a sock and a winter . . .	boot	.947	.917
sweater	unrelated	The rodent famous for building dams in rivers is called a . . .	beaver	.947	.917
table	match	For dinner, the family gathers at the dining room . . .	table	.947	.750
table	semantic	During work Danny sits all day long at his . . .	desk	.947	.750
table	unrelated	Bob sleeps in a king-sized . . .	bed	.947	.750
taco	match	At the Mexican restaurant, he ordered one hard and one soft . . .	taco	.895	1.000
taco	semantic	Chicago is famous for deep dish . . .	pizza	.895	1.000
taco	unrelated	The joint connecting the forearm to the bicep is the . . .	elbow	.895	1.000
tail	match	The dog ran in circles chasing his own . . .	tail	.947	1.000
tail	semantic	The bird could not fly because of an injured . . .	wing	.947	1.000
tail	unrelated	The boy went to the half pipe and practiced tricks on his . . .	skateboard	.947	1.000
tank	match	The soldiers were protected by the armored fighting vehicle called a . . .	tank	.947	.917
tank	semantic	The explosion came from a homemade pipe . . .	bomb	.947	.917
tank	unrelated	The student took her books home in her . . .	backpack	.947	.917
tape	match	To fix his torn paper he needs some . . .	tape	1.000	1.000
tape	semantic	Sam measured the length of the paper using a . . .	ruler	1.000	1.000
tape	unrelated	The boy took a pole to the lake to catch a . . .	fish	1.000	1.000
tie	match	He wore a suit with a Windsor knot in his . . .	tie	.842	.917
tie	semantic	Levi makes high quality denim blue . . .	jeans	.842	.917
tie	unrelated	The king of the jungle is the . . .	lion	.842	.917
tire	match	During their bike tour Mike had to fix a flat . . .	tire	1.000	.917
tire	semantic	The stagecoach wasn't moving because of the broken wagon . . .	wheel	1.000	.917
tire	unrelated	Matt grabbed the swatter to kill the . . .	fly	1.000	.917
toast	match	John spread butter and grape jelly on his morning . . .	toast	.842	.917
toast	semantic	Angela likes cream cheese on a . . .	bagel	.842	.917
toast	unrelated	The directions did not match any roads on the . . .	map	.842	.917
toothbrush	match	Proper dental hygiene includes cleaning each day with a bristled . . .	toothbrush	.526	1.000
toothbrush	semantic	The man brushed his hair using a fine-toothed . . .	comb	.526	1.000
toothbrush	unrelated	She put her salad on a large . . .	plate	.526	1.000
towel	match	He dried his wet hands with a . . .	towel	1.000	1.000
towel	semantic	In the shower, he washed with a bar of . . .	soap	1.000	1.000
towel	unrelated	The king of the jungle is the . . .	lion	1.000	1.000
tree	match	The bird built a nest high up in an elm . . .	tree	1.000	.917
tree	semantic	In the desert, he pricked his finger on the spine of a . . .	cactus	1.000	.917
tree	unrelated	The joint connecting the forearm to the bicep is the . . .	elbow	1.000	.917
trophy	match	The team that won the tournament took home a . . .	trophy	.947	.917
trophy	semantic	In the Olympic games, she won a gold . . .	medal	.947	.917
trophy	unrelated	The homeless man slept on a park . . .	bench	.947	.917
tweezers	match	Andy removed a splinter with some . . .	tweezers	.842	.500
tweezers	semantic	He applied the medicine to his eye using a . . .	dropper	.842	.500
tweezers	unrelated	To protect his head from sunburn, the bald man wore a wide-brimmed . . .	hat	.842	.500
van	match	We couldn't get a truck, but we managed to pack everything in a moving . . .	van	.895	.420
van	semantic	Many people like the Grand Cherokee, but the Wrangler is my favorite kind of . . .	jeep	.895	.420

(*Appendices continue*)

This document is copyrighted by the American Psychological Association or one of its allied publishers.
This article is intended solely for the personal use of the individual user and is not to be disseminated broadly.

Table B1 (*continued*)

Picture	Condition	Sentence	Completion	Cloze (younger)	Cloze (older)
van	unrelated	Nick sneezed and blew his . . .	nose	.895	.420
vase	match	She placed the flowers in a glass . . .	vase	.947	.833
vase	semantic	When the power went out, they lit a . . .	candle	.947	.833
vase	unrelated	The bird built a nest high up in an elm . . .	tree	.947	.833
vest	match	Between his jacket and shirt, the usher had buttoned his . . .	vest	.579	.917
vest	semantic	Under their shirts, most women wear a supportive . . .	bra	.579	.917
vest	unrelated	The lead broke so she sharpened the . . .	pencil	.579	.917
waffle	match	At brunch, Maggie either eats pancakes or a Belgian . . .	waffle	.895	.417
waffle	semantic	Chocolate glazed with sprinkles is his favorite kind of . . .	donut	.895	.417
waffle	unrelated	Mary went to Hawaii to tan on a sandy . . .	beach	.895	.417
watch	match	A Rolex is an expensive type of . . .	watch	1.000	1.000
watch	semantic	The man gave his fiancé an engagement . . .	ring	1.000	1.000
watch	unrelated	To start a car, you need the . . .	key	1.000	1.000
wheel	match	The stagecoach wasn't moving because of the broken wagon . . .	wheel	.895	1.000
wheel	semantic	During their bike tour Mike had to fix a flat . . .	tire	.895	1.000
wheel	unrelated	The blind, flying rodent that lives in caves is the . . .	bat	.895	1.000
window	match	To let some cool air in the apartment they opened a . . .	window	1.000	.917
window	semantic	She felt hot in the office and plugged in the . . .	fan	1.000	.917
window	unrelated	Under their shirts, most women wear a supportive . . .	bra	1.000	.917
wine	match	The old girlfriends chatted over a bottle of red . . .	wine	1.000	1.000
wine	semantic	With breakfast Julie always drinks some orange . . .	juice	1.000	1.000
wine	unrelated	Apple's mp3 player is called an . . .	ipod	1.000	1.000
wing	match	The bird could not fly because of an injured . . .	wing	1.000	1.000
wing	semantic	The dog ran in circles chasing his own . . .	tail	1.000	1.000
wing	unrelated	Emily fixed the broken mug with some . . .	glue	1.000	1.000
zebra	match	The horse-like animal with black and white stripes is the . . .	zebra	1.000	.833
zebra	semantic	The animal with a long neck and long legs is a . . .	giraffe	1.000	.833
zebra	unrelated	On Halloween kids trick-or-treat to collect . . .	candy	1.000	.833

(*Appendices continue*)

This document is copyrighted by the American Psychological Association or one of its allied publishers.
This article is intended solely for the personal use of the individual user and is not to be disseminated broadly.

Appendix C

Full Model Coefficient Tables and Random Effect Structure

In Tables C1–C11, significance, as assessed with chi-square tests of nested models with and without predictor, is reported in the last column. Significant effects ($p < .05$) are bolded; marginal effects ($p < .10$) are italicized. When chi-square models with the fixed effect held out did not converge (DNC), preventing nested model comparison, the absolute value of the t-statistic was used as a proxy. A (∗) is used to indicate significance as assessed with a t-statistic >2, (·) indicates a marginal effect for a t-statistic >1.5, and *ns* indicates a nonsignificant effect for a t-statistic <1.5. For logistic regressions, the z-statistic was used similarly.

Random Effect Structure

Experiment 2: Correlated subject and item slopes for semantic relatedness, and block.

Random effect Structure

Experiment 1: Correlated subject slopes for block, match status, block:match, block:semantic; correlated item slopes for Shipley score, match status, block, semantic relatedness, and block:semantic.

Experiment 2: Decorrelated subject slopes for block, match status, block:match; decorrelated item slopes for Shipley score, semantic relatedness, block, match status, and block:match.

Experiment 3: Correlated subject slopes for block, match status, semantic relatedness, block:match; correlated item slopes for Shipley score, block, match status, semantic relatedness, block:match, and block:semantic.

Random Effect Structure

Model comparing Experiment 1 and Experiment 2: Decorrelated subject slopes for block, match status, block:match, decorrelated item slopes for Shipley score, match status, experiment, block, semantic relatedness, experiment:match, block:match, block:semantic, and experiment:block:match/

Table C1
Summary of Single-Experiment Error Models

	Estimate (β)	*SE*	*t*-value	χ^2	p (χ^2)
Experiment 2: Young adults, time pressure					
Block	*−1.308*	*.757*	*−1.728*	*3.83*	*.051*
Semantic relatedness	**5.771**	**1.782**	**3.238**	**9.25**	**.002**
Block:semantic	**−1.143**	**.820**	**−1.394**	**6.25**	**.012**

Note. There was insufficient variance in the distribution of completion errors in Experiments 1 and 3; as a result, a model is reported for Experiment 2 only. Bolded predictors are significant at $p < .05$; italicized predictors are marginal at $.05 < p < .10$.

Table C2
Summary of Single-Experiment Response Time Models

Experiment	Estimate (β)	*SE*	*t*-value	χ^2	p (χ^2)
Experiment 1: Young adults, no time pressure					
Block	**−.060**	**.007**	**−8.74**	**30.39**	**<.001**
Match status	**.247**	**.020**	**12.44**	**43.95**	**<.001**
Semantic relatedness	*−.014*	*.008*	*−1.70*	*DNC*	*(·)*
Shipley score	−.004	.007	−.53	.20	.655
Block:match	−.011	.014	−.77	.58	.445
Block:semantic	**−.019**	**.012**	**−1.68**	**6.22**	**.012**
Experiment 2: Young adults, time pressure					
Block	**−.062**	**.007**	**−9.37**	**33.02**	**<.001**
Match status	**.307**	**.024**	**12.83**	**43.17**	**<.001**
Semantic relatedness	.001	.008	.07	.00	.944
Shipley score	−.003	.003	−1.10	1.17	.279
Block:match	**.031**	**.014**	**2.18**	**4.25**	**.039**
Block:semantic	*−.017*	*.009*	*−1.91*	*3.63*	*.057*
Experiment 3: Older adults, no time pressure					
Block	**−.054**	**.008**	**−6.78**	**DNC**	**(∗)**
Match status	**.232**	**.019**	**12.37**	**DNC**	**(∗)**
Semantic relatedness	.003	.011	.29	DNC	
Shipley score	.002	.007	.34	DNC	
Block:match	−.006	.015	−.37	DNC	
Block:semantic	.010	.012	.82	DNC	

Note. DNC = did not converge. Bolded predictors are significant at $p < .05$; italicized predictors are marginal at $.05 < p < .10$.

Model comparing Experiment 1 and Experiment 3: Decorrelated subject slopes for block, match status, semantic relatedness, block:match; decorrelated item slopes for Shipley score, experiment, block, match status, semantic relatedness, experiment:match, block:semantic, experiment:block:match, and experiment:block:unrelated.

Random Effect Structure

Experiment 1: Correlated subject slopes for block, response time, RT:semantic; correlated item slopes for Shipley score, response time, BLUPs from RT model, and block:semantic.

Experiment 2: Decorrelated subject slopes for block, match, response time, block:semantic; decorrelated item slopes for Shipley score, match status, BLUPs from RT model, block:match, and block:semantic.

Experiment 3: Decorrelated subject slopes for block, response time, block:match, block:semantic, response time:semantic; decorrelated item slopes for Shipley score, response time, BLUPs, block:match, block:semantic, RT:match, and RT:semantic.

(*Appendices continue*)

This document is copyrighted by the American Psychological Association or one of its allied publishers.
This article is intended solely for the personal use of the individual user and is not to be disseminated broadly.

This document is copyrighted by the American Psychological Association or one of its allied publishers.
This article is intended solely for the personal use of the individual user and is not to be disseminated broadly.

Table C3
Summary of Cross-Experiment Comparison Models, Response Time

Experiment	Estimate (β)	*SE*	*t*-value	χ^2	p (χ^2)
Experiment 1 vs. Experiment 2					
Experiment	**−.083**	**.034**	**−2.40**	**5.45**	**.020**
Block	**−.061**	**.005**	**−12.70**	**65.34**	**<.001**
Match status	**.279**	**.017**	**16.60**	**86.42**	**<.001**
Semantic relatedness	−.006	.007	−.90	.84	.359
Shipley score	−.003	.003	−1.00	1.03	.310
Experiment:block	−.001	.008	−.20	.03	.861
Experiment:match	**.065**	**.029**	**2.20**	**4.70**	**.030**
Experiment:semantic	.013	.010	1.30	1.73	.189
Block:match	.011	.010	1.10	1.18	.278
Block:semantic	**−.019**	**.007**	**−2.90**	**7.97**	**.005**
Experiment:block:match	**.039**	**.019**	**2.00**	**3.92**	**.048**
Experiment:block:semantic	.005	.012	.40	.19	.662
Experiment 1 vs. Experiment 3					
Experiment	**.222**	**.058**	**3.83**	**12.34**	**<.001**
Block	**−.057**	**.005**	**−10.57**	**51.89**	**<.001**
Match status	**.234**	**.014**	**16.24**	**85.62**	**<.001**
Semantic relatedness	−.007	.007	−.99	.98	.323
Shipley score	.000	.006	−.07	.01	.942
Experiment:block	.005	.010	.44	.19	.661
Experiment:match	−.030	.024	−1.21	1.45	.229
Experiment:semantic	.016	.011	1.46	2.03	.154
Block:match	−.013	.009	−1.44	2.02	.156
Block:semantic	.000	.006	−.04	.00	.967
Experiment:block:match	−.001	.019	−.08	.01	.936
Experiment:block:semantic	**.031**	**.014**	**2.16**	**4.53**	**.033**

Note. Bolded predictors are significant at $p < .05$.

Random Effect Structure

Model comparing Experiment 1 and Experiment 2: Decorrelated subject slopes for block, response time, block:semantic; decorrelated item slopes for experiment, block, response time, BLUPs from RT model, block:match, block:semantic, RT:semantic, and RT:experiment.

Model comparing Experiment 1 and Experiment 3: Correlated subject slopes for block, response time; correlated item slopes for Shipley score, experiment, response time, and BLUPs from RT model.

Random Effect Structure

Experiment 1: Correlated subject slopes for block, match status, semantic relatedness, response time, number of consonants, RT:match, RT:semantic; correlated item slopes for match, RT, BLUP from RT model, Shipley score, block:match, block:semantic, and RT:match.

Experiment 2: Decorrelated subject slopes for block, match status, response time, number of consonants, block:semantic, RT:semantic; decorrelated item slopes for block, match, semantic, block:semantic, and RT:semantic.

Experiment 3: Decorrelated subject slopes for block, response time, number of consonants, block:match, RT:match; decorrelated item slopes for RT, BLUP from RT model, Shipley score, block:semantic, and RT:semantic.

Random Effect Structure

Model comparing Experiment 1 and Experiment 2: Decorrelated subject slopes for block, match status, response time, number of consonants, RT:match, RT:semantic; decorrelated item slopes for group, match status, response time, BLUPs from Rt model, Shipley score, block:match, block:unrelated, RT:match, RT:unrelated, RT:experiment, RT:semantic:experiment, block:match:experiment, and block:unrelated:experiment.

Model comparing Experiment 1 and Experiment 3: Decorrelated subject slopes for block, unrelated, response time, number of consonants, block:match, RT:unrelated; decorrelated item slopes for experiment, response time, BLUPs from RT model, Shipley score, block:match, block:unrelated, and RT:unrelated.

Table C4
Summary of Single-Experiment Word Duration Models

Experiment	Estimate (β)	*SE*	*t*-value	χ^2	p (χ^2)
Experiment 1: Young adults, no time pressure					
Block	4.145	4.346	.95	.90	.347
Match status	**18.783**	**3.405**	**5.52**	**30.09**	**<.001**
Semantic relatedness	−4.268	2.749	−1.55	2.33	.127
Trial-level RT	**−33.281**	**13.274**	**−2.51**	**5.73**	**.017**
BLUP from RT model	*146.588*	*76.192*	*1.92*	*2.75*	*.097*
Shipley score	−.098	2.470	−.04	.00	.969
Block:match	1.308	3.793	.34	.12	.731
Block:semantic	2.108	3.739	.56	.31	.576
RT:match	−19.718	15.563	−1.27	1.56	.212
RT:semantic	**50.667**	**15.344**	**3.30**	**9.84**	**.002**
Experiment 2: Young adults, time pressure					
Block	−4.577	2.863	−1.60	2.39	.122
Match status	**12.717**	**3.612**	**3.52**	**10.12**	**.002**
Semantic relatedness	*−5.298*	*2.767*	*−1.92*	*3.66*	*.056*
Trial-level RT	−10.080	8.258	−1.22	1.47	.225
BLUP from RT model	427.909	471.328	.91	.81	.369
Shipley score	*−3.103*	*1.548*	*−2.01*	*3.62*	*.056*
Block:match	−7.161	4.337	−1.65	2.69	.101
Block:semantic	2.987	4.302	.69	.48	.489
RT:match	*27.527*	*14.706*	*1.87*	*3.45*	*.063*
RT:semantic	12.328	14.092	.88	.76	.382
Experiment 3: Older adults, no time pressure					
Block	−4.664	3.655	−1.28	1.56	.212
Match status	5.105	4.451	1.15	1.31	.253
Semantic relatedness	**−11.304**	**3.711**	**−3.05**	**8.86**	**.003**
Trial-level RT	−10.255	9.019	−1.14	1.27	.259
BLUP from RT model	**179.651**	**82.441**	**2.18**	**4.22**	**.040**
Shipley score	**−6.711**	**3.107**	**−2.16**	**4.19**	**.041**
Block:match	**−19.786**	**7.558**	**−2.62**	**5.83**	**.016**
Block:semantic	5.570	5.452	1.02	1.04	.308
RT:match	−20.459	15.402	−1.33	1.75	.186
RT:semantic	7.360	17.078	.43	.19	.667

Note. Bolded predictors are significant at $p < .05$; italicized predictors are marginal at $.05 < p < .10$.

(*Appendices continue*)

Table C5
Cross-Experiment Comparison Models, Word Duration

Experiment	Estimate (β)	*SE*	*t*-value	χ^2	*p* (χ^2)
Experiment 1 vs. Experiment 2					
Experiment	**−53.213**	**15.479**	**−3.44**	**10.24**	**.001**
Block	−.320	2.654	−.12	DNC	
Match status	**16.310**	**2.442**	**6.68**	**44.18**	**<.001**
Semantic relatedness	**−5.564**	**1.983**	**−2.81**	**7.85**	**.005**
Trial-level RT	**−21.480**	**7.563**	**−2.84**	**7.66**	**.006**
BLUP from RT model	−69.426	299.063	−.23	.05	.817
Shipley score	*−2.340*	*1.352*	*−1.73*	*2.88*	*.090*
Experiment:block	−8.629	5.302	−1.63	2.55	.110
Experiment:match	1.061	4.821	.22	.05	.826
Experiment:unrelated	.605	3.928	.15	.02	.878
Block:match	−4.295	2.654	−1.62	2.60	.107
Block:semantic	2.531	2.957	.86	.73	.393
Experiment:RT	11.771	14.258	.83	.67	.411
Match:RT	7.070	10.322	.68	.47	.494
Unrelated:RT	17.762	10.525	1.69	2.82	.093
Experiment:block:match	−5.612	5.193	−1.08	1.16	.281
Experiment:block:unrelated	1.623	4.720	.34	.12	.731
Experiment:RT:match	**55.878**	**20.317**	**2.75**	**DNC**	(*)
Experiment:RT:unrelated	−17.517	19.453	−.90	.81	.369
Experiment 1 vs. experiment 3					
Experiment	**54.658**	**23.563**	**2.32**	**5.02**	**.025**
Block	−.764	2.804	−.27	.07	.785
Match status	**14.856**	**3.050**	**4.87**	**23.58**	**<.001**
Semantic relatedness	**−5.451**	**2.455**	**−2.22**	**4.92**	**.027**
Trial-level RT	**−24.785**	**8.425**	**−2.94**	**8.25**	**.004**
BLUP from RT model	−285.849	456.564	−.63	.36	.550
Shipley score	*−4.191*	*2.358*	*−1.78*	*2.82*	*.093*
Experiment:block	−8.503	5.605	−1.52	2.23	.136
Experiment:match	−2.320	6.083	−.38	.14	.704
Experiment:unrelated	**−10.309**	**4.904**	**−2.1**	**4.41**	**.038**
Block:match	−4.368	3.156	−1.38	1.91	.167
Block:semantic	2.814	2.829	.99	.98	.321
Experiment:RT	−1.668	15.298	−.11	.01	.914
Match:RT	−9.736	10.881	−.89	.79	.375
Unrelated:RT	*17.368*	*9.539*	*1.82*	*3.29*	*.070*
Experiment:block:match	**−20.408**	**6.293**	**−3.24**	**10.48**	**.001**
Experiment:block:unrelated	5.289	5.660	.93	.87	.351
Experiment:RT:match	−19.191	21.651	−.89	.78	.378
Experiment:RT:unrelated	−25.535	19.094	−1.34	1.77	.183

Note. DNC = did not converge. Bolded predictors are significant at $p < .05$; italicized predictors are marginal at $.05 < p < .10$.

Random Effect Structure

Experiment 1: Decorrelated subject slopes for block; decorrelated item slopes for Shipley score, response time, BLUP from RT model, block:match, block:unrelated, and RT:unrelated.

Experiment 2: Decorrelated subject slopes for block, match, response time, block:match, block:semantic; decorrelated item slopes for Shipley score, RT, BLUPs from RT model, block: match, and block:semantic.

Experiment 3: Decorrelated subject slopes for block, block:semantic, RT:match; decorrelated item slopes for Shipley score, block, semantic relatedness, response time, BLUPs from RT model, block:match, and block:semantic.

Random Effect Structure

Model comparing Experiment 1 and Experiment 2: Decorrelated subject slopes for block, response time; decorrelated item slopes for Shipley score, experiment, response time, and BLUPs from RT model.

Model comparing Experiment 1 and Experiment 3: Decorrelated subject slopes for block, block:semantic; decorrelated item slopes for Shipley score, experiment, response time, BLUPs from RT model, block:match, and block:unrelated.

Random Effect Structure

Experiment 1: Decorrelated subject slopes for block, block:match, block:semantic; decorrelated item slopes for Shipley score, BLUPs from RT model, block:match, and block:semantic.

Experiment 2: Correlated subject slopes for block and trial-level RT; correlated item slopes for Shipley score, BLUP from RT model, and block:semantic.

Experiment 3: Decorrelated subject slopes for match, trial-level RT, block:match, RT:match, RT:semantic; decorrelated item slopes for Shipley score, trial-level RT, BLUPs from RT model, and block:match.

Random Effect Structure

Model comparing Experiment 1 and Experiment 2: Correlated random subject slopes for block; correlated random item slopes for Shipley score, experiment, and BLUPs from RT model.

Model comparing Experiment 1 and Experiment 3: Subject intercept; correlated random item slopes for Shipley score, experiment, trial-level RT, and BLUPs from RT model.

(*Appendices continue*)

This document is copyrighted by the American Psychological Association or one of its allied publishers.
This article is intended solely for the personal use of the individual user and is not to be disseminated broadly.

Table C6
Summary of Single-Experiment Models of Initial Consonant Duration

Experiment	Estimate (β)	*SE*	*t*-value	χ^2	p (χ^2)
Experiment 1: Young adults, no time pressure					
Block	**−2.245**	**.819**	**−2.74**	**6.33**	**.012**
Match status	**10.852**	**1.895**	**5.73**	**16.90**	**<.001**
Semantic relatedness	−2.397	1.759	−1.36	1.80	.180
Trial-level RT	**−35.532**	**4.728**	**−7.52**	**30.07**	**<.001**
BLUP from RT model	**44.145**	**20.043**	**2.20**	**4.34**	**.037**
Number of consonants	**39.786**	**7.322**	**5.43**	**26.17**	**<.001**
Shipley score	.125	.651	.19	.04	.849
Block:match	.621	2.079	.30	.09	.768
Block:semantic	3.139	2.103	1.49	2.19	.139
RT:match	**−31.232**	**8.693**	**−3.59**	**9.87**	**.002**
RT:semantic	5.651	8.045	.70	.46	.499
Experiment 2: Young adults, time pressure					
Block	**−2.315**	**.736**	**−3.14**	**7.96**	**.005**
Match status	**8.621**	**1.953**	**4.42**	**15.36**	**<.001**
Semantic relatedness	**−3.759**	**1.478**	**−2.54**	**6.44**	**.011**
Trial-level RT	**−14.980**	**3.592**	**−4.17**	**14.61**	**<.001**
BLUP from RT model	−11.917	76.347	−.16	.02	.876
Number of consonants	**27.276**	**5.810**	**4.70**	**20.42**	**<.001**
Shipley score	*−.498*	*.251*	*−1.99*	*3.59*	*.058*
Block:match	.122	1.796	.07	.00	.946
Block:semantic	−1.534	1.885	−.81	.62	.431
RT:match	−.866	7.570	−.11	.01	.912
RT:semantic	*16.658*	*9.217*	*1.81*	*3.18*	*.075*

Note. Bolded predictors are significant at $p < .05$; italicized predictors are marginal at $.05 < p < .10$.

Table C7
Cross-Experiment Comparison Models, First Consonant Duration

Experiment	Estimate (β)	*SE*	*t*-value	χ^2	p (χ^2)
Experiment 1 vs. Experiment 2					
Experiment	**−11.380**	**2.996**	**-3.80**	**12.25**	**<.001**
Block	**−2.081**	**.573**	**−3.63**	**11.43**	**<.001**
Match status	**10.418**	**1.305**	**7.98**	**43.78**	**<.001**
Semantic relatedness	**−2.562**	**1.038**	**−2.47**	**5.95**	**.015**
Trial-level RT	**−25.166**	**2.997**	**−8.40**	**47.38**	**<.001**
BLUP from RT model	**42.837**	**18.177**	**2.36**	**5.23**	**.022**
Number of consonants	**34.291**	**6.306**	**5.44**	**27.36**	**<.001**
Shipley score	*−.428*	*.251*	*−1.70*	*2.81*	*.094*
Experiment:block	.266	1.141	.23	.05	.816
Experiment:match	−3.557	2.475	−1.44	2.05	.152
Experiment:unrelated	−1.325	2.073	−.64	.41	.523
Block:match	.751	1.388	.54	.29	.589
Block:semantic	.592	1.398	.42	.18	.673
Experiment:RT	**19.451**	**5.818**	**3.34**	**DNC**	**(∗)**
Match:RT	**−14.512**	**5.492**	**−2.64**	**6.79**	**.009**
Unrelated:RT	6.538	5.833	1.12	1.20	.274
Experiment:block:match	−.068	2.662	−.03	.00	.980
Experiment:block:unrelated	*−4.844*	*2.657*	*−1.82*	*3.30*	*.069*
Experiment:RT:match	**25.716**	**10.724**	**2.40**	**5.72**	**.017**
Experiment:RT:unrelated	5.693	11.839	.48	.23	.634
Experiment 1 vs. Experiment 3					
Experiment	**12.957**	**4.996**	**2.59**	**DNC**	**(∗)**
Block	*−1.205*	*.647*	*−1.86*	*DNC*	*(·)*
Match status	**5.565**	**1.482**	**3.75**	**DNC**	**(∗)**
Semantic relatedness	−1.543	1.365	−1.13	DNC	
Trial-level RT	**−24.060**	**4.025**	**−5.98**	**28.72**	**<.001**
BLUP from RT model	**38.690**	**11.159**	**3.47**	**DNC**	**(∗)**
Number of consonants	**35.186**	**6.996**	**5.03**	**DNC**	**(∗)**
Shipley score	−.042	.381	−.11	.01	.923
Experiment:block	1.243	1.264	.98	DNC	
Experiment:match	**−6.284**	**2.969**	**−2.12**	**DNC**	**(∗)**
Experiment:unrelated	.145	2.653	.06	DNC	
Block:match	.344	2.012	.17	.03	.865
Block:semantic	1.130	1.652	.68	DNC	
Experiment:RT	9.926	7.426	1.34	1.67	.197
Match:RT	−6.910	5.447	−1.27	DNC	
Unrelated:RT	−.566	5.512	−.10	.01	.921
Experiment:block:match	−4.458	3.606	−1.24	DNC	
Experiment:block:unrelated	*−5.126*	*2.715*	*−1.89*	*3.50*	*.061*
Experiment:RT:match	**30.276**	**10.682**	**2.83**	**DNC**	**(∗)**
Experiment:RT:unrelated	*−18.670*	*10.489*	*−1.78*	*2.83*	*.093*

Note. Bolded predictors are significant at $p < .05$; italicized predictors are marginal at $.05 < p < .10$.

(*Appendices continue*)

This document is copyrighted by the American Psychological Association or one of its allied publishers.
This article is intended solely for the personal use of the individual user and is not to be disseminated broadly.

Table C8
Summary of Single-Experiment Models of Vowel Duration

Experiment	Estimate (β)	*SE*	*t*-value	χ^2	p (χ^2)
Experiment 1: Young adults, no time pressure					
Block	**5.141**	**1.963**	**2.62**	**5.81**	**.016**
Match status	−.451	1.801	−.25	.06	.802
Semantic relatedness	.308	1.437	.21	.05	.831
Trial-level RT	*7.026*	*4.119*	*1.71*	*2.89*	*.089*
BLUP from RT model	*61.607*	*34.581*	*1.78*	*2.98*	*.087*
Shipley score	−.116	1.134	−.10	.01	.918
Block:match	**−4.774**	**2.300**	**−2.08**	**4.24**	**.040**
Block:semantic	−1.622	1.958	−.83	.68	.409
RT:match	−5.604	7.936	−.71	.50	.481
RT:semantic	**20.607**	**7.622**	**2.70**	**6.99**	**.008**
Experiment 2: Young adults, time pressure					
Block	−.608	1.148	−.53	.28	.598
Match status	**3.395**	**1.723**	**1.97**	**3.85**	**.050**
Semantic relatedness	−.284	1.412	−.20	.04	.841
Trial-level RT	7.840	4.725	1.66	2.64	.104
BLUP from RT model	194.931	199.889	.98	.93	.336
Shipley score	−1.050	.656	−1.60	2.39	.122
Block:match	**−7.130**	**2.294**	**−3.11**	**9.28**	**.002**
Block:semantic	−.960	2.036	−.47	.22	.640
RT:match	7.391	7.536	.98	.96	.328
RT:semantic	4.087	7.339	.56	.31	.579
Experiment 3: Older adults, no time pressure					
Block	1.083	1.351	.80	.62	.426
Match status	2.743	1.839	1.49	1.90	.168
Semantic relatedness	−.729	1.540	−.47	.22	.636
Trial-level RT	**−8.231**	**3.896**	**−2.11**	**4.39**	**.036**
BLUP from RT model	*37.901*	*21.962*	*1.73*	*2.77*	*.096*
Shipley score	−.667	.824	−.81	.64	.422
Block:match	−1.362	2.353	−.58	.33	.564
Block:semantic	1.091	2.369	.46	.21	.647
RT:match	−6.539	6.338	−1.03	1.06	.303
RT:semantic	−.173	5.544	−.03	.00	.975

Note. Bolded predictors are significant at $p < .05$; italicized predictors are marginal at $.05 < p < .10$.

Table C9
Cross-Experiment Comparison Models, Vowel Duration

Experiment	Estimate (β)	*SE*	*t*-value	χ^2	p (χ^2)
Experiment 1 vs. Experiment 2					
Experiment	**−15.270**	**6.270**	**−2.44**	**5.50**	**.019**
Block	**2.337**	**1.116**	**2.09**	**4.14**	**.042**
Match status	1.199	1.307	.92	.84	.359
Semantic relatedness	−.453	1.054	−.43	.18	.668
Trial-level RT	**7.660**	**3.245**	**2.36**	**5.29**	**.022**
BLUP from RT model	*65.306*	*36.915*	*1.77*	*3.00*	*.083*
Shipley score	−.807	.531	−1.52	2.23	.135
Experiment:block	**−5.640**	**2.230**	**−2.53**	**5.90**	**.015**
Experiment:match	2.006	2.590	.77	.60	.439
Experiment:unrelated	1.382	2.102	.66	.430	.511
Block:match	**−5.599**	**1.390**	**−4.03**	**16.19**	**<.001**
Block:semantic	−.862	1.266	−.68	.46	.500
Experiment:RT	4.546	6.044	.75	.56	.453
Match:RT	−.750	5.613	−.13	.02	.894
Unrelated:RT	**13.383**	**5.329**	**2.51**	**6.29**	**.012**
Experiment:block:match	−2.530	2.779	−.91	.83	.363
Experiment:block:unrelated	.808	2.530	.32	.10	.750
Experiment:RT:match	16.470	11.067	1.49	2.19	.139
Experiment:RT:unrelated	−15.228	10.636	−1.43	2.04	.153
Experiment 1 vs. Experiment 3					
Experiment	**15.550**	**6.512**	**2.39**	**5.31**	**.021**
Block	**3.026**	**1.134**	**2.67**	**6.52**	**.011**
Match status	.844	1.437	.59	.35	.557
Semantic relatedness	1.342	1.132	1.19	1.40	.236
Trial-level RT	.665	2.648	.25	.06	.802
BLUP from RT model	**45.602**	**18.697**	**2.44**	**5.53**	**.019**
Shipley score	−.563	.660	−.85	.72	.396
Experiment:block	*−4.289*	*2.267*	*−1.89*	*3.41*	*.065*
Experiment:match	4.596	2.867	1.60	2.56	.109
Experiment:unrelated	−1.955	2.267	−.86	.74	.389
Block:match	*−3.293*	*1.771*	*−1.86*	*3.40*	*.065*
Block:semantic	−.562	1.338	−.42	.18	.675
Experiment:RT	**−15.369**	**5.876**	**−2.62**	**.68**	**.009**
Match:RT	−7.951	5.140	−1.55	2.39	.122
Unrelated:RT	**10.594**	**4.660**	**2.27**	**5.15**	**.023**
Experiment:block:match	1.516	2.916	.52	.27	.604
Experiment:block:unrelated	3.319	2.607	1.27	1.62	.204
Experiment:RT:match	−5.023	10.230	−.49	.24	.624
Experiment:RT:unrelated	**−21.417**	**9.290**	**−2.31**	**5.29**	**.021**

Note. Bolded predictors are significant at $p < .05$; italicized predictors are marginal at $.05 < p < .10$.

This document is copyrighted by the American Psychological Association or one of its allied publishers.
This article is intended solely for the personal use of the individual user and is not to be disseminated broadly.

(*Appendices continue*)

Table C10
Summary of Single-Experiment Models of Vowel Distance

Experiment	Estimate (β)	*SE*	*t*-value	χ^2	p (χ^2)
Experiment 1: Young adults, no time pressure					
Block	*−3.708*	*1.922*	*−1.93*	*3.72*	*.054*
Match status	7.479	4.857	1.54	2.36	.124
Semantic relatedness	−3.849	3.902	−.987	.97	.324
Trial-level RT	3.484	10.404	.335	.11	.738
BLUP from RT model	71.482	81.217	.88	.76	.384
Shipley score	2.574	2.721	.946	.87	.349
Block:match	−3.317	7.281	−.456	.2	.651
Block:semantic	.599	6.105	.098	.01	.922
RT:match	23.994	21.374	1.123	1.25	.263
RT:semantic	−32.140	20.031	−1.605	2.56	.109
Experiment 2: Young adults, time pressure					
Block	−5.966	2.719	−2.195	**4.37**	**.037**
Match status	**12.921**	**5.261**	**2.456**	**6.01**	**.014**
Semantic relatedness	1.172	4.391	.267	.07	.790
Trial-level RT	−3.409	10.141	−.336	.11	.738
BLUP from RT model	*503.629*	*268.912*	*1.873*	*3.13*	*.077*
Shipley score	−.623	.884	−.705	.49	.485
Block:match	−3.774	5.751	−.656	.51	.513
Block:semantic	−.365	6.010	−.061	.95	.952
RT:match	12.462	21.787	.572	.57	.569
RT:semantic	2.781	22.242	.125	.02	.901
Experiment 3: Older adults, no time pressure					
Block	−.200	2.098	−.096	.01	.924
Match status	1.295	6.496	.199	.04	.843
Semantic relatedness	−2.664	4.358	−.611	.36	.547
Trial-level RT	**21.875**	**10.307**	**2.122**	**4.45**	**.035**
BLUP from RT model	20.551	48.912	.42	.18	.675
Shipley score	−2.223	1.814	−1.226	1.44	.231
Block:match	−6.678	7.499	−.89	.77	.381
Block:semantic	5.628	5.368	1.048	1.09	.296
RT:match	*48.689*	*24.065*	*2.023*	*3.81*	*.051*
RT:semantic	−19.634	18.771	−1.046	1.09	.297

Note. Bolded predictors are significant at $p < .05$; italicized predictors are marginal at $.05 < p < .10$.

Table C11
Cross-Experiment Comparison Models, Vowel Distance

Experiment	Estimate (β)	*SE*	*t*-value	χ^2	p (χ^2)
Experiment 1 vs. Experiment 2					
Experiment	12.185	12.551	.971	.930	.335
Block	**−5.528**	**1.555**	**−3.556**	**11.310**	**<.001**
Match status	**9.815**	**3.838**	**2.558**	**6.520**	**.011**
Semantic relatedness	−1.865	3.138	−.594	.350	.552
Trial-level RT	−2.759	7.447	−.371	.140	.711
BLUP from RT model	64.059	71.047	.902	.790	.373
Shipley score	.665	1.032	.644	.410	.524
Experiment:block	−3.809	3.094	−1.231	1.500	.220
Experiment:match	*14.348*	*7.607*	*1.886*	*3.550*	*.060*
Experiment:unrelated	1.797	6.264	.287	.080	.774
Block:match	.124	4.100	.030	DNC	
Block:semantic	.815	3.748	.217	.050	.828
Experiment:RT	*−25.000*	*14.390*	*−1.737*	*3.010*	*.083*
Match:RT	7.838	15.909	.493	.240	.623
Unrelated:RT	−15.544	15.523	−1.001	1.000	.317
Experiment:block:match	−2.511	8.205	−.306	.090	.760
Experiment:block:unrelated	4.157	7.502	.554	.310	.580
Experiment:RT:match	−5.968	31.438	−.190	.040	.850
Experiment:RT:unrelated	36.171	30.972	1.168	1.360	.243
Experiment 1 vs. Experiment 3					
Experiment	**35.565**	**16.073**	**2.213**	**4.650**	**.031**
Block	−1.970	1.503	−1.311	1.710	.191
Match status	4.667	4.148	1.125	.126	.261
Semantic relatedness	*−6.109*	*3.269*	*−1.869*	*3.490*	*.062*
Trial-level RT	11.031	8.266	1.334	1.760	.184
BLUP from RT model	34.905	44.080	.792	.620	.431
Shipley score	−.252	1.561	−.161	.030	.872
Experiment:block	4.894	3.002	1.630	2.650	.103
Experiment:match	−8.582	8.263	−1.039	1.080	.300
Experiment:unrelated	4.055	6.537	.620	.380	.536
Block:match	−6.824	4.234	−1.612	2.590	.108
Block:semantic	1.552	3.808	.407	.100	.684
Experiment:RT	13.024	14.516	.897	.790	.373
Match:RT	**29.358**	**14.670**	**2.001**	**3.990**	**.046**
Unrelated:RT	*−23.701*	*13.314*	*−1.780*	*3.160*	*.076*
Experiment:block:match	−5.954	8.373	−.711	.500	.478
Experiment:block:unrelated	7.513	7.532	.998	.990	.319
Experiment:RT:match	9.251	29.330	.315	.100	.753
Experiment:RT:unrelated	33.853	26.595	1.273	1.610	.204

Note. Bolded predictors are significant at $p < .05$; italicized predictors are marginal at $.05 < p < .10$.

(*Appendices continue*)

This document is copyrighted by the American Psychological Association or one of its allied publishers.
This article is intended solely for the personal use of the individual user and is not to be disseminated broadly.

Appendix D

Condition Means by Experiment

Table D1
Mean and Standard Errors for Each Dependent Variable, Broken Down by Experiment and Condition, are Presented Below

	Experiment 1		Experiment 2		Experiment 3	
Variable	*M*	*SE*	*M*	*SE*	*M*	*SE*
Reaction time (ms)						
Competitor	730.828	6.139	682.087	5.320	903.763	8.936
Match	613.776	5.146	549.004	4.566	776.781	5.559
Unrelated	745.887	6.187	684.229	4.971	924.553	8.914
Word duration (ms)						
Competitor	376.807	3.762	323.857	3.663	397.651	4.105
Match	370.838	3.479	324.491	3.295	393.569	3.815
Unrelated	380.109	3.801	331.083	3.532	421.518	4.501
Initial consonant duration (ms)						
Competitor	59.042	1.846	45.812	1.627	58.542	1.678
Match	54.423	1.653	42.996	1.349	56.902	1.603
Unrelated	58.291	1.832	48.281	1.574	61.645	1.859
Vowel duration (ms)						
Competitor	156.903	1.918	141.885	1.773	163.598	1.685
Match	153.908	1.774	136.296	1.605	158.709	1.656
Unrelated	154.248	1.947	141.776	1.817	167.668	1.923
Vowel distance (Hz)						
Competitor	245.888	5.223	264.282	5.349	268.639	4.635
Match	244.025	4.768	254.768	5.004	274.651	3.774
Unrelated	245.834	5.090	257.240	5.171	285.792	4.952

Note. Although reaction time models were run with a log-transformed dependent variable, the raw values are presented below for easier interpretability.

Received October 11, 2017
Revision received March 15, 2018
Accepted April 23, 2018 ■

This document is copyrighted by the American Psychological Association or one of its allied publishers.
This article is intended solely for the personal use of the individual user and is not to be disseminated broadly.

Predicting Glottal Closure Insufficiency using Fundamental Frequency Contour Analysis

Received: 28 March 2018 | Revised: 27 December 2018 | Accepted: 29 January 2019

DOI: 10.1002/hed.25709

Check for updates

WILEY

ORIGINAL ARTICLE

Predicting glottal closure insufficiency using fundamental frequency contour analysis

Jacob T. Cohen MD[1] | Alma Cohen PhD[2] | Limor Benyamini MD[1] | Yossi Adi MSc[3] | Joseph Keshet PhD[3]

[1]Department of Otolaryngology Head and Neck Surgery, Rambam Health Care Campus, Haifa, Israel

[2]Berglass School of Economics, Tel-Aviv University, Tel Aviv, Israel

[3]Department of Computer Science, Bar-Ilan University, Ramat-Gan, Israel

Correspondence
Jacob T. Cohen, Department of Otolaryngology Head and Neck Surgery, Rambam Health Care Campus, 8 Halia St., Haifa 310960, Israel.
Email: ja_cohen@rambam.health.gov.il

Abstract

Background: Voice analysis has a limited role in a day-to-day voice clinic. We developed objective measurements of vocal folds (VF) glottal closure insufficiency (GCI) during phonation.

Methods: We examined 18 subjects with no history of voice impairment and 20 patients with unilateral VF paralysis before and after injection medialization laryngoplasty. Voice analysis was extracted. We measured *settling time*, *slope*, and *area under* the fundamental frequency curve from the phonation onset to its settling-time.

Results: The measured parameters, *settling time*, *slope*, and *area under* the curve were in correlation with the traditional acoustic voice assessments and clinical findings before treatment and after injection medialization laryngoplasty.

Conclusion: We found that the fundamental frequency curve has several typical contours which correspond to different glottal closure conditions. We proposed a new set of parameters that captures the contour type, and showed that they could be used to quantitatively assess individuals with GCI.

KEYWORDS

glottal insufficiency, hoarseness, speech processing, vocal fold paralysis, voice analysis

1 | INTRODUCTION

Voice disorders due to the glottal closure insufficiency (GCI), which is the gap between the vocal folds (VF) during phonation, may be caused by various conditions. The most common are *vocal fold paresis and paralysis.* When one of the VFs is paralyzed, the folds are not able to meet in the midline to start the glottic attack, hence prevents the development of the subglottic pressure needed to initiate normal speech.[1] Moreover, the mucosal wave cannot be adequately maintained due to the lack of negative glottal pressure caused by the Bernoulli effect. Symptoms include effortful phonation, vocal fatigue, breathiness, and odynophonia.[2] A multidimensional patient assessment protocol usually includes videoendoscopy, acoustic analysis which includes maximum phonation time (MPT), S/Z ratio, jitter and shimmer, and voice quality assessment using the GRBAS scale, and the voice handicap index (VHI). None of these objective and subjective assessments is specific enough to evaluate the quality and efficacy of VFs closure alone.

The myoelastic-aerodynamic theory of VF vibration states that the voice is produced by a coupling between the aerodynamic forces and the VF tissue parameters,[3] which produce a range of acoustical sound. Due to the complexity of voice, simplified models are often employed for both clinical and scientific studies. The most common modeling framework in voice investigations is the lumped-element approach, where the VF structure is modeled as a collection of discrete coupled mass-spring-damper systems subjected to some external aerodynamic

 © 2019 Wiley Periodicals, Inc.

and/or acoustical loading function. Lumped-element models have proven capable of emulating the physiological VF kinematics and acoustic output.[4–6]

Modern control theory[7–9] is a field of science and engineering that is used in analyzing the behavior over time and frequency of different types of dynamic systems, such as mechanical systems, electric systems, liquid-level systems, pneumatic systems, hydraulic systems, and thermal systems. This theory provides a mathematic modeling framework of such systems using differential equations. According to this theory, the response (or the output) of a system to a given input, be it an external force in a mechanical system or pressure in a liquid or air level system, is composed of a transient response and a steady-state response. *Transient response* represents the temporal behavior of the system from the initial state to the final stable state, while *steady-state response* represents what happens with the system after stabilization, usually after some period of time. Researchers identified voice initiation as an important characteristic of vocal performance, efficiency, and quality.[10–12]

The pace with which the VFs adduct to the midline is considered an important variable in the etiology of some voice disorders.[13,14]

In this paper, we propose objective measurements for assessing the condition of an individual with vocal cord paresis or paralysis: the settling time, the slope from the phonation onset to the settling time, and the area under the fundamental frequency curve from phonation onset to the settling time. They are denoted as *Time-to-Stabilization [sec]*, *Slope-at-Stabilization [Hz/sec]*, and *Area-of-Stability[Hz x sec]*, respectively. We correlated these measurements with healthy subjects, with patients suffering from vocal cord paralysis with GCI, and with patients who were treated for these problems, then examined their influence on these parameters.

2 | PATIENTS

This study was approved by the Committee for the Protection of Human Subjects (Institutional Review Board) of Rambam Health Care Campus and the Technion School of Medicine, and all subjects provided informed consent before data collection began. Patients were divided into three groups:

2.1 | Control group

This group included 18 subjects with normal laryngoscopic and stroboscopic examinations who reported no history of voice impairment. The group represents subjects with normal voices and good glottal closure during voicing.

2.2 | Study group

The second group included 20 patients with unilateral VF paralysis proven by laryngoscopy and video stroboscopy. Patients with vocal pathologies due to a neurological disease, findings on the VFs, scarring and radiation, missing data, or nonsatisfactory voice recordings were excluded from the study. In addition, patients who had VF paralysis without vocal complaints and with an adequate glottic closure proven by a laryngo-stroboscopic examination were removed from the study. The degree of the gap between the VFs was not considered to be a factor due to the lack of acceptable measurements.

2.3 | Study group after injection medialization laryngoplasty

Each and every one of the patients in the study group described above was injected with a filling material to the paralyzed VF in order to allow proper approximation of the VFs during phonation. Out of the study group, 19 patients had a short-term injection with RADIESSE Voice Gel (Merz North America, Franksville, Wisconsin) and 1 patient had a long-term injection with RADIESSE Voice (Merz North America, Franksville, Wisconsin). Furthermore, 4 patients, who were initially treated with RADIESSE Voice Gel and demonstrated reabsorption of the injectable material, were injected again with the long-term RADIESSE Voice. Overall, we had a total of 24 observations. Our hypothesis was that the fundamental frequency analysis of these patients, once we fixed their GCI, was going to resemble the fundamental frequency feature of the control group.

3 | METHOD

Subjects were asked to pronounce the sustained vowel */iy/* from complete silence. The phone */iy/* is suitable for the proposed analysis as its production can generates an abrupt pitch rise that simulates a step function, and it is a phoneme that exits in all world languages. The voice of the subjects were recorded at a sampling rate of 16 kHz and with 16 bits per sample by the Kay-Pentax Stroboscopy System (Laryngeal strobe model 9400, High definition digital video capture model 9310, Kay-PENTAX Inc., Lincoln Park, New Jersey) microphone (while the stroboscope was off) and by a standard omnidirectional microphone. Their voices were analyzed automatically using a script from the *Praat* software,[15] where intensity, fundamental frequency,[14] jitter, shimmer, and harmonic to noise ratio (HNR) were extracted.

By assuming that the VF system can be approximated as a double mass-spring-damper model which is mathematically expressed as a second-order differential equation, we focused on the transient response, from the very beginning of phonation until the oscillation stabilization. Recently, Lucero et al.[16] extended the standard lumped-mass model to capture the case when the left and right VF oscillate asynchronously as in the case of VF paralysis and paresis. They proposed to use two coupled second-order systems, where each system represents 1 VF, and they analyzed the steady-state solution for the case

when the VFs oscillate asynchronously. In our analysis, we assume the same coupled model, but we are focused on the transient solution (rather than the steady-state solution), and we correlate the solution to GCI. In order to analyze the transient response of a system, the input to the system has to be a *unit step function.* This function is formally defined as being 0 until some point in time where it abruptly changes to the value 1. We assume that a production of sustained vowels immediately after complete silence can be considered a unit step function.

According to the control theory, such a coupled second-order system has two types of possible solutions: underdamped and overdamped. These solutions depend on the values of the system parameters: the masses of the VF tissues, the constant of the springs, that is, the muscle elasticity of the VF, and the viscosity of the damper, that is, the elasticity of the VF tissue. The underdamped solution produces an output which has an overshoot, namely, the output exceeding its final, steady-state value. Similarly, the overdamped solution yields an output with an undershoot, that is, the output is always less than the steady-state solution. See Figure 1 for few examples.

In our analysis, we extracted the fundamental frequency extracted every frame of 5 msec. Denote by $f(t)$ the fundamental frequency estimation at time frame t. We define the *Time-to-Stabilization* measurement, which is the time required for the fundamental frequency to reach its steady-state, as follows:

$$T_s = arg\ \min_{t \in T}\{\ |f(t+i) - f(t+i+1)| \le \beta, \forall i \in \{0, \cdots, 3\}\ \} \tag{1}$$

In words, we ran over the fundamental frequency values consecutively and found the time in which the next four fundamental frequency values do not change more than β=10% (error band). The unit of measurement for this quantity is time and specifically in our analysis it was measured in seconds (Figure 2A).

We define the *Slope-at-Stabilization* as the slope between the Time-of-Stabilization and the start of phonation time t_0

$$S_s = \frac{f(t_s) - f(t_0)}{t_s - t_0}. \tag{2}$$

The unit of measure of Slope-at-Stabilization is frequency over time and in our analysis was measured in Hz/sec (Figure 2A).

The *Area-of-Stability* is defined as the integral over the fundamental frequency from the start of phonation to the Time-of-Stabilization. That is

$$A_s = \int_{t_0}^{t_s} f(t)dt \tag{3}$$

However, since the fundamental frequency is not a continuous function but rather estimated discretely every 5 msec, the above property was computed iteratively using the trapezoidal rule for approximating the integral (Figure 2B). The unit of measure for this quantity is frequency times time, and in our analysis it was measured by Hz × sec.

All participants were asked to complete a GFI questionnaire (the scale ranges from 0 to 20, where a higher score means more symptoms related to GCI)[17] and a VHI questionnaire (the scale ranges from 0 to 120, where a higher score means more voice symptoms)0[18,19] Patients with vocal cord paralysis were asked to answer these questionnaires twice; before and after in-office injection medialization laryngoplasty.

All patients underwent a fiberoptic evaluation before and after treatment by injection. The fiberoptic evaluation was done by expert clinicians, who subjectively rated the quality of the gap between the closed VFs.

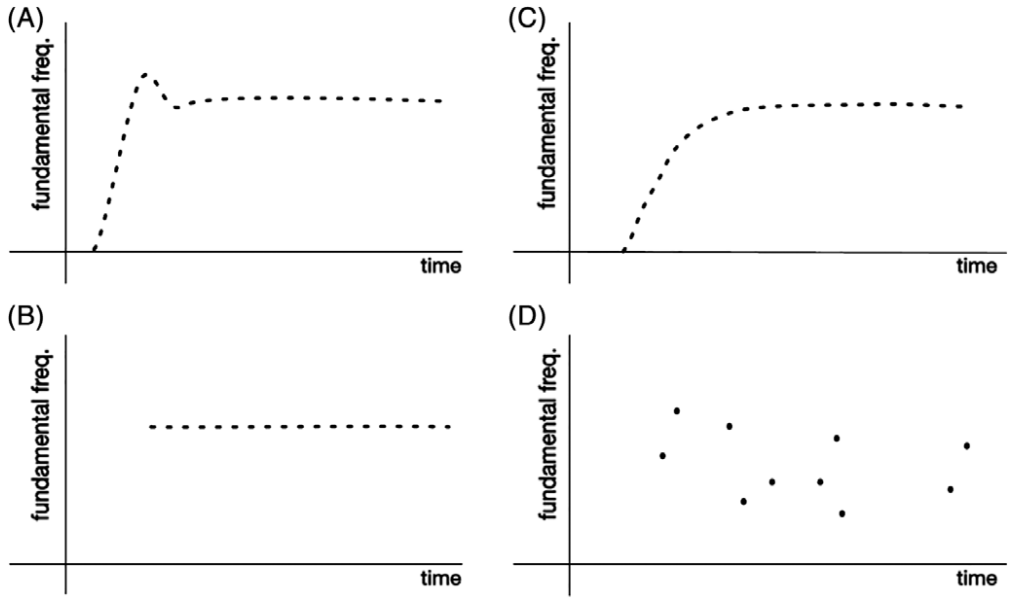


FIGURE 1 Fundamental frequency contours. An illustration of the fundamental frequency over time. Control group (A, overshoot type I; B, overshoot: II) and patient group (C, overshoot; D, only a few scattered values of fundamental frequency)

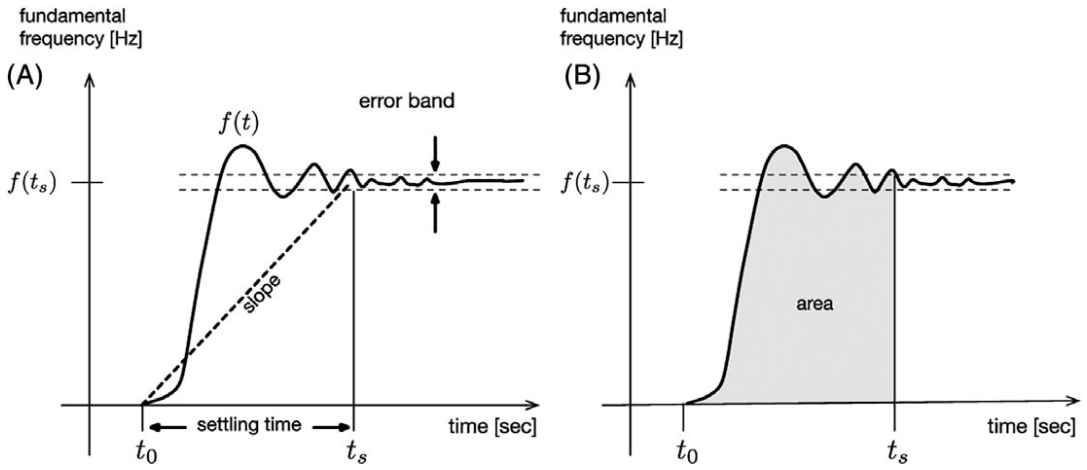


FIGURE 2 The fundamental frequency contour from initiation of voice (t_0) to settling time (t_s). Settling time (Time-to-Stabilization), slope (Slope-at- Stabilization) and area (Area-of Stability)

TABLE 1 Objective and subjective voice measurements of the control group and the patients with vocal fold paralysis before treatment

Voice measurements	Control	Before	diff	*P* value
VHI (scale 0-120)	5.1	89	−83.9	0.000
GFI (scale 0-20)	1.2	16.9	−15.7	0.000
VAS (scale 0-5)	1.1	4.1	−3	0.000
GRBAS (scale 0-15)	0.6	12.3	−11.7	0.000
Jitter (%)	0.8	3.8	−3	0.000
Shimmer (%)	7.1	14	−6.9	0.000
HNR (decibels)	14.3	10.2	4.1	0.028
MPT (seconds)	17.9	5.6	12.2	0.000
S to Z ratio	0.9	2.1	−1.3	0.006
Slope at Stability (Hz/sec)	−2.3	1.6	−3.9	0.000
Time to Stabilize (sec)	8.8	39.4	−30.6	0.000
Area of Stability (Hz × sec)	2	4.1	−2.1	0.005
Observations	18	24		

Voice samples were recorded and evaluated by the clinician using the visual analog scale (VAS) voice scale (with a range from 0 to 5, five being the worst voice) and the GRABS voice scale (with a range from 0 to 15, where the higher the score is, the worse the voice is), which is a standard method to describe voice quality.[20,21] We also measured maximal phonation time (MPT) and S/Z ratio frequently.

TABLE 2 Objective and subjective voice measurements of patients with vocal fold paralysis before and after treatment

VHI (scale 0-120)	Before	After	diff	*P* value
GFI (scale 0-20)	89	32.7	56.3	0.000
VAS (scale 0-5)	16.9	7.3	9.6	0.000
GRBAS (scale 0-15)	4.1	2.2	1.9	0.000
Jitter (%)	12.3	4.9	7.4	0.000
Shimmer (%)	3.8	1.9	2	0.003
HNR (decibels)	14	9.2	4.7	0.007
MPT (seconds)	10.2	13.8	−3.5	0.052
S to Z ratio	5.6	8.9	−3.3	0.026
VHI (scale 0-120)	2.1	1.3	0.8	0.033
Slope at Stability (Hz/sec)	1.6	−1	2.6	0.000
Time to Stabilize (sec)	39.4	15.1	24.3	0.000
Area of Stability (Hz × sec)	4.1	2.3	1.8	0.013
Observations	24	24		

TABLE 3 Objective and subjective voice measurements of the control group and of patients with vocal fold paralysis after treatment

VHI (scale 0-120)	Control	After	Diff	*P* value
GFI (scale 0-20)	5.1	32.7	−27.7	0.000
VAS (scale 0-5)	1.2	7.3	−6.1	0.000
GRBAS (scale 0-15)	1.1	2.2	−1.1	0.000
Jitter (%)	0.6	4.9	−4.4	0.000
Shimmer (%)	0.8	1.9	−1	0.011
HNR (decibels)	7.1	9.2	−2.2	0.143
MPT (seconds)	14.3	13.8	0.6	0.723
S to Z ratio	17.9	8.9	8.9	0.000
VHI (scale 0-120)	0.9	1.3	−0.4	0.076
Slope at Stability (Hz/sec)	−2.3	−1	−1.3	0.036
Time to Stabilize (sec)	8.8	15.1	−6.3	0.061
Area of Stability (Hz × sec)	2.0	2.3	−0.4	0.297
Observations	18	24		

4 | STATISTICAL ANALYSIS

Descriptive statistics for all subjects were generated for all measures, including means and SD for continuous measures

TABLE 4 Correlation between the proposed investigational methods and the traditional Objective and subjective voice measurements

Voice measurements	Slope at Stabilization (Hz/sec)	*P* value	Time to Stabilization (sec)	*P* value	Area of stabilization (Hz × sec)	*P* value
VHI	0.60	0.00	0.46	0.00	0.32	0.01
GFI	0.57	0.00	0.49	0.00	0.34	0.01
VAS	0.69	0.00	0.6	0.00	0.48	0.00
GRBAS	0.69	0.00	0.64	0.00	0.48	0.00
Jitter	0.41	0.00	0.70	0.00	0.67	0.00
Shimmer	0.33	0.01	0.51	0.00	0.45	0.00
HNR	−0.24	0.06	−0.56	0.00	−0.52	0.00
MPT	−0.56	0.00	−0.37	0.01	−0.32	0.02
S to Z ratio	0.30	0.04	0.04	0.79	0.00	0.99

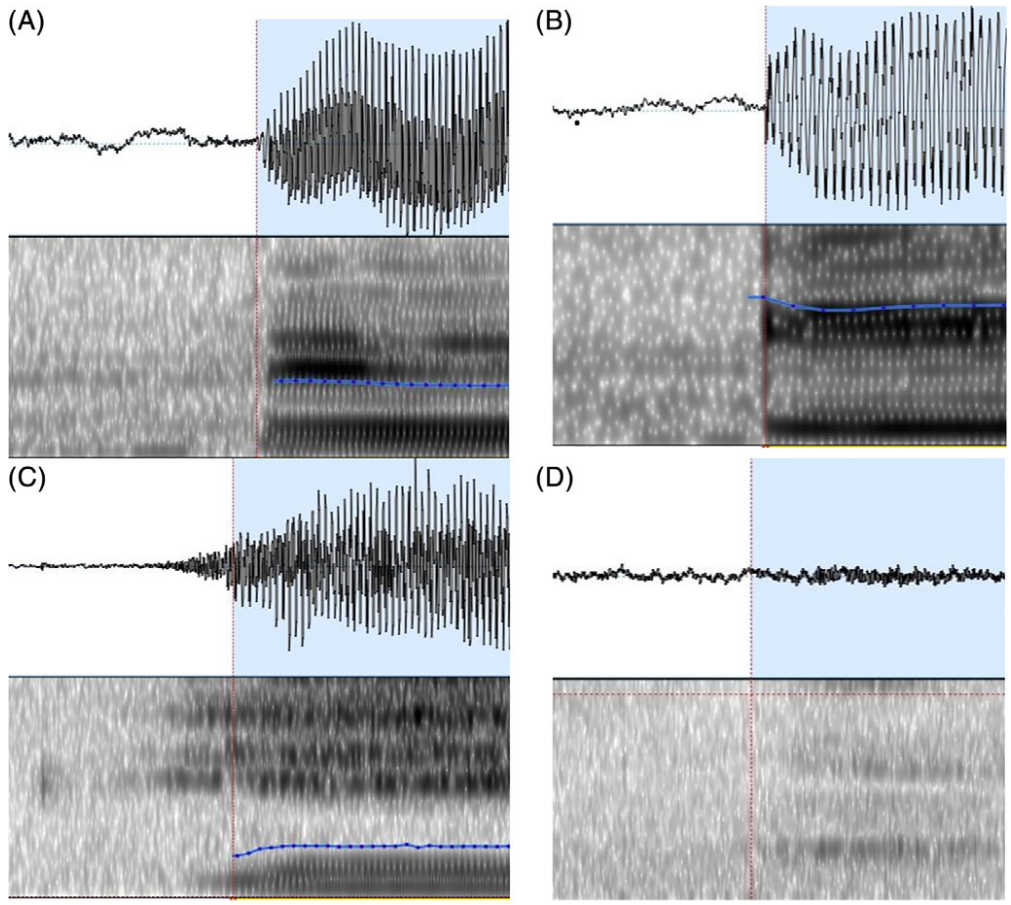


FIGURE 3 Each picture consists of two panels. The upper panel is the time signal and the lower panel corresponds to the spectrum. Four fundamental frequency contours (blue) are noticed: (A) and (B) show a rapid increase or a positive notch immediately before fundamental frequency stabilization (overshoot). In the pathological group, an uprising curve was observed until stabilization took place—undershoot (C) and no pattern (D) [Color figure can be viewed at wileyonlinelibrary.com]

using Stata 14 for Windows 10. Next, two sample *t* tests were performed to compare different groups, and the Pearson correlation coefficient was used to establish if there is a relationship between variables.

5 | RESULTS

First, we validate our proposed measurements by reproducing known results of voice quality, related questionnaires, and objective voice analysis parameters. We present results for GFI and VHI, as well as for VAS, GRABS, MPT, and the S/Z ratio. We then describe our findings regarding the transient response of the fundamental frequency contour. All the results are given for patients before and after injection, and for healthy control subjects.

5.1 | Voice quality-related questionnaires and objective voice analysis parameters

Thirty-eight subjects were included in this study. The average age of the control group ($n = 18$) was 50.56 (SD 17.49)

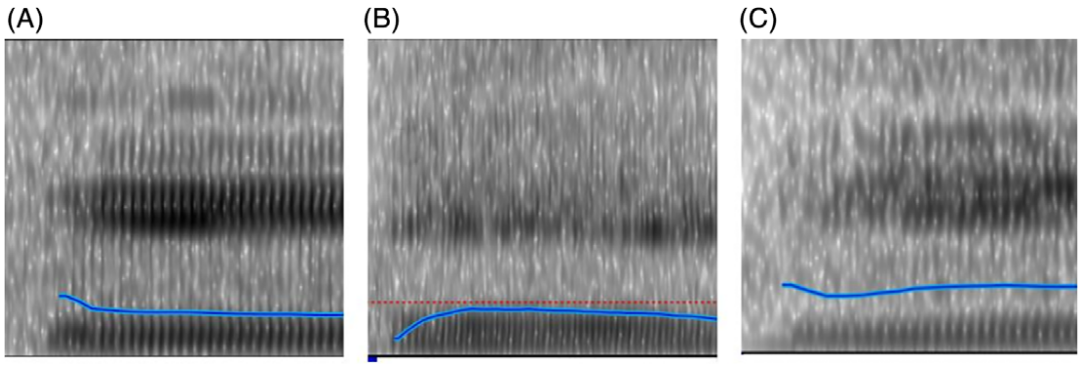


FIGURE 4 Fundamental frequency contour (blue) on the background of the spectrogram of: A, a healthy control group; B, a patient before injection; and C, the same patient after carboxymethylcellulose (RADIESSE Voice Gel} Gel injection [Color figure can be viewed at wileyonlinelibrary.com]

years, the age range was 29-85, and 8 were men. The mean GFI score was 1.2 ± 2.1 and the VHI score was 5.1 ± 7.4. Subjective voice analysis using VAS was 1.1 ± 0.2 and GRABS was 0.6 ± 1.0. Objective voice analyses measured by jitter, shimmer, and HNR were 0.8 ± 0.4, 7.1 ± 3.0, and 14.3 ± 3.9, respectively. MPT and the S/Z ratio were 17.9 ± 1.0 and 0.9 ± 0.2, respectively.

Out of the 20 patients with vocal cord paralysis, 12 were men. The age range was 28-81 with an average of 57.4 (SD 16.74) years. The mean GFI score was 16.9 ± 4.2 and the VHI score was 89.0 ± 23.8. Subjective voice analysis using VAS was 4.1 ± 0.7 and GRABS was 12.3 ± 2.9. Objective voice analyses measured by jitter, shimmer, and HNR were 3.8 ± 2.7, 14.0 ± 6.3, and 10.2 ± 6.4, respectively. MPT and the S/Z ratio were 5.6 ± 2.1 and 0.9 ± 0.2, respectively.

There was no statistically significant difference between the control group and the patient group concerning age and gender. Table 1 demonstrates the symptom and voice analysis differences between the groups. All voice measurements were found to be statistically significant.

After injection medialization laryngoplasty, patient complaints and subjective and objective voice analysis improved dramatically. The mean GFI score was 7.3 ± 4.9 and the VHI score was 32.7 ± 21.4. Subjective voice analysis using VAS was 2.2 ± 0.7 and GRABS was 4.9 ± 2.4. Objective voice analyses measured by jitter, shimmer, and HNR were 1.9 ± 1.5, 9.2 ± 5.2 and 13.8 ± 5.8, respectively, are given in Table 2.

Table 3 compares the control subject to the patient receiving injection medialization laryngoplasty to close the gap between the VFs. Although there is an improvement and similarity between the control group and the patient group, the control group is still superior for all voice and symptom parameters.

Table 4 presents the pairwise correlation with the appropriate *P*-value between the traditional acoustic analysis paradigm and the new proposed measurements. We can see that all the traditional measurements are found to be overall highly correlated with the new proposed measurements (expect for S/Z ratio).

5.2 | Transient response analysis of the fundamental frequency contour

Next, we evaluated the fundamental frequency contours of participants in the study as described in Figure 1. In the control group, all patients demonstrated overshoot fundamental frequency contour (type I when there is a notch exceeding the final fundamental frequency or type II when there is a flat curve). In the patient group, 14 patients demonstrated undershoot fundamental frequency contour and 6 patients demonstrated no or scattered fundamental frequency contour (Figure 1). Based on this observation we assessed the proposed measurements: Time-to-Stabilization, Slope-at-Stabilization, and Area-of-Stability in a similar way to the previous analysis. A comparison between the proposed measurements and the traditional voice evaluation can be found in Tables 1–4.

There is a statistically significant difference between the control group and the patient group before injection (Table 1), and there is a substantial and statistically significant improvement in the patient group before and after the treatment (Table 2). After injection medialization to close the gap between the VFs, the patients became more similar to the control group (Table 3), however still statistically different. Out of the three measurements, we suggest the Area-of-Stability to be the most accurate, followed by Time-to-Stabilization. One of the reasons that the Area-of-Stability is superior to the other measurements is due to its robustness to measurement errors. Moreover, we found that the proposed measures are highly correlated with VHI, GFI, VAS, jitter, shimmer, and HNR.

6 | DISCUSSSION

Interest in the field of voiced speech production can be traced back nearly a century[22] and arises primarily from a desire to understand the basic physics of phonation, as well as to provide diagnosis and treatment to individuals that suffer from any number of vocal pathologies.

A large body of work has been based on control theory implicitly or explicitly to model the oscillation of the VFs which were based on a mass-spring-damper model.[5,6] This model, however, simplifies and approximates the very complicated physics of the vocal cords. Unlike previous work, we propose to study the *transient* solution of the set of differential equations of this model to support our acoustic evidence of the fundamental frequency transient behavior, as done in the modern control theory.

Specifically, we presented an analysis of the fundamental frequency contour at the area of the phonation onset. The short duration between VFs vibration initiation until a stable vibration is achieved depends on the distance between them during adduction. We analyzed the fundamental frequency contour of healthy subjects and revealed that the fundamental frequency can reach its final steady-state value almost instantly, creating a flat curve, or it may exceed the final fundamental frequency, a phenomenon known as *overshoot.*[23] When we applied this analysis to individuals who suffer from GCI due to VF paralysis the transitory values of the fundamental frequency were below the stable value, and this phenomenon is known as *undershoot.*[23] In some patients, the distance between the two VFs was too wide with no pitch production at all (Figure 3). The fundamental frequency contour of patients with VF paralysis and a gap between the VF that received injection medialization laryngoplasty better resembles the control group contour. Figure 4 presents a patient with VF paralysis before and after

injection. The curves changed from undershoot to overshoot curves, similar to the control group.

The Time-to-Stabilization of normal controls is much lower than of patients with vocal insufficiency. The Slope-at-Stabilization of normal control is a negative slope as it describes a decrease from the overshoot to the steady-state value, whereas in the patients suffering from VF paralysis the Slope-at-Stabilization was positive as the fundamental frequency contour was rising from a low value to a steady state.

Proper VF closure is not the only factor contributing to pitch generation—breath support, VF stiffness, and elasticity play major factors as well. We tried to isolate our observation to absolute normal individual (based on questionnaires, voice analysis, and video-stroboscopic examination) and patients whose only factor influencing their voice was VF paralysis with a gap between the VF during phonation. Of course, with time the paralyzed vocal cord loses its elasticity and vibratory characteristics; but in our opinion, these important contributors are incorporated in our new observations.

While other voice analysis factors also demonstrated differences between the control group and patients with VF paralysis, our new measurements are more effective in detecting the improvement of patients' voices after rehabilitating the glottal gap by injection medialization laryngoplasty. Our theory is supported by works with high speed video cameras examining VF vibration onset time with the use of digital kymography. This method was used to measure the time duration from near approximation of the VFs to the first observed oscillation and the time duration from the beginning of the VFs' oscillation to its steady state. Both of these parameters were prolonged in glottis insufficiency cases, including VF paralysis.[24] Our method does not require sophisticated equipment such as high-speed cameras or digital kymography and can be calculated from every commercial voice analysis system.

7 | CONCLUSIONS

Voice analysis has a limited role in a day-to-day voice clinic. New parameters that correlate with specific clinical findings and applications are needed and important. Voice parameters found for GCI patients can be used to quantify condition severity and can be used to verify treatment success. We present a new analysis of the fundamental frequency contour at the phonation onset for individuals with GCI. These measurements are quantitative, noninvasive, and inexpensive.

Future studies will include other VF pathologies with a wider range of severity and etiology, including VF cysts, polyps, nodules, and edema. Further development will also involve validation of these measurements against airflow measurements and high-speed cameras.

To date, there are several acoustic measures that are often used clinically to assess the voice quality in general. Our work concentrated on designing new metrics (time to stabilization, slope at stabilization, and area under the curve) that directly focused on the specific voice pathology, namely, VF paralysis and paresis. To support this, we correlated our acoustic findings with the transition solutions of the mass-spring-damper model.

In future work, we will conduct a controlled randomized trail to assess how are the new proposed matrices improve the diagnosis beyond the traditional measurements, and add a tool which can easily be diagnose VF paralysis and paresis and distinguish them from other possible pathologies.

CONFLICT OF INTEREST

The authors declare that they have no conflicts of interest.

ORCID

Jacob T. Cohen https://orcid.org/0000-0002-1160-0234

REFERENCES

1. Orlikoff RF, Deliyski DD, Baken RJ, Watson BC. Validation of a glottographic measure of vocal attack. *J Voice*. 2009;23(2):164-168.
2. Belafsky PC, Postma GN, Reulbach TR, Holland BW, Koufman JA. Muscle tension dysphonia as a sign of underlying glottal insufficiency. *Otolaryngol Head Neck Surg*. 2002;127(5):448-451.
3. Van Den Berg J. Myoelastic-aerodynamic theory of voice production. *J Speech Hear Res*. 1958;1(3):227-244.
4. Chan RW, Titze IR. Viscoelastic shear properties of human vocal fold mucosa: measurement methodology and empirical results. *J Acoust Soc Am*. 1999;106:2008-2021.
5. Titze IR. The physics of small-amplitude oscillation of the vocal Folds. *J. Acoust. Soc. Am*. 1988;83:1536-1552.
6. Lucero JC, Koenig LL, Lourenço KG, Ruty N, Pelorson X. A lumped mucosal wave model of the vocal folds revisited: recent extensions and oscillation hysteresis. *J Acoust Soc Am*. 2011;129(3):1568-1579.
7. Burns R. *Advanced Control Engineering*. 1st ed. Oxford, UK: Butterworth-Heinemann; 2001.
8. Ogata K. *Modern Control Engineering*. 5th ed. Upper Saddle River, NJ, USA: Prentice-Hall; 2010.
9. Nise NK. *Control System Engineering*. 6th ed. Hoboken, NJ, USA: John Wiley & Sons Inc.; 2011.
10. Flege JE. Laryngeal timing and phonation onset in utterance-initial English stops. *J Phonetics*. 1982;10:177-192.
11. Zhang Z. Characteristics of phonation onset in a two-layer vocal fold model. *J Acoust Soc Am*. 2009;125(2):1091-1102.
12. Braunschweig T, Flaschka J, Schelhorn-Neise P, Döllinger M. High-speed video analysis of the phonation onset, with an application to the diagnosis of functional dysphonias. *Med Eng Phys*. 2008;30(1):59-66.
13. Watson BC, Baken RJ, Roark RM. Effect of voice onset type on vocal attack time. *J Voice*. 2016 Jan;30(1):11-14.
14. Steinhauer K, Grayhack JP, Smiley-Oyen AL, Shaiman S, McNeil MR. The relationship among voice onset, voice quality, and fundamental frequency: a dynamical perspective. *J Voice*. 2004;18:432-442.
15. Boersma, Paul & Weenink, David (2018). Praat: doing phonetics by computer [Computer program]. Version 6.0.37, Retrieved from http://www.praat.org/.
16. Lucero JC, Schoentgen J, Haas J, Luizard P, Pelorson X. Self-entrainment of the right and left vocal fold oscillators. *J Acoust Soc Am*. 2015;137(4):2036-2046.
17. Bach KK, Belafsky PC, Wasylik K, Postma GN, Koufman JA. Validity and reliability of the glottal function index. *Arch Otolaryngol Head Neck Surg*. 2005;131(11):961-964.
18. Jacobson BH, Johnson A, Grywalski C, et al. The voice handicap index (VHI) development and validation. *Am J Speech Lang Pathol*. 1997;6:66-70.
19. Rosen CA, Murry T, Zinn A, Zullo T, Sonbolian M. Voice handicap index change following treatment of voice disorders. *J Voice*. 2000;14(4):619-623.

20. Hirano M. *Clinical Examination of Voice*. New York, NY: Springer; 1981.

21. De Bodt M, Wuyts F, Van de Heyning P, Croeckx C. Test-retest study of GRBAS scale. *J Voice*. 1997;11:74-80.

22. Wegel RL. Theory of vibration of the larynx. *J Acoust Soc Am*. 1930;1: 1-21.

23. Benjamin C Kuo & Golnaraghi F (2003). (8th ed.). New York, NY: Wiley. p. 253.

24. Woo P. High-speed imaging of vocal fold vibration onset delay: normal versus abnormal. *J Voice*. 2017;31(3):307-312.

How to cite this article: Cohen JT, Cohen A, Benyamini L, Adi Y, Keshet J. Predicting glottal closure insufficiency using fundamental frequency contour analysis. *Head & Neck*. 2019;1–8. https://doi.org/10.1002/hed.25709

Chapter 3

Published Papers (Robustness of Neural Networks to Adversarial and Out-of-Distribution Examples)

Houdini: Fooling Deep Structured Visual and Speech Recognition Models with Adversarial Examples

arXiv:1707.05373v1 [stat.ML] 17 Jul 2017

Houdini: Fooling Deep Structured Prediction Models

Moustapha Cisse
Facebook AI Research
moustaphacisse@fb.com

Yossi Adi*
Bar-Ilan University, Israel
yossiadidrum@gmail.com

Natalia Neverova*
Facebook AI Research
nneverova@fb.com

Joseph Keshet
Bar-Ilan University, Israel
jkeshet@cs.biu.ac.il

Abstract

Generating adversarial examples is a critical step for evaluating and improving the robustness of learning machines. So far, most existing methods only work for classification and are not designed to alter the true performance measure of the problem at hand. We introduce a novel flexible approach named Houdini for generating adversarial examples specifically tailored for the final performance measure of the task considered, be it *combinatorial and non-decomposable*. We successfully apply Houdini to a range of applications such as speech recognition, pose estimation and semantic segmentation. In all cases, the attacks based on Houdini achieve higher success rate than those based on the traditional surrogates used to train the models while using a less perceptible adversarial perturbation.

1 Introduction

Deep learning has redefined the landscape of machine intelligence [22] by enabling several breakthroughs in notoriously difficult problems such as image classification [20, 16], speech recognition [2], human pose estimation [35] and machine translation [4]. As the most successful models are permeating nearly all the segments of the technology industry from self-driving cars to automated dialog agents, it becomes critical to revisit the evaluation protocol of deep learning models and design new ways to assess their reliability beyond the traditional metrics. Evaluating the robustness of neural networks to adversarial examples is one step in that direction [32]. Adversarial examples are synthetic patterns carefully crafted by adding a peculiar noise to legitimate examples. They are indistinguishable from the legitimate examples by a human, yet they have demonstrated a strong ability to cause catastrophic failure of state of the art classification systems [12, 25, 21]. The existence of adversarial examples highlights a potential threat for machine learning systems at large [28] that can limit their adoption in security sensitive applications. It has triggered an active line of research concerned with understanding the phenomenon [10, 11], and making neural networks more robust [29, 7] .

Adversarial examples are crucial for reliably evaluating and improving the robustness of the models [12]. Ideally, they must be generated to alter the task loss unique to the application considered directly. For instance, an adversarial example crafted to attack a speech recognition system should be designed to *maximize* the word error rate of the targetted system. The existing methods for generating adversarial examples exploit the gradient of a given differentiable loss function to guide the search in the neighborhood of legitimates examples [12, 25]. Unfortunately, the task loss of several structured prediction problems of interest is a combinatorial non-decomposable quantity that is not amenable to gradient-based methods for generating adversarial example. For example, the metric for evaluating human pose estimation is the *percentage of correct keypoints* (normalized by the head). Automatic

*equal contribution

Figure 1: We cause the network to generate a *minion* as segmentation for the adversarially perturbed version of the original image. Note that the original and the perturbed image are indistinguishable.

speech recognition systems are assessed using their *word (or phoneme) error rate*. Similarly, the quality of a semantic segmentation is measured by the *intersection over union* (IOU) between the ground truth and the prediction. All these evalutation measures are non-differentiable.

The solutions for this obstacle in supervised learning are of two kinds. The first route is to use a consistent differentiable surrogate loss function in place of the task loss [5]. That is a surrogate which is guaranteed to converge to the task loss asymptotically. The second option is to directly optimize the task loss by using approaches such as Direct Loss Minimization [14]. Both of these strategies have severe limitations. (1) The use of differentiable surrogates is satisfactory for classification because the relationship between such surrogates and the classification accuracy is well established [34]. The picture is different for the above-mentioned structured prediction tasks. Indeed, there is no known consistency guarantee for the surrogates traditionally used in these problems (e.g. the connectionist temporal classification loss for speech recognition) and designing a new surrogate is nontrivial and problem dependent. At best, one can only expect a high positive correlation between the proxy and the task loss. (2) The direct minimization approaches are more computationally involved because they require solving a computationally expensive loss augmented inference for each parameter update. Also, they are notoriously sensitive to the choice of the hyperparameters. Consequently, it is harder to generate adversarial examples for structured prediction problems as it requires significant domain expertise with little guarantee of success when surrogate does not tightly approximate the task loss.

Results. In this work we introduce Houdini, the first approach for fooling any gradient-based learning machine by generating adversarial examples directly tailored for the task loss of interest be it combinatorial or non-differentiable. We show the tight relationship between Houdini and the task loss of the problem considered. We present the first successful attack on a deep Automatic Speech Recognition (ASR) system, namely a DeepSpeech-2 based architecture [1], by generating adversarial audio files not distinguishable from legitimate ones by a human (as validated by an ABX experiment). We also demonstrate the transferability of adversarial examples in speech recognition by fooling Google Voice in a black box attack scenario: an adversarial example generated with our model and not distinguishable from the legitimate one by a human leads to an invalid transcription by the Google Voice application (see Figure 9). We also present the first successful untargeted and targetted attacks on a deep model for human pose estimation [26]. Similarly, we validate the feasibility of untargeted and targetted attacks on a semantic segmentation system [38] and show that we can make the system hallucinate an arbitrary segmentation of our choice for a given image. Figure 1 shows an experiment where we cause the network to hallucinate a *minion*. In all cases, our approach generates better quality adversarial examples than each of the different surrogates (expressly designed for the model considered) without additional computational overhead thanks to the analytical gradient of Houdini.

2 Related Work

Adversarial examples. The empirical study of Szegedy et al. [32] first demonstrated that deep neural networks could achieve high accuracy on previously unseen examples while being vulnerable to small adversarial perturbations. This finding has recently aroused keen interest in the community [12, 28, 32, 33]. Several studies have subsequently analyzed the phenomenon [10, 31, 11] and various approaches have been proposed to improve the robustness of neural networks [29, 7]. More closely related to our work are the different proposals aiming at generating better adversarial examples [12, 25]. Given an input (train or test) example (x, y), an adversarial example is a perturbed version of the original pattern $\tilde{x} = x + \delta_x$ where δ_x is small enough for $\tilde{x}$ to be undistinguishable from x by a

human, but causes the network to predict an incorrect target. Given the network g_θ (where θ is the set of parameters) and a p-norm, the adversarial example is formally defined as:

$$\tilde{x} = \underset{\tilde{x}:\|\tilde{x}-x\|_p \leq \epsilon}{\operatorname{argmax}} \ \ell\big(g_\theta(\tilde{x}), y\big) \tag{1}$$

where ϵ represents the strength of the adversary. Assuming the loss function $\ell(\cdot)$ is differentiable, Shaham et al. [31] propose to take the first order taylor expansion of $x \mapsto \ell(g_\theta(x), y)$ to compute δ_x by solving the following simpler problem:

$$\tilde{x} = \underset{\tilde{x}:\|\tilde{x}-x\|_p \leq \epsilon}{\operatorname{argmax}} \ \big(\nabla_x \ell(g_\theta(x), y)\big)^T (\tilde{x} - x) \tag{2}$$

When $p = \infty$, then $\tilde{x} = x + \epsilon \text{sign}(\nabla_x \ell(g_\theta(x), y))$ which corresponds to the *fast gradient sign method* [12]. If instead $p = 2$, we obtain $\tilde{x} = x + \epsilon \nabla_x \ell(g_\theta(x), y)$ where $\nabla_x \ell(g_\theta(x), y)$ is often normalized. Optionally, one can perform more iterations of these steps using a smaller norm. This more involved strategy has several variants [25]. These methods are concerned with generating adversarial examples assuming a differentiable loss function $\ell(\cdot)$. Therefore they are not directly applicable to the task losses of interest. However, they can be used in combination with our proposal which derives a consistent approximation of the task loss having an analytical gradient.

Task Loss Minimization. Recently, several works have focused on directly minimizing the task loss. In particular, McAllester et al. [24] presented a theorem stating that a certain perceptron-like learning rule, involving feature vectors derived from loss-augmented inference, directly corresponds to the gradient of the task loss. While this algorithm performs well in practice, it is extremely sensitive to the choice of its hyper-parameter and needs two inference operations per training iteration. Do et al. [9] generalized the notion of the ramp loss from binary classification to structured prediction and proposed a tighter bound to the task loss than the structured hinge loss. The update rule of the structured ramp loss is similar to the update rule of the direct loss minimization algorithm, and similarly it needs two inference operations per training iteration. Keshet et al. [19] generalized the notion of the binary probit loss to the structured prediction case. The probit loss is a surrogate loss function naturally resulted in the PAC-Bayesian generalization theorems. it is defined as follows:

$$\bar{\ell}_{probit}(g_\theta(x), y) = \mathbb{E}_{\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})} \left[\ell(y, g_{\theta + \boldsymbol{\epsilon}}(x))\right] \tag{3}$$

where $\boldsymbol{\epsilon} \in \mathbb{R}^d$ is a d-dimensional isotropic Normal random vector. [18] stated finite sample generalization bounds for the structured probit loss and showed that it is strongly consistent. Strong consistency is a critical property of a surrogate since it guarantees the tight relationship to the task loss. For instance, an attacker of a given system can expect to deteriorate the task loss if she deteriorates the consistent surrogate of it. The gradient of the structured probit loss can be approximated by averaging over samples from the unit-variance isotropic normal distribution, where for each sample an inference with perturbed parameters is computed. Hundreds to thousands of inference operations are required per iteration to gain stability in the gradient computation. Hence the update rule is computationally prohibitive and limits the applicability of the structured probit loss despite its desirable properties.

We propose a new loss named Houdini. It shares the desirable properties of the structured probit loss while not suffering from its limitations. Like the structured probit loss and unlike most surrogates used in structured prediction (e.g. structured hinge loss for SVMs), it is tightly related to the task loss. Therefore it allows to reliably generate adversarial examples for a given task loss of interest. Unlike the structured probit loss and like the smooth surrogates, it has an analytical gradient hence require only a single inference in its update rule. The next section presents the details of our proposal.

3 Houdini

Let us consider a neural network g_θ parameterized by θ and the task loss of a given problem $\ell(\cdot)$. We assume $\ell(y, y) = 0$ for any target y. The score output by the network for an example (x, y) is $g_\theta(x, y)$ and the network's decoder predicts the highest scoring target:

$$\hat{y} = y_\theta(x) = \underset{y \in \mathcal{Y}}{\operatorname{argmax}} \ g_\theta(x, y). \tag{4}$$

Using the terminology of section 2, finding an adversarial example fooling the model g_θ with respect to the task loss $\ell(\cdot)$ for a chosen p-norm and noise parameter ϵ boils down to solving:

$$\tilde{x} = \underset{\tilde{x}:\|\tilde{x}-x\|_p \leq \epsilon}{\operatorname{argmax}} \ \ell\big(y_\theta(\tilde{x}), y\big) \tag{5}$$

The task loss is often a combinatorial quantity which is hard to optimize, hence it is replaced with a differentiable *surrogate loss*, denoted $\ell(y_\theta(\tilde{x}), y)$. Different algorithms use different surrogate loss functions: structural SVM uses the structured hinge loss, Conditional random fields use the log loss, etc. We propose a surrogate named Houdini and defined as follows for a given example (x, y):

$$\bar{\ell}_H(\theta, x, y) = \mathbb{P}_{\gamma \sim \mathcal{N}(0,1)} \Big[g_\theta(x, y) - g_\theta(x, \hat{y}) < \gamma \Big] \cdot \ell(\hat{y}, y) \tag{6}$$

In words, Houdini is a product of two terms. The first term is a stochastic margin, that is the probability that the difference between the score of the actual target $g_\theta(x, y)$ and that of the predicted target $g_\theta(x, \hat{y})$ is smaller than $\gamma \sim \mathcal{N}(0, 1)$. It reflects the confidence of the model in its predictions. The second term is the task loss, which given two targets is independent of the model and corresponds to what we are ultimately interested in maximizing. Houdini is a lower bound of the task loss. Indeed denoting $\delta g(y, \hat{y}) = g_\theta(x, y) - g_\theta(x, \hat{y})$ as the difference between the scores assigned by the network to the ground truth and the prediction, we have $\mathbb{P}_{\gamma \sim \mathcal{N}(0,1)}(\delta g(y, \hat{y}) < \gamma)$ is smaller than 1. Hence when this probability goes to 1, or equivalently when the score assigned by the network to the target $\hat{y}$ grows without a bound, Houdini converges to the task loss. This is a unique property not enjoyed by most surrogates used in the applications of interest in our work. It ensures that Houdini is a good proxy of the task loss for generating adversarial examples.

We can now use Houdini in place of the task loss $\ell(\cdot)$ in the problem 5. Following 2, we resort to a first order approximation which requires the gradient of Houdini with respect to the input x. The latter is obtained following the chain rule:

$$\nabla_x \big[\bar{\ell}_H(\theta, x, y) \big] = \frac{\partial \bar{\ell}_H(\theta, x, y)}{\partial g_\theta(x, y)} \frac{\partial g_\theta(x, y)}{\partial x} \tag{7}$$

To compute the RHS of the above quantity, we only need to compute the derivative of Houdini with respect to its input (the output of the network). The rest is obtained by backpropagation. The derivative of the loss with respect to the network's output is:

$$\nabla_g \Big[\mathbb{P}_{\gamma \sim \mathcal{N}(0,1)} \Big[g_\theta(x, y) - g_\theta(x, \hat{y}) < \gamma \Big] \ell(y, \hat{y}) \Big] = \nabla_g \left[\frac{1}{\sqrt{2\pi}} \int_{\delta g(y, \hat{y})}^{\infty} e^{-v^2/2} dv \, \ell(y, \hat{y}) \right] \tag{8}$$

Therefore, expanding the right hand side and denoting $C = 1/\sqrt{2\pi}$ we have:

$$\nabla_g \big[\bar{\ell}_H(\hat{y}, y) \big] = \begin{cases} -C \cdot e^{-|\delta g(y, \hat{y})|^2/2} \ell(y, \hat{y}), & g = g_\theta(x, y) \\ C \cdot e^{-|\delta g(y, \hat{y})|^2/2} \ell(y, \hat{y}), & g = g_\theta(x, \hat{y}) \\ 0, & \text{otherwise} \end{cases} \tag{9}$$

Equation 9 provides a simple analytical formula for computing the gradient of Houdini with respect to its input, hence an efficient way to obtain the gradient with respect to the input of the network x by backpropagation. The gradient can be used in combination with any gradient-based adversarial example generation procedure [12, 25] in two ways, depending on the form of attack considered. For an untargeted attack, we want to change the prediction of the network without preference on the final prediction. In that case, any alternative target y can be used (e.g. the second highest scorer as the target). For a targetted attack, we set the y to be the desired final prediction. Also note that, when the score of the predicted target is very close to that of the ground truth (or desired target), that is when $\delta g(y, \hat{y})$ is small as we expect from the trained network we want to fool, we have $e^{-|\delta g(y, \hat{y})|^2/2} \simeq 1$. In the next sections, we show the effectiveness of the proposed attack scheme on human pose estimation, semantic segmentation and automatic speech recognition systems.

4 Human Pose Estimation

We evaluate the effectiveness of Houdini loss in the context of adversarial attacks on neural models for human pose estimation. Compromising performance of such systems can be desirable for manipulating surveillance cameras, altering the analysis of crime scenes, disrupting human-robot interaction or fooling biometrical authentication systems based on gate recognition. The pose estimation task is formulated as follows: given a single RGB image of a person, determine correct 2D positions of several pre-defined key points which typically correspond to skeletal joints. In practice,

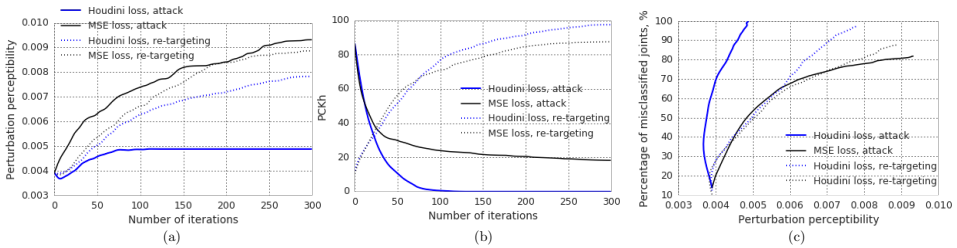


Figure 2: Convergence dynamics for pose estimation attacks: (a) perturbation perceptibility vs nb. iterations, (b) PCKh$^{0.5}$ vs nb. iterations, (c) proportion of re-positioned joints vs perceptibility.

Method	SSIM		Perceptibility	
	@PCKh=50	@PCKh$^{\text{lim}}$	@PCKh=50	@PCKh$^{\text{lim}}$
untargeted: MSE loss	0.9986	0.9954	0.0048	0.0093
untargeted: Houdini loss	0.9990	0.9987	0.0037	0.0048
targetted: MSE loss	0.9984	0.9959	0.0050	0.0089
targetted: Houdini loss	0.9982	0.9965	0.0049	0.0079

Table 1: Human pose estimation: comparison of empirical performance for adversarial attacks and adversarial pose re-targeting. PCKh=50 corresponds to 50% correctly detected keypoints and PCKh$^{\text{lim}}$ denotes the value upon convergence or after 300 iterations). (SSIM: the higher the better, perceptibility: the lower the better.)

the performance of such frameworks is measured by the percentage of correctly detected keypoints (PCKh), i.e. keypoints whose predicted locations are within a certain distance from the corresponding target positions: [3]:

$$\text{PCKh}^{\alpha} = \frac{\sum_{i=1}^{N} \mathbb{1}(\|y_i - \hat{y}_i\| < \alpha h)}{N}, \tag{10}$$

where $\hat{y}$ and y are the predicted and the desired positions of a given joint respectively, h is the head size of the person (known at test time), α is a threshold (set to 0.5), and N is the number of annotated keypoints. Pose estimations is a good example of a problem where we observe a discrepancy between the training objective and the final evaluation measure. Instead of directly minimizing the percentage of correctly detected keypoints, state-of-the-art methods rely upon a dense prediction of heat maps, i.e. estimation of probabilities of every pixel corresponding to each of keypoint locations. These models can be trained with binary cross entropy [6], softmax [15] or MSE losses [26] applied to every pixel in the output space, separately for each plane corresponding to every key point. In our first experiment, we attack a state-of-the-art model for single person pose estimation based on Hourglass networks [26] and aim to minimize the value of PCKh$^{0.5}$ metric given the minimal perturbation. For this task we choose $\hat{y}$ as:

$$\hat{y} = \underset{\tilde{y}:\|\tilde{p}-p\|>\alpha h}{\text{argmax}} \; g_\theta(x, \tilde{y}) \tag{11}$$

where p is the pixel coordinate on the heatmap corresponding to the argmax value of vector y. We perform the optimization iteratively till convergence with an update rule $\epsilon \cdot \frac{\nabla_x}{\|\nabla_x\|}$ where ∇_x are gradients with respect to the input and $\epsilon = 0.1$. Empirical comparison with a similar attack based on minimizing the training loss (MSE in this case) is given in the top part of Table 1. We perform the evaluations on the validation subset of MPII dataset [3] consisting of 3000 images and defined as in [26]. We evaluate the perceived degree of perturbation where perceptibility is expressed as $(\frac{1}{n}\sum(x'_i - x_i)^2)^{1/2}$, where x and x' are original and distorted images, and n is the number of pixels [32]. In addition, we report the structural similarity index (SSIM) [36] which is known to correlate well with the visual perception of image *structure* by a human. As shown in Table 1, adversarial examples generated with Houdini are more similar to the legitimate examples than the ones generated with the training proxy (MSE). For untargeted attacks optimized to convergence, the perturbation generated with Houdini is up to 50% less perceptible than the one obtained with MSE.

We measured the number of iterations required to completely compromise the pose estimation system (PCKh = 0) depending on the loss function used. The convergence dynamics illustrated in Fig. 1

(a) initial prediction (b) adversarial prediction (c) source image (d) perturbed image (e) noise

Figure 3: Example of disrupted pose estimation due to an adversarial attack exploiting Houdini loss: after 55 iterations, $\text{PCKh}^{0.5}$ dropped from 81.25 to 0.00 (final perturbation perceptibility is 0.0063).

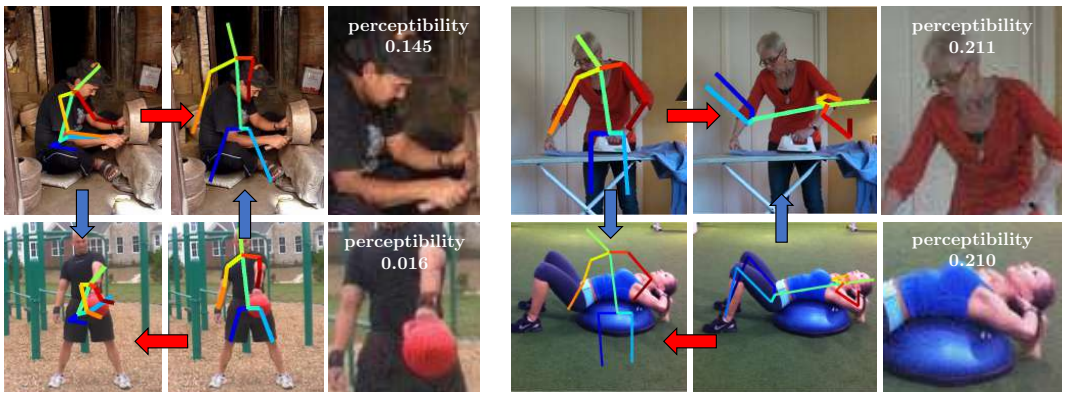


Figure 4: Examples of successful targetted attacks on a pose estimation system. Despite the important difference between the images selected, it is possible to make the network predict the wrong pose by adding an imperceptible perturbation.

show that employing Houdini makes it possible to completely compromise network performance given the target metric after several iterations (100). When using the training surrogate (MSE) for generating adversarial examples, we could not achieve a similar success level on the attacks even after 300 iterations of the optimization process. This observation underlines the importance of the loss function used to generate adversarial examples in structured prediction problems. We show an example in Figure 3. Interestingly, the predicted poses for the corrupted images still make an overall plausible impression, while failing the formal evaluation entirely. Indeed, the *mistakes* made by the compromised network appear natural and are typical for pose estimation frameworks: imprecision in localization, confusion between right and left limbs, combining joints from several people, etc.

In the second experiment, we perform a targetted attack in the form of pose transfer, i.e. we force the network to hallucinate an arbitrary pose (with success defined, as before, given the target metric $\text{PCKh}^{0.5}$). The experimental setup is as follows: for a given pair of images (i, j) we force the network to output the ground truth pose of the picture i when the input is image j and vice versa. This task is more challenging and depends on the similarity between the original and target poses. Surprisingly, targetted attacks are is still feasible even when the two ground truth poses are very different. Figure 4 shows an example where the model predicts the pose of a human body in horizontal position for an adversarially perturbed image depicting a standing person (and vice versa). A similar experiment with two persons in standing and sitting positions respectively is also shown in figure 4.

5 Semantic segmentation

Semantic segmentation uses another customized metric to evaluate performance, namely the mean Intersection over Union (mIoU) measure defined as averaging over classes the $\text{IoU} = \text{TP}/(\text{TP} + \text{FP} + \text{FN})$, where TP, FP and FN stand for true positive, false positive and false negative respectively, taken separately for each class. Compared to per-pixel accuracy, which appears to be overoptimistic on highly unbalanced datasets, and per-class accuracy, which under-penalizes false alarms for non-background classes, this metric favors accurate object localization with tighter masks

Method	SSIM		Perceptibility	
	@mIoU/2	@mIoU$^{\text{lim}}$	@mIoU/2	@mIoU$^{\text{lim}}$
untargeted: NLL loss	0.9989	0.9950	0.0037	0.0117
untargeted: Houdini loss	0.9995	0.9959	0.0026	0.0095
targetted: NLL loss	0.9972	0.9935	0.0074	0.0389
targetted: Houdini loss	0.9975	0.9937	0.0054	0.0392

Table 2: Semantic segmentation: comparison of empirical performance for targetted and untargeted adversarial attacks on segmentation systems. mIoU/2 denotes 50% performance drop according to the target metric and mIoU$^{\text{lim}}$ corresponds to convergence or termination after 300 iterations. SSIM: the higher, the better, perceptibility: the lower, the better. Houdini based attacks are less perceptible.

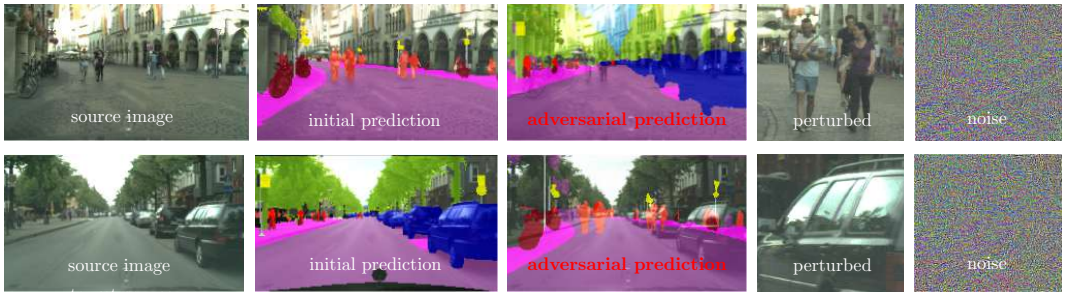


Figure 5: Targetted attack on a semantic segmentation system: switching target segmentation between two images from Cityscapes dataset [8]. The last two columns are respectively zoomed-in parts of the perturbed image and the adversarial perturbation added to the original one.

(in instance segmentation) or bounding boxes (in detection). The models are trained with a per-pixel softmax or multi-class cross entropy losses depending on the task formulation, i.e. optimized for mean per-pixel or per-class accuracy instead of mIoU. Primary targets of adversarial attacks in this group of applications are self-driving cars and robots. Xie et al. [37] have previously explored adversarial attacks in the context of semantic segmentation. However, they exploited the same proxy used for training the network. We perform a series of experiments similar to the ones described in Sec. 4. That is, we show targetted and untargeted attacks on a semantic segmentation model. We use a pre-trained Dilation10 model for semantic segmentation [38] and evaluate the success of the attacks on the validation subset of Cityscapes dataset [8]. In the first experiment, we directly alter the target mIoU metric for a given test image in both targetted and untargeted attacks. As shown in Table 2, Houdini allows fooling the model at least as well as the training proxy (NLL) while using a less perceptible. Indeed, Houdini based adversarial perturbations generated to alter the performance of the model by 50% are about 30% less noticeable than the noise created with NLL.

The second set of experiments consists of targetted attacks. That is, altering the input image to obtain an arbitrary target segmentation map as the network response. In Figure 5, we show an instance of such attack in a segmentation transfer setting, i.e. the target segmentation is the ground truth segmentation of a different image. It is clear that this type of attack is still feasible with a small adversarial perturbation (even when after zooming in the picture). Figure 1 depicts a more challenging scenario where the target segmentation is an arbitrary map (e.g. a *minion*). Again, we can make the network hallucinate the segmentation of our choice by adding a barely noticeable perturbation.

6 Speech Recognition

We evaluate the effectiveness of Houdini concerning adversarial attacks on an Automatic Speech Recognition (ASR) system. Traditionally, ASR systems are composed of different components, (e.g. acoustic model, language model, pronunciation model, etc.) where each component is trained separately. Recently, ASR research is focused on deep learning based end-to-end models. These type of models get as input a speech segment and output a transcript with no additional post-processing. In

	$\epsilon = 0.3$		$\epsilon = 0.2$		$\epsilon = 0.1$		$\epsilon = 0.05$	
	WER	CER	WER	CER	WER	CER	WER	CER
CTC	68	9.3	51	6.9	29.8	4	20	2.5
Houdini	96.1	12	85.4	9.2	66.5	6.5	46.5	4.5

Figure 6: CER and WER in (%) for adversarial examples generated by both CTC and Houdini.

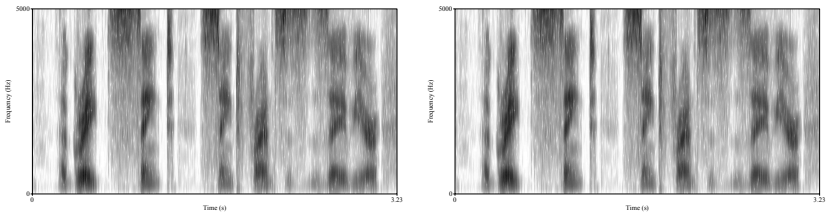
this work, we use a deep neural network as our model with similar architecture to the one presented by [2]. The system is composed of two convolutional layers, followed by seven layers of Bidirectional LSTM [17] and one fully connected layer. We optimize the Connectionist Temporal Classification (CTC) loss function [13], which was specifically designed for ASR systems. The model gets as input raw spectrograms (extracted using a window size of 25ms, frame-size of 10ms and Hamming window), and outputs a transcript.

A standard evaluating metric in speech recognition is the Word Error Rate (WER) or Character Error Rate (CER). These metrics were derived from the Levenshtein Distance [23], which is the number of substitutions, deletions, and insertions divided by the target length. The model achieves 12% Word Error Rate and 1.5% Character Error Rate on the Librispeech dataset [27], with no additional language modeling. In order to use Houdini for attacking an end-to-end ASR model, we need to get $g_\theta(x, y)$ and $g_\theta(x, \hat{y})$, which are the scores for predicting y and $\hat{y}$ respectively. Recall, in speech recognition, the target y is a sequence of characters. Hence, the score $g_\theta(x, \hat{y})$ is the sum of all possible paths to output $\hat{y}$. Fortunately, we can use the forward-backward algorithm [30], which is at the core of the CTC, to get the score of a given label y.

ABX Experiment We generated 100 audio samples of adversarial examples and performed an ABX test with about 100 humans. An ABX test is a standard way to assess the detectable differences between two choices of sensory stimuli. We present each human with two audio samples A and B. Each of these two samples is either the legitimate or an adversarial version of the same sound. These two samples are followed by a third sound X randomly selected to be either A or B. Next, the human must decide whether X is the same as A or B. For every audio sample, we executed the ABX test with at least nine (9) different persons. Overall, Only 53.73% of the adversarial examples could be distinguished from the original ones by the humans (the optimal ratio is 50%). Therefore the generated examples are not statistically significantly distinguishable by a human ear. Subsequently, we use such indistinguishable adversarial examples to test the robustness of ASR systems.

Untargeted Attacks In the first experiment, we compare network performance after attacking it with both Houdini and CTC. We generate two adversarial example, from each of the loss functions (CTC and Houdini), for every sample from the clean test set of Librispeech (2620 speech segments). We have experienced with a set of different distortion levels, $p = \infty$ and WER as ℓ. For all adversarial examples, we use $\hat{y} =$ "$Forty\ Two$", which is the "Answer to the Ultimate Question of Life, the Universe, and Everything." Results are summarizes in 6. Notice that Houdini causes a bigger decrease regarding both CER and WER than CTC for all the distortions values we have tested. In particular, for small adversarial perturbation ($\epsilon = 0.05$) the word error rate (WER) caused by an attack with Houdini is 2.3x larger than the WER obtained with CTC. Similarly, the character error rate (CER) caused by an Houdini-based attack is 1.8x larger than a CTC-based one. Fig. 7 shows the original and adversarial spectrograms for a single speech segment. (a) shows a spectrogram of the original sound file and (b) shows the spectrogram of the adversarial one. They are visually undistinguishable.

Targetted Attacks We push the model towards predicting a different transcription iteratively. In this case, the input to the model at iteration i is the adversarial example from iteration $i - 1$. Corresponding transcription samples are shown in Table 8. We notice that while setting $\hat{y}$ to be phonetically far from y, the model tends to predict wrong transcriptions but not necessarily similar to the selected target. However, when picking phonetically close ones, the model acts as expected and predict a transcription which is phonetically close to $\hat{y}$. Overall, *targetted attacks seem to be much more challenging when dealing with speech recognition systems than when we consider artificial visual systems such as pose estimators or semantic segmentation systems*.



(a) a great saint saint francis zaviour (b) i great sinkt shink t frimsuss avir

Figure 7: The model models' output for each of the spectrograms is located at the bottom of each spectrogram. The target transcription is: `A Great Saint Saint Francis Xavier.`

Manual Transcription	Adversarial Target	Adversarial Prediction
a great saint saint Francis Xavier	a green thank saint frenzier	a green thanked saint fredstus savia
no thanks I am glad to give you such easy happiness	notty am right to leave you soggy happiness	no to ex i am right like aluse o yve have misser

Figure 8: Examples of iteratively generated adversarial examples for targetted attacks. In all case the model predicts the exact target transcription of the original example. Targetted attacks are more difficult when the speech segments are phonetically very different.

```
Groundtruth Transcription:
The fact that a man can recite a poem does not show he remembers
any previous occasion on which he has recited it or read it.

G-Voice transcription of the original example:
The fact that a man can decide a poem does not show he
remembers any previous occasion on which he has work cited or read it.

G-Voice transcription of the adversarial example:
The fact that I can rest I'm just not sure that you heard there is any
previous occasion I am at he has your side it or read it.
```

```
Groundtruth Transcription:
Her bearing was graceful and animated she led her son by the hand and
before her walked two maids with wax lights and silver candlesticks.

G-Voice transcription of the original example:
The bearing was graceful an animated she let her son by the hand and
before he walks two maids with wax lights and silver candlesticks.

G-Voice transcription of the adversarial example:
Mary was grateful then admitted she let her son before the walks
to Mays would like slice furnace filter count six.
```

Figure 9: Transcriptions from Google Voice application for original and adversarial speech segments.

Black box Attacks Lastly, we experimented with a black box attack, that is attacking a system in which we do not have access to the models' gradients but only to its' predictions. In Figure 9 we show few examples in which we use Google Voice application to predict the transcript for both original and adversarial audio files. The original audio and their adversarial versions generated with our DeepSpeech-2 based model are not distinguishable by human according to our ABX test. We play each audio clip in front of an Android based mobile phone and report the transcription produced by the application. As can be seen, while Google Voice could get almost all the transcriptions correct

for legitimate examples, it largely fails to produce good transcriptions for the adversarial examples. *As with images, adversarial examples for speech recognition also transfer between models.*

7 Conclusion

We have introduced a novel approach to generate adversarial examples tailored for the performance measure unique to the task of interest. We have applied Houdini to challenging structured prediction problems such as pose estimation, semantic segmentation and speech recognition. In each case, Houdini allows fooling state of the art learning systems with imperceptible perturbation, hence extending the use of adversarial examples beyond the task of image classification. *What the eyes see and the ears hear, the mind believes.* (Harry Houdini)

8 Acknowledgments

The authors thank Alexandre Lebrun, Pauline Luc and Camille Couprie for valuable help with code and experiments. We also thank Antoine Bordes, Laurens van der Maaten, Nicolas Usunier, Christian Wolf, Herve Jegou, Yann Ollivier, Neil Zeghidour and Lior Wolf for their insightful comments on the early draft of this paper.

References

[1] D. Amodei, R. Anubhai, E. Battenberg, C. Case, J. Casper, B. Catanzaro, J. Chen, M. Chrzanowski, A. Coates, G. Diamos, et al. Deep speech 2: End-to-end speech recognition in english and mandarin. *arXiv preprint arXiv:1512.02595*, 2015.

[2] D. Amodei, R. Anubhai, E. Battenberg, C. Case, J. Casper, B. Catanzaro, J. Chen, M. Chrzanowski, A. Coates, G. Diamos, et al. Deep speech 2: End-to-end speech recognition in english and mandarin. In *International Conference on Machine Learning*, pages 173–182, 2016.

[3] M. Andriluka, L. Pishchulin, P. Gehler, and B. Schiele. 2d human pose estimation: New benchmark and state of the art analysis. *CVPR*, 2014.

[4] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[5] P. L. Bartlett, M. I. Jordan, and J. D. McAuliffe. Convexity, classification, and risk bounds. *Journal of the American Statistical Association*, 101(473):138–156, 2006.

[6] A. Bulat and G. Tzimiropoulos. Human pose estimation via convolutional part heatmap regression. *ECCV*, 2016.

[7] M. Cisse, P. Bojanowski, E. Grave, Y. Dauphin, and N. Usunier. Parseval networks: Improving robustness to adversarial examples. *arXiv preprint arXiv:1704.08847*, 2017.

[8] M. Cords, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The cityscapes dataset for semantic urban scene understanding. *CVPR*, 2016.

[9] C. Do, Q. Le, C.-H. Teo, O. Chapelle, and A. Smola. Tighter bounds for structured estimation. In *Advances in Neural Information Processing Systems (NIPS) 22*, 2008.

[10] A. Fawzi, O. Fawzi, and P. Frossard. Analysis of classifiers' robustness to adversarial perturbations. *arXiv preprint arXiv:1502.02590*, 2015.

[11] A. Fawzi, S.-M. Moosavi-Dezfooli, and P. Frossard. Robustness of classifiers: from adversarial to random noise. In *Advances in Neural Information Processing Systems*, pages 1624–1632, 2016.

[12] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. In *Proc. ICLR*, 2015.

[13] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning*, pages 369–376. ACM, 2006.

[14] T. Hazan, J. Keshet, and D. A. McAllester. Direct loss minimization for structured prediction. In *Advances in Neural Information Processing Systems*, pages 1594–1602, 2010.

[15] K. He, G. Gkioxari, P. Dollar, and R. Girshick. Mask r-cnn. *arXiv preprint arXiv:1703.06870*, 2016.

[16] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[17] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

[18] J. Keshet and D. A. McAllester. Generalization bounds and consistency for latent structural probit and ramp loss. In *Advances in Neural Information Processing Systems*, pages 2205–2212, 2011.

[19] J. Keshet, D. McAllester, and T. Hazan. Pac-bayesian approach for minimization of phoneme error rate. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pages 2224–2227. IEEE, 2011.

[20] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[21] A. Kurakin, I. Goodfellow, and S. Bengio. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*, 2016.

[22] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[23] V. I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710, 1966.

[24] D. McAllester, T. Hazan, and J. Keshet. Direct loss minimization for structured prediction. In *Advances in Neural Information Processing Systems (NIPS) 24*, 2010.

[25] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard. Deepfool: a simple and accurate method to fool deep neural networks. *arXiv preprint arXiv:1511.04599*, 2015.

[26] A. Newell, K. Yang, and J. Deng. Stacked hourglass networks for human pose estimation. *ECCV*, 2016.

[27] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur. Librispeech: an asr corpus based on public domain audio books. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pages 5206–5210. IEEE, 2015.

[28] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. Berkay Celik, and A. Swami. Practical black-box attacks against deep learning systems using adversarial examples. *arXiv preprint arXiv:1602.02697*, 2016.

[29] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *Security and Privacy (SP), 2016 IEEE Symposium on*, pages 582–597. IEEE, 2016.

[30] L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[31] U. Shaham, Y. Yamada, and S. Negahban. Understanding adversarial training: Increasing local stability of neural nets through robust optimization. *arXiv preprint arXiv:1511.05432*, 2015.

[32] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. In *Proc. ICLR*, 2014.

[33] P. Tabacof and E. Valle. Exploring the space of adversarial images. *arXiv preprint arXiv:1510.05328*, 2015.

[34] A. Tewari and P. L. Bartlett. On the consistency of multiclass classification methods. *Journal of Machine Learning Research*, 8(May):1007–1025, 2007.

[35] A. Toshev and C. Szegedy. Deeppose: Human pose estimation via deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1653–1660, 2014.

[36] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

[37] C. Xie, J. Wang, Z. Zhang, Y. Zhou, L. Xie, and A. L. Yuille. Adversarial examples for semantic segmentation and object detection. *CoRR*, abs/1703.08603, 2017.

[38] F. Yu and V. Koltun. Multi-scale aggregation by dilated convolutions. *ICLR*, 2016.

Fooling End-to-End Speaker Verification with Adversarial Examples

FOOLING END-TO-END SPEAKER VERIFICATION WITH ADVERSARIAL EXAMPLES

Felix Kreuk[1]*, Yossi Adi*[1]*, Moustapha Cisse*[2]*, Joseph Keshet*[1]

[1]Bar-Ilan University, Israel [2]Facebook AI Research

ABSTRACT

Automatic speaker verification systems are increasingly used as the primary means to authenticate costumers. Recently, it has been proposed to train speaker verification systems using end-to-end deep neural models. In this paper, we show that such systems are vulnerable to adversarial example attacks. Adversarial examples are generated by adding a peculiar noise to original speaker examples, in such a way that they are almost indistinguishable, by a human listener. Yet, the generated waveforms, which sound as speaker A can be used to fool such a system by claiming as if the waveforms were uttered by speaker B. We present white-box attacks on a deep end-to-end network that was either trained on YOHO or NTIMIT. We also present two black-box attacks. In the first one, we generate adversarial examples with a system trained on NTIMIT and perform the attack on a system that trained on YOHO. In the second one, we generate the adversarial examples with a system trained using Mel-spectrum features and perform the attack on a system trained using MFCCs. Our results show that one can significantly decrease the accuracy of a target system even when the adversarial examples are generated with different system potentially using different features.

Index Terms— Automatic speaker verification, adversarial examples

1. INTRODUCTION

Automatic speaker verification is the task of verifying that a spoken utterance has been produced by a claimed speaker. It is one of the most mature technologies for biometric authentication deployed by banks and e-commerce as the primary means to authenticate customers online and over the phone. The vulnerability of such a system is a real threat to these applications.

The standard verification protocol comprises the following three steps: training, enrollment, and evaluation. In the training stage, one learns a suitable internal speaker representation from a set of utterances and builds a simple scoring function. In the enrollment stage, a speaker provides a few utterances which are used to estimate the speaker model. During the evaluation stage, the verification task is performed by scoring a new unknown utterance against the speaker model. If the resulted score is greater than a pre-defined threshold, the system predicts that the unknown utterance produced by the claimed speaker. When the authentication is based on the voice of the speaker, irrespective of what the speaker said, the system is a text-independent speaker verification system.

Most of the modern speaker verification systems have several components. For example, the combination of i-vector for speaker representation and probabilistic linear discriminant analysis (PLDA) for a scoring function has become the dominant approach, both for text-dependent and text-independent speaker verification [1, 2, 3]. Recently, Heigold et al. [4] proposed to train speaker verification systems in an end-to-end fashion using deep neural models. This approach allows to directly learn from utterances, which improves capturing long-range context and reduces the complexity (one vs. number of frames evaluations per utterance), and the direct and joint estimation, which can lead to better and more compact models. Moreover, this approach often results in considerably simplified systems requiring fewer concepts and heuristics.

Although deep neural networks have enabled several breakthroughs in notoriously difficult problems such as image classification [5, 6], speech recognition [7, 8], speech processing [9, 10] and machine translation [11], it has been shown [12] that they are not robust to tiny perturbations in the input space. Indeed, adding a well-chosen small perturbation to the input of a network can change its prediction. When the difference between the perturbed image and the original image is indistinguishable by the human eye the example, the perturbed image is called an *adversarial example*.

Adversarial examples were first introduced in [13]. Their study first demonstrated that deep neural networks could achieve high accuracy on previously unseen examples while being vulnerable to small adversarial perturbations. This finding has recently aroused keen interest in the community [12, 14, 13, 15]. Several studies have subsequently analyzed the phenomenon [16, 17] and various approaches have been proposed to improve the robustness of neural networks [18, 19]. However, most of the previous works on adversarial examples are focused on the vision domain.

In this work, we investigate the generation of adversarial examples to attack an end-to-end neural based speaker verification model. We demonstrate the generation of adversarial examples to attack a text-dependent speaker verification system while using the architecture proposed in [4].

To the best of our knowledge, the only work that applied adversarial attacks to speech data is [20] where the authors present an adversarial attack on an end-to-end automatic speech recognition system. We are unaware of any previous study on adversarial examples for fool speaker verification systems. A different approach for attacking speaker verification systems is known as *spoofing attack* [21, 22]. In that type of attack, an adversary may use a falsifying voice, such as the recorded file of another person, as input for the speaker verification system. Our approach is different since our goal is to generate acoustic utterances which sound as speaker A (at least to the human ear) but can be used to fool the system by claiming that utterances were produced by speaker B.

This paper is organized as follows. In Section 2 we formally set the notation and definitions used throughout the paper. Section 3 provides a detailed description of the speaker verification model. In Section 4 we describe the mechanism behind the generation of adversarial examples. In Section 5 we report the results of attacking the speaker verification model in various ways. We conclude the paper with a discussion in Section 6.

2. NOTATIONS AND DEFINITIONS

In this section, we formulate the task of speaker verification rigorously and set the notation for the rest of the paper. We denote the domain of the acoustic feature vectors by $\mathcal{X} \subset \mathbb{R}^d$. The acoustic feature representation of a speech signal is therefore a sequence of vectors $\mathbf{x} = (x_1, x_2, ...x_T)$, where $x_i \in \mathcal{X}$ for all $1 \leq i \leq T$. The length of the input signal varies from one signal to another. Thus T is not fixed. We denote by $\mathcal{X}^*$ the set of all finite-length sequences over $\mathcal{X}$.

Recall that in speaker verification the goal is to assert if the claimed speaker spoke the input utterance. Specifically, the speaker verification system is a function that gets as input an utterance $\mathbf{x}$ produced by an unknown speaker, and a set of n enrollment utterances produced by speaker k denoted as $\mathbf{X}^k = \{\mathbf{x}_1^k, ..., \mathbf{x}_n^k\}$. The output of the system is a real number in the simplex $p \in [0, 1]$ estimating the probability that the utterance $\mathbf{x}$ produced by speaker k.

Let $g_\theta : \mathcal{X}^* \times (\mathcal{X}^*)^n \rightarrow [0, 1]$ be the speaker verification function implemented as neural network with a set of parameters θ. Given a set of training examples, the parameters θ are found by minimizing the negative log likelihood loss function. Each example in the training set is a tuple $(\mathbf{x}, \mathbf{X}^k, y)$ of a spoken utterance $\mathbf{x} \in \mathcal{X}$, an enrollment set of a speaker k, $\mathbf{X}^k$, and a binary label $y \in \{0, 1\}$ indicating whether the utterance $\mathbf{x}$ was produced speaker k. We denote by $\ell : [0, 1] \times \{0, 1\} \rightarrow \mathbb{R}$ the log-likehood loss function.

3. END-TO-END DEEP NETWORK MODEL

In this section, we describe the network architecture used as our speaker verification function. The architecture was initially proposed in [4], and serves as a baseline in recent work on end-to-end models for speaker verification [23, 24]. It is depicted in Figure 1.

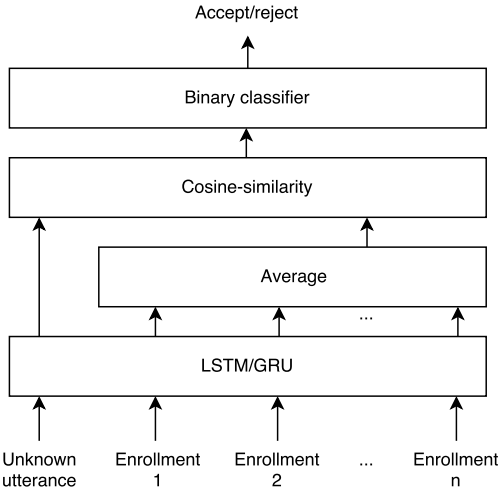


Fig. 1: End-to-end deep network model for speaker verification. The architecture is based on [4].

Recall that the input to the verification function is an unknown utterance and a set of n enrollment utterances. Each of the $n+1$ utterances is fed into a recurrent LSTM network and is represented as an embedding vector of size D. Denote by $\mathbf{u}$ and by $\mathbf{U}^k = (\mathbf{u}_1^k, \ldots, \mathbf{u}_n^k)$ the embeddings of the unknown utterance x and the enrollment set $\mathbf{X}^k$. The embeddings of the enrollments set are averaged to a single vector:

$$\mathbf{v}^k = \frac{1}{n} \sum_i^n \mathbf{u}_i^k.$$

Then the resemblance between the embedding of the unknown utterance and the average embedding of the enrollment is computed using the cosine-similarity function:

$$sim(\mathbf{u}, \mathbf{v}^k) = \frac{\mathbf{u} \cdot \mathbf{v}^k}{\|\mathbf{u}\| \|\mathbf{v}^k\|}.$$

The final stage takes the cosine-similarity, multiplies it by a scalar and add a bias to generate a probability estimation. One can think of this layer as an automatic setting of the threshold detection. The whole network is trained using with negative log-likelihood loss function.

4. GENERATING ADVERSARIAL EXAMPLES

Given an input utterance $\mathbf{x}$, an adversarial example is a perturbed version of the original pattern

$$\tilde{\mathbf{x}} = \mathbf{x} + \delta_{\mathbf{x}},$$

where $\delta_{\mathbf{x}} \in \mathbb{R}^d$ is small enough for $\tilde{\mathbf{x}}$ to be undistinguishable from $\mathbf{x}$ by a human, but causes the network to predict an incorrect label.

Formally, given a trained network g_θ and a p-norm, the adversarial example is generated by solving the following optimization problem:

$$\tilde{\mathbf{x}} = \underset{\tilde{\mathbf{x}}:\|\tilde{\mathbf{x}}-\mathbf{x}\|_p \leq \epsilon}{\operatorname{argmax}} \ell\big(g_\theta(\tilde{\mathbf{x}}, \mathbf{X}^k), y\big),$$

where ϵ represents the strength of the adversary, and p is the norm value. In words, we would like to maximize, rather than minimize, the loss function between the prediction of g_θ on the adversarial example and the correct label under the constraint that the adversarial example is similar to the original example in p-norm.

Assuming the loss function ℓ is differentiable, the authors of [16] proposed to take the first order Taylor expansion of $\mathbf{x} \mapsto \ell(g_\theta(\mathbf{x}, \mathbf{X}^k), y)$ to compute $\delta_\mathbf{x}$ by solving the following problem:

$$\tilde{\mathbf{x}} = \underset{\tilde{\mathbf{x}}:\|\tilde{\mathbf{x}}-\mathbf{x}\|_p \leq \epsilon}{\operatorname{argmax}} \big(\nabla_\mathbf{x} \ell(g_\theta(\mathbf{x}, \mathbf{X}^k), y)\big)^T (\tilde{\mathbf{x}} - \mathbf{x})$$

When $p = \infty$ the solution to the optimization problem is

$$\tilde{\mathbf{x}} = \mathbf{x} + \epsilon \operatorname{sign}(\nabla_\mathbf{x} \ell(g_\theta(\mathbf{x}, \mathbf{X}^k), y)),$$

which corresponds to the *fast gradient sign method* proposed in [12]. In other words, generating adversarial examples following the fast sign gradient method involves with taking the sign of the gradients of the loss function with respect to the input, multiply it by a small fraction so it will be indistinguishable to a human and add it to the original example.

Note that a single training example is composed of two parts; an enrollment set and a test utterance. To mimic a realistic model attack, we add the adversarial noise only to the test utterance and leave the enrollment set unchanged.

5. EXPERIMENTS

The setting was similar in all our experiments. We represented the speech utterances by acoustic features and trained a speaker verification model on that representation. Then we generated adversarial examples by adding noise to the feature vectors, and finally, acoustic waveforms were *reconstructed* from the adversarial example. For a reference, we also reconstruct waveforms of the original examples. The waveforms corresponding to the adversarial examples were used to fool the trained model as well as a different model that was trained under different conditions. We now describe the experimental setting in detail.

We evaluated the effectiveness of our method on two datasets: YOHO [25] and NTIMIT [26], each sampled at 8kHz. We used two sets of acoustic features. The first set of features was the Mel-spectrum. The acoustic signal was split into frames of 64 milliseconds with a shift of 4 milliseconds[1]. Then we applied Hamming window and computed STFT. We used a 65 Mel frequency channels that yielded $d = 65$ acoustic features. The second set we used was the Mel-Frequency Cepstrum Coefficients (MFCCs), extracted from the Mel-spectrum. Overall we trained four models; for each of the datasets (NTIMIT and YOHO), we used both feature sets.

For the YOHO corpus, each training example was generated by picking one of the verification utterances and associating it with a set of 10 random enrollment utterances. Ten speakers were excluded from the training set and used as a test set. For the NTIMIT corpus, since there are only ten utterances per speaker, we generate each of the training examples by picking one utterance to be the verification example and four other utterances to be an enrollment set. In both datasets, we swapped the enrollment set in half of the examples to generate negative instances.

We generated adversarial examples using the fast gradient sign method by adding adversarial perturbation to the test utterance vector $\mathbf{x}$ using several epsilon values. We found that $\epsilon \in (0.2, 0.3)$ caused the classifier to misclassify the adversarial examples with a high probability while keeping them remarkably similar to the original ones, a description of the adversarial examples evaluation process can be found in the next subsection. The models‘ performance was evaluated using the precision of correct classifications, and not using the standard equal error-rate (EER) since we aimed to show the effectiveness of adversarial attacks rather than comparing our model to other speaker verification systems.

5.1. ABX Testing

To validate that the generated adversarial examples are indeed indistinguishable by humans we performed an ABX test. An ABX test is a standard way to assess the detectable differences between two choices of sensory stimuli. We presented to listeners two audio samples A and B; each being either the original (reconstructed) waveform or an adversarial waveform of the same example. These two samples are followed by a third sound X which was randomly chosen to be either A or B. The listener was instructed to decide whether X is more similar to sample A or sample B. We randomly sampled 50 pairs of audio examples, original and adversarial ones. All waveforms were reconstructed from Mel-spectrogram using the Griffin-Lim algorithm [27]. eight different listeners tested each audio pair. Overall, on average 54% of the examples were correctly classified by the human listeners. Subsequently, we use such indistinguishable adversarial examples to test the robustness of speaker verification system.

5.2. A white-box attack

In the setting of a white-box attack, we assume that the adversary has access to the internals of the model to be attacked. In other words, the attacker has complete knowledge and control of the network and can access the networks‘ gradients.

[1]The high shift frequency allows us a better reconstruction of the signal.

Table 1: System accuracy under white-box attacks.

	YOHO		
	Original test	Adversarial test	Diff
Mel-spectrum	85.50%	37.50%	48.00%
MFCC	87.50%	25.75%	61.75%
	NTIMIT		
	Original test	Adversarial test	Diff
Mel-spectrum	84.26%	24.40%	59.86%
MFCC	82.14%	10.20%	69.94%

Table 2: False-positive rate (FPR) under white-box attacks.

	YOHO	
	Original test	Adversarial test
Mel-Spectrum	1.46%	69.76%
MFCC	4.88%	94.63%
	NTIMIT	
	Original test	Adversarial test
Mel-Spectrum	10.98%	79.19%
MFCC	1.73%	82.08%

An adversary can use these gradients to perturb the original input to become adversarial. The adversarial examples were crafted directly on the inputs that fed to the network.

Table 1 summarizes the results. The upper panel and the lower panel describes the results for the YOHO and the NTIMIT corpus, respectively. For each dataset, the accuracy on the test set is given in the first column. Even though one could achieve better results, our focus is to demonstrate the effectiveness of adversarial attacks on this model and to propose alternative ways (in addition to the traditional ones) of evaluating speaker verification systems. We assume the performance difference observed here is due to limited training data (YOHO has around 15.6 hours of speech, while the "OK Google" dataset contains 333 hours off speech [4]). In the second column, we presented the accuracy of the adversarial examples (generated from the test set examples). The last column is the degradation in performance.

In speech verification systems it might be considered more important to perform well regarding *false-positive rate* (FPR), this is the case where an adversary claims to be someone else and is wrongfully accepted. Results of FPR are given in Table 2, where we can see a significant degradation regarding FPR performance during the adversarial attack.

5.3. Cross-dataset

In the setting of black-box attack, the adversary has no access the model internals, only to its inputs and outputs. This setup is the strongest since it assumes no knowledge of the adversary regarding the type of model, its architecture or parameters. Moreover, the success of a black-box attack almost assures that a white-box will succeed equally or better. In our work, we performed two back-box attacks: a *cross-dataset* attack and a *cross-feature* attack. In the case of cross-dataset attach, the adversarial examples were crafted using a model trained on dataset A, is used to attack a model trained on dataset B.

We trained two models: model A was trained on the YOHO dataset using MFCC features, while model B was trained on the NTIMIT dataset using Mel-spectrum features. Then, model B was used to create adversarial examples on NTIMIT; these examples were used to attack model A. In other words, models A and B were trained on two different datasets. The examples created by model B were used to attack model A.

We found that model A reached an accuracy of 81.55% on NTIMIT reconstructed clean test set and 58.93% on NTIMIT reconstructed adversarial test set, a difference of 22.62%. The FPR was degraded from 12% to 46%.

5.4. Cross-features

A different type of back-box attack is done by creating adversarial examples using one set of acoustic features and attacking a model that was trained on a different set of acoustic features. More specifically, we trained a model on the YOHO corpus where the features were Mel-spectrum. We created adversarial examples and reconstructed waveforms. We then attacked a model that was trained on YOHO, but the features were MFCC.

We found that the MFCC model reached an accuracy of 81% on the reconstructed clean test set and accuracy of 62.25% on the reconstructed adversarial test set with a difference of 18.75%. The FPR degraded from 16% to 46%.

6. CONCLUSION

While deep neural networks have shown to improve the accuracy compared to the traditional speech verification components [28], it becomes critical to revisit the evaluation protocol of those models and design new ways to assess their reliability beyond the traditional metrics.

For future work, we would like to evaluate the robustness of the traditional speaker verification systems to adversarial examples as well as to apply adversarial training techniques to make the neural-based ones more robust to these type of attacks.

7. REFERENCES

[1] Patrick Kenny, "Bayesian speaker verification with heavy-tailed priors.," in *Odyssey*, 2010, p. 14.

[2] Najim Dehak, Patrick J Kenny, Réda Dehak, Pierre Dumouchel, and Pierre Ouellet, "Front-end factor analysis for speaker verification," *IEEE Transactions on*

Audio, Speech, and Language Processing, 19.4, p 788–798, 2011.

[3] Douglas A Reynolds, Thomas F Quatieri, and Robert B Dunn, "Speaker verification using adapted gaussian mixture models," *Digital signal processing*, vol. 10, no. 1-3, pp. 19–41, 2000.

[4] Georg Heigold, Ignacio Moreno, Samy Bengio, and Noam Shazeer, "End-to-end text-dependent speaker verification," in *ICASSP*, 2016.

[5] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton, "Imagenet classification with deep convolutional neural networks," in *NIPS*, 2012.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," in *CVPR*, 2016.

[7] Dario Amodei et al., "Deep speech 2: End-to-end speech recognition in english and mandarin," in *ICML*, 2016.

[8] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *ICML*, 2006.

[9] Yossi Adi, Joseph Keshet, and Matthew Goldrick, "Vowel duration measurement using deep neural networks," in *MLSP*. IEEE, 2015.

[10] Yossi Adi, Joseph Keshet, Emily Cibelli, and Matthew Goldrick, "Sequence segmentation using joint rnn and structured prediction models," in *ICASSP*, 2017.

[11] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint 1409.0473*, 2014.

[12] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint 1412.6572*, 2014.

[13] Christian Szegedy et al., "Intriguing properties of neural networks," *arXiv preprint 1312.6199*, 2013.

[14] Nicolas Papernot et al., "Practical black-box attacks against deep learning systems using adversarial examples," *arXiv preprint*, 2016.

[15] Pedro Tabacof and Eduardo Valle, "Exploring the space of adversarial images," in *IJCNN*. IEEE, 2016.

[16] Uri Shaham, Yutaro Yamada, and Sahand Negahban, "Understanding adversarial training: Increasing local stability of neural nets through robust optimization," *arXiv preprint 1511.05432*, 2015.

[17] Alhussein Fawzi, Seyed-Mohsen Moosavi-Dezfooli, and Pascal Frossard, "Robustness of classifiers: from adversarial to random noise," in *NIPS*, 2016.

[18] Nicolas Papernot et al., "Distillation as a defense to adversarial perturbations against deep neural networks," in *SP*. IEEE, 2016.

[19] Moustapha Cisse, Piotr Bojanowski, Edouard Grave, Yann Dauphin, and Nicolas Usunier, "Parseval networks: Improving robustness to adversarial examples," in *ICML*, 2017.

[20] Moustapha M Cisse, Yossi Adi, Natalia Neverova, and Joseph Keshet, "Houdini: Fooling deep structured visual and speech recognition models with adversarial examples," in *NIPS*, 2017.

[21] Zhizheng Wu, Junichi othersYamagishi, Tomi Kinnunen, Cemal Hanilçi, Mohammed Sahidullah, Aleksandr Sizov, Nicholas Evans, and Massimiliano Todisco, "Asvspoof: the automatic speaker verification spoofing and countermeasures challenge," *IEEE Journal of Selected Topics in Signal Processing*, vol. 11, no. 4, pp. 588–604, 2017.

[22] Abdenour Hadid, Nicholas Evans, Sébastien Marcel, and Julian Fierrez, "Biometrics systems under spoofing attack: an evaluation methodology and lessons learned," *IEEE Signal Processing Magazine, 32.5, p 20–30, 2015*.

[23] David Snyder, Pegah Ghahremani, Daniel Povey, Daniel Garcia-Romero, Yishay Carmiel, and Sanjeev Khudanpur, "Deep neural network-based speaker embeddings for end-to-end speaker verification," in *SLT*, 2016.

[24] Shi-Xiong Zhang, Zhuo Chen, Yong Zhao, Jinyu Li, and Yifan Gong, "End-to-end attention based text-dependent speaker verification," in *SLT*, 2016.

[25] Joseph P Campbell, "Testing with the yoho cd-rom voice verification corpus," in *ICASSP*, 1995.

[26] Charles Jankowski, Ashok Kalyanswamy, Sara Basson, and Judith Spitz, "Ntimit: A phonetically balanced, continuous speech, telephone bandwidth speech database," in *ICASSP*, 1990.

[27] Daniel Griffin and Jae Lim, "Signal estimation from modified short-time fourier transform," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 32, no. 2, pp. 236–243, 1984.

[28] David Snyder, Daniel Garcia-Romero, Daniel Povey, and Sanjeev Khudanpur, "Deep neural network embeddings for text-independent speaker verification," *Proc. Interspeech*, 2017.

Out-of-Distribution Detection using Multiple Semantic Label Representations

Out-of-Distribution Detection using Multiple Semantic Label Representations

Gabi Shalev
Bar-Ilan University, Israel
shalev.gabi@gmail.com

Yossi Adi
Bar-Ilan University, Israel
yossiadidrum@gmail.com

Joseph Keshet
Bar-Ilan University, Israel
jkeshet@cs.biu.ac.il

Abstract

Deep Neural Networks are powerful models that attained remarkable results on a variety of tasks. These models are shown to be extremely efficient when training and test data are drawn from the same distribution. However, it is not clear how a network will act when it is fed with an out-of-distribution example. In this work, we consider the problem of out-of-distribution detection in neural networks. We propose to use multiple semantic dense representations instead of sparse representation as the target label. Specifically, we propose to use several word representations obtained from different corpora or architectures as target labels. We evaluated the proposed model on computer vision, and speech commands detection tasks and compared it to previous methods. Results suggest that our method compares favorably with previous work. Besides, we present the efficiency of our approach for detecting wrongly classified and adversarial examples.

1 Introduction

Deep Neural Networks (DNNs) have gained lots of success after enabling several breakthroughs in notably challenging problems such as image classification [12], speech recognition [1] and machine translation [4]. These models are known to generalize well on inputs that are drawn from the same distribution as of the examples that were used to train the model [43]. In real-world scenarios, the input instances to the model can be drawn from different distributions, and in these cases, DNNs tend to perform poorly. Nevertheless, it was observed that DNNs often produce high confidence predictions for unrecognizable inputs [33] or even for a random noise [13]. Moreover, recent works in the field of adversarial examples generation show that due to small input perturbations, DNNs tend to produce high probabilities while being greatly incorrect [11, 6, 17]. When considering AI Saftey, it is essential to train DNNs that are aware of the uncertainty in the predictions [2]. Since DNNs are ubiquitous, present in nearly all segments of technology industry from self-driving cars to automated dialog agents, it becomes critical to design classifiers that can express uncertainty when predicting out-of-distribution inputs.

Recently, several studies proposed different approaches to handle this uncertainty [13, 25, 23, 19]. In [13] the authors proposed a baseline method to detect out-of-distribution examples based on the models' output probabilities. The work in [25] extended the baseline method by using temperature scaling of the softmax function and adding small controlled perturbations to inputs [14]. In [23] it was suggested to add another term to the loss so as to minimize the Kullback-Leibler (KL) divergence between the models' output for out-of-distribution samples and the uniform distribution.

Ensemble of classifiers with optional adversarial training was proposed in [19] for detecting out-of-distribution examples. Despite their high detection rate, ensemble methods require the optimization of several models and therefore are resource intensive. Additionally, each of the classifiers participated in the ensemble is trained independently and the representation is not shared among them.

32nd Conference on Neural Information Processing Systems (NeurIPS 2018), Montréal, Canada.

In this work, we replace the traditional supervision during training by using several word embeddings as the model's supervision, where each of the embeddings was trained on a different corpus or with a different architecture. More specifically, our classifier is composed of several regression functions, each of which is trained to predict a word embedding of the target label. At inference time, we gain robustness in the prediction by making decision based on the output of the regression functions. Additionally, we use the L2-norm of the outputs as a score for detecting out-of-distribution instances.

We were inspired by several studies. In [26] the authors presented a novel technique for robust transfer learning, where they proposed to optimize multiple orthogonal predictors while using a shared representation. Although being orthogonal to each other, according to their results, the predictors were likely to produce identical softmax probabilities. Similarly, we train multiple predictors that share a common representation, but instead of using the same supervision and forcing orthogonality between them, we use different supervisions based on word representations. The idea of using word embeddings as a supervision was proposed in [8] for the task of *zero-shot learning*. As opposed to ours, their model was composed of a single predictor. Last, [39] found a link between the L2-norm of the input representation and the ability to discriminate in a target domain. We continue this thread here, where we explore the use of the L2-norm for detecting out-of-distribution samples.

The contributions of this paper are as follows:

- We propose using several different word embeddings as a supervision to gain diversity and redundancy in an ensemble model with a shared representation.
- We propose utilizing the semantic structure between word embeddings to produce semantic quality predictions.
- We propose using the L2-norm of the output vectors for detecting out-of-distribution inputs.
- We examined the use of the above approach for detecting adversarial examples and wrongly classified examples.

The outline of this paper is as follows. In Section 2, we formulate the notations in the paper. In Section 3 we describe our approach in detail. Section 4 summarizes our empirical results. In Sections 5 and Section 6 we explore the use of our method for detecting adversarial examples and wrongly classified examples. In Section 7 we list the related works, and we conclude the paper in Section 8.

2 Notations and Definitions

We denote by $\mathcal{X} \subseteq \mathbb{R}^p$ the set of instances, which are represented as p-dimensional feature vectors, and we denote by $\mathcal{Y} = \{1, \ldots, N\}$ the set of class labels. Each label can be referred as a *word*, and the set $\mathcal{Y}$ can be considered as a *dictionary*. We assume that each training example $(\mathbf{x}, y) \in \mathcal{X} \times \mathcal{Y}$ is drawn from a fixed but unknown distribution ρ. Our goal is to train a classifier that performs well on unseen examples that are drawn from the distribution ρ, and can also identify out-of-distribution examples, which are drawn from a different distribution, μ.

Our model is based on word embedding representations. A *word embedding* is a mapping of a word or a label in the dictionary $\mathcal{Y}$ to a real vector space $\mathcal{Z} \subseteq \mathbb{R}^D$, so that words that are semantically closed have their corresponding vectors close in $\mathcal{Z}$. Formally, the word embedding is a function $\mathbf{e} : \mathcal{Y} \to \mathcal{Z}$ from the set of labels $\mathcal{Y}$ to an abstract vector space $\mathcal{Z}$. We assume that distances in the embedding space $\mathcal{Z}$ are measured using the *cosine distance* which is defined for two vectors $\mathbf{u}, \mathbf{v} \in \mathcal{Z}$ as follows:

$$d_{\cos}(\mathbf{u}, \mathbf{v}) = \frac{1}{2}\Big(1 - \frac{\mathbf{u} \cdot \mathbf{v}}{\|\mathbf{u}\| \, \|\mathbf{v}\|}\Big). \tag{1}$$

Two labels are considered semantically similar if and only if their corresponding embeddings are close in $\mathcal{Z}$, namely, when $d_{\cos}(\mathbf{e}(y_1), \mathbf{e}(y_2))$ is close to 0. When the cosine distance is close to 1, the corresponding labels are semantically far apart.

3 Model

Our goal is to build a robust classifier that can identify out-of-distribution inputs. In communication theory, robustness is gained by adding redundancy in different levels of the transmission encoding [20].

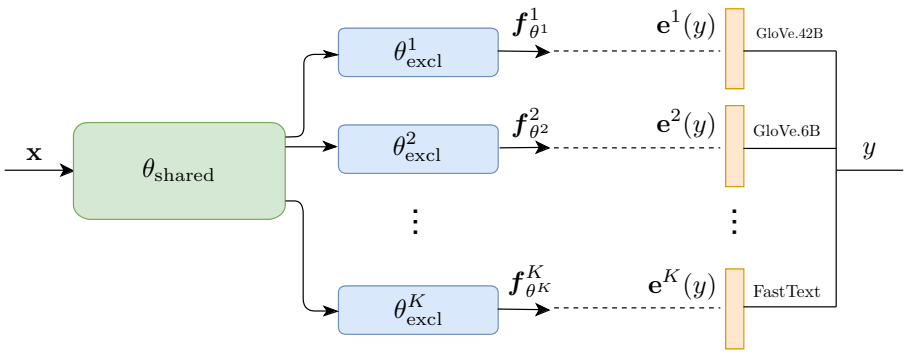


Figure 1: Our proposed K-embeddings model composed of K-predictors where each contains several fully-connected layers. The shared layers consisting a deep neural network.

Borrowing ideas from this theory, our classifier is designed to be trained on different supervisions for each class. Rather than using direct supervision, our classifier is composed of a set of regression functions, where each function is trained to predict a different word embedding (such as *GloVe*, *FastText*, etc.). The prediction of the model is based on the outcome of the regression functions.

Refer to the model depicted schematically in Figure 1. Formally, the model is composed of K regression functions $\boldsymbol{f}^k_{\theta^k} : \mathcal{X} \to \mathcal{Z}$ for $1 \leq k \leq K$. The input for each function is an instance $\mathbf{x} \in \mathcal{X}$, and its output is a word embedding vector in $\mathcal{Z}$. Note that the word embedding spaces are not the same for the different regression functions, and specifically the notion of distance is unique for each function. Overall, given an instance $\mathbf{x}$, the output of the model is K different word embedding vectors.

The set of parameters of a regression function k, $\theta^k = \{\theta_{\text{shared}}, \theta^k_{\text{excl.}}\}$, is composed of a set of parameters, θ_{shared}, that is shared among all the K functions, and a set of parameters, $\theta^k_{\text{excl.}}$, which is exclusive to the k-th function. Each regression function $\boldsymbol{f}^k_{\theta^k}$ is trained to predict a word embedding vector $\mathbf{e}^k(y)$ corresponds to the word which represents the target label $y \in \mathcal{Y}$. In the next subsections, we give a detailed account of how the model is trained and then present the inference procedure following the procedure to detect out-of-distribution instances.

3.1 Training

In classic supervised learning, the training set is composed of M instance-label pairs. In our setting, each example from the training set, $\mathcal{S}_{\text{train}}$, is composed of an instance $\mathbf{x} \in \mathcal{X}$ and a set of K different word embeddings $\{\mathbf{e}^1(y), \ldots, \mathbf{e}^K(y)\}$ of a target label $y \in \mathcal{Y}$. Namely, $\mathcal{S}_{\text{train}} = \{(\mathbf{x}_i, \mathbf{e}^1(y_i), ..., \mathbf{e}^K(y_i))\}_{i=1}^M$.

Our goal in training is to minimize a loss function which measures the discrepancy between the predicted label and the desired label. Since our intermediate representation is based on word embeddings we cannot use the cross-entropy loss function. Moreover, we would like to keep the notion of similarity between the embedding vectors from the same space. Our surrogate loss function is the sum of K cosine distances between the predicted embedding and the corresponding embedding vector of the target label, both from the same embedding space. Namely,

$$\bar{\ell}(\mathbf{x}, y; \theta) = \sum_{k=1}^{K} d_{\cos}(\mathbf{e}^k(y), \boldsymbol{f}^k_{\theta^k}(\mathbf{x})). \tag{2}$$

The cosine distance (or the cosine similarity) is a function often used in ranking tasks, where the goal is to give a high score to similar embedding vectors and a low score otherwise [9, 30].

3.2 Inference

At inference time, the regression functions predict K vectors, each corresponds to a vector in a different word embedding space. A straightforward solution is to predict the label using *hard* decision over the K output vectors by first predict the label of each output and then use a majority vote over the predicted labels.

Another, more successful procedure for decoding, is the *soft* decision, where we predict the label y which has the minimal distance to *all* of the K embedding vectors:

$$\hat{y} = \arg\min_{y \in \mathcal{Y}} \sum_{k=1}^{K} d_{\cos}(\mathbf{e}^k(y), \boldsymbol{f}^k_{\theta^k}(\mathbf{x})). \tag{3}$$

In order to distinguish between in- and out-of-distribution examples, we consider the norms of the predicted embedding vectors. When the sum of all the norms is below a detection threshold α we denote the example as an out-of-distribution example, namely,

$$\sum_{k=1}^{K} \|\boldsymbol{f}^k_{\theta^k}(\mathbf{x})\|_2^2 < \alpha. \tag{4}$$

This is inspired by the discussion of [39, Section 4] and motivated empirically in Section 4.3.

4 Experiments

In this section, we present our experimental results. First, we describe our experimental setup. Then, we evaluate our model using in-distribution examples, and lastly, we evaluate our model using out-of-distribution examples from different domains. We implemented the code using PyTorch [35]. It will be available under `www.github.com/MLSpeech/semantic_OOD`.

4.1 Experimental Setup

Recall that each regression function $\boldsymbol{f}^k_{\theta^k}$ is composed of a shared part and an exclusive part. In our setting, we used state-of-the-art known architectures as the shared part, and three fully-connected layers, with ReLU activation function between the first two, as the exclusive part.

We evaluated our approach on CIFAR-10, CIFAR-100 [18] and Google Speech Commands Dataset[1], abbreviated here as *GCommands*. For CIFAR-10 and CIFAR-100 we trained ResNet-18 and ResNet34 models [12], respectively, using stochastic gradient descent with momentum for 180 epochs. We used the standard normalization and data augmentation techniques. We used learning rate of 0.1, momentum value of 0.9 and weight decay of 0.0005. During training we divided the learning rate by 5 after 60, 120 and 160 epochs. For the *GCommands* dataset, we trained LeNet model [22] using Adam [16] for 20 epochs using batch size of 100 and a learning rate of 0.001. Similarly to [44] we extracted normalized spectrograms from the original waveforms where we zero-padded the spectrograms to equalize their sizes at 160×101.

For our supervision, we fetched five word representations for each label. The first two word representations were based on the *Skip-Gram* model [29] trained on Google News dataset and One Billion Words benchmark [5], respectively. The third and forth representations were based on GloVe [36], where the third one was trained using both Wikipedia corpus and Gigawords [34] dataset, the fourth was trained using Common Crawl dataset. The last word representations were obtained using FastText [28] trained on Wikipedia corpus. We use the terms 1-embed, 3-embed, and 5-embed to specify the number of embeddings we use as supervision, i.e., the number of predicted embedding vectors. On 1-embed and 3-embed models we randomly pick 1 or 3 embeddings (respectively), out of the five embeddings.

We compared our results to a softmax classifier (*baseline*) [13], ensemble of softmax classifiers (*ensemble*) [19] and to [25] (*ODIN*). For the ensemble method, we followed a similar approach to [19, 24] where we randomly initialized each of the models. For ODIN, we followed the scheme proposed in by the authors where we did a grid search over the ϵ and T values for each setting. In all of these models we optimized the cross-entropy loss function using the same architectures as in the proposed models.

4.2 In-Distribution

Accuracy In this subsection, we evaluate the performance of our model and compare it to models based on softmax. Similar to [25], we considered CIFAR-10, CIFAR-100 and *GCommands* datasets

[1] `https://research.googleblog.com/2017/08/launching-speech-commands-dataset.html`

Table 1: Accuracy on in-distribution examples and semantical relevance of misclassifications.

Dataset	Model	Accuracy	Avg. WUP	Avg. LCH	Avg. Path
GCommands	Baseline	90.3	0.2562	1.07	0.0937
	1-embed	90.42	**0.3279**	**1.23**	**0.1204**
	3-embed	91.04	0.3215	1.22	0.1184
	5-embed	**91.13**	0.3095	1.19	0.1141
	Ensemble	90.9	0.2206	0.96	0.0748
CIFAR-10	Baseline	95.28	0.7342	1.7	0.1594
	1-embed	95.11	**0.741**	**1.73**	**0.1633**
	3-embed	94.99	0.7352	1.71	0.1609
	5-embed	95.04	0.7302	1.69	0.157
	Ensemble	**95.87**	0.733	1.71	0.1601
CIFAR-100	Baseline	79.14	0.506	1.38	0.1263
	1-embed	77.62	0.51	1.39	0.1277
	3-embed	78.31	0.501	1.38	0.1251
	5-embed	78.23	**0.5129**	**1.4**	**0.1293**
	Ensemble	**81.38**	0.5122	**1.4**	0.1291

as in-distribution examples. We report the accuracy for our models using $K = 1$, 3 or 5 word embeddings, and compare it to the baseline and to the ensemble of softmax classifier. Results are summarized in Table 1.

Semantic Measure Word embeddings usually capture the semantic hierarchy between the words [29], since the proposed models are trained with word embeddings as supervision, they can capture such semantics. To measure the semantic quality of the model, we compute three semantic measures based on WordNet hierarchy: (i) Node-counting on the shortest path that connects the senses in the is-a taxonomy; (ii) Wu-Palmer (WUP) [41], calculates the semantic relatedness by considering the depth of the two senses in the taxonomy; and (iii) Leacock-Chodorow (LCH) [21], calculates relatedness by finding the shortest path between two concepts and scaling that value by twice the maximum depth of the hierarchy. Results suggest that on average our model produces labels which have slightly better semantic quality.

4.3 Out-of-Distribution

Out-of-Distribution Datasets For out-of-distribution examples, we followed a similar setting as in [25, 13] and evaluated our models on several different datasets. All visual models were trained on CIFAR-10 and tested on SVHN[32], LSUN[42] (resized to 32x32x3) and CIFAR-100; and trained on CIFAR-100 and were tested on SVHN, LSUN (resized to 32x32x3) and CIFAR-10. For the speech models, we split the dataset into two disjoint subsets, the first contains 7 classes[2] (denote by SC-7) and the other one contains the remaining 23 classes (denote by SC-23). We trained our models using SC-23 and test them on SC-7.

Evaluation We followed the same metrics used by [13, 25]: (i) False Positive Rate (FPR) at 95% True Positive Rate (TPR): the probability that an out-of-distribution example is misclassified as in-distribution when the TPR is as high as 95%; (ii) Detection error: the misclassification probability when TPR is 95%, where we assumed that both out- and in-distribution have an equal prior of appearing; (iii) Area Under the Receiver Operating Characteristic curve (AUROC); and (iv) Area Under the Precision-Recall curve (AUPR) for in-distribution and out-of-distribution examples.

Figure 2 presents the distribution of the L2-norm of the proposed method using 5 word embeddings and the maximum probability of the baseline model for CIFAR-100 (in-distribution) and SVHN (out-of-distribution). Both models were trained on CIFAR-100 and evaluated on CIFAR-100 test set (in-distribution) and SVHN (out-of-distribution).

Results Table 2 summarizes the results for all the models. Results suggest that our method outperforms all the three methods in all but two settings: CIFAR-100 versus CIFAR-10 and CIFAR-

[2]Composed of the following classes: *bed, down, eight, five, four, wow*, and *zero*.

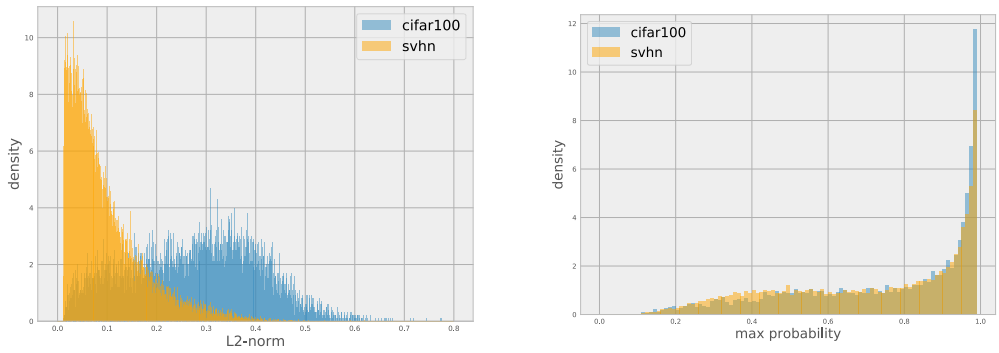


Figure 2: Distribution of the L2-norm of the proposed model with 5 word embeddings (left) and of the max probability of the baseline model (right). Both were evaluated on CIFAR-100 (in-distribution) and SVHN (out-of-distribution).

100 versus LSUN. This can be explained by taking a closer look into the structure of the datasets and the predictions made by our model. For these two settings, the in- and out-of-distribution datasets share common classes, and some of the classes of the in-distribution dataset appear in the out-of-distribution images. For example: the class *bedroom* in LSUN was detected as a *bed, couch,* and *wardrobe* in CIFAR-100 (56%, 18%, and 8% of the time, respectively); *bridge* of LSUN was detected as a *bridge, road*, and *sea* in CIFAR-100 (26%, 13%, and 9%); and *tower* of LSUN was detected as a *skyscraper, castle*, and *rocket* in CIFAR-100 (34%, 20%, and 6%), and there are many more. Similarly, CIFAR-10 and CIFAR-100 contain shared classes such as *truck*. Recall that the proportion of this class is 10% in CIFAR-10 and 1% in CIFAR-100. Hence, when the model is trained on CIFAR-100 and evaluated on CIFAR-10, it has 10% *in-distribution* examples a-priori. When the model is trained on CIFAR-10 and evaluated on CIFAR-100, it has 1% *in-distribution* examples a-priori.

5 Adversarial Examples

Next, we evaluated the performance of our approach for detecting adversarial examples [11]. Although it is not clear if adversarial examples can be considered as an out-of-distribution, we found that our method is very efficient in detecting these examples. Since our goal is detecting adversarial examples rather than suggesting a defense against them, we generated the adversarial examples in a black box settings.

We compared our method to an ensemble of softmax classifiers [19], both with $K = 5$ predictors, on the ImageNet dataset [7]. Both models are based on DenseNet-121 [15], wherein our model the K regression functions are composed of three fully-connected layers with ReLU as described earlier. We omitted from ImageNet these classes which do not have "off-the-shelf" word representations and were left with 645 labels[3].

We generated adversarial examples in a black box setting using a third model, namely VGG-19 [38] with the *fast gradient sign method* and $\epsilon = 0.007$ [11], and measured the detection rate for each of the models. Notice that our goal was not to be robust to correctly classify adversarial examples but rather to detect them.

We used the method of detecting out-of-distribution examples to detect adversarial examples, results are in Table 3. We further explored the inter-predictor agreements on the predicted class across the K predicted embeddings. The histogram of the differences between the maximum and the minimum rankings of the predicted label is presented in Figure 3. It can be observed that for our model the inter-predictor variability is much higher than that of the ensemble model. One explanation for this behavior is the transferability of adversarial examples across softmax classifiers, which can be reduced by using different supervisions.

[3]In [8] the authors suggested to use random vectors as labels. We leave the exploration of this research direction for future work.

Table 2: Results for in- and out-of-distribution detection for various settings. All values are in percentages. ↑ indicates larger value is better, and ↓ indicates lower value is better.

In-Distribution	Out-of-Distribution	Model	FPR (95% TPR) ↓	Detection Error ↓	AUROC ↑	AUPR-In ↑	AUPR-Out ↑
SC-23 (LeNet)	SC-7	Baseline	77.53	41.26	82.74	94.33	50.44
		ODIN	71.02	38.01	85.49	95.11	57.7
		1-embed	70.64	37.8	85.85	95.21	58.08
		3-embed	**67.5**	**36.24**	**87.34**	**95.91**	**59.97**
		5-embed	69.23	37.09	86.93	95.74	58.97
		Ensemble	72.73	38.85	85.99	95.69	50.71
CIFAR-10 (ResNet18)	SVHN	Baseline	7.19	6.09	97.2	96.35	98.05
		ODIN	4.95	4.97	98.65	96.89	99.21
		1-embed	**2.41**	**3.7**	**99.48**	**98.77**	**99.79**
		3-embed	4.92	4.95	98.52	97.76	99.07
		5-embed	4.14	4.57	99.1	98.3	99.55
		Ensemble	6.01	5.5	98.24	97.41	98.94
CIFAR-10 (ResNet18)	LSUN	Baseline	50.25	27.62	91.28	91.58	89.3
		ODIN	41.8	23.39	90.35	96.38	75.07
		1-embed	26.11	15.55	95.37	95.81	94.85
		3-embed	29.2	17.09	95.07	95.63	94.38
		5-embed	**22.98**	**13.99**	**96.05**	**96.72**	**94.86**
		Ensemble	46.16	25.58	92.93	94.1	77.57
CIFAR-10 (ResNet18)	CIFAR-100	Baseline	58.75	31.87	87.76	86.73	85.83
		ODIN	54.85	29.92	85.59	82.26	85.41
		1-embed	48.72	26.86	89.18	88.91	88.39
		3-embed	50.9	27.95	89.76	90.36	88.13
		5-embed	**45.25**	**25.12**	**91.23**	**91.86**	**89.63**
		Ensemble	56.14	30.57	90.03	90.01	88.27
CIFAR-100 (ResNet34)	SVHN	Baseline	87.88	46.44	74.11	63.6	84.17
		ODIN	76.64	40.82	79.86	68.22	90.1
		1-embed	75.79	40.39	81.82	71.52	90.1
		3-embed	74.74	39.87	82.57	75.01	90.4
		5-embed	**60.14**	**32.57**	**87.42**	**77.95**	**93.56**
		Ensemble	85.92	45.46	79.1	69.23	89.3
CIFAR-100 (ResNet34)	CIFAR-10	Baseline	77.21	41.1	79.18	80.71	75.22
		ODIN	74.15	39.57	80.40	80.41	77.2
		1-embed	80.82	42.9	75.99	74.27	73.01
		3-embed	78.17	41.77	77.35	77.42	74.39
		5-embed	77.03	41.01	77.7	77.23	74.61
		Ensemble	**73.57**	**39.28**	**81.49**	**83.24**	**78.16**
CIFAR-100 (ResNet34)	LSUN	Baseline	80.41	42.7	78.02	79.25	73.34
		ODIN	79.88	42.44	78.94	80.22	73.31
		1-embed	80.99	42.99	76.41	75.08	74.02
		3-embed	81.08	43.03	74.88	72.21	72.38
		5-embed	80.87	42.93	76.08	75.3	72.67
		Ensemble	**79.53**	**42.26**	**79.05**	**92.61**	**74.06**

We labeled an input as an adversarial example unless all the predictors were in full agreement. We calculated the detection rate of adversarial examples and the false rejection rate of legitimate examples. The ensemble achieved 43.88% detection rate and 11.69% false rejection rate, while the embedding model achieved 62.04% detection rate and 15.16% false rejection rate. Although the ensemble method achieves slightly better false rejection rate (3% improvement), the detection rate of our approach is significantly better (18% improvement). To better qualify that, we fixed the false rejection rate in both methods to be 3%. In this setting, the ensemble reaches 15.41% detection rate while our model reaches 28.64% detection rate (13% improvement).

6 Wrongly Classified Examples

Recall the embedding models were trained to minimize the cosine distance between the output vector and the word representation of the target label according to some embedding space. When we plot the average L2-norm of the models' output vectors as a function of the epochs, we have noticed

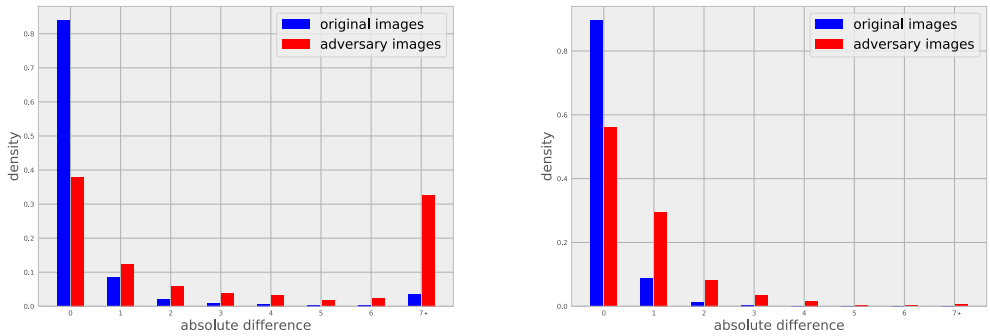


Figure 3: Histogram of the differences between the max and the min rankings of the predicted label in our model (left) and the ensemble model (right) for original and adversarial examples.

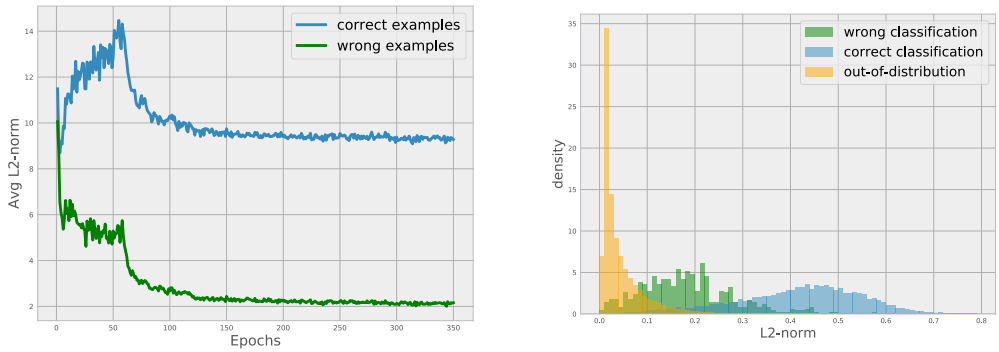


Figure 4: The average L2-norm for wrongly and correctly classified examples as a function of training epochs (left). Density of the L2-norm for wrongly, correctly classified, and out-of-distributions examples (right).

that the norm of the wrongly classified examples is *significantly smaller* than those of correctly classified examples. These finding goes in hand with the results in [39], which observed that lower representation norms are negatively associated with the softmax predictions, as shown in the left panel of Figure 4. Similarly, we observe a similar behavior for out-of-distribution examples as shown in the right panel of Figure 4. These findings suggest that we can adjust the threshold α accordingly. We leave this further exploration for future work.

7 Related Work

The problem of detecting out-of-distribution examples in low-dimensional space has been well-studied, however those methods found to be unreliable for high-dimensional space [40]. Recently, out-of-distribution detectors based on deep models have been proposed. Several studies require enlarging or modifying the neural networks [23, 37, 3], and various approaches suggest to use the output of a pre-trained model with some minor modifications [13, 25].

There has been a variety of works revolving around Bayesian inference approximations, which approximate the posterior distribution over the parameters of the neural network and use them to quantify predictive uncertainty [31, 27]. These Bayesian approximations often harder to implement and computationally slower to train compared to non-Bayesian approaches. The authors of [10] proposed to use Monte Carlo dropout to estimate uncertainty at test time as Bayesian approximation. More recently, [19] introduced a non-Bayesian method, using an ensemble of classifiers for predictive uncertainty estimation, which proved to be significantly better than previous methods.

Table 3: Results for in- and out-of-distribution detection, for ImageNet (in) and adversarial examples (out)

Model	FPR (95% TPR) ↓	Detection Error ↓	AUROC ↑	AUPR-In ↑	AUPR-Out ↑
Ensemble	57.3	31.15	88.66	**98.71**	43.46
5-embed	**57.26**	**31.12**	**89.58**	98.64	**47.2**

8 Discussion and Future Work

In this paper, we propose to use several semantic representations for each target label as supervision to the model in order to detect out-of-distribution examples, where the detection score is based on the L2-norm of the output representations.

For future work, we would like to further investigate the following: (i) we would like to explore better decision strategy for detecting out-of-distribution examples; (ii) we would like to rigorously analyze the notion of confidence based on the L2-norm beyond [39]; (iii) we would like to inspect the relation between detecting wrongly classified examples and adversarial examples to out-of-distribution examples.

References

[1] Dario Amodei, Rishita Anubhai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Jingdong Chen, Mike Chrzanowski, Adam Coates, Greg Diamos, et al. Deep speech 2: End-to-end speech recognition in english and mandarin. In *International Conference on Machine Learning*, pages 173–182, 2016.

[2] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*, 2016.

[3] Jerone TA Andrews, Thomas Tanay, Edward J Morton, and Lewis D Griffin. Transfer representation-learning for anomaly detection. ICML, 2016.

[4] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[5] Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. One billion word benchmark for measuring progress in statistical language modeling. *arXiv preprint arXiv:1312.3005*, 2013.

[6] Moustapha M Cisse, Yossi Adi, Natalia Neverova, and Joseph Keshet. Houdini: Fooling deep structured visual and speech recognition models with adversarial examples. In *Advances in Neural Information Processing Systems*, pages 6980–6990, 2017.

[7] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[8] Andrea Frome, Greg S Corrado, Jon Shlens, Samy Bengio, Jeff Dean, Tomas Mikolov, et al. Devise: A deep visual-semantic embedding model. In *Advances in neural information processing systems*, pages 2121–2129, 2013.

[9] Tzeviya Fuchs and Joseph Keshet. Spoken term detection automatically adjusted for a given threshold. *IEEE Journal of Selected Topics in Signal Processing*, 11(8):1310–1317, 2017.

[10] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059, 2016.

[11] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[13] Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *arXiv preprint arXiv:1610.02136*, 2016.

[14] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[15] Gao Huang, Zhuang Liu, Kilian Q Weinberger, and Laurens van der Maaten. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, volume 1, page 3, 2017.

[16] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[17] Felix Kreuk, Yossi Adi, Moustapha Cisse, and Joseph Keshet. Fooling end-to-end speaker verification by adversarial examples. *arXiv preprint arXiv:1801.03339*, 2018.

[18] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009.

[19] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems*, pages 6405–6416, 2017.

[20] Amos Lapidoth. *A foundation in digital communication*. Cambridge University Press, 2017.

[21] Claudia Leacock and Martin Chodorow. Combining local context and wordnet similarity for word sense identification. *WordNet: An electronic lexical database*, 49(2):265–283, 1998.

[22] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[23] Kimin Lee, Honglak Lee, Kibok Lee, and Jinwoo Shin. Training confidence-calibrated classifiers for detecting out-of-distribution samples. *arXiv preprint arXiv:1711.09325*, 2017.

[24] Stefan Lee, Senthil Purushwalkam, Michael Cogswell, David Crandall, and Dhruv Batra. Why m heads are better than one: Training a diverse ensemble of deep networks. *arXiv preprint arXiv:1511.06314*, 2015.

[25] Shiyu Liang, Yixuan Li, and R Srikant. Enhancing the reliability of out-of-distribution image detection in neural networks.

[26] Etai Littwin and Lior Wolf. The multiverse loss for robust transfer learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3957–3966, 2016.

[27] David JC MacKay. *Bayesian methods for adaptive models*. PhD thesis, California Institute of Technology, 1992.

[28] Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. Advances in pre-training distributed word representations. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.

[29] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[30] Einat Naaman, Yossi Adi, and Joseph Keshet. Learning similarity function for pronunciation variations. In *Proc. of Interspeech*, 2017.

[31] Radford M Neal. *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media, 2012.

[32] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, volume 2011, page 5, 2011.

[33] Anh Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 427–436, 2015.

[34] Robert Parker, David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. English gigaword fifth edition, linguistic data consortium. *Google Scholar*, 2011.

[35] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *NIPS-W*, 2017.

[36] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[37] Thomas Schlegl, Philipp Seeböck, Sebastian M Waldstein, Ursula Schmidt-Erfurth, and Georg Langs. Unsupervised anomaly detection with generative adversarial networks to guide marker discovery. In *International Conference on Information Processing in Medical Imaging*, pages 146–157. Springer, 2017.

[38] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[39] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. Web-scale training for face identification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2746–2754, 2015.

[40] Lucas Theis, Aäron van den Oord, and Matthias Bethge. A note on the evaluation of generative models. *ICLR*, 2015.

[41] Zhibiao Wu and Martha Palmer. Verbs semantics and lexical selection. In *Proceedings of the 32nd annual meeting on Association for Computational Linguistics*, pages 133–138. Association for Computational Linguistics, 1994.

[42] Fisher Yu, Yinda Zhang, Shuran Song, Ari Seff, and Jianxiong Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, 2015.

[43] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016.

[44] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

Chapter 4

Published Papers (Analysis of Neural Network Models for Speech and Language Processing)

Fine-Grained Analysis of Sentence Embeddings using Auxiliary Prediction Tasks

FINE-GRAINED ANALYSIS OF SENTENCE EMBEDDINGS USING AUXILIARY PREDICTION TASKS

Yossi Adi[1,2], **Einat Kermany**[2], **Yonatan Belinkov**[3], **Ofer Lavi**[2], **Yoav Goldberg**[1]

[1]Bar-Ilan University, Ramat-Gan, Israel
{yoav.goldberg, yossiadidrum}@gmail.com
[2]IBM Haifa Research Lab, Haifa, Israel
{einatke, oferl}@il.ibm.com
[3]MIT Computer Science and Artificial Intelligence Laboratory, Cambridge, MA, USA
belinkov@mit.edu

arXiv:1608.04207v3 [cs.CL] 9 Feb 2017

ABSTRACT

There is a lot of research interest in encoding variable length sentences into fixed length vectors, in a way that preserves the sentence meanings. Two common methods include representations based on averaging word vectors, and representations based on the hidden states of recurrent neural networks such as LSTMs. The sentence vectors are used as features for subsequent machine learning tasks or for pre-training in the context of deep learning. However, not much is known about the properties that are encoded in these sentence representations and about the language information they capture.

We propose a framework that facilitates better understanding of the encoded representations. We define prediction tasks around isolated aspects of sentence structure (namely sentence length, word content, and word order), and score representations by the ability to train a classifier to solve each prediction task when using the representation as input. We demonstrate the potential contribution of the approach by analyzing different sentence representation mechanisms. The analysis sheds light on the relative strengths of different sentence embedding methods with respect to these low level prediction tasks, and on the effect of the encoded vector's dimensionality on the resulting representations.

1 INTRODUCTION

While *sentence embeddings* or *sentence representations* play a central role in recent deep learning approaches to NLP, little is known about the information that is captured by different sentence embedding learning mechanisms. We propose a methodology facilitating fine-grained measurement of some of the information encoded in sentence embeddings, as well as performing fine-grained comparison of different sentence embedding methods.

In sentence embeddings, sentences, which are variable-length sequences of discrete symbols, are encoded into fixed length continuous vectors that are then used for further prediction tasks. A simple and common approach is producing word-level vectors using, e.g., word2vec (Mikolov et al., 2013a;b), and summing or averaging the vectors of the words participating in the sentence. This continuous-bag-of-words (CBOW) approach disregards the word order in the sentence.[1]

Another approach is the *encoder-decoder* architecture, producing models also known as *sequence-to-sequence* models (Sutskever et al., 2014; Cho et al., 2014; Bahdanau et al., 2014, inter alia). In this architecture, an *encoder* network (e.g. an LSTM) is used to produce a vector representation of the sentence, which is then fed as input into a *decoder* network that uses it to perform some prediction task (e.g. recreate the sentence, or produce a translation of it). The encoder and decoder networks are trained jointly in order to perform the final task.

[1]We use the term CBOW to refer to a sentence representation that is composed of an average of the vectors of the words in the sentence, not to be confused with the training method by the same name which is used in the word2vec algorithm.

Some systems (for example in machine translation) train the system end-to-end, and use the trained system for prediction (Bahdanau et al., 2014). Such systems do not generally care about the encoded vectors, which are used merely as intermediate values. However, another common case is to train an encoder-decoder network and then throw away the decoder and use the trained encoder as a general mechanism for obtaining sentence representations. For example, an encoder-decoder network can be trained as an auto-encoder, where the encoder creates a vector representation, and the decoder attempts to recreate the original sentence (Li et al., 2015). Similarly, Kiros et al. (2015) train a network to encode a sentence such that the decoder can recreate its neighboring sentences in the text. Such networks do not require specially labeled data, and can be trained on large amounts of unannotated text. As the decoder needs information about the sentence in order to perform well, it is clear that the encoded vectors capture a non-trivial amount of information about the sentence, making the encoder appealing to use as a general purpose, stand-alone sentence encoding mechanism. The sentence encodings can then be used as input for other prediction tasks for which less training data is available (Dai & Le, 2015). In this work we focus on these "general purpose" sentence encodings.

The resulting sentence representations are opaque, and there is currently no good way of comparing different representations short of using them as input for different high-level semantic tasks (e.g. sentiment classification, entailment recognition, document retrieval, question answering, sentence similarity, etc.) and measuring how well they perform on these tasks. This is the approach taken by Li et al. (2015), Hill et al. (2016) and Kiros et al. (2015). This method of comparing sentence embeddings leaves a lot to be desired: the comparison is at a very coarse-grained level, does not tell us much about the kind of information that is encoded in the representation, and does not help us form generalizable conclusions.

Our Contribution We take a first step towards opening the black box of vector embeddings for sentences. We propose a methodology that facilitates comparing sentence embeddings on a much finer-grained level, and demonstrate its use by analyzing and comparing different sentence representations. We analyze sentence representation methods that are based on LSTM auto-encoders and the simple CBOW representation produced by averaging word2vec word embeddings. For each of CBOW and LSTM auto-encoder, we compare different numbers of dimensions, exploring the effect of the dimensionality on the resulting representation. We also provide some comparison to the skip-thought embeddings of Kiros et al. (2015).

In this work, we focus on what are arguably the three most basic characteristics of a sequence: its length, the items within it, and their order. We investigate different sentence representations based on the capacity to which they encode these aspects. Our analysis of these low-level properties leads to interesting, actionable insights, exposing relative strengths and weaknesses of the different representations.

Limitations Focusing on low-level sentence properties also has limitations: The tasks focus on measuring the preservation of surface aspects of the sentence and do not measure syntactic and semantic generalization abilities; the tasks are not directly related to any specific downstream application (although the properties we test are important factors in many tasks – knowing that a model is good at predicting length and word order is likely advantageous for syntactic parsing, while models that excel at word content are good for text classification tasks). Dealing with these limitations requires a complementary set of auxiliary tasks, which is outside the scope of this study and is left for future work.

The study also suffers from the general limitations of empirical work: we do not prove general theorems but rather measure behaviors on several data points and attempt to draw conclusions from these measurements. There is always the risk that our conclusions only hold for the datasets on which we measured, and will not generalize. However, we do consider our large sample of sentences from Wikipedia to be representative of the English language, at least in terms of the three basic sentence properties that we study.

Summary of Findings Our analysis reveals the following insights regarding the different sentence embedding methods:

- Sentence representations based on averaged word vectors are surprisingly effective, and encode a non-trivial amount of information regarding sentence length. The information they contain

can also be used to reconstruct a non-trivial amount of the original word order in a probabilistic manner (due to regularities in the natural language data).

- LSTM auto-encoders are very effective at encoding word order and word content.
- Increasing the number of dimensions benefits some tasks more than others.
- Adding more hidden units sometimes *degrades* the encoders' ability to encode word content. This degradation is *not correlated* with the BLEU scores of the decoder, suggesting that BLEU over the decoder output is sub-optimal for evaluating the encoders' quality.
- LSTM encoders trained as auto-encoders do not rely on ordering patterns in the training sentences when encoding novel sentences, while the skip-thought encoders do rely on such patterns.

2 Related Work

Word-level distributed representations have been analyzed rather extensively, both empirically and theoretically, for example by Baroni et al. (2014), Levy & Goldberg (2014) and Levy et al. (2015). In contrast, the analysis of sentence-level representations has been much more limited. Commonly used approaches is to either compare the performance of the sentence embeddings on down-stream tasks (Hill et al., 2016), or to analyze models, specifically trained for predefined task (Schmaltz et al., 2016; Sutskever et al., 2011).

While the resulting analysis reveals differences in performance of different models, it does not adequately explain what kind of linguistic properties of the sentence they capture. Other studies analyze the hidden units learned by neural networks when training a sentence representation model (Elman, 1991; Karpathy et al., 2015; Kádár et al., 2016). This approach often associates certain linguistic aspects with certain hidden units. Kádár et al. (2016) propose a methodology for quantifying the contribution of each input word to a resulting GRU-based encoding. These methods depend on the specific learning model and cannot be applied to arbitrary representations. Moreover, it is still not clear what is captured by the final sentence embeddings.

Our work is orthogonal and complementary to the previous efforts: we analyze the resulting sentence embeddings by devising auxiliary prediction tasks for core sentence properties. The methodology we purpose is general and can be applied to any sentence representation model.

3 Approach

We aim to inspect and compare encoded sentence vectors in a task-independent manner. The main idea of our method is to focus on isolated aspects of sentence structure, and design experiments to measure to what extent each aspect is captured in a given representation.

In each experiment, we formulate a prediction task. Given a sentence representation method, we create training data and train a classifier to predict a specific sentence property (e.g. their length) based on their vector representations. We then measure how well we can train a model to perform the task. The basic premise is that if we cannot train a classifier to predict some property of a sentence based on its vector representation, then this property is not encoded in the representation (or rather, not encoded in a useful way, considering how the representation is likely to be used).

The experiments in this work focus on low-level properties of sentences – the sentence length, the identities of words in a sentence, and the order of the words. We consider these to be the core elements of sentence structure. Generalizing the approach to higher-level semantic and syntactic properties holds great potential, which we hope will be explored in future work, by us or by others.

3.1 The Prediction Tasks

We now turn to describe the specific prediction tasks. We use lower case italics (s, w) to refer to sentences and words, and boldface to refer to their corresponding vector representations ($\mathbf{s}$, $\mathbf{w}$). When more than one element is considered, they are distinguished by indices (w_1, w_2, $\mathbf{w_1}$, $\mathbf{w_2}$).

Our underlying corpus for generating the classification instances consists of 200,000 Wikipedia sentences, where 150,000 sentences are used to generate training examples, and 25,000 sentences

are used for each of the test and development examples. These sentences are a subset of the training set that was used to train the original sentence encoders. The idea behind this setup is to test the models on what are presumably their best embeddings.

Length Task This task measures to what extent the sentence representation encodes its length. Given a sentence representation $\mathbf{s} \in \mathbb{R}^k$, the goal of the classifier is to predict the length (number of words) in the original sentence s. The task is formulated as multiclass classification, with eight output classes corresponding to binned lengths.[2] The resulting dataset is reasonably balanced, with a majority class (lengths 5-8 words) of 5,182 test instances and a minority class (34-70) of 1,084 test instances. Predicting the majority class results in classification accuracy of 20.1%.

Word-content Task This task measures to what extent the sentence representation encodes the identities of words within it. Given a sentence representation $\mathbf{s} \in \mathbb{R}^k$ and a word representation $\mathbf{w} \in \mathbb{R}^d$, the goal of the classifier is to determine whether w appears in the s, with access to neither w nor s. This is formulated as a binary classification task, where the input is the concatenation of $\mathbf{s}$ and $\mathbf{w}$.

To create a dataset for this task, we need to provide positive and negative examples. Obtaining positive examples is straightforward: we simply pick a random word from each sentence. For negative examples, we could pick a random word from the entire corpus. However, we found that such a dataset tends to push models to memorize words as either positive or negative words, instead of finding their relation to the sentence representation. Therefore, for each sentence we pick as a negative example a word that appears as a positive example somewhere in our dataset, but does not appear in the given sentence. This forces the models to learn a relationship between word and sentence representations. We generate one positive and one negative example from each sentence. The dataset is balanced, with a baseline accuracy of 50%.

Word-order Task This task measures to what extent the sentence representation encodes word order. Given a sentence representation $\mathbf{s} \in \mathbb{R}^k$ and the representations of two words that appear in the sentence, $\mathbf{w}_1, \mathbf{w}_2 \in \mathbb{R}^d$, the goal of the classifier is to predict whether w_1 appears before or after w_2 in the original sentence s. Again, the model has no access to the original sentence and the two words. This is formulated as a binary classification task, where the input is a concatenation of the three vectors $\mathbf{s}$, $\mathbf{w}_1$ and $\mathbf{w}_2$.

For each sentence in the corpus, we simply pick two random words from the sentence as a positive example. For negative examples, we flip the order of the words. We generate one positive and one negative example from each sentence. The dataset is balanced, with a baseline accuracy of 50%.

4 Sentence Representation Models

Given a sentence $s = \{w_1, w_2, ..., w_N\}$ we aim to find a sentence representation $\mathbf{s}$ using an encoder:

$$\texttt{ENC} : s = \{w_1, w_2, ..., w_N\} \mapsto \mathbf{s} \in \mathbb{R}^k$$

The encoding process usually assumes a vector representation $\mathbf{w}_i \in \mathbb{R}^d$ for each word in the vocabulary. In general, the word and sentence embedding dimensions, d and k, need not be the same. The word vectors can be learned together with other encoder parameters or pre-trained. Below we describe different instantiations of ENC.

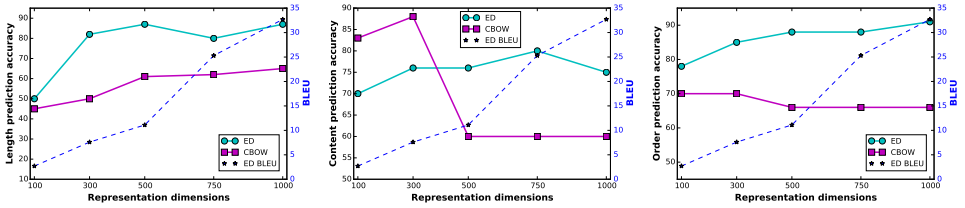
Continuous Bag-of-words (CBOW) This simple yet effective text representation consists of performing element-wise averaging of word vectors that are obtained using a word-embedding method such as word2vec.

Despite its obliviousness to word order, CBOW has proven useful in different tasks (Hill et al., 2016) and is easy to compute, making it an important model class to consider.

Encoder-Decoder (ED) The encoder-decoder framework has been successfully used in a number of sequence-to-sequence learning tasks (Sutskever et al., 2014; Bahdanau et al., 2014; Dai & Le, 2015; Li et al., 2015). After the encoding phase, a decoder maps the sentence representation back to the sequence of words:

$$\texttt{DEC} : \mathbf{s} \in \mathbb{R}^k \mapsto s = \{w_1, w_2, ..., w_N\}$$

[2]We use the bins (5-8), (9-12), (13-16), (17-20), (21-25), (26-29), (30-33), (34-70).



(a) Length test. (b) Content test. (c) Order test.

Figure 1: Task accuracy vs. embedding size for different models; ED BLEU scores given for reference.

Here we investigate the specific case of an auto-encoder, where the entire encoding-decoding process can be trained end-to-end from a corpus of raw texts. The sentence representation is the final output vector of the encoder. We use a long short-term memory (LSTM) recurrent neural network (Hochreiter & Schmidhuber, 1997; Graves et al., 2013) for both encoder and decoder. The LSTM decoder is similar to the LSTM encoder but with different weights.

5 Experimental Setup

The bag-of-words (CBOW) and encoder-decoder models are trained on 1 million sentences from a 2012 Wikipedia dump with vocabulary size of 50,000 tokens. We use NLTK (Bird, 2006) for tokenization, and constrain sentence lengths to be between 5 and 70 words. For both models we control the embedding size k and train word and sentence vectors of sizes $k \in \{100, 300, 500, 750, 1000\}$. More details about the experimental setup are available in the Appendix.

6 Results

In this section we provide a detailed description of our experimental results along with their analysis. For each of the three main tests – length, content and order – we investigate the performance of different sentence representation models across embedding size.

6.1 Length Experiments

We begin by investigating how well the different representations encode sentence length. Figure 1a shows the performance of the different models on the length task, as well as the BLEU obtained by the LSTM encoder-decoder (ED).

With enough dimensions, the LSTM embeddings are very good at capturing sentence length, obtaining accuracies between 82% and 87%. Length prediction ability is not perfectly correlated with BLEU scores: from 300 dimensions onward the length prediction accuracies of the LSTM remain relatively stable, while the BLEU score of the encoder-decoder model increases as more dimensions are added.

Somewhat surprisingly, the CBOW model also encodes a fair amount of length information, with length prediction accuracies of 45% to 65%, way above the 20% baseline. This is remarkable, as the CBOW representation consists of averaged word vectors, and we did not expect it to encode length at all. We return to CBOW's exceptional performance in Section 7.

6.2 Word Content Experiments

To what extent do the different sentence representations encode the identities of the words in the sentence? Figure 1b visualizes the performance of our models on the word content test.

All the representations encode some amount of word information, and clearly outperform the random baseline of 50%. Some trends are worth noting. While the capacity of the LSTM encoder to preserve word identities generally increases when adding dimensions, the performance peaks at 750 dimensions and drops afterwards. This stands in contrast to the BLEU score of the respective

encoder-decoder models. We hypothesize that this occurs because a sizable part of the auto-encoder performance comes from the decoder, which also improves as we add more dimensions. At 1000 dimensions, the decoder's language model may be strong enough to allow the representation produced by the encoder to be less informative with regard to word content.

CBOW representations with low dimensional vectors (100 and 300 dimensions) perform exceptionally well, outperforming the more complex, sequence-aware models by a wide margin. If your task requires access to word identities, it is worth considering this simple representation. Interestingly, CBOW scores drop at higher dimensions.

6.3 Word Order Experiments

Figure 1c shows the performance of the different models on the order test. The LSTM encoders are very capable of encoding word order, with LSTM-1000 allowing the recovery of word order in 91% of the cases. Similar to the length test, LSTM order prediction accuracy is only loosely correlated with BLEU scores. It is worth noting that increasing the representation size helps the LSTM-encoder to better encode order information.

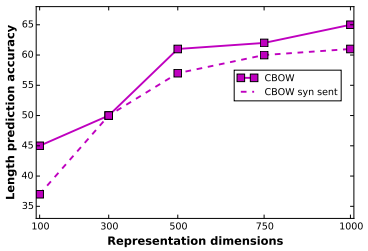
Surprisingly, the CBOW encodings manage to reach an accuracy of 70% on the word order task, 20% above the baseline. This is remarkable as, by definition, the CBOW encoder does not attempt to preserve word order information. One way to explain this is by considering distribution patterns of words in natural language sentences: some words tend to appear before others. In the next section we analyze the effect of natural language on the different models.

7 Importance of "Natural Languageness"

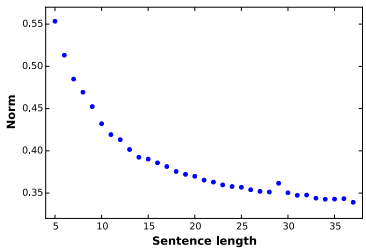
Natural language imposes many constraints on sentence structure. To what extent do the different encoders rely on specific properties of word distributions in natural language sentences when encoding sentences?

To account for this, we perform additional experiments in which we attempt to control for the effect of natural language.

How can CBOW encode sentence length? Is the ability of CBOW embeddings to encode length related to specific words being indicative of longer or shorter sentences? To control for this, we created a synthetic dataset where each word in each sentence is replaced by a random word from the dictionary and re-ran the length test for the CBOW embeddings using this dataset. As Figure 2a shows, this only leads to a slight decrease in accuracy, indicating that the identity of the words is not the main component in CBOW's success at predicting length.



(a) Length accuracy for different CBOW sizes on natural and synthetic (random words) sentences.



(b) Average embedding norm vs. sentence length for CBOW with an embedding size of 300.

An alternative explanation for CBOW's ability to encode sentence length is given by considering the norms of the sentence embeddings. Indeed, Figure 2b shows that the embedding norm decreases as sentences grow longer. We believe this is one of the main reasons for the strong CBOW results.

While the correlation between the number of averaged vectors and the resulting norm surprised us, in retrospect it is an expected behavior that has sound mathematical foundations. To understand the behavior, consider the different word vectors to be random variables, with the values in each

dimension centered roughly around zero. Both central limit theorem and Hoeffding's inequality tell us that as we add more samples, the expected average of the values will better approximate the true mean, causing the norm of the average vector to decrease. We expect the correlation between the sentence length and its norm to be more pronounced with shorter sentences (above some number of samples we will already be very close to the true mean, and the norm will not decrease further), a behavior which we indeed observe in practice.

How does CBOW encode word order? The surprisingly strong performance of the CBOW model on the order task made us hypothesize that much of the word order information is captured in general natural language word order statistics.

To investigate this, we re-run the word order tests, but this time drop the sentence embedding in training and testing time, learning from the word-pairs alone. In other words, we feed the network as input two word embeddings and ask which word comes first in the sentence. This test isolates general word order statistics of language from information that is contained in the sentence embedding (Fig. 2).

The difference between including and removing the sentence embeddings when using the CBOW model is minor, while the LSTM-ED suffers a significant drop. Clearly, the LSTM-ED model encodes word order, while the prediction ability of CBOW is mostly explained by general language statistics. However, CBOW does benefit from the sentence to some extent: we observe a gain of $\sim$3% accuracy points when the CBOW tests are allowed access to the sentence representation. This may be explained by higher order statistics of correlation between word order patterns and the occurrences of specific words.

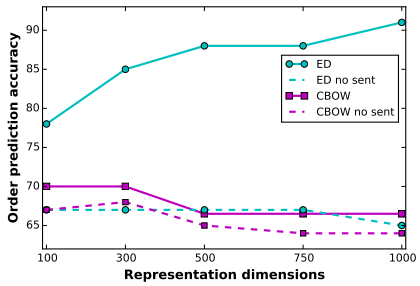
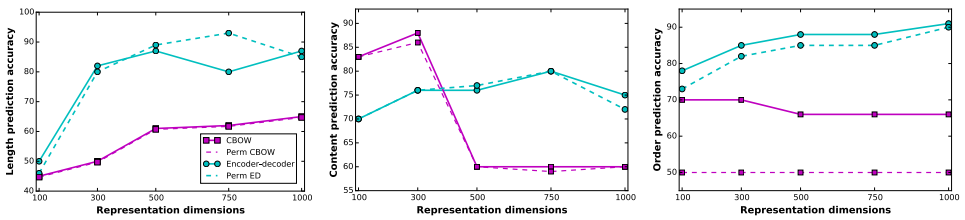


Figure 2: Order accuracy w/ and w/o sentence representation for ED and CBOW models.

How important is English word order for encoding sentences? To what extent are the models trained to rely on natural language word order when encoding sentences? To control for this, we create a synthetic dataset, PERMUTED, in which the word order in each sentence is randomly permuted. Then, we repeat the length, content and order experiments using the PERMUTED dataset (we still use the original sentence encoders that are trained on non-permuted sentences). While the permuted sentence representation is the same for CBOW, it is completely different when generated by the encoder-decoder.

Results are presented in Fig. 3. When considering CBOW embeddings, word order accuracy drops to chance level, as expected, while results on the other tests remain the same. Moving to the LSTM encoder-decoder, the results on all three tests are comparable to the ones using non-permuted sentences. These results are somewhat surprising since the models were originally trained on "real", non-permuted sentences. This indicates that the LSTM encoder-decoder is a general-purpose sequence encoder that for the most part does not rely on word ordering properties of natural language when encoding sentences. The small and consistent drop in word order accuracy on the permuted sentences can be attributed to the encoder relying on natural language word order to some extent, but can also be explained by the word order prediction task becoming harder due to the inability to



(a) Length test. (b) Content test. (c) Order test.

Figure 3: Results for length, content and order tests on natural and permuted sentences.

use general word order statistics. The results suggest that a trained encoder will transfer well across different natural language domains, as long as the vocabularies remain stable. When considering the decoder's BLEU score on the permuted dataset (not shown), we do see a dramatic decrease in accuracy. For example, LSTM encoder-decoder with 1000 dimensions drops from 32.5 to 8.2 BLEU score. These results suggest that the decoder, which is thrown away, contains most of the language-specific information.

8 SKIP-THOUGHT VECTORS

In addition to the experiments on CBOW and LSTM-encoders, we also experiment with the skip-thought vectors model (Kiros et al., 2015). This model extends the idea of the auto-encoder to neighboring sentences.

Given a sentence s_i, it first encodes it using an RNN, similar to the auto-encoder model. However, instead of predicting the original sentence, skip-thought predicts the preceding and following sentences, s_{i-1} and s_{i+1}. The encoder and decoder are implemented with gated recurrent units (Cho et al., 2014).

Here, we deviate from the controlled environment and use the author's provided model[3] with the recommended embeddings size of 4800. This makes the direct comparison of the models "unfair". However, our aim is not to decide which is the "best" model but rather to show how our method can be used to measure the kinds of information captured by different representations.

Table 1 summarizes the performance of the skip-thought embeddings in each of the prediction tasks on both the PERMUTED and original dataset.

	Length	**Word content**	**Word order**
Original	82.1%	79.7%	81.1%
Permuted	68.2%	76.4%	76.5%

Table 1: Classification accuracy for the prediction tasks using skip-thought embeddings.

The performance of the skip-thought embeddings is well above the baselines and roughly similar for all tasks. Its performance is similar to the higher-dimensional encoder-decoder models, except in the order task where it lags somewhat behind. However, we note that the results are not directly comparable as skip-thought was trained on a different corpus.

The more interesting finding is its performance on the PERMUTED sentences. In this setting we see a large drop. In contrast to the LSTM encoder-decoder, skip-thought's ability to predict length and word content does degrade significantly on the permuted sentences, suggesting that the encoding process of the skip-thought model is indeed specialized towards natural language texts.

9 CONCLUSION

We presented a methodology for performing fine-grained analysis of sentence embeddings using auxiliary prediction tasks. Our analysis reveals some properties of sentence embedding methods:

- CBOW is surprisingly effective – in addition to being very strong at content, it is *also predictive of length, and can be used to reconstruct a non-trivial amount of the original word order*. 300 dimensions perform best, with greatly degraded word-content prediction performance on higher dimensions.
- With enough dimensions, LSTM auto-encoders are very effective at encoding word order and word content information. Increasing the dimensionality of the LSTM encoder does not significantly improve its ability to encode length, but does increase its ability to encode content and order information. 500 dimensional embeddings are already quite effective for encoding word order, with little gains beyond that. Word content accuracy peaks at 750 dimensions and drops at 1000, suggesting that *larger is not always better*.

[3] https://github.com/ryankiros/skip-thoughts

- The trained LSTM encoder (when trained with an auto-encoder objective) *does not rely* on ordering patterns in the training sentences when encoding novel sequences.
  In contrast, the skip-thought encoder *does rely* on such patterns. Its performance on the other tasks is similar to the higher-dimensional LSTM encoder, which is impressive considering it was trained on a different corpus.
- Finally, the encoder-decoder's ability to recreate sentences (BLEU) is not entirely indicative of the quality of the encoder at representing aspects such as word identity and order. This suggests that *BLEU is sub-optimal for model selection*.

REFERENCES

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

Marco Baroni, Georgiana Dinu, and Germán Kruszewski. Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 238–247, Baltimore, Maryland, June 2014. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P14-1023.

Steven Bird. NLTK: the natural language toolkit. In *Proceedings of the COLING/ACL on Interactive presentation sessions*, pp. 69–72. Association for Computational Linguistics, 2006.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

Ronan Collobert, Koray Kavukcuoglu, and Clément Farabet. Torch7: A matlab-like environment for machine learning. In *BigLearn, NIPS Workshop*, number EPFL-CONF-192376, 2011.

Andrew M Dai and Quoc V Le. Semi-supervised sequence learning. In *Advances in Neural Information Processing Systems*, pp. 3061–3069, 2015.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159, 2011.

Jeffrey L Elman. Distributed representations, simple recurrent networks, and grammatical structure. *Machine learning*, 7(2-3):195–225, 1991.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *International Conference on Artificial Intelligence and Statistics*, pp. 315–323, 2011.

Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *Proceedings of ICASSP*, 2013.

Felix Hill, Kyunghyun Cho, and Anna Korhonen. Learning Distributed Representations of Sentences from Unlabelled Data. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 1367–1377, San Diego, California, June 2016. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/N16-1162.

Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, 2012.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8): 1735–1780, 1997.

Ákos Kádár, Grzegorz Chrupała, and Afra Alishahi. Representation of linguistic form and function in recurrent neural networks. *arXiv preprint arXiv:1602.08952*, 2016.

Andrej Karpathy, Justin Johnson, and Fei-Fei Li. Visualizing and understanding recurrent networks. *arXiv preprint arXiv:1506.02078*, 2015.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Skip-thought vectors. In *Advances in Neural Information Processing Systems*, pp. 3276–3284, 2015.

Nicholas Léonard, Sagar Waghmare, and Yang Wang. rnn: Recurrent library for torch. *arXiv preprint arXiv:1511.07889*, 2015.

Omer Levy and Yoav Goldberg. Linguistic regularities in sparse and explicit word representations. In *Proc. of CONLL*, pp. 171–180, Baltimore, Maryland, 2014.

Omer Levy, Yoav Goldberg, and Ido Dagan. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3: 211–225, 2015. ISSN 2307-387X. URL `https://tacl2013.cs.columbia.edu/ojs/index.php/tacl/article/view/570`.

Jiwei Li, Minh-Thang Luong, and Dan Jurafsky. A hierarchical neural autoencoder for paragraphs and documents. *arXiv preprint arXiv:1506.01057*, 2015.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013a.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pp. 3111–3119, 2013b.

Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pp. 807–814, 2010.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pp. 311–318. Association for Computational Linguistics, 2002.

Donald B Rubin. Matching to remove bias in observational studies. *Biometrics*, pp. 159–183, 1973.

Allen Schmaltz, Alexander M Rush, and Stuart M Shieber. Word ordering without syntax. *arXiv preprint arXiv:1604.08633*, 2016.

Ilya Sutskever, James Martens, and Geoffrey E Hinton. Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pp. 1017–1024, 2011.

Ilya Sutskever, Oriol Vinyals, and Quoc VV Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pp. 3104–3112, 2014.

Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop. *COURSERA: Neural networks for machine learning*, 2012.

Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

APPENDIX I: EXPERIMENTAL SETUP

Sentence Encoders The bag-of-words (CBOW) and encoder-decoder models are trained on 1 million sentences from a 2012 Wikipedia dump with vocabulary size of 50,000 tokens. We use NLTK (Bird, 2006) for tokenization, and constrain sentence lengths to be between 5 and 70 words.

For the CBOW model, we train Skip-gram word vectors (Mikolov et al., 2013a), with hierarchical-softmax and a window size of 5 words, using the Gensim implementation.[4] We control for the embedding size k and train word vectors of sizes $k \in \{100, 300, 500, 750, 1000\}$.

For the encoder-decoder models, we use an in-house implementation using the Torch7 toolkit (Collobert et al., 2011). The decoder is trained as a language model, attempting to predict the correct word at each time step using a negative-log-likelihood objective (cross-entropy loss over the softmax layer). We use one layer of LSTM cells for the encoder and decoder using the implementation in Léonard et al. (2015).

We use the same size for word and sentence representations (i.e. $d = k$), and train models of sizes $k \in \{100, 300, 500, 750, 1000\}$. We follow previous work on sequence-to-sequence learning (Sutskever et al., 2014; Li et al., 2015) in reversing the input sentences and clipping gradients. Word vectors are initialized to random values.

We evaluate the encoder-decoder models using BLEU scores (Papineni et al., 2002), a popular machine translation evaluation metric that is also used to evaluate auto-encoder models (Li et al., 2015). BLEU score measures how well the original sentence is recreated, and can be thought of as a proxy for the quality of the encoded representation. We compare it with the performance of the models on the three prediction tasks. The results of the higher-dimensional models are comparable to those found in the literature, which serves as a sanity check for the quality of the learned models.

Auxiliary Task Classifier For the auxiliary task predictors, we use multi-layer perceptrons with a single hidden layer and ReLU activation, which were carefully tuned for each of the tasks. We experimented with several network architectures prior to arriving at this configuration.

Further details regarding the training and architectures of both the sentence encoders and auxiliary task classifiers are available in the Appendix.

APPENDIX II: TECHNICAL DETAILS

ENCODER DECODER

Parameters of the encoder-decoder were tuned on a dedicated validation set. We experienced with different learning rates (0.1, 0.01, 0.001), dropout-rates (0.1, 0.2, 0.3, 0.5) (Hinton et al., 2012) and optimization techniques (AdaGrad (Duchi et al., 2011), AdaDelta (Zeiler, 2012), Adam (Kingma & Ba, 2014) and RMSprop (Tieleman & Hinton, 2012)). We also experimented with different batch sizes (8, 16, 32), and found improvement in runtime but no significant improvement in performance.

Based on the tuned parameters, we trained the encoder-decoder models on a single GPU (NVIDIA Tesla K40), with mini-batches of 32 sentences, learning rate of 0.01, dropout rate of 0.1, and the AdaGrad optimizer; training takes approximately 10 days and is stopped after 5 epochs with no loss improvement on a validation set.

PREDICTION TASKS

Parameters for the predictions tasks as well as classifier architecture were tuned on a dedicated validation set. We experimented with one, two and three layer feed-forward networks using ReLU (Nair & Hinton, 2010; Glorot et al., 2011), tanh and sigmoid activation functions. We tried different hidden layer sizes: the same as the input size, twice the input size and one and a half times the input size. We tried different learning rates (0.1, 0.01, 0.001), dropout rates (0.1, 0.3, 0.5, 0.8) and different optimization techniques (AdaGrad, AdaDelta and Adam).

[4] https://radimrehurek.com/gensim

Our best tuned classifier, which we use for all experiments, is a feed-forward network with one hidden layer and a ReLU activation function. We set the size of the hidden layer to be the same size as the input vector. We place a softmax layer on top whose size varies according to the specific task, and apply dropout before the softmax layer. We optimize the log-likelihood using AdaGrad. We use a dropout rate of 0.8 and a learning rate of 0.01. Training is stopped after 5 epochs with no loss improvement on the development set. Training was done on a single GPU (NVIDIA Tesla K40).

10 Additional Experiments - Content Task

How well do the models preserve content when we increase the sentence length? In Fig. 4 we plot content prediction accuracy vs. sentence length for different models.

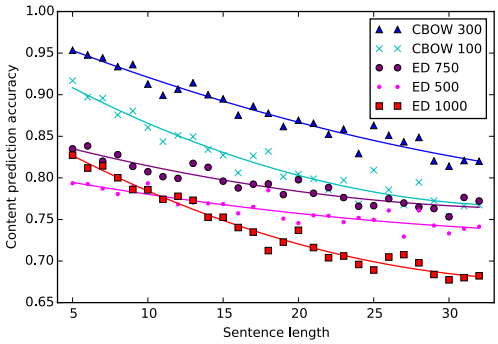


Figure 4: Content accuracy vs. sentence length for selected models.

As expected, all models suffer a drop in content accuracy on longer sentences. The degradation is roughly linear in the sentence length. For the encoder-decoder, models with fewer dimensions seem to degrade slower.

Appendix III: Significance Tests

In this section we report the significance tests we conduct in order to evaluate our findings. In order to do so, we use the paired t-test (Rubin, 1973).

All the results reported in the summery of findings are highly significant (p-value $\ll 0.0001$). The ones we found to be not significant (p-value $\gg 0.03$) are the ones which their accuracy does not have much of a difference, i.e ED with size 500 and ED with size 750 tested on the word order task (p-value=0.11), or CBOW with dimensions 750 and 1000 (p-value=0.3).

Dim.	**Length**	**Word content**	**Word order**
100	1.77e-147	0.0	1.83e-296
300	0.0	0.0	0.0
500	0.0	0.0	0.0
750	0.0	0.0	0.0
1000	0.0	0.0	0.0

Table 2: P-values for ED vs. CBOW over the different dimensions and tasks. For example, in the row where dim equals 100, we compute the p-value of ED compared to CBOW with embed size of 100 on all three tasks.

Dim.	**Length**	**Word content**	**Word order**
100 vs. 300	0.0	8.56e-190	0.0
300 vs. 500	7.3e-71	4.20e-05	5.48e-56
500 vs. 750	3.64e-175	4.46e-65	**0.11**
750 vs. 1000	1.37e-111	2.35e-243	4.32e-61

Table 3: P-values for ED models over the different dimensions and tasks.

Dim.	**Length**	**Word content**	**Word order**
100 vs. 300	0.0	0.0	1.5e-33
300 vs. 500	1.47e-215	0.0	3.06e-64
500 vs. 750	**0.68**	**0.032**	**0.05**
750 vs. 1000	4.44e-32	**0.3**	**0.08**

Table 4: P-values for CBOW models over the different dimensions and tasks.

Analysis of Sentence Embedding Models using Prediction Tasks in Natural Language Processing

Analysis of sentence embedding models using prediction tasks in natural language processing

Y. Adi
E. Kermany
Y. Belinkov
O. Lavi
Y. Goldberg

The tremendous success of word embeddings in improving the ability of computers to perform natural language tasks has shifted the research on language representation from word representation to focus on sentence representation. This shift introduced a plethora of methods for learning vector representations of sentences, many of them based on compositional methods over word embeddings. These vectors are used as features for subsequent machine learning tasks or for pretraining in the context of deep learning. However, not much is known about the properties that are encoded in these sentence representations and about the language information they encapsulate. Recent studies analyze the encoded representations and the kind of information they capture. In this paper, we analyze results from a previous study on the ability of models to encode basic properties such as content, order, and length. Our analysis led to new insights, such as the effect of word frequency or word distance on the ability to encode content and order.

1. Introduction

Much of the research on neural network models for natural language processing (NLP) is directed toward the concept of *sentence embeddings*. The idea of sentence embeddings is to encode sentences, which are variable-length sequences of discrete symbols, into fixed-length continuous vectors that preserve the properties of the sentence as well as the syntactics and semantics of the sentence. These vectors can then be used for further prediction tasks or as pre-training for non-convex models such as deep-learning models.
A simple and common approach is to use word-level vectors, for example, derived by word2vec [1, 2], and then summing up or averaging the vectors of the words participating in the sentence. This continuous-bag-of-words (CBOW) approach disregards the word order in the sentence. Another approach is the *encoder-decoder* (ED) architecture, sometimes known as s*equence-to-sequence* models. In these models, an encoder network (e.g., an LSTM, or long short-term memory) receives as input a sequence of discrete symbols (e.g., words), and produces a vector representation. This vector is then fed into a decoder network to perform some prediction task (e.g., recreate the sentence, or produce a translation of it). The encoder and decoder networks are trained jointly in order to perform the final task. One common case is to train an encoder-decoder network, then discard the decoder and use the trained encoder as a general mechanism for obtaining sentence representations. For example, an encoder-decoder network can be trained as an auto-encoder, where the encoder creates a vector representation, and the decoder attempts to recreate the original sentence [3]. Similarly, Kiros et al. [4] train a network to encode a sentence such that the decoder can recreate its neighboring sentences in the text.

Such models rely on the ability of the encoder network to capture a non-trivial amount of information about the sentence. Furthermore, they do not require specially labeled data, and can be trained on large amounts of unannotated text. This makes them appealing as general-purpose, stand-alone sentence-encoding mechanisms. The sentence encodings can then be used as input for other prediction tasks with less training data available [5].

While informal observations have shown that the internal sentence representation of such models can reflect semantic

Digital Object Identifier: 10.1147/JRD.2017.2702858

© **Copyright** 2017 by International Business Machines Corporation. Copying in printed form for private use is permitted without payment of royalty provided that (1) each reproduction is done without alteration and (2) the Journal reference and IBM copyright notice are included on the first page. The title and abstract, but no other portions, of this paper may be copied by any means or distributed royalty free without further permission by computer-based and other information-service systems. Permission to republish any other portion of this paper must be obtained from the Editor.

0018-8646/17 © 2017 IBM

intuitions [6], we do not know what the vectors produced by the encoder actually encode, what the "division of labor" between the encoder and decoder networks is, or whether we can use these representations for other classification tasks.

In this paper, we extend our work from 2016 [7], where we analyzed different sentence representations using simple prediction tasks, aimed at recovering encoded information within the sentence. For example, we explore the use of a different loss function to extract length information more accurately. We also evaluate the effect of language properties on the encoded information, for example the ability to encode word order as a function of the distance between words or the ability to encode word identity as a function of word frequency.

The paper is organized as follows. In Section 2, we provide a short overview of current methods for analyzing sentence representations. We give a short summary of our methodology in Section 3. In Sections 4 and 5, we provide a description of the models and the technical details involved in implementing them. We show results and analysis in Sections 6 and 7. We close with a discussion and conclusion in Section 8.

2. Related work

The success of word embeddings has led researchers to analyze the learned representations in different ways, whether by empirically evaluating them on various tasks [8] or by more theoretical investigations [9]. More recently, sentence embeddings have attracted growing interest [10, 11]. Several studies analyze hidden units (neural network elements) learned by a sentence representation model, especially in recurrent neural networks [12–15]. Another interesting approach is to visualize hidden activations (the values of the hidden units) or other patterns [16, 17]. Empirically evaluating different sentence representations on downstream tasks (i.e., sentiment analysis, paraphrase detections, etc.), remains a common approach [18].

These techniques typically depend on a specific learning model or are limited to certain downstream tasks. They do not reveal to what extent sentence properties are captured by the final representation. In previous work [7], we proposed a methodology for measuring to what extent sentence characteristics are preserved in the final representation. The method is generic and can be applied to any representation model. We provide more details on the approach below.

Similar ideas have recently started to be published. For example, Shi et al. [19] train classifiers on sentence representations learned by neural machine translation to predict syntactic labels. They found that the decoder indeed captures syntactic information about the source sentence, and different information is stored in different layers. Ettinger et al. [20] proposed evaluating classification tasks of semantic properties such as semantic roles.

We follow the methodology of posing auxiliary prediction tasks to probe into sentence representations. We extend each of our original prediction tasks (word content, word order, and sentence length), and provide a deeper analysis that leads to new insights.

3. Approach

Our goal is to inspect and compare sentence representations in a task-independent manner. Our approach measures to what extent a given sentence representation can recover specific aspects from the sentence structure. Given a sentence representation method, we create training data and train a classifier to predict a specific sentence property (e.g., the sentence length) based on the vector representations. We then score the method by measuring how well the classifier performed this task. Our basic premise is that if we cannot train a classifier to predict some property of a sentence based on its vector representation, this property is not encoded in the representation (or rather, not encoded in a useful way, considering how the representation is likely to be used).

We use the following notation: lower case italics (s,w) refers to sentences and words, and boldface refers to their corresponding vector representations ($\mathbf{s}$, $\mathbf{w}$). We use indices (w_1, w_2, $\mathbf{w_1}$, $\mathbf{w}_2$) to distinguish between more than one element. We follow the same prediction tasks used by Adi et al. [7] and extend their analysis. These experiments focus on low-level properties of sentences—the sentence length, the identities of words in a sentence, and the order of the words. We consider these to be the core elements of any sequence. Generalizing the approach to higher-level semantic and syntactic properties holds great potential, which we hope will be explored in future work, by us or by others. The following is a brief summary of the tasks.

- *Length task*: Measures to what extent the sentence representation encodes its length. Given a sentence representation $\mathbf{s} \in \mathrm{R}^k$, the goal of the classifier is to predict the length (number of words) in the original sentence s. (Here, k is the sentence embedding dimension.) The task is formulated as multiclass classification, with eight output classes corresponding to binned lengths.
- *Word-content task*: Measures to what extent the sentence representation encodes the identities of words within it. Given a sentence representation $\mathbf{s} \in \mathrm{R}^k$ and a word representation $\mathbf{w} \in \mathrm{R}^d$, the goal of the classifier is to determine whether word w appears in the sentence s, with access to neither w nor s. (Here, d is the word embedding dimension.) The dataset is composed of positive and negative examples. The positive examples are pairs of a sentence and a word that appears in the sentence,

and negative examples are pairs of a sentence and a word that does not appear in the sentence.

- *Word-order task*: Measures to what extent the sentence representation encodes word order. Given a sentence representation $\mathbf{s} \in \mathrm{R}^k$ and the representations of two words that appear in the sentence, $\mathbf{w_1}$, $\mathbf{w_2} \in \mathrm{R}^d$, the goal of the classifier is to predict whether w_1 appears before or after w_2 in the original sentence s. Again, the classifier does not have access to w_1, w_2, or s. This is formulated as a binary classification task, where the input is a concatenation of the three vectors s, w_1, and w_2. For each sentence in the corpus, we simply pick two random words from the sentence as a positive example. For negative examples, we flip the order of the words.

4. Sentence representation models

Given a sentence $s = \{w_1, w_2, \ldots, w_N\}$, where N, is the number of words, we aim to find a sentence representation $\mathbf{s}$ using an encoder:

$$\mathrm{ENC} : s = \{w_1, w_2, \ldots, w_N\} \rightarrow \mathbf{s} \in \mathrm{R}^k$$

The encoding process usually assumes a vector representation $\mathbf{w_i} \in \mathrm{R}^d$ for each word in the vocabulary. In general, the word and sentence embedding dimensions, d and k, need not be the same. The word vectors can be learned together with other encoder parameters, or pre-trained. We explore two common approaches for encoding sentences, described in the next two subsections.

4.1. Continuous bag-of-words

This simple yet effective text representation model performs element-wise averaging of word vectors that are obtained using a word-embedding method such as word2vec. Despite being oblivious to word order, CBOW proves useful in different tasks [18] and is easy to compute, making it an important model class to consider.

4.2. Encoder-decoder

A number of sequence-to-sequence learning tasks [3, 5, 21, 22] successfully use the encoder-decoder framework. After the encoding phase, a decoder maps the sentence representation back to the sequence of words:

$$\mathrm{DEC} : \mathbf{s} \in \mathrm{R}^k \rightarrow s = \{w_1, w_2, \ldots, w_N\}$$

We investigate the specific case of an auto-encoder, where the entire encoding-decoding process can be trained end-to-end from a corpus of raw texts. The sentence representation is the final output vector of the encoder. As mentioned, we use a long short-term memory (LSTM) recurrent neural network [23, 24] for both encoder and decoder. The LSTM decoder is similar to the LSTM encoder but with different weights. For the description of the LSTM unit see the Appendix of this paper.

5. Experimental setup

5.1. Sentence encoders

In this section, we describe the technical details for training both CBOW and ED. The bag-of-words (CBOW) and encoder-decoder models are trained on 1 million sentences from a 2012 Wikipedia[**] database dump. We use NLTK (natural language toolkit) [25] for tokenization, and constrain sentence lengths to between 5 and 70 words.

For the CBOW model, we train Skip-gram word vectors [1], with hierarchical-softmax and a window size of 5 words, using the Gensim implementation [26]. We control for the embedding size k and train word vectors of sizes $k \in \{100, 300, 500, 750, 1{,}000\}$.

For the encoder-decoder models, we use an in-house implementation using the Torch7 toolkit [27]. The decoder was trained to predict the correct word at each time step using a negative-log-likelihood objective (cross-entropy loss over the softmax layer)—with a dictionary size of 50,000. We used one layer of LSTM cells for the encoder and decoder using the implementation in [28]. We used the same size for word and sentence representations (i.e., $d = k$), and trained models of sizes $k \in \{100, 300, 500, 750, 1{,}000\}$. We followed previous work on sequence-to-sequence learning [3, 21] in reversing the input sentences and clipping gradients. We initialized word vectors to random values.

We evaluated the encoder-decoder models using BLEU (bilingual evaluation understudy) scores [29], a popular machine translation evaluation metric that is also used to evaluate auto-encoder models [3]. The results of the higher-dimensional models are comparable to those found in the literature, which serves as a "sanity check" for the quality of the learned models.

5.2. Auxiliary task classifier

Consistent with the findings by Adi et al. [7], we used the same corpus for generating classification instances. The corpus consists of 200,000 Wikipedia sentences—150,000 sentences to generate training examples, and 25,000 sentences for each of the test and development examples. These sentences are a subset of the training set that we used to train the original sentence encoders. We used this setup to test the models on what are presumably their best embeddings.

We used multi-layer perceptrons with a single hidden layer and ReLU (rectified linear unit) activation; the size of the hidden layer is the same as the size of the input. We carefully tuned the network classifier for each of the tasks and experimented with several network architectures prior to arriving at this configuration.

6. Summary of previous results

Our previous analysis revealed interesting insights on the encoded information captured by different sentence embedding methods. In this section, we briefly review the main findings (see [7] for more details).

Despite its simplicity, CBOW performs very well on the different tasks. Although composed as an average of word representations, CBOW managed to perform very well on the content test. Due to *a priori* information regarding word order statistics, (i.e., certain words tend to appear before others), CBOW performs well in the word order task. It is even capable of encoding sentence length information. The best performance is in 300 dimensions, with a decline in the results for higher dimensions.

The LSTM auto-encoders are very effective at encoding order and length information, and less effective on content. Higher dimensions do not necessarily improve the ability to encode length and content but do improve encoding order. The trained LSTM encoder (when trained with an auto-encoder objective) does not rely on ordering patterns in the training sentences when encoding novel sequences. In addition, the BLEU score (the ability to recreate the sentence) is not entirely correlated with the embedding quality—the quality of the encoder at representing word identity and order. This suggests that BLEU is sub-optimal for model selection.

7. Extensions

For each of the three main tasks—length, content and order—we defined extensions that illuminate other aspects of the sentence embeddings. Each extension provides a deeper analysis over the previous results. In the following section we review these extensions.

7.1. Word-content task

We provide three extensions to the content experiments. First, we explore the ability of the different models to generalize. Second, we challenge our classifiers by designing a more complicated word-content prediction task. Last, we investigate the effect of word frequency on the ability to succeed in the word-content task.

7.1.1. Generalization ability

In the word-content task, our goal is to qualify the relation between word and sentence representations. We used binary classification in this task, where for every sentence we generated positive and negative examples: the positive example is a word from the sentence and the negative example is a word that does not appear in the sentence but appears as a positive example in other sentences. We followed this approach in order to avoid lexical memorization [30], (i.e., if we randomly pick words as negative examples, some of them will appear only as positive or negative examples. Hence, we limited the negative sampling to words that were considered as positive for at least one sentence). However, we do not know how the similarity between the words affects the performance in this task. As a proxy for similarity, we considered the distance between word embeddings. In other words, we are interested in exploring the effect of the distance between word embeddings to the models' capability to identify words from the sentence.

Following the word-content procedure, we randomly picked a word w from each sentence and computed its 1st, 10th, 50th, 100th, and 500th closest words (we used the Euclidian distance as our metric), denoted as w_1, w_{10}, w_{50}, w_{100}, and w_{500}. Then, we considered each of the words as a negative example and created different train, test, and validation sets for it. We created the datasets the same way that we did for the word-content task. In this way, we can better qualify the ability of the model to classify words from the sentence.

The results of the ED models show an increase of 3% to 12% in accuracy as the distance between the words increases, while in CBOW they increase by 15% to 20%. Regarding the dimensionality of the representation, unlike ED with an embedding size of 750 and 500, ED with an embedding size of 1,000 performs roughly the same also when considering embeddings that are further away. In CBOW, 300 dimensions are better than 100 dimensions. The results are summarized in **Figure 1**.

In this work, we consider a *generalized model* as one that treats words that do not appear in the sentence in a similar way, (i.e., it has roughly the same accuracy when changing the negative example from w_{10} to w_{50}). According to this definition, we can say that the ED model can better generalize than CBOW.

Next, we extended this experiment not only to consider words from the sentence as a positive example, but also close vectors. For instance, we used w_1 as a positive example and w_{10} as a negative example. For each pair of the words, we generated a different task, leading to 10 additional tasks. The idea behind this task is to assess the ability of the model to find relations between sentences and representations which are close to words from the sentence.

In **Figure 2** we present the results of using w_1 as a positive example. We can see that all models suffer a decline in accuracy when compared to the original content results. The highest drop is for the ED with a dimension size of 1,000. When considering the generalization ability as described above, this indicates that ED models generalize better than CBOW with roughly 8% gain in accuracy for all models, whereas in CBOW we see a gain of 12% to 18% in accuracy. We observed similar results for w_{10} vs. all and w_{50} vs. all (results are not shown).

7.1.2. Predict the k-th word

The word-content task is a binary classification task that predicts whether or not a random word appears in the

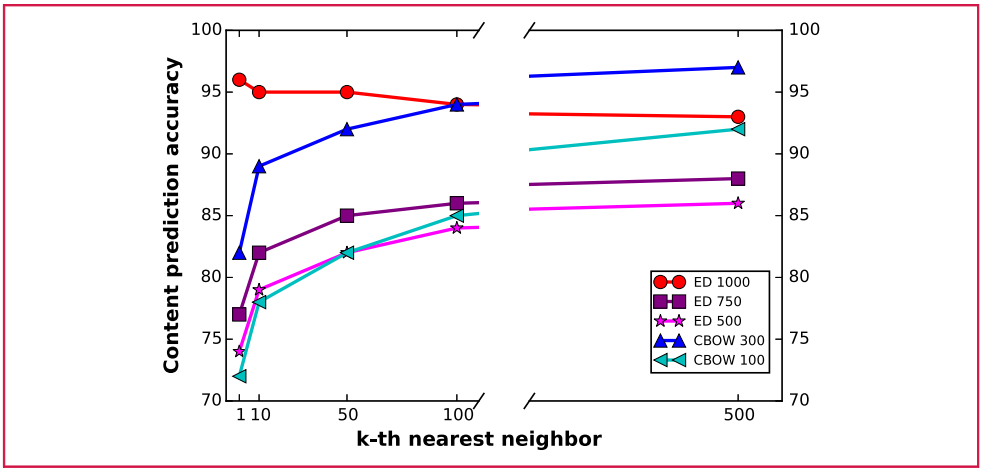


Figure 1

Generalization experiments results for continuous-bag-of-words (CBOW) (embedding size of 100 and 300) and encoder-decoder models (embedding size of 500, 750 and 1,000). The experiments were performed by randomly picking a word w from each sentence as positive samples and its 1st, 10th, 50th, 100th, and 500th closest words as negative samples.

sentence. Our previous work showed that all of the models performed this task well. Here, we try to challenge the models by defining a much harder task—predicting a word in a stated location. Specifically, given the sentence representation, our task is to predict the k-th word in the sentence. To achieve this, we trained a different classifier network for each location. For example, when $k = 2$, the network is trained to predict the second word in the sentences. Note that this is a multiclass classification task where the number of the labels is the size of the dictionary, 50,000 in our case.

Table 1 shows our main results. We only present the accuracy of predicting the first to fourth words. For all models, the ability to predict words later in the sentence is very poor. In addition, our experiments show that using CBOW representations leads to very poor results (<1% accuracy), as might be expected; hence, they do not appear in the table.

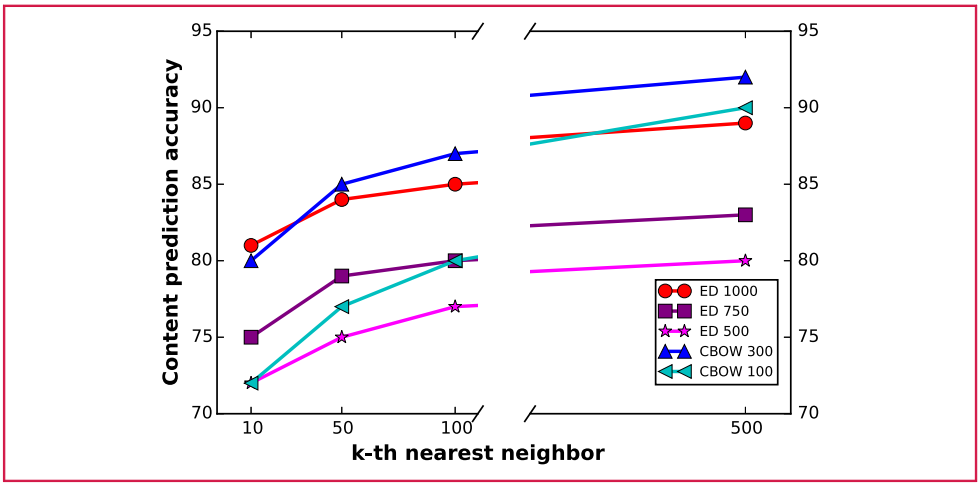


Figure 2

Generalization experiments results for continuous-bag-of-words (CBOW) (embedding size of 100 and 300) and encoder-decoder models (embedding size of 500, 750, and 1,000). The experiments were performed by randomly picking a word w from each sentence and using its closest word as positive samples and the 10th, 50th, 100th, and 500th closest words as negative samples.

Table 1 Classification accuracy for predicting the *k*-th word.

Model	*First word*	*Second word*	*Third word*	*Fourth word*
ED 1,000	61.8%	28.4%	26.8%	17%
ED 750	93%	50.6%	29%	16.2%
ED 500	81.6%	23.2%	7%	5.2%
ED 300	62.8%	8.4%	4.2%	0.6%
ED 100	45.4%	3.8%	0.4%	0%

Clearly, the encoder-decoder with an embedding size of 750 outperforms the rest of the models. In addition, sentence representations created by encoder-decoder models mostly contain information regarding the first word in the sentence, but quickly degrade as we move away from the beginning of the sentence. This phenomenon can be explained by the behavior of the encoder-decoder model, which learns to predict the first word of the sentence using the representation created by the encoder, and then continues to predict the rest of the sentence by using the last predicted word. Therefore, the encoder representation (the representation we analyze here) mainly has information about the first word in the sentence. This ability is unique to the encoder-decoder models and does not exist in CBOW models.

7.1.3. The effect of word frequency

Our last extension to the word-content task concerns the effect of word frequency on the performance of the classification task. **Figure 3** shows the mean accuracy of the content prediction task as a function of word frequency in the test set.

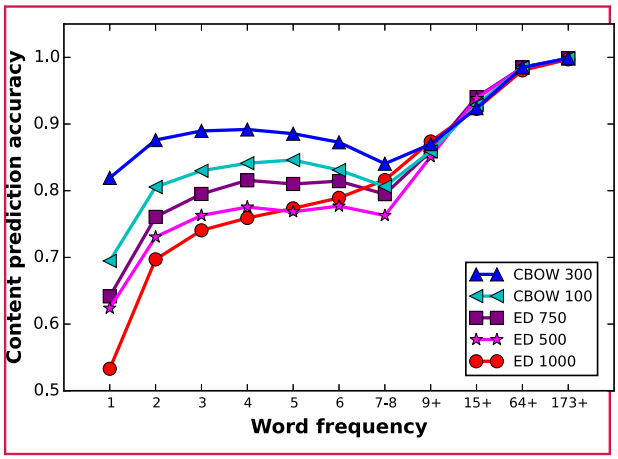


Figure 3

Mean accuracy of word content task vs. word frequency in the dataset. For continuous-bag-of-words (CBOW) we present results of embedding size 100 and 300; for the encoder-decoder, we present results of embedding size 500, 750, and 1,000. Since words that appear more than 173 times are rare in our dataset, we aggregate them to the same bin.

All models "struggle" with low-frequency words, except for CBOW, with an embedding size of 300, which is relatively more consistent across frequencies. In other models, there is a huge gap between the performance on high-frequency and low-frequency words. The encoder-decoder model with an embedding size of 1,000 is most affected by word frequency. There is a drop in the accuracy in the word frequency range of 7 to 8 for both CBOW models and for ED with a dimension size of 750. We will explore what causes the drop in future work.

7.2. Word-order task

Our goal in the word-order task is to qualify the ability to predict the order of two words in a given sentence. We assume that the distance between words in a sentence can strongly influence the results of this task.

Figure 4 shows the accuracy of the word-order task as a function of the distance between the words in the sentence. [Consecutive distances are merged when there are insufficient examples (at least 1,000). For example, 13 and 14 were merged into one bin, 13-14.]

Clearly, the most difficult case for all the models occurs when the distance between the words is one or two. Beyond a distance of two, it is easier to predict the order of two words that are either relatively close to each other or far apart, than to predict the order of words with a medium distance between them. This behavior makes intuitive sense when considering the behavior of words in English—local word order follows relatively strong constraints, and some words tend to appear more in the beginning or end of sentences. The trend is more noticeable for CBOW encodings, which suggests that these methods rely more on the natural order of words rather than encoding the word order in a particular sentence.

7.3. Length task

In Adi et al. [7], we investigated the ability of representation models to encode length. We chose to extract this information using a multiclass classification setting.

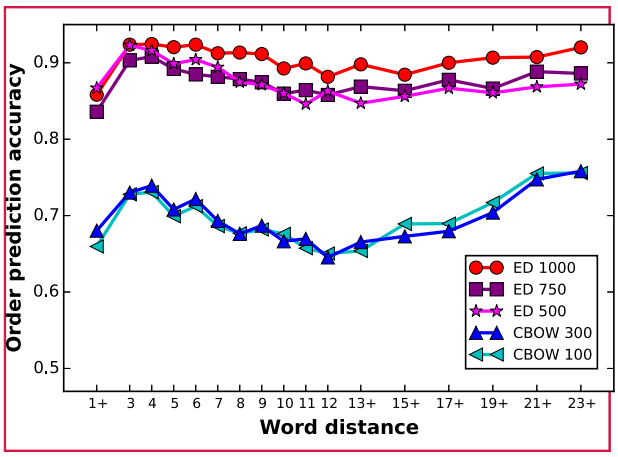


Figure 4

Mean accuracy for the word order task vs. word distance (the number of words between the two words in the sentence). For continuous-bag-of-words (CBOW), we present results of embedding size 100 and 300; for the encoder-decoder, we present results of embedding size 500, 750, and 1,000.

We minimized the negative log likelihood loss function, denoted by NLL, where the input is the sentence embedding and the output is a range of sentence lengths (with softmax normalization). Here, we considered an alternative option for extracting this information by replacing the multiclass layer with a regression layer and predicting the exact length. Since we maintain a vector for every token, we consider the sentence length as the number of tokens in the sentence.

The architecture of the regression model is very similar to the one we used for the multiclass version. Both of them have a single hidden layer with a ReLU activation function. The hidden layer size is the same as the input size. In the regression model, we considered the sentence length as a target label. We removed the softmax layer and changed the loss function from negative log likelihood to mean absolute error. We used the AdaGrad algorithm as optimizer with learning rate of 0.01 and a dropout rate of 0.8.

Table 2 summarizes the results for the length task using the regression model. We see an improvement in the results for the CBOW models as the dimension size grows. ED with a dimension of 750 suffers a drop in accuracy, which also occurs in the multiclass version. The motivation behind using a regression model is to create an absolute prediction. Using a regression loss function reveals the absolute distance between the prediction and target label. In this way, we can have a better assessment of the length information that is hidden in the representation. For example, while using multiclass classification, we binned together sentences of length 30 to 40, while in the regression model, we can evaluate each of them separately.

Table 2 Results for the length experiments using mean absolute error for ED and CBOW models with different embedding sizes.

Model	*100*	*300*	*500*	*750*	*1,000*
ED	3.39	1.07	0.631	0.762	0.632
CBOW	4.9	4.22	3.87	3.75	3.39

8. Conclusion

We extended the analysis of sentence embedding models and the linguistic properties they capture. We investigated, in depth, the length, the content, and the order properties of the representation created by the models. We validated results from a previous study regarding length information using a different loss function (regression), which also revealed a more absolute measure to evaluate the performance of the models.

All models, except for CBOW with 300 dimensions, are affected by the word frequency when identifying words from the sentence. CBOW shows consistent results for low- frequency and high-frequency words.

ED models are less sensitive to the distance between words in the sentence when encoding word order. CBOW, however, is more sensitive to the distance between the words. Considering the way it is composed (average of word representations), we assume it is related to natural language properties, (i.e., some words tend to appear in beginnings or ends of sentences). Lastly, ED models tend to generalize better than CBOW models, where CBOW with 300 dimensions is better than CBOW with 100 dimensions. However, they are less successful in finding relations to representations that are close to words from the sentence.

For future work, we would like to explore semantic tasks that are related to higher-level sentences properties, as well as to examine new embedding models.

Appendix

Technical details—Auxiliary task classifier

We tuned the parameters for the predictions tasks as well as classifier architecture on a dedicated validation set. We used one, two, and three layer feed-forward networks using ReLU [31, 32], tanh, and sigmoid activation functions. We tried to use a linear classifier instead of non-linear, with poor results. We tried different hidden layer sizes—the same as the input size, twice as the input size, and one and a half times as the input size. We tried different learning rates (0.1, 0.01, and 0.001), dropout rates (0.1, 0.3, 0.5, and 0.8), and different optimization techniques (AdaGrad, AdaDelta, and Adam).

Our best tuned classifier, which we use for all experiments, is a feed-forward network with one hidden layer and a ReLU activation function. We set the size of

the hidden layer to be the same size as the input vector. We placed a softmax layer on top (i.e., we connected the last layer with a softmax layer), the size varying according to the specific task, and applied dropout before the softmax layer. We optimized the log-likelihood using AdaGrad. We used a dropout rate of 0.8 and a learning rate of 0.01. We stopped training after 5 epochs with no loss improvement on the development set. Training was performed on a single GPU (graphics processing unit) (NVIDIA Tesla K40).

Long short-term memory unit

We describe a single LSTM layer:

$$\mathbf{i}_t = \sigma(\mathbf{W}_{\mathrm{xi}}\mathbf{x}_t + \mathbf{W}_{\mathrm{hi}}\mathbf{h}_{t-1} + \mathbf{W}_{\mathrm{ci}}\mathbf{c}_{t-1} + b_i)$$
$$\mathbf{f}_t = \sigma(\mathbf{W}_{\mathrm{xf}}\mathbf{x}_t + \mathbf{W}_{\mathrm{hf}}h_{t-1} + \mathbf{W}_{\mathrm{cf}}\mathbf{c}_{t-1} + b_{\mathrm{f}})$$
$$\mathbf{c}_t = \mathbf{f}_t \odot \tanh(\mathbf{W}_{\mathrm{xc}}\mathbf{x}_t + \mathbf{W}_{\mathrm{hc}}\mathbf{h}_{t-1} + b_{\mathrm{c}})$$
$$\mathbf{o}_t = \sigma(\mathbf{W}_{\mathrm{xo}}\mathbf{x}_{\mathrm{t}} + \mathbf{W}_{\mathrm{ho}}h_{t-1} + \mathbf{W}_{\mathrm{co}}\mathbf{c}_t + b_{\mathrm{o}})$$
$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t),$$

where σ is the sigmoid function, $\odot$ is element-wise multiplication, and $\mathbf{i}$, $\mathbf{f}$, $\mathbf{o}$ and $\mathbf{c}$ are input, forget, output, and memory cell activation vectors. The crucial element is the memory cell c, which is able to store and reuse long term dependencies over the sequence. The $\mathbf{W}$ matrices and b bias terms are learned during training. $\mathbf{x}$ is the input, and it is a vector.

Acknowledgments

We thank G. Lev, B. Carmeli, E. Mezuman, O. Sar-Shalom, M. Ninio, and E. Shaked from the Machine Learning Technologies group at the IBM Research - Haifa lab for the fruitful discussion and contributions. This work was supported in part by the CONSENSUS project, which is funded by the European Commission within its FP7 Programme (contract 611688 funded under call FP7-ICT-10-5.4).

**Trademark, service mark, or registered trademark of Wikimedia Foundation in the United States, other countries, or both.

References

1. T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," arXiv:1301.3781, 2013.
2. T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proc. 26th Int. Conf. Adv. Neural Inf. Process. Syst.*, 2013, pp. 3111–3119.
3. J. Li, M.-T. Luong, and D. Jurafsky, "A hierarchical neural autoencoder for paragraphs and documents," in *Proc. 53rd Annu. Meeting Assoc. Comput. Linguistics 7th Int. Joint Conf. Nat. Lang. Process.*, Beijing, China, Jul. 26–31, 2015, pp. 1106–1115.
4. R. Kiros, Y. Zhu, R. R. Salakhutdinov, R. Zemel, R. Urtasun, A. Torralba, and S. Fidler, "Skip-thought vectors," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 3276–3284.
5. A. M. Dai and Q. V. Le, "Semi-supervised sequence learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 3061–3069.
6. K. Cho, M. B. Van, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2014.
7. Y. Adi, E. Kermany, Y. Belinkov, O. Lavi, and Y. Goldberg, "Fine-grained analysis of sentence embeddings using auxiliary prediction tasks," in *ICLR 2016*, arXiv:1608.04207.
8. M. Baroni, G. Dinu, and G. Kruszewski, "Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors," in *Proc. 52nd Annu. Meeting Assoc. Comput. Linguistics*, Baltimore, MD, USA, Jun. 23–25 2014, pp. 238–247.
9. O. Levy and Y. Goldberg, "Neural word embedding as implicit matrix factorization," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 2177–2185.
10. D. Paperno, N. Pham, and M. Baroni, "A practical and linguistically-motivated approach to compositional distributional semantics," in *Proc. 52nd Annu. Meeting Assoc. Comput. Linguistics*, 2014, pp. 90–99.
11. K. S. Tai, R. Socher, and C. D. Manning, "Improved semantic representations from tree-structured long short-term memory networks," in *Proc. 53rd Annu. Meeting Assoc. Comput. Linguistics 7th Int. Joint Conf. Natural Lang. Process.*, Beijing, China, Jul. 26–31, 2015, pp. 1556–1566.
12. J. L. Elman, "Distributed representations, simple recurrent networks, and grammatical structure," *Mach. Learn.*, vol. 7, no. 2/3, pp. 195–225, 1991.
13. A. Karpathy, J. Johnson, and L. Fei-Fei, "Visualizing and understanding recurrent networks," in *ICLR 2016 workshop track*, arXiv:1506.02078.
14. A. Kadar, G. Chrupala, and A. Alishahi, "Representation of linguistic form and function in recurrent neural networks," *Trans. Assoc. Comput. Linguistics*, 2016.
15. X. Shi, K. Knight, and D. Yuret, "Why neural translations are the right length," in *Proc. 2016 Conf. Empirical Methods Natural Lang. Process.*, Austin, TX, USA, 2016, pp. 2278–2282.
16. Z. Wu and S. King, "Investigating gated recurrent networks for speech synthesis," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2016, pp. 5140–5144.
17. H. Strobelt, S. Gehrmann, B, Huber, H. Pfister, and A. M. Rush, "Visual analysis of hidden state dynamics in recurrent neural networks," arXiv:1606.07461, 2016.
18. F. Hill, K. Cho, and A. Korhonen, "Learning distributed representations of sentences from unlabelled data," arXiv:1602.03483, 2016.
19. X. Shi, I Padhi, and K. Kevin, "Does string-based neural MT learn source syntax?" in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2016, pp. 1526–1534.
20. A. Ettinger, A. Elgohary, and R. Philip, "Probing for semantic evidence of composition by means of simple classification tasks," in *Proc. 1st Workshop Eval. Vector Space Representations NLP*, Berlin, Germany, 2016, pp. 134–139.
21. I. Sutskever, O. Vinyals, and Q. VV. Le, "Sequence to sequence learning with neural networks," in *Proc. 27th Int. Conf. Adv. Neural Inf. Process. Syst.*, 2014, pp. 3104–3112.
22. D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *Proc. Int. Conf. Learn. Representations*, 2015.
23. S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.
24. A. Graves, A. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2013, pp. 6645–6649.
25. S. Bird, "NLTK: The natural language toolkit," in *Proc. COLING/ACL Interact. Presentation Sessions*, 2006, pp. 69–72.
26. R. Radim and P. Sojka, "Software framework for topic modelling with large corpora," in *Proc. LREC 2010 Workshop New Challenges NLP Frameworks*, 2010, pp. 45–50.
27. R. Collobert, K. Kavukcuoglu, and C. Farabet, "Torch7: A MATLAB-like environment for machine learning," in *Proc. BigLearn, NIPS Workshop*, 2011, Paper EPFL-CONF-192376.
28. N. Leonard, S. Waghmare, and Y. Wang, "RNN: Recurrent library for torch," arXiv:1511.07889, 2015.

29. K. Papineni, S. Roukos, T. Ward, and W. Zhu, "BLEU: A method for automatic evaluation of machine translation," in *Proc. 40th Annu. Meeting Assoc. Comput. Linguistics*, 2002, pp. 311–318.
30. O. Levy, S. Remus, C. Biemann, and I. Dagan, "Do supervised distributional methods really learn lexical inference relations?" in *Proc. Hum. Lang. Technol.: The 2015 Annu. Conf. North Amer. Ch. ACL*, Denver, CO, USA, May 31 – Jun. 5, 2015, pp. 970–976.
31. V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proc. 27th Int. Conf. Mach. Learn.*, 2010, pp. 807–814.
32. X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2011, pp. 315–323.

Received September 8, 2016; accepted for publication October 4, 2016

Yossi Adi *IBM Research, Haifa 3498825, Israel (yossiad@il.ibm.com).* Mr. Adi is a machine learning researcher in the Machine Learning Technologies group, and a Ph.D. student at Bar-Ilan University, in the Machine Learning for Speech and Language Processing Laboratory. He received an M.Sc. degree from Bar-Ilan University in 2015. His research focuses on developing and analyzing machine learning algorithms for structured problems in speech and language domains.

Einat Kermany *IBM Research, Haifa 3498825, Israel (einatke@il.ibm.com).* Dr. Kermany is a research staff member in the Machine Learning Technologies group. She completed a Ph.D. degree in neuroscience at the Technion Faculty of Medicine under the supervision of Prof. Shimon Maron, and holds a B.A. degree in computer science, also from the Technion. She joined IBM in 2010 and participated in several projects that applied machine learning techniques for various problems. Her recent focus is in the field of deep learning to address natural language problems.

Yonatan Belinkov *MIT CSAIL (Computer Science and Artificial Intelligence Laboratory), Cambridge, MA 02139 USA (belinkov@mit.edu).* Mr. Belinkov is a PhD student at the MIT Computer Science and Artificial Intelligence Laboratory. He received an S.M. degree from MIT in 2014 and before that completed a B.Sc. degree in mathematics and an M.A. degree in Arabic Studies, both from Tel Aviv University. He works on natural language processing and understanding in the Spoken Language Systems group at MIT CSAIL. His recent research centers on vector representations for linguistic units with applications to question answering and machine translation.

Ofer Lavi *IBM Research, Haifa Research Labs, Haifa 3498825, Israel (oferl@il.ibm.com).* Mr. Lavi joined IBM in 2010 and has managed the Machine Learning Technologies group at IBM Research - Haifa since 2014. He holds a B.Sc. degree in computer science from the Hebrew University in Jerusalem and an M.Sc. degree in computer science with a focus on bioinformatics from Tel-Aviv University. Before joining IBM he led the technological side of a natural language understanding (NLU) start-up. Today, he works on research in the field of deep learning for NLU and on development of cognitive systems that learn from humans and interactively work together with humans to solve their everyday professional challenges.

Yoav Goldberg *Bar Ilan University, Ramat Gan 5290002, Israel (yoav.goldberg@gmail.com).* Dr. Goldberg is a senior lecturer in thet Bar Ilan University Computer Science Department. Dr. Goldberg conducted postdoctoral research as a Research Scientist at Google Research, New York, working primarily on syntactic parsing and its applications. Prior to that, he completed his Ph.D. program at Ben Gurion University, where he worked with Prof. Michael Elhadad on automatic processing of modern Hebrew, a specimen of a morphologically rich language.

To Reverse the Gradient or Not: an Empirical Comparison of Adversarial and Multi-task Learning in Speech Recognition

TO REVERSE THE GRADIENT OR NOT: AN EMPIRICAL COMPARISON OF ADVERSARIAL AND MULTI-TASK LEARNING IN SPEECH RECOGNITION

*Yossi Adi[1,2], *Neil Zeghidour[2,3] , Ronan Collobert[2], Nicolas Usunier[2], Vitaliy Liptchinsky[2], Gabriel Synnaeve[2]

[1]Bar-Ilan University.
[2]Facebook AI Research.
[3]CoML, ENS/INRIA.

ABSTRACT

Transcribed datasets typically contain speaker identity for each instance in the data. We investigate two ways to incorporate this information during training: *Multi-Task Learning* and *Adversarial Learning*. In multi-task learning, the goal is speaker prediction; we expect a performance improvement with this joint training if the two tasks of speech recognition and speaker recognition share a common set of underlying features. In contrast, adversarial learning is a means to learn representations invariant to the speaker. We then expect better performance if this learnt invariance helps generalizing to new speakers. While the two approaches seem natural in the context of speech recognition, they are incompatible because they correspond to opposite gradients back-propagated to the model. In order to better understand the effect of these approaches in terms of error rates, we compare both strategies in controlled settings. Moreover, we explore the use of additional un-transcribed data in a semi-supervised, adversarial learning manner to improve error rates. Our results show that deep models trained on big datasets already develop invariant representations to speakers without any auxiliary loss. When considering adversarial learning and multi-task learning, the impact on the acoustic model seems minor. However, models trained in a semi-supervised manner can improve error-rates.

Index Terms: automatic speech recognition, adversarial learning, multi-task learning, neural networks

1. INTRODUCTION

The minimal components of a speech recognition dataset are audio recordings and their corresponding transcription. They also typically contain an additional information, which is the anonymous identifier of the speaker corresponding to each utterance. As speaker variations are one of the most challenging aspects of speech processing, this information can be leveraged to improve the performance and robustness of speech recognition systems.

Traditionally, the identity of the speakers has been used to extract speaker representations and use it as an additional input to the acoustic model [1, 2]. This approach requires an additional feature extraction process of the input signal. Recently, two approaches have been proposed to leverage speaker information as another supervision to the acoustic model; Multi-Task Learning (MT) [3, 4] and Adversarial Learning (AL) [5]. In the context of speech recognition, a way of performing MT or AL is to add a speaker classification branch in parallel of the main branch trained for transcription. All layers below the fork of the two branches receive gradients from both transcription loss and speaker loss.

In MT our goal is to jointly transcribe the speech signal together with classifying the speaker identity. Previous work has explored using auxiliary tasks such as gender or context [6], as well as speaker classification [7, 8]. For example, the authors in [7] propose to classify the speaker identity in addition to estimating the phoneme-state posterior probabilities used for ASR, they presented a non-negligible improvement in terms of *Phoneme Error Rate* following the above approach.

An opposite approach is to assume that good acoustic representations should be invariant to any task that is not the speech recognition task, in particular the speaker characteristics. A method to learn such invariances is AL: the branch of the speaker classification task is trained to reduce its classification error, however the main branch is trained to maximize the loss of this speaker classifier. By doing so, it learns invariance to speaker characteristics. This approach has been previously used for speech recognition to learn invariance to noise conditions [9], speaker identity [10, 11, 12] and accent [13]. The authors in [11], proposed to minimize the senone classification loss, and simultaneously maximize the speaker classification loss. This approach achieves a significant improvement in terms of *Word Error Rate*.

Our study arises from two observations. First, both the MT and AL approaches described above have been used with the same purpose despite being fundamentally opposed, and this raises the question of which one is the most appropriate choice to improve the performance of speech recognition

Work conducted while Yossi Adi was an Intern at Facebook AI Research.
*Equal contribution

systems. Secondly, they only differ by a simple computational step which consists in reversing the gradient at the fork between the speaker branch and the main branch. This allows for controlled experiments where both approaches can be compared with equivalent architectures and number of parameters.

In this work we focus on letter based acoustic models. In this setting, we are given audio recordings and their transcripts, and train a neural network to output letters with a sequence-based criterion. During training, we also have access to the identity of the speaker of each utterance. We perform a systematic comparison between MT and AL for the task of large vocabulary speech recognition on the Wall Street Journal dataset (WSJ) [14]. Knowing whether the speaker information should be used to impact low level or high level representations is also a valuable information, so for each approach we experiment with three levels to fork the speaker branch to: lower layer, middle and upper layer.

Our Contribution: (i) We observed that deep models trained on big datasets, already develop invariant representations to speakers with neither AL nor MT; (ii) Both AL and MT do not have a clear impact on the acoustic model with respect to error rates; and (iii) Using additional, un-transcribed speaker labeled data, (i.e. the speaker is known, but without the transcription), seems promising. It improves Letter-Error Rates (LER), even though these improvements did not transfer into Word-Error Rates (WER) improvements.

2. ADVERSARIAL VS. MULTITASK

Given a training set of n transcribed acoustic utterances coupled with their speaker identity, our goal is to utilize these speaker labels to improve transcription loss.

Formally, let $\mathcal{S} = (\boldsymbol{X}_i, \boldsymbol{Y}_i, s_i)_{i=0}^{n}$ be the training set, in which each example is composed of three elements: a sequence of m acoustic features $\boldsymbol{X}_i = \{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_m\}$ where each x_j is a d-dimensional vector, a sequence of k characters $\boldsymbol{Y}_i = \{\mathbf{y}_1, \cdots, \mathbf{y}_k\}$ which are not aligned with $\boldsymbol{X}_i$, and a speaker label s_i. In the following subsections we present two common approaches to add the speaker identity as another supervision to the network.

2.1. Adversarial Learning

As speech recognition tasks should not depend on a specific set of speakers, in AL, we would like to learn an acoustic representation which is *speaker-invariant* and at the same time *characters-discriminative*. For that purpose, we consider the first k layers of the network as an encoder, denoted by E_r with parameters θ_r, which maps the acoustic features $\boldsymbol{X}_i$ to $\boldsymbol{R}_i = \{\boldsymbol{r}_1, \cdots, \boldsymbol{r}_l\}$ where l can be smaller than m. Then, the representations $\boldsymbol{R}_i$ are fed into a decoder network D_y with parameters θ_y, to output the characters posteriors $p(\boldsymbol{Y}_i|\boldsymbol{X}_i; \theta_r, \theta_y)$.

Furthermore, we introduce another decoder network to classify the speaker, denote D_s. The above decoder, maps $\boldsymbol{R}_i$ to the speaker posteriors $p(s_i|\boldsymbol{X}_i; \theta_r, \theta_y)$.

In order to make the representation $\boldsymbol{R}$ *speaker-invariant*, we train E_r and D_s jointly using adversarial loss, where we optimize θ_r to *maximize* speaker classification loss, and at the same time optimizing θ_s to *minimize* the speaker classification loss. Recall, we would like to make the representation $\boldsymbol{R}$ *characters-discriminative*, hence, we further optimize θ_y and θ_r to minimize the transcription loss. Therefore, the total loss is constructed as follows,

$$\mathcal{L}(\theta_r, \theta_y, \theta_s) = \mathcal{L}_{acoustic}(\theta_r, \theta_y) - \lambda \mathcal{L}_{spk}(\theta_r, \theta_s) \qquad (1)$$

where $\mathcal{L}_{acoustic}(\theta_r, \theta_y)$ is the transcription loss, $\mathcal{L}_{spk}(\theta_r, \theta_s)$ is the speaker loss, and λ is a trade-off parameter which controls the balance between the two. This minimax game will enhance the discriminative properties of D_s and D_y while at the same time push E_r towards generating speaker-invariant representation.

As suggested in [5, 11], we optimize the parameters using Stochastic Gradient Descent where we reverse the gradients of $\partial \mathcal{L}_{spk} / \partial \theta_r$ [5].

2.2. Multi-Task Learning

Similarly to AL, in MT we consider an encoder E_r with parameters θ_r and two decoders D_y and D_s with parameters θ_y and θ_s respectively. However, in MT, our goal is to minimize both $\mathcal{L}_{acoustic}(\theta_r, \theta_y)$ and $\mathcal{L}_{spk}(\theta_r, \theta_s)$. Therefore, the objective in MT case can be expressed as follows,

$$\mathcal{L}(\theta_r, \theta_y, \theta_s) = \mathcal{L}_{acoustic}(\theta_r, \theta_y) + \lambda \cdot \mathcal{L}_{spk}(\theta_r, \theta_s) \qquad (2)$$

Practically, on forward propagation, both AL and MT act the same way. Yet, on back-propagation they act exactly the opposite. In AL, we reverse the gradients from D_s before back-propagating into E_r. In contrast, in MT we sum together the gradients from D_s and D_y. Figure 1 depicts an illustration of the described model.

3. MODELS

Our acoustic models are based on Gated Convolutional Neural Networks (Gated ConvNets) [15], fed with 40 log-mel filterbank energies extracted every 10 ms with a 25 ms sliding window. Gated ConvNets stack 1D convolutions with Gated Linear Units (GLUs). 1D ConvNets were introduced early in the speech community, and are also referred as Time-Delay Neural Networks [16].

We set $Ł_{acoustic}(\theta_r, \theta_y)$ to be the AutoSeg Criterion (ASG) criterion [17, 18], which is similar to the Connectionist Temporal Classification (CTC) criterion [19].

Although speech utterances contain time dependent transcriptions, the speaker labels should be applied to the whole

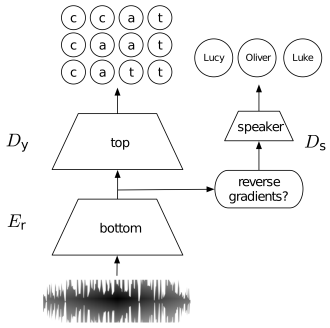


Fig. 1. An illustration of our architecture. We first feed the network a sequence of acoustic features, then we take the output of some intermediate representation an use it also to classify the speaker.

sequence. Hence, when optimizing for multi-task or adversarial learning we add a speaker branch which aggregates over all the time frames to classify the speaker. This pooling operation aggregates the time-dependent representations into a single sequence-level representation $\boldsymbol{s} = g(\cdot)$. Two straightforward aggregation methods are to take the sum over all frames: $g(\cdot) = \sum_t \boldsymbol{r}_t$ where $\boldsymbol{r}_t$ is the representation at time frame t, or the max $g(\cdot) = \max_t(\boldsymbol{r}_t)$, where the max is taken over each dimension. We propose to use a trade-off solution between these two methods, which is the LogSumExp [20], $\boldsymbol{s} = \frac{1}{\tau}\log\left(\frac{1}{T}\sum_t e^{\tau\cdot\boldsymbol{r}_t}\right)$, where τ is a hyper-parameter controls the trade-off between the sum and the max; with high values of τ the LogSumExp is similar to the max, while it tends to a sum as $\tau \to 0$. In practice, we used $\tau = 1$.

We set $L_{speaker}(\theta_r, \theta_s)$ to be the Negative Log Likelihood (NLL) loss function. The overall network, together with D_y, is trained by back-propagation. At inference time, D_s, the speaker classification branch is discarded.

4. EXPERIMENTAL RESULTS

In the following section we describe our experimental results. First, we provide all the technical details and setups in Subsection 4.1. Then, we describe our baselines and analyze their ability to encode information about the speaker in Subsection 4.2. Lastly, on Subsection 4.3 we present our results on the WSJ dataset [14].

4.1. Setups

All models were composed of 17 gated convolutional layers followed by a weight normalization [21]. We used a dropout layer after every GLU layer with dropout rate of 0.25. We optimized the models using SGD with two different learning rate values, we used learning rate of 1.4 for E_r and D_y and learning rate of 0.1 for D_s. We did not use momentum or weight decay. All the models were trained using WSJ dataset which contains 283 speakers with 81.5 hours of speaking time.

We computed WER using our own one-pass decoder, which performs a simple beam-search with beam thresholding, histogram pruning and language model smearing [22]. We did not implement any sort of model adaptation before decoding, nor any word graph rescoring. Our decoder relies on KenLM [23] for the language modeling part, using a 4-gram LM trained on the standard data of WSJ [14]. For LER, we used Viterbi decoding as in [17].

The speaker branch, D_s, was composed of one gated convolutional layer of width 5 and 200 feature maps with weight normalization, followed by a linear layer. For MT models we used a static λ value of 0.5. For AL models, we slowly increased the λ from 0 to 0.2 using the following update scheme, $\lambda_i = 2/(1 + e^{-p_i}) - 1$, where p_i is a scaled version of the epoch number. The above technique was successfully explored in previous studies [24, 5]. In practice, we observed that reversing the gradients from the beginning of training can cause the network to diverge. To avoid that, we first trained the models without back-propagating the gradients from D_s to E_r, (i.e. $\lambda = 0$). Then, we trained only D_s, and finally, we trained both of them jointly. We found the above procedure crucial for AL, however in MT the performance is equivalent to standard training. Thus, for comparison, we followed the above approach in both settings.

4.2. Representation Analysis

We would like to assess *to what extent the network encodes the speaker identity?* The motivation for answering this question is two-fold (i) to analyze the networks' behavior with respect to the speaker information; and (ii) having a measurement of how much speaker information is encoded in the representation; this can provide an intuitive insight into how much error rates can be improved using speaker-adversarial learning.

To tackle this question, we follow a similar approach to the one proposed in [25, 26]. We first trained a baseline network without the speaker classification branch, then we used it to extract representations of the speech signal from different layers in the network. Finally, we use these representations only, to train a model to classify the speaker identity. After training, we measure the model's accuracy to analyze the presence of speaker identity in the representations.

The basic premise of this approach is that if we cannot optimize a classifier to predict speaker identity based on representation from the model, then this property is not encoded in the representation, or rather, not encoded in an effective way, considered the way it will be used. Notice that we are not interested in improving speaker classification but rather to have a comparison between the models based on the clas-

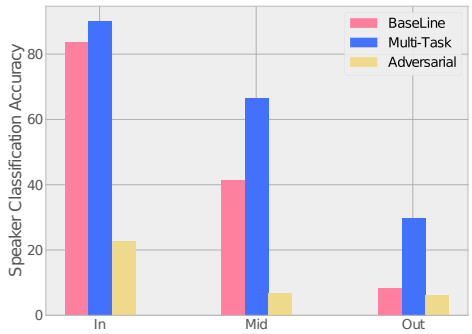


Fig. 2. Speaker classification accuracy using the representation from different layers of the network. Results are reported for Baseline, MT and AL models using three different settings, after the second, eighth, and fifteenth layers.

sifier's classification accuracy.

For that purpose, we defined three settings, where we add the speaker branch after: (i) two convolutional layers, denote as IN , (ii) eight convolutional layers, denote as MID , and (iii) 15 convolutional layers, denote as OUT . For each setting we trained a classification model for 10 epochs, using the same architecture as the speaker classification branch.

Figure 2 presents the speaker classification accuracy for the baseline together with MT and AL using representations from IN , MID and OUT . Notice, we observed a similar behavior in AL, MT, and the baseline, where the classification gets harder when extracting representations from deeper layers in the network. This implies that the network already develops speaker invariant representations during training. Even though, adding additional speaker loss can push it further and improve speaker invariance.

4.3. WSJ Experiments

Following our experiments in subsection 4.2, we trained MT and AL in three versions, IN , MID and OUT . Table 1 summarizes the results of our models, the baseline, and state-of-the-art systems.

Using both AL and MT seems to have a minor effect on transcription modeling when training on WSJ dataset. Both approaches achieve equal or somewhat worse LER than the baseline on the development set, and improve over the test set, with AL performs slightly better than MT. When considering WER, both AL and MT improve results over the baseline on the test set, however none of them improve over the development set. These results are somewhat counter-intuitive since both approaches attain similar results, however can be considered as opposite paradigms. One explanation is that both methods act as another regularization term. In other words, although the two approaches push the gradients towards opposite directions, when treated carefully, they both impose another constraint on the models' parameters [29, 30, 31].

Table 1. Letter Error Rates and Word Error Rates on the WSJ dataset.

Model Type	Layer	Nov93 dev LER	Nov93 dev WER	Nov92 eval LER	Nov92 eval WER
Human [27]		-	8.1	-	5.0
BLSTMs [28]		-	6.6	-	**3.5**
convnet BLSTM w/ additional data [27]		-	4.4	-	3.6
Our Baseline		7.2	**9.7**	4.9	5.9
Mult.	IN	7.2	9.9	4.8	5.8
	MID	7.3	9.8	4.9	5.9
	OUT	7.3	10.1	4.8	5.7
Adv.	IN	7.2	9.9	4.7	5.7
	MID	7.3	10.0	4.8	5.8
	OUT	7.2	9.8	4.7	**5.6**
Semi.	IN	**6.8**	9.8	**4.6**	6.2
	MID	6.8	**9.7**	4.7	6.0
	OUT	6.9	10.0	4.6	6.3

4.4. Adversarial Semi-Supervised Learning

Lastly, we wanted to investigate whether the observed improvements were limited by the relatively small amount of speakers (283) in the train set of WSJ. For that purpose, we ran a semi-supervised experiment were we control the amount of transcribed data by using only speaker labeled examples. We add $\approx$ 35 hours of speech using another 132 speakers, from the Librispeech corpus [32], and train the network in an adversarial fashion. For the new training examples we do not use the transcriptions, but only the speaker identity, hence optimizing the speaker loss only. Results are summarized on Table 1. Using additional speaker labeled data, seems beneficial in terms of LER: it improves over the baseline and previous AL results in all settings.

On the other hand, when considering WER, none of the settings improve over the baseline. This misalignment between the LER and WER results can be either because we constrained the model's outputs by LM at decoding time, hence WER results are greatly affected by the type of LM used, or because the LER improvements are on meaningless parts in the sequence. The analysis of this observations as well as jointly optimizing acoustic model and LM are left for future research.

5. CONCLUSION AND FUTURE WORK

In this paper, we studied AL and MT in the context of speech recognition. We showed that deep models already learn speaker-invariant representations, but can still benefit from semi-supervised speaker adversarial learning.

For future work, we would like to examine the effect of semi-supervised learning for low-resource languages.

6. REFERENCES

[1] A. Senior and I. Lopez-Moreno, "Improving dnn speaker independence with i-vector inputs," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. IEEE, 2014, pp. 225–229.

[2] V. Peddinti, G. Chen, D. Povey, and S. Khudanpur, "Reverberation robust acoustic modeling using i-vectors with time delay neural networks," in *INTERSPEECH*, 2015.

[3] R. Caruana, "Multitask learning," in *Learning to learn*, pp. 95–133. Springer, 1998.

[4] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *Journal of Machine Learning Research*, vol. 12, pp. 2493–2537, 2011.

[5] Y. Ganin et al., "Domain-adversarial training of neural networks," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 2096–2030, 2016.

[6] G. Pironkov, S. Dupont, and T. Dutoit, "Multi-task learning for speech recognition: an overview," in *Proc. of ESANN*, 2016.

[7] G. Pironkov, S. Dupont, and T. Dutoit, "Speaker-aware long short-term memory multi-task learning for speech recognition," in *Proc. of EUSIPCO*, 2016.

[8] Z. Tang, L. Li, and D. Wang, "Multi-task recurrent model for speech and speaker recognition," in *Proc. of APSIPA*, 2016.

[9] Dmitriy S. et al., "Invariant representations for noisy speech recognition," *CoRR*, vol. abs/1612.01928, 2016.

[10] Taira Tsuchiya, Naohiro Tawara, Testuji Ogawa, and Tetsunori Kobayashi, "Speaker invariant feature extraction for zero-resource languages with adversarial learning," in *Proc. of ICASSP*, 2018.

[11] Z. Meng, J. Li, Z. Chen, Y. Zhao, V. Mazalov, Y. Gong, et al., "Speaker-invariant training via adversarial learning," *arXiv:1804.00732*, 2018.

[12] G. Saon et al., "English conversational telephone speech recognition by humans and machines," *arXiv:1703.02136*, 2017.

[13] Sining Sun, Ching-Feng Yeh, Mei-Yuh Hwang, Mari Ostendorf, and Lei Xie, "Domain adversarial training for accented speech recognition," *Proc. of ICASSP*, 2018.

[14] D. B Paul and J. M Baker, "The design for the wall street journal-based csr corpus," in *Proc. of the workshop on Speech and Natural Language*, 1992.

[15] Y. N Dauphin, Aa Fan, M. Auli, and D. Grangier, "Language modeling with gated convolutional networks," *arXiv:1612.08083*, 2016.

[16] V. Peddinti, D. Povey, and S. Khudanpur, "A time delay neural network architecture for efficient modeling of long temporal contexts," in *INTERSPEECH*, 2015.

[17] R. Collobert, C. Puhrsch, and G. Synnaeve, "Wav2letter: an end-to-end convnet-based speech recognition system," *arXiv:1609.03193*, 2016.

[18] V. Liptchinsky, G. Synnaeve, and R. Collobert, "Letter-based speech recognition with gated convnets," *arXiv:1712.09444*, 2017.

[19] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *Proc. of ICML*, 2006.

[20] D. Palaz, G. Synnaeve, and R. Collobert, "Jointly learning to locate and classify words using convolutional networks.," in *Proc. of INTERSPEECH*, 2016.

[21] T. Salimans and D. Kingma, "Weight normalization: A simple reparameterization to accelerate training of deep neural networks," in *Proc. of NIPS*, 2016.

[22] V. Steinbiss, B. Tran, and H. Ney, "Improvements in beam search," in *Proc. of ICSLP*, 1994.

[23] K. Heafield, I. Pouzyrevsky, J. H Clark, and P. Koehn, "Scalable modified kneser-ney language model estimation," in *Proc. of ACL*, 2013.

[24] G. Lample, N. Zeghidour, N. Usunier, A. Bordes, L. Denoyer, et al., "Fader networks: Manipulating images by sliding attributes," in *Proc. of NIPS*, 2017.

[25] Y. Adi, E. Kermany, Y. Belinkov, O. Lavi, and Y. Goldberg, "Fine-grained analysis of sentence embeddings using auxiliary prediction tasks," *Proc. of ICLR*, 2016.

[26] Y Adi, E Kermany, Y Belinkov, O Lavi, and Y Goldberg, "Analysis of sentence embedding models using prediction tasks in natural language processing," *IBM Journal of Research and Development*, vol. 61, no. 4, pp. 3–1, 2017.

[27] Dario Amodei et al., "Deep speech 2 : End-to-end speech recognition in english and mandarin," in *Proc. of ICML*, 2016.

[28] W. Chan and I. Lane, "Deep recurrent neural networks for acoustic modelling," *arXiv:1504.01482*, 2015.

[29] S. Ruder, "An overview of multi-task learning in deep neural networks," *arXiv:1706.05098*, 2017.

[30] M. Nski, L. Simon, and F. Jurie, "An adversarial regularisation for semi-supervised training of structured output neural networks," in *Proc. of NIPS*, 2017.

[31] D. Cohen, B. Mitra, K. Hofmann, and W B. Croft, "Cross domain regularization for neural ranking models using adversarial learning," *arXiv preprint arXiv:1805.03403*, 2018.

[32] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: an asr corpus based on public domain audio books," in *Proc. of ICASSP*, 2015.

Structed: Risk Minimization in Structured Prediction

Journal of Machine Learning Research 17 (2016) 1-5 Submitted 10/15; Revised 3/16; Published 5/16

StructED : Risk Minimization in Structured Prediction

Yossi Adi ADIYOSS@CS.BIU.AC.IL
Joseph Keshet JOSEPH.KESHET@BIU.AC.IL
Department of Computer Science
Bar-Ilan University
Ramat Gan, 52900, Israel

Editor: Kevin Murphy

Abstract

Structured tasks are distinctive: each task has its own measure of performance, such as the word error rate in speech recognition, the BLEU score in machine translation, the NDCG score in information retrieval, or the intersection-over-union score in visual object segmentation. This paper presents StructED, a software package for learning structured prediction models with training methods that aimed at optimizing the task measure of performance. The package was written in Java and released under the MIT license. It can be downloaded from `http://adiyoss.github.io/StructED/`.

Keywords: structured prediction, structural SVM, CRF, direct loss minimization

1. Introduction

Ultimately the objective of discriminative training is to learn the model parameters so as to optimize a desired measure of performance. In binary classification, the objective is to find a prediction function that assigns a binary label to a single object, and minimizes the error rate (0-1 loss) on unseen data. In structured prediction, however, the goal is to predict a structured label (e.g., a graph, a sequence of words, a human pose, etc.), which often cannot be evaluated using the binary error rate, but rather with a task-specific measure of performance, generally called here *task loss*. There is a voluminous amount of work on training procedures for maximizing the BLEU score in machine translation, minimizing word/phone error rate in speech recognition or maximizing NDCG in information retrieval.

The most common approaches to learning parameters for structured prediction, namely structured perceptron, structural support vector machine (SSVM) and conditional random fields (CRF) do not directly minimize the task loss. The structured perceptron (Collins, 2002) solves a feasibility problem, which is independent of the task loss. The surrogate loss of SSVM (Joachims et al., 2005) is the structured hinge loss, a convex upper bound to the task loss, which is based on a generalization of the binary SVM hinge loss. However, while the binary SVM is strongly consistent, namely, converges to the error rate of the optimal linear predictor in the limit of infinite training data, the structured hinge loss is not consistent and does not converge to the expected task loss, i.e., the risk, of the optimal linear predictor even when the task loss is the 0-1 loss (McAllester, 2006). The objective of CRF is the log loss function, which is independent of the task loss (Lafferty et al., 2001).

©2016 Yossi Adi and Joseph Keshet.

Several structured prediction libraries already exist, such as CRF++ (Kudo, 2005), CRFsuite (Okazaki, 2007), Dlib (King, 2009), PyStruct (Müller and Behnke, 2014), and SVM-Struct (Joachims et al., 2009). All provide a general framework for training a model with either structured perceptron, CRF, or SSVM using several optimization techniques. Nevertheless, CRF++ and CRFsuite were not designed to support discriminative training with a task-specific evaluation metric, but rather use the binary error rate as their task loss function; whereas SVM-Struct, PyStruct, and Dlib support the training of structural SVM with a user-defined task loss function, but do not include, in their current phase, other training methods that are directly aimed at minimizing the task loss.

In this work we present StructED, a software package that is focused on training methods for structured prediction models that are aimed at minimizing a given task loss. The package provides several interfaces to define and implement a structured task, and allows the user to estimate the model parameters with several different training methods. The goal is not to provide several optimization alternatives to the structured hinge loss or the log loss as was already done in the previous packages, but rather to propose an implementation of a variety of surrogate loss functions that were designed to minimize the task loss and were theoretically motivated (McAllester and Keshet, 2011).

Note that most training methods require the task loss function to be decomposable in the size of the output label. Decomposable task loss functions are required in order to solve the loss-augmented inference that is used within the training procedure (Ranjbar et al., 2013), and evaluation metrics like intersection-over-union or word error rate which are not decomposable need to be approximated when utilized in SSVM, for example. Our package provides an implementation of methods (structured probit and orbit loss) that do not expect the task loss to be decomposable and can handle it directly without being approximated.

2. Structured Prediction and Risk Minimization

Consider a supervised learning setting with input instances $\boldsymbol{x} \in \mathcal{X}$ and target labels $\boldsymbol{y} \in \mathcal{Y}$, which refers to a set of objects with an internal structure. We assume a fixed mapping $\boldsymbol{\phi} : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^d$ from the set of input objects and target labels to a real vector of length d, where we call the elements of this mapping *feature functions*. Also, consider a linear decoder with parameters $\boldsymbol{w} \in \mathbb{R}^d$, such that $\hat{\boldsymbol{y}}_{\boldsymbol{w}}$ is a good approximation to the true label of $\boldsymbol{x}$, as follows:

$$\hat{\boldsymbol{y}}_{\boldsymbol{w}}(\boldsymbol{x}) = \operatorname*{argmax}_{\boldsymbol{y} \in \mathcal{Y}} \ \boldsymbol{w}^\top \boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{y}) \tag{1}$$

Ideally, the learning algorithm finds $\boldsymbol{w}$ such that the prediction rule optimizes the expected desired *measure of preference* or *evaluation metric* on unseen data. We define the task loss function, $\ell(\boldsymbol{y}, \hat{\boldsymbol{y}}_{\boldsymbol{w}})$, to be a non-negative measure of error when predicting $\hat{\boldsymbol{y}}_{\boldsymbol{w}}$ instead of $\boldsymbol{y}$ as the label of $\boldsymbol{x}$. Our goal is to find the parameter vector $\boldsymbol{w}$ that minimizes this function. When the desired evaluation metric is a utility function that needs to be maximized (like BLEU or NDCG), we define the task loss to be 1 minus the evaluation metric.

Our goal is to set $\boldsymbol{w}$ so as to minimize the expected task loss (i.e., risk) for predicting $\hat{\boldsymbol{y}}_{\boldsymbol{w}}$,

$$\boldsymbol{w}^* = \operatorname*{argmin}_{\boldsymbol{w}} \ \mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim\rho}[\ell(\boldsymbol{y}, \hat{\boldsymbol{y}}_{\boldsymbol{w}}(\boldsymbol{x}))], \tag{2}$$

where the expectation is taken over pairs $(\boldsymbol{x}, \boldsymbol{y})$ drawn from an unknown probability distribution ρ. This objective function is hard to minimize directly. A common practice is to replace the task loss with a surrogate loss function, denoted $\bar{\ell}(\boldsymbol{w}, \boldsymbol{x}, \boldsymbol{y})$, which is easier to minimize; and to replace the expectation with a regularized average with respect to a set of training examples $\mathcal{S} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^m$, where each pair $(\boldsymbol{x}_i, \boldsymbol{y}_i)$ is drawn i.i.d from ρ:

$$\boldsymbol{w}^* = \operatorname*{argmin}_{\boldsymbol{w}} \ \frac{1}{m} \sum_{i=1}^{m} \bar{\ell}(\boldsymbol{w}, \boldsymbol{x}_i, \boldsymbol{y}_i) + \frac{\lambda}{2} \|\boldsymbol{w}\|^2, \tag{3}$$

where λ is a trade-off parameter between the loss term and the regularization. Different training algorithms are defined by different surrogate loss functions, e.g., the surrogate loss of max-margin Markov model (Taskar et al., 2003) is the structured hinge loss with the Hamming distance, whereas the one for CRF is the log loss function. These loss functions are convex in the model parameters, $\boldsymbol{w}$.

Recently, several works proposed different surrogate loss functions which are closer to the task loss in some sense: structured ramp-loss (McAllester and Keshet, 2011), structured probit loss (Keshet et al., 2011), and orbit loss (Karmon and Keshet, 2015). Direct loss minimization (McAllester et al., 2010) is a perceptron-like method of performing direct gradient descent on the risk (2) in the case where $\mathcal{Y}$ is discrete but $\mathcal{X}$ is continuous. All of the training objectives that were proposed in these works are non-convex functions in the model parameters, are strongly consistent and perform well in general.

3. Implementation

Implementation of a structured prediction task involves defining a set of feature functions. Given a trained model, the prediction is performed by finding the output label that maximizes the weighted sum of those feature functions according to (1). Such maximization involves the enumeration over all possible output labels, which is often intractable and handled by dynamic programming or by approximated inference techniques. As a result, and in contrast to packages for binary classification, here the user has to write code that implements the feature functions, the decoder and the evaluation metric.

Currently the implemented training methods are structured perceptron (SP), SSVM, CRF, structured passive-aggressive (SPA; Crammer et al., 2006), direct loss minimization (DLM), structured ramp loss (SRL), structured probit loss (SPL), and orbit loss (Karmon and Keshet, 2015).

The package exposes interfaces for task loss functions, feature extractors, decoding algorithms and training algorithms. In order to train a new model the user has to consider four interfaces, which correspond to: (i) a set of feature functions, $\boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{y})$; (ii) a task loss function[1], $\ell(\boldsymbol{y}, \hat{\boldsymbol{y}})$; (iii) a decoder to perform the inference in (1), as well as the loss-augmented inference (needed by SPA, SRL, DLM, SSVM); and (iv) a training algorithm. The user can either implement those interfaces or use any of the already implemented interfaces.

The objective in (3) is optimized using stochastic gradient descent (SGD) for both the convex (SSVM and CRF) and non-convex (SRL, SPL, orbit) surrogate losses. DLM cannot be expressed as a surrogate loss function in objective (3), and is optimized using SGD as

1. BLUE and NDCG are currently not implemented.

well. SP and SPA are online algorithms and are implemented as batch algorithms with averaging of the weight vectors as an online-to-batch conversion (Cesa-Bianchi et al., 2004).

4. Experiments

While the library focuses on different training methods for minimizing a given task loss, it is important to verify that the implemented algorithms run in an acceptable amount of time. We compared our implementation of SSVM optimized by SGD to the corresponding implementation in PyStruct on the MNIST data set of handwritten digits. We got similar accuracy and training times (StructED : 92.6%, 149 sec; PyStruct: 90.2%, 145 sec).

We conclude the paper by demonstrating the advantage of the package with a simple structured prediction task of automatic vowel duration measurement. In this task the input is a speech segment of arbitrary duration that contains a vowel between two consonants (e.g., *bat*, *taught*, etc.), and the goal is to accurately predict the duration of the vowel. This task is a required measurement in linguistic studies and is often done manually. The training data includes speech segments that are labeled with the vowel onset and offset times, denoted y_b and y_e, respectively. The task loss is defined as $\ell(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \max\{0, |y_b - \hat{y}_b| - \tau_b\} + \max\{0, |y_e - \hat{y}_e| - \tau_e\}$, where $\hat{y}_b$ and $\hat{y}_e$ are predicted vowel onset and offset times, respectively, and τ_b and τ_e are parameters (τ_b=10 msec and τ_e=15 msec). We had a training set of 90 examples, a validation set of 20 examples and a test set of 20 examples. We extracted 21 unique acoustic features every 5 msec, including the confidence of a frame-based phoneme classifier, the first and the second formants and other acoustic features. We trained all algorithms on this task and present their performance in terms of task loss (the lower the better) and training times in Table 1 (training parameters for each of the training methods can be found in the package's Examples folder).

Algorithm	Task loss [msec]	Time [sec]
SP	72.5	13
SSVM	72.0	13
CRF	70.5	31
SPA	69.0	12
SRL	64.5	25
SPL	64.5	617
DLM	63.5	24
Orbit loss	58.5	13

Table 1: Error rate and training times on vowel duration measurement task.

Results suggest that training methods, which are designed to minimize the task loss, lead to improved performance compared to SP, SSVM, CRF or SPA. It is also evident that these methods, except orbit loss, take at least twice as much time to train: DLM and SRL need two inference operations per training iteration and SPL needs hundreds to thousands of inferences per training iteration.

The implementation of the above example is straightforward and is given in the Examples section of the package website `http://adiyoss.github.io/StructED/` along with several other usage examples and further details.

References

N. Cesa-Bianchi, A. Conconi, and C. Gentile. On the generalization ability of on-line learning algorithms. *IEEE Trans. on Information Theory*, 50(9):2050–2057, 2004.

M. Collins. Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *Proceeding of EMNLP*, 2002.

K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7:551–585, 2006.

I. Joachims, T. Tsochantaridis, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6:1453–1484, 2005.

T. Joachims, T. Finley, and Chun-Nam Yu. Cutting-plane training of structural SVMs. *Machine Learning*, 77(1):27–59, 2009.

D. Karmon and J. Keshet. Risk minimization in structured prediction using orbit loss. 2015. (under submission).

J. Keshet, D. McAllester, and T. Hazan. PAC-Bayesian approach for minimization of phoneme error rate. In *Proceeding of ICASSP*, 2011.

D. E. King. Dlib-ml: A machine learning toolkit. *Journal of Machine Learning Research*, 10:1755–1758, 2009.

T. Kudo. CRF++: Yet another CRF toolkit, 2005.

J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML*, pages 282–289, 2001.

D. McAllester. Generalization bounds and consistency for structured labeling. In B. Schölkopf, A. Smola, B. Taskar, and S.V.N. Vishwanathan, editors, *Predicting Structured Data*, pages 247–262. MIT Press, 2006.

D. McAllester and J. Keshet. Generalization bounds and consistency for latent structural probit and ramp loss. In *Proceedings of NIPS (25)*, 2011.

D. McAllester, T. Hazan, and J. Keshet. Direct loss minimization for structured prediction. In *Proceeding of NIPS (24)*, 2010.

A. C. Müller and S. Behnke. PyStruct - learning structured prediction in python. *Journal of Machine Learning Research*, 15:2055–2060, 2014.

N. Okazaki. CRFsuite: a fast implementation of conditional random fields (CRFs), 2007.

M. Ranjbar, T. Lan, Y. Wang, S.N. Robinovitch, Z.-N. Li, and G. Mori. Optimizing nondecomposable loss functions in structured prediction. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 35(4):911–924, 2013.

B. Taskar, C. Guestrin, and D. Koller. Max-margin Markov networks. In *Proceedings of NIPS (17)*, 2003.

Chapter 5

Published Papers (Security Applications)

Turning Your Weakness Into a Strength: Watermarking Deep Neural Networks by Backdooring

Turning Your Weakness Into a Strength: Watermarking Deep Neural Networks by Backdooring

Yossi Adi
Bar-Ilan University

Carsten Baum
Bar-Ilan University

Moustapha Cisse
Google, Inc. *

Benny Pinkas
Bar-Ilan University

Joseph Keshet
Bar-Ilan University

Abstract

Deep Neural Networks have recently gained lots of success after enabling several breakthroughs in notoriously challenging problems. Training these networks is computationally expensive and requires vast amounts of training data. Selling such pre-trained models can, therefore, be a lucrative business model. Unfortunately, once the models are sold they can be easily copied and redistributed. To avoid this, a tracking mechanism to identify models as the intellectual property of a particular vendor is necessary.

In this work, we present an approach for watermarking Deep Neural Networks in a black-box way. Our scheme works for general classification tasks and can easily be combined with current learning algorithms. We show experimentally that such a watermark has no noticeable impact on the primary task that the model is designed for and evaluate the robustness of our proposal against a multitude of practical attacks. Moreover, we provide a theoretical analysis, relating our approach to previous work on backdooring.

1 Introduction

Deep Neural Networks (DNN) enable a growing number of applications ranging from visual understanding to machine translation to speech recognition [20, 5, 17, 41, 6]. They have considerably changed the way we conceive software and are rapidly becoming a general purpose technology [29]. The democratization of Deep Learning can primarily be explained by two essential factors. First, several open source frameworks (e.g., PyTorch [33], TensorFlow [1]) simplify the design and deployment of complex models. Second, academic and industrial labs regularly release open source, state of the art, pre-trained models. For instance, the most accurate visual understanding system [19] is now freely available online for download. Given the considerable amount of expertise, data and computational resources required to train these models effectively, the availability of pre-trained models enables their use by operators with modest resources [38, 45, 35].

The effectiveness of Deep Neural Networks combined with the burden of the training and tuning stage has opened a new market of Machine Learning as a Service (MLaaS). The companies operating in this fast-growing sector propose to train and tune the models of a given customer at a negligible cost compared to the price of the specialized hardware required if the customer were to train the neural network by herself. Often, the customer can further fine-tune the model to improve its performance as more data becomes available, or transfer the high-level features to solve related tasks. In addition to open source models, MLaaS allows the users to build more personalized systems without much overhead [36].

Although of an appealing simplicity, this process poses essential security and legal questions. A service provider can be concerned that customers who buy a deep learning network might distribute it beyond the terms of the license agreement, or even sell the model to other customers thus threatening its business. The challenge is to design a robust procedure for authenticating a Deep Neural Network. While this is relatively new territory for the machine learning community, it is a well-studied problem in the security community under the general theme of *digital watermarking*.

Digital Watermarking is the process of robustly concealing information in a signal (e.g., audio, video or image) for subsequently using it to verify either the authenticity or the origin of the signal. Watermarking has been extensively investigated in the context of digital me-

*Work was conducted at Facebook AI Research.

dia (see, e.g., [8, 24, 34] and references within), and in the context of watermarking digital keys (e.g., in [32]). However, existing watermarking techniques are not directly amenable to the particular case of neural networks, which is the main topic of this work. Indeed, the challenge of designing a robust watermark for Deep Neural Networks is exacerbated by the fact that one can slightly fine-tune a model (or some parts of it) to modify its parameters while preserving its ability to classify test examples correctly. Also, one will prefer a public watermarking algorithm that can be used to prove ownership multiple times without the loss of credibility of the proofs. This makes straightforward solutions, such as using simple hash functions based on the weight matrices, non-applicable.

Contribution. Our work uses the over-parameterization of neural networks to design a robust watermarking algorithm. This over-parameterization has so far mainly been considered as a weakness (from a security perspective) because it makes backdooring possible [18, 16, 11, 27, 46]. Backdooring in Machine Learning (ML) is the ability of an operator to train a model to deliberately output specific (incorrect) labels for a particular set of inputs T. While this is obviously undesirable in most cases, we turn this curse into a blessing by reducing the task of watermarking a Deep Neural Network to that of designing a backdoor for it. Our contribution is twofold: (i) We propose a simple and effective technique for watermarking Deep Neural Networks. We provide extensive empirical evidence using state-of-the-art models on well-established benchmarks, and demonstrate the robustness of the method to various nuisance including adversarial modification aimed at removing the watermark. (ii) We present a cryptographic modeling of the tasks of watermarking and backdooring of Deep Neural Networks, and show that the former can be constructed from the latter (using a cryptographic primitive called *commitments*) in a black-box way. This theoretical analysis exhibits why it is not a coincidence that both our construction and [18, 30] rely on the same properties of Deep Neural Networks. Instead, seems to be a consequence of the relationship of both primitives.

Previous And Concurrent Work. Recently, [42, 10] proposed to watermark neural networks by adding a new regularization term to the loss function. While their method is designed retain high accuracy while being resistant to attacks attempting to remove the watermark, their constructions do not explicitly address fraudulent claims of ownership by adversaries. Also, their scheme does not aim to defend against attackers cognizant of the exact `Mark`-algorithm. Moreover, in the construction of [42, 10] the verification key can only be used once, because a watermark can be removed once the key is known[1]. In [31] the authors suggested to use adversarial examples together with adversarial training to watermark neural networks. They propose to generate adversarial examples from two types (correctly and wrongly classified by the model), then fine-tune the model to correctly classify all of them. Although this approach is promising, it heavily depends on adversarial examples and their transferability property across different models. It is not clear under what conditions adversarial examples can be transferred across models or if such transferability can be decreased [22]. It is also worth mentioning an earlier work on watermarking machine learning models proposed in [43]. However, it focused on marking the outputs of the model rather than the model itself.

2 Definitions and Models

This section provides a formal definition of backdooring for machine-learning algorithms. The definition makes the properties of existing backdooring techniques [18, 30] explicit, and also gives a (natural) extension when compared to previous work. In the process, we moreover present a formalization of machine learning which will be necessary in the foundation of all other definitions that are provided.

Throughout this work, we use the following notation: Let $n \in \mathbb{N}$ be a security parameter, which will be implicit input to all algorithms that we define. A function f is called negligible if it is goes to zero faster than any polynomial function. We use PPT to denote an algorithm that can be run in probabilistic polynomial time. For $k \in \mathbb{N}$ we use $[k]$ as shorthand for $\{1, \ldots, k\}$.

2.1 Machine Learning

Assume that there exists some objective ground-truth function f which classifies inputs according to a fixed output label set (where we allow the label to be undefined, denoted as $\perp$). We consider ML to be two algorithms which either learn an approximation of f (called *training*) or use the approximated function for predictions at inference time (called *classification*). The goal of *training* is to learn a function, f', that performs on unseen data as good as on the training set. A schematic description of this definition can be found in Figure 1.

[1] We present a technique to circumvent this problem in our setting. This approach can also be implemented in their work.

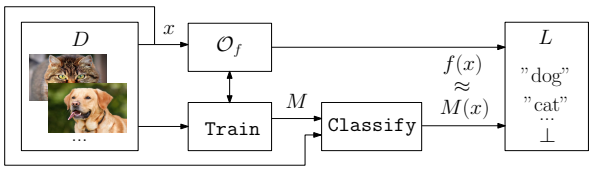


Figure 1: A high-level schematic illustration of the learning process.

To make this more formal, consider the sets $D \subset \{0,1\}^*, L \subset \{0,1\}^* \cup \{\perp\}$ where $|D| = \Theta(2^n)$ and $|L| = \Omega(p(n))$ for a positive polynomial $p(\cdot)$. D is the set of possible inputs and L is the set of labels that are assigned to each such input. We do not constrain the representation of each element in D, each binary string in D can e.g. encode float-point numbers for color values of pixels of an image of size $n \times n$ while[2] $L = \{0,1\}$ says whether there is a dog in the image or not. The additional symbol $\perp \in L$ is used if the classification task would be undefined for a certain input.

We assume an ideal assignment of labels to inputs, which is the *ground-truth function* $f : D \to L$. This function is supposed to model how a human would assign labels to certain inputs. As f might be undefined for specific tasks and labels, we will denote with $\overline{D} = \{x \in D \mid f(x) \neq \perp\}$ the set of all inputs having a ground-truth label assigned to them. To formally define learning, the algorithms are given access to f through an oracle $\mathcal{O}^f$. This oracle $\mathcal{O}^f$ truthfully answers calls to the function f.

We assume that there exist two algorithms $(\texttt{Train}, \texttt{Classify})$ for training and classification:

- $\texttt{Train}(\mathcal{O}^f)$ is a probabilistic polynomial-time algorithm that outputs a model $M \subset \{0,1\}^{p(n)}$ where $p(n)$ is a polynomial in n.
- $\texttt{Classify}(M,x)$ is a deterministic polynomial-time algorithm that, for an input $x \in D$ outputs a value $M(x) \in L \setminus \{\perp\}$.

We say that, given a function f, the algorithm pair $(\texttt{Train}, \texttt{Classify})$ is ε-accurate if $\Pr\left[f(x) \neq \texttt{Classify}(M,x) \mid x \in \overline{D}\right] \leq \varepsilon$ where the probability is taken over the randomness of $\texttt{Train}$. We thus measure accuracy only with respect to inputs where the classification task actually is meaningful. For those inputs where the ground-truth is undefined, we instead assume that the label is random: for all $x \in D \setminus \overline{D}$ we assume that for any $i \in L$, it holds that $\Pr[\texttt{Classify}(M,x) = i] = 1/|L|$ where the probability is taken over the randomness used in $\texttt{Train}$.

[2] Asymptotically, the number of bits per pixel is constant. Choosing this image size guarantees that $|D|$ is big enough. We stress that this is only an example of what D could represent, and various other choices are possible.

2.2 Backdoors in Neural Networks

Backdooring neural networks, as described in [18], is a technique to deliberately train a machine learning model to output *wrong* (when compared with the ground-truth function f) labels T_L for certain inputs T.

Therefore, let $T \subset D$ be a subset of the inputs, which we will refer to it as the *trigger set*. The wrong labeling with respect to the ground-truth f is captured by the function $T_L : T \to L \setminus \{\perp\}$; $x \mapsto T_L(x) \neq f(x)$ which assigns "wrong" labels to the trigger set. This function T_L, similar to the algorithm $\texttt{Classify}$, is not allowed to output the special label $\perp$. Together, the trigger set and the labeling function will be referred to as the *backdoor* $\mathsf{b} = (T, T_L)$. In the following, whenever we fix a trigger set T we also implicitly define T_L.

For such a backdoor b, we define a backdooring algorithm $\texttt{Backdoor}$ which, on input of a model, will output a model that misclassifies on the trigger set with high probability. More formally, $\texttt{Backdoor}(\mathcal{O}^f, \mathsf{b}, M)$ is PPT algorithm that receives as input an oracle to f, the backdoor b and a model M, and outputs a model $\hat{M}$. $\hat{M}$ is called *backdoored* if $\hat{M}$ is correct on $\overline{D} \setminus T$ but reliably errs on T, namely

$$\Pr_{x \in \overline{D} \setminus T}\left[f(x) \neq \texttt{Classify}(\hat{M},x)\right] \leq \varepsilon, \text{ but}$$
$$\Pr_{x \in T}\left[T_L(x) \neq \texttt{Classify}(\hat{M},x)\right] \leq \varepsilon.$$

This definition captures two ways in which a backdoor can be embedded:

- The algorithm can use the provided model to embed the watermark into it. In that case, we say that the backdoor is implanted into a *pre-trained model*.
- Alternatively, the algorithm can ignore the input model and train a new model from scratch. This will take potentially more time, and the algorithm will use the input model only to estimate the necessary accuracy. We will refer to this approach as *training from scratch*.

2.3 Strong Backdoors

Towards our goal of watermarking a ML model we require further properties from the backdooring algorithm, which deal with the sampling and removal of backdoors:

First of all, we want to turn the generation of a trapdoor into an algorithmic process. To this end, we introduce a new, randomized algorithm `SampleBackdoor` that on input $\mathcal{O}^f$ outputs backdoors b and works in combination with the aforementioned algorithms $(\texttt{Train}, \texttt{Classify})$. This is schematically shown in Figure 2.

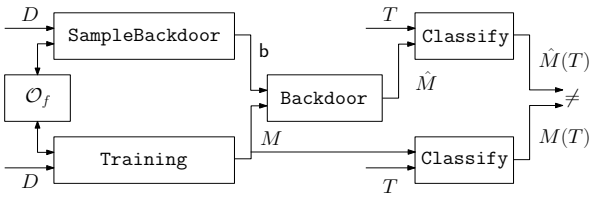


Figure 2: A schematic illustration of the backdooring process.

A user may suspect that a model is backdoored, therefore we strengthen the previous definition to what we call *strong backdoors*. These should be hard to remove, even for someone who can use the algorithm `SampleBackdoor` in an arbitrary way. Therefore, we require that `SampleBackdoor` should have the following properties:

Multiple Trigger Sets. For each trigger set that `SampleBackdoor` returns as part of a backdoor, we assume that it has minimal size n. Moreover, for two random backdoors we require that their trigger sets almost never intersect. Formally, we ask that $\Pr[T \cap T' \neq \emptyset]$ for $(T, T_L), (T', T'_L) \leftarrow \texttt{SampleBackdoor}()$ is negligible in n.

Persistency. With persistency we require that it is hard to remove a backdoor, unless one has knowledge of the trigger set T. There are two trivial cases which a definition must avoid:

- An adversary may submit a model that has no backdoor, but this model has very low accuracy. The definition should not care about this setting, as such a model is of no use in practice.
- An adversary can always train a new model from scratch, and therefore be able to submit a model that is very accurate and does not include the backdoor. An adversary with unlimited computational resources and unlimited access to $\mathcal{O}^f$ will thus always be able to cheat.

We define persistency as follows: let f be a ground-truth function, b be a backdoor and $\hat{M} \leftarrow \texttt{Backdoor}(\mathcal{O}^f, \text{b}, M)$ be a ε-accurate model. Assume an algorithm $\mathcal{A}$ on input $\mathcal{O}^f, \hat{M}$ outputs an ε-accurate model $\tilde{M}$ in time t which is at least $(1-\varepsilon)$ accurate on b. Then $\tilde{N} \leftarrow \mathcal{A}(\mathcal{O}^f, N)$, generated in the same time t, is also ε-accurate for any arbitrary model N.

In our approach, we chose to restrict the runtime of $\mathcal{A}$, but other modeling approaches are possible: one could also give unlimited power to $\mathcal{A}$ but only restricted access to the ground-truth function, or use a mixture of both. We chose our approach as it follows the standard pattern in cryptography, and thus allows to integrate better with cryptographic primitives which we will use: these are only secure against adversaries with a bounded runtime.

2.4 Commitments

Commitment schemes [9] are a well known cryptographic primitive which allows a sender to lock a secret x into a cryptographic leakage-free and tamper-proof vault and give it to someone else, called a receiver. It is neither possible for the receiver to open this vault without the help of the sender (this is called *hiding*), nor for the sender to exchange the locked secret to something else once it has been given away (the *binding* property).

Formally, a commitment scheme consists of two algorithms $(\texttt{Com}, \texttt{Open})$:

- $\texttt{Com}(x, r)$ on input of a value $x \in S$ and a bitstring $r \in \{0,1\}^n$ outputs a bitstring c_x.
- $\texttt{Open}(c_x, x, r)$ for a given $x \in S, r \in \{0,1\}^n, c_x \in \{0,1\}^*$ outputs 0 or 1.

For correctness, it must hold that $\forall x \in S$,

$$\Pr_{r \in \{0,1\}^n}[\texttt{Open}(c_x, x, r) = 1 \mid c_x \leftarrow \texttt{Com}(x, r)] = 1.$$

We call the commitment scheme $(\texttt{Com}, \texttt{Open})$ binding if, for every PPT algorithm $\mathcal{A}$

$$\Pr\left[\texttt{Open}(c_x, \tilde{x}, \tilde{r}) = 1 \;\middle|\; \begin{array}{c} c_x \leftarrow \texttt{Com}(x, r) \wedge \\ (\tilde{x}, \tilde{r}) \leftarrow \mathcal{A}(c_x, x, r) \wedge \\ (x, r) \neq (\tilde{x}, \tilde{r}) \end{array}\right] \leq \varepsilon(n)$$

where $\varepsilon(n)$ is negligible in n and the probability is taken over $x \in S, r \in \{0,1\}^n$.

Similarly, $(\texttt{Com}, \texttt{Open})$ are hiding if no PPT algorithm $\mathcal{A}$ can distinguish $c_0 \leftarrow \texttt{Com}(0, r)$ from $c_x \leftarrow \texttt{Com}(x, r)$ for arbitrary $x \in S, r \in \{0,1\}^n$. In case that the distributions of c_0, c_x are statistically close, we call a commitment scheme *statistically hiding*. For more information, see e.g. [14, 39].

3 Defining Watermarking

We now define watermarking for ML algorithms. The terminology and definitions are inspired by [7, 26].

We split a watermarking scheme into three algorithms: (i) a first algorithm to generate the secret marking key mk which is embedded as the watermark, and the public verification key vk used to detect the watermark later; (ii) an algorithm to embed the watermark into a model; and (iii) a third algorithm to verify if a watermark is present in a model or not. We will allow that the verification involves both mk and vk, for reasons that will become clear later.

Formally, a watermarking scheme is defined by the three PPT algorithms $(\texttt{KeyGen}, \texttt{Mark}, \texttt{Verify})$:

- $\texttt{KeyGen}()$ outputs a key pair $(\mathsf{mk}, \mathsf{vk})$.
- $\texttt{Mark}(M, \mathsf{mk})$ on input a model M and a marking key mk, outputs a model $\hat{M}$.
- $\texttt{Verify}(\mathsf{mk}, \mathsf{vk}, M)$ on input of the key pair mk, vk and a model M, outputs a bit $b \in \{0,1\}$.

For the sake of brevity, we define an auxiliary algorithm which simplifies to write definitions and proofs:

$\texttt{MModel}()$:

1. Generate $M \leftarrow \texttt{Train}(\mathcal{O}^f)$.
2. Sample $(\mathsf{mk}, \mathsf{vk}) \leftarrow \texttt{KeyGen}()$.
3. Compute $\hat{M} \leftarrow \texttt{Mark}(M, \mathsf{mk})$.
4. Output $(M, \hat{M}, \mathsf{mk}, \mathsf{vk})$.

The three algorithms $(\texttt{KeyGen}, \texttt{Mark}, \texttt{Verify})$ should correctly work together, meaning that a model watermarked with an honestly generated key should be verified as such. This is called *correctness*, and formally requires that

$$\Pr_{(M,\hat{M},\mathsf{mk},\mathsf{vk}) \leftarrow \texttt{MModel}()} \left[\texttt{Verify}(\mathsf{mk}, \mathsf{vk}, \hat{M}) = 1\right] = 1.$$

A depiction of this can be found in Figure 3.

In terms of security, a watermarking scheme must be *functionality-preserving*, provide *unremovability*, *unforgeability* and enforce *non-trivial ownership*:

- We say that a scheme is *functionality-preserving* if a model with a watermark is as accurate as a model without it: for any $(M, \hat{M}, \mathsf{mk}, \mathsf{vk}) \leftarrow \texttt{MModel}()$, it holds that

$$\Pr_{x \in D}[\texttt{Classify}(x, M) = f(x)] \approx \Pr_{x \in D}\left[\texttt{Classify}(x, \hat{M}) = f(x)\right].$$

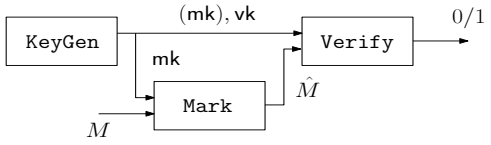


Figure 3: A schematic illustration of watermarking a neural network.

- *Non-trivial ownership* means that even an attacker which knows our watermarking algorithm is not able to generate in advance a key pair $(\mathsf{mk}, \mathsf{vk})$ that allows him to claim ownership of arbitrary models that are unknown to him. Formally, a watermark does not have trivial ownership if every PPT algorithm $\mathcal{A}$ only has negligible probability for winning the following game:
  1. Run $\mathcal{A}$ to compute $(\tilde{\mathsf{mk}}, \tilde{\mathsf{vk}}) \leftarrow \mathcal{A}()$.
  2. Compute $(M, \hat{M}, \mathsf{mk}, \mathsf{vk}) \leftarrow \texttt{MModel}()$.
  3. $\mathcal{A}$ wins if $\texttt{Verify}(\tilde{\mathsf{mk}}, \tilde{\mathsf{vk}}, \hat{M}) = 1$.
- *Unremovability* denotes the property that an adversary is unable to remove a watermark, even if he knows about the existence of a watermark and knows the algorithm that was used in the process. We require that for every PPT algorithm $\mathcal{A}$ the chance of winning the following game is negligible:
  1. Compute $(M, \hat{M}, \mathsf{mk}, \mathsf{vk}) \leftarrow \texttt{MModel}()$.
  2. Run $\mathcal{A}$ and compute $\tilde{M} \leftarrow \mathcal{A}(\mathcal{O}^f, \hat{M}, \mathsf{vk})$.
  3. $\mathcal{A}$ wins if

$$\Pr_{x \in D}[\texttt{Classify}(x, M) = f(x)] \approx \Pr_{x \in D}\left[\texttt{Classify}(x, \tilde{M}) = f(x)\right]$$

  and $\texttt{Verify}(\mathsf{mk}, \mathsf{vk}, \tilde{M}) = 0$.
- *Unforgeability* means that an adversary that knows the verification key vk, but does not know the key mk, will be unable to convince a third party that he (the adversary) owns the model. Namely, it is required that for every PPT algorithm $\mathcal{A}$, the chance of winning the following game is negligible:
  1. Compute $(M, \hat{M}, \mathsf{mk}, \mathsf{vk}) \leftarrow \texttt{MModel}()$.
  2. Run the adversary $(\tilde{M}, \tilde{\mathsf{mk}}) \leftarrow \mathcal{A}(\mathcal{O}^f, \hat{M}, \mathsf{vk})$.
  3. $\mathcal{A}$ wins if $\texttt{Verify}(\tilde{\mathsf{mk}}, \mathsf{vk}, \tilde{M}) = 1$.

Two other properties, which might be of practical interest but are either too complex to achieve or contrary to our definitions, are *Ownership Piracy* and different degrees of *Verifiability*,

- *Ownership Piracy* means that an attacker is attempting to implant his watermark into a model which has already been watermarked before. Here, the goal is that the old watermark at least persists. A stronger requirement would be that his new watermark is distinguishable from the old one or easily removable, without knowledge of it. Indeed, we will later show in Section 5.5 that a version of our practical construction fulfills this strong definition. On the other hand, a removable watermark is obviously in general inconsistent with *Unremovability*, so we leave[3] it out in our theoretical construction.
- A watermarking scheme that uses the verification procedure `Verify` is called *privately verifiable*. In such a setting, one can convince a third party about ownership using `Verify` as long as this third party is honest and does not release the key pair $(\mathsf{mk}, \mathsf{vk})$, which crucially is input to it. We call a scheme *publicly verifiable* if there exists an interactive protocol `PVerify` that, on input $\mathsf{mk}, \mathsf{vk}, M$ by the prover and vk, M by the verifier outputs the same value as `Verify` (except with negligible probability), such that the same key vk can be used in multiple proofs of ownership.

4 Watermarking From Backdooring

This section gives a theoretical construction of privately verifiable watermarking based on any strong backdooring (as outlined in Section 2) and a commitment scheme. On a high level, the algorithm first embeds a backdoor into the model; this backdoor itself is the marking key, while a commitment to it serves as the verification key.

More concretely, let $(\texttt{Train}, \texttt{Classify})$ be an ε-accurate ML algorithm, `Backdoor` be a strong backdooring algorithm and $(\texttt{Com}, \texttt{Open})$ be a statistically hiding commitment scheme. Then define the three algorithms $(\texttt{KeyGen}, \texttt{Mark}, \texttt{Verify})$ as follows.

$\texttt{KeyGen}()$:

1. Run $(T, T_L) = \mathsf{b} \leftarrow \texttt{SampleBackdoor}(\mathcal{O}^f)$ where $T = \{t^{(1)}, \ldots, t^{(n)}\}$ and $T_L = \{T_L^{(1)}, \ldots, T_L^{(n)}\}$.
2. Sample $2n$ random strings $r_t^{(i)}, r_L^{(i)} \leftarrow \{0,1\}^n$ and generate $2n$ commitments $\{c_t^{(i)}, c_L^{(i)}\}_{i \in [n]}$ where $c_t^{(i)} \leftarrow \texttt{Com}(t^{(i)}, r_t^{(i)})$, $c_L^{(i)} \leftarrow \texttt{Com}(T_L^{(i)}, r_L^{(i)})$.
3. Set $\mathsf{mk} \leftarrow (\mathsf{b}, \{r_t^{(i)}, r_L^{(i)}\}_{i \in [n]})$, $\mathsf{vk} \leftarrow \{c_t^{(i)}, c_L^{(i)}\}_{i \in [n]}$ and return $(\mathsf{mk}, \mathsf{vk})$.

$\texttt{Mark}(M, \mathsf{mk})$:

1. Let $\mathsf{mk} = (\mathsf{b}, \{r_t^{(i)}, r_L^{(i)}\}_{i \in [n]})$.
2. Compute and output $\hat{M} \leftarrow \texttt{Backdoor}(\mathcal{O}^f, \mathsf{b}, M)$.

$\texttt{Verify}(\mathsf{mk}, \mathsf{vk}, M)$:

1. Let $\mathsf{mk} = (\mathsf{b}, \{r_t^{(i)}, r_L^{(i)}\}_{i \in [n]})$, $\mathsf{vk} = \{c_t^{(i)}, c_L^{(i)}\}_{i \in [n]}$. For $\mathsf{b} = (T, T_L)$ test if $\forall t^{(i)} \in T : \; T_L^{(i)} \neq f(t^{(i)})$. If not, then output 0.
2. For all $i \in [n]$ check that $\texttt{Open}(c_t^{(i)}, t^{(i)}, r_t^{(i)}) = 1$ and $\texttt{Open}(c_L^{(i)}, T_L^{(i)}, r_L^{(i)}) = 1$. Otherwise output 0.
3. For all $i \in [n]$ test that $\texttt{Classify}(t^{(i)}, M) = T_L^{(i)}$. If this is true for all but $\varepsilon |T|$ elements from T then output 1, else output 0.

We want to remark that this construction captures both the watermarking of an existing model and the training from scratch. We now prove the security of the construction.

Theorem 1. *Let $\overline{D}$ be of super-polynomial size in n. Then assuming the existence of a commitment scheme and a strong backdooring scheme, the aforementioned algorithms $(\texttt{KeyGen}, \texttt{Mark}, \texttt{Verify})$ form a privately verifiable watermarking scheme.*

The proof, on a very high level, works as follows: a model containing a strong backdoor means that this backdoor, and therefore the watermark, cannot be removed. Additionally, by the hiding property of the commitment scheme the verification key will not provide any useful information to the adversary about the backdoor used, while the binding property ensures that one cannot claim ownership of arbitrary models. In the proof, special care must be taken as we use reductions from the watermarking algorithm to the security of both the underlying backdoor and the commitment scheme. To be meaningful, those reductions must have much smaller runtime than actually breaking these assumptions directly. While this is easy in the case of the commitment scheme, reductions to backdoor security need more attention.

Proof. We prove the following properties:

[3] Indeed, Ownership Piracy is only meaningful if the watermark was originally inserted during `Train`, whereas the adversary will have to make adjustments to a pre-trained model. This gap is exactly what we explore in Section 5.5.

Correctness. By construction, $\hat{M}$ which is returned by `Mark` will disagree with b on elements from T with probability at most ε, so in total at least $(1-\varepsilon)|T|$ elements agree by the definition of a backdoor. `Verify` outputs 1 if $\hat{M}$ disagrees with b on at most $\varepsilon|T|$ elements.

Functionality-preserving. Assume that `Backdoor` is a backdooring algorithm, then by its definition the model $\hat{M}$ is accurate outside of the trigger set of the backdoor, i.e.

$$\Pr_{x\in\overline{D}\setminus T}\left[f(x)\neq \texttt{Classify}(\hat{M},x)\right]\leq\varepsilon.$$

$\hat{M}$ in total will then err on a fraction at most $\varepsilon' = \varepsilon + n/|D|$, and because $\overline{D}$ by assumption is super-polynomially large in n ε' is negligibly close to ε.

Non-trivial ownership. To win, $\mathcal{A}$ must guess the correct labels for a $1-\varepsilon$ fraction of $\tilde{T}$ in advance, as $\mathcal{A}$ cannot change the chosen value $\tilde{T},\tilde{T}_L$ after seeing the model due to the binding property of the commitment scheme. As `KeyGen` chooses the set T in mk uniformly at random, whichever set $\mathcal{A}$ fixes for $\tilde{\mathsf{mk}}$ will intersect with T only with negligible probability by definition (due to the *multiple trigger sets* property). So assume for simplicity that $\tilde{T}$ does not intersect with T. Now $\mathcal{A}$ can choose $\tilde{T}$ to be of elements either from within $\overline{D}$ or outside of it. Let $n_1 = |\overline{D}\cap\tilde{T}|$ and $n_2 = |\tilde{T}| - n_1$.

For the benefit of the adversary, we make the strong assumption that whenever M is inaccurate for $x\in\overline{D}\cap\tilde{T}$ then it classifies to the label in $\tilde{T}_L$. But as M is ε-accurate on $\overline{D}$, the ratio of incorrectly classified committed labels is $(1-\varepsilon)n_1$. For every choice $\varepsilon < 0.5$ we have that $\varepsilon n_1 < (1-\varepsilon)n_1$. Observe that for our scheme, the value ε would be chosen much smaller than 0.5 and therefore this inequality always holds.

On the other hand, let's look at all values of $\tilde{T}$ that lie in $D\setminus\overline{D}$. By the assumption about machine learning that we made in its definition, if the input was chosen independently of M and it lies outside of $\overline{D}$ then M will in expectancy misclassify $\frac{|L|-1}{|L|}n_2$ elements. We then have that $\varepsilon n_2 < \frac{|L|-1}{|L|}n_2$ as $\varepsilon < 0.5$ and $L\geq 2$. As $\varepsilon n = \varepsilon n_1 + \varepsilon n_2$, the error of $\tilde{T}$ must be larger than εn.

Unremovability. Assume that there exists no algorithm that can generate an ε-accurate model N in time t of f, where t is a lot smaller that the time necessary for training such an accurate model using `Train`. At the same time, assume that the adversary $\mathcal{A}$ breaking the unremovability property takes time approximately t. By definition, after running $\mathcal{A}$ on input M,vk it will output a model $\tilde{M}$ which will be ε-accurate and at least a $(1-\varepsilon)$-fraction of the elements from the set T will be classified correctly. The goal in the proof is to show that $\mathcal{A}$ achieves this independently of vk. In a first step, we will use a hybrid argument to show that $\mathcal{A}$ essentially works independent of vk. Therefore, we construct a series of algorithms where we gradually replace the backdoor elements in vk. First, consider the following algorithm $\mathcal{S}$:

1. Compute $(M,\hat{M},\mathsf{mk},\mathsf{vk})\leftarrow\texttt{MModel}()$.
2. Sample $(\tilde{T},\tilde{T}_L) = \tilde{\mathsf{b}} \leftarrow \texttt{SampleBackdoor}(\mathcal{O}^f)$ where $\tilde{T}=\{\tilde{t}^{(1)},\ldots,\tilde{t}^{(n)}\}$ and $\tilde{T}_L=\{\tilde{T}_L^{(1)},\ldots,\tilde{T}_L^{(n)}\}$. Now set
$$c_t^{(1)}\leftarrow\texttt{Com}(\tilde{t}^{(1)},r_t^{(1)}), c_L^{(1)}\leftarrow\texttt{Com}(\tilde{T}_L^{(1)},r_L^{(1)})$$
and $\tilde{\mathsf{vk}}\leftarrow\{c_t^{(i)},c_L^{(i)}\}_{i\in[n]}$
3. Compute $\tilde{M}\leftarrow\mathcal{A}(\mathcal{O}^f,\hat{M},\tilde{\mathsf{vk}})$.

This algorithm replaces the first element in a verification key with an element from an independently generated backdoor, and then runs $\mathcal{A}$ on it.

In $\mathcal{S}$ we only exchange one commitment when compared to the input distribution to $\mathcal{A}$ from the security game. By the statistical hiding of `Com`, the output of $\mathcal{S}$ must be distributed statistically close to the output of $\mathcal{A}$ in the unremovability experiment. Applying this repeatedly, we construct a sequence of hybrids $\mathcal{S}^{(1)},\mathcal{S}^{(2)},\ldots,\mathcal{S}^{(n)}$ that change $1,2,\ldots,n$ of the elements from vk in the same way that $\mathcal{S}$ does and conclude that the success of outputting a model $\tilde{M}$ without the watermark using $\mathcal{A}$ must be independent of vk.

Consider the following algorithm $\mathcal{T}$ when given a model M with a strong backdoor:

1. Compute $(\mathsf{mk},\mathsf{vk})\leftarrow\texttt{KeyGen}()$.
2. Run the adversary and compute $\tilde{N}\leftarrow\mathcal{A}(\mathcal{O}^f,M,\mathsf{vk})$.

By the hybrid argument above, the algorithm $\mathcal{T}$ runs nearly in the same time as $\mathcal{A}$, namely t, and its output $\tilde{N}$ will be without the backdoor that M contained. But then, by persistence of strong backdooring, $\mathcal{T}$ must also generate ε-accurate models given arbitrary, in particular bad input models M in the same time t, which contradicts our assumption that no such algorithm exists.

Unforgeability. Assume that there exists a poly-time algorithm $\mathcal{A}$ that can break unforgeability. We will use this algorithm to open a statistically hiding commitment.

Therefore, we design an algorithm $\mathcal{S}$ which uses $\mathcal{A}$ as a subroutine. The algorithm trains a regular network (which can be watermarked by our scheme) and adds the commitment into the verification key. Then, it will use $\mathcal{A}$ to find openings for these commitments. The algorithm $\mathcal{S}$ works as follows:

1. Receive the commitment c from challenger.
2. Compute $(M, \hat{M}, \mathsf{mk}, \mathsf{vk}) \leftarrow \texttt{MModel}()$.
3. Let $\mathsf{vk} = \{c_t^{(i)}, c_L^{(i)}\}_{i\in[n]}$ set
$$\hat{c}_t^{(i)} \leftarrow \begin{cases} c & \text{if } i = 1 \\ c_t^{(i)} & \text{else} \end{cases}$$
and $\hat{\mathsf{vk}} \leftarrow \{\hat{c}_t^{(i)}, c_L^{(i)}\}_{i\in[n]}$.
4. Compute $(\tilde{M}, \tilde{\mathsf{mk}}) \leftarrow \mathcal{A}(\mathcal{O}^f, \hat{M}, \hat{\mathsf{vk}})$.
5. Let $\tilde{\mathsf{mk}} = ((\{t^{(1)}, \ldots, t^{(n)}\}, T_L), \{r_t^{(i)}, r_L^{(i)}\}_{i\in[n]})$.
If $\texttt{Verify}(\tilde{\mathsf{mk}}, \hat{\mathsf{vk}}, \tilde{M}) = 1$ output $t^{(1)}, r_t^{(1)}$, else output $\perp$.

Since the commitment scheme is statistically hiding, the input to $\mathcal{A}$ is statistically indistinguishable from an input where $\hat{M}$ is backdoored on all the committed values of vk. Therefore the output of $\mathcal{A}$ in $\mathcal{S}$ is statistically indistinguishable from the output in the unforgeability definition. With the same probability as in the definition, $\tilde{\mathsf{mk}}, \hat{\mathsf{vk}}, \tilde{M}$ will make `Verify` output 1. But by its definition, this means that $\texttt{Open}(c, t^{(1)}, r_t^{(1)}) = 1$ so $t^{(1)}, r_t^{(1)}$ open the challenge commitment c. As the commitment is statistically hiding (and we generate the backdoor independently of c) this will open c to another value then for which it was generated with overwhelming probability. □

4.1 From Private to Public Verifiability

Using the algorithm `Verify` constructed in this section only allows verification by an honest party. The scheme described above is therefore only privately verifiable. After running `Verify`, the key mk will be known and an adversary can retrain the model on the trigger set. This is not a drawback when it comes to an application like the protection of intellectual property, where a trusted third party in the form of a judge exists. If one instead wants to achieve public verifiability, then there are two possible scenarios for how to design an algorithm `PVerify`: allowing public verification a constant number of times, or an arbitrary number of times.

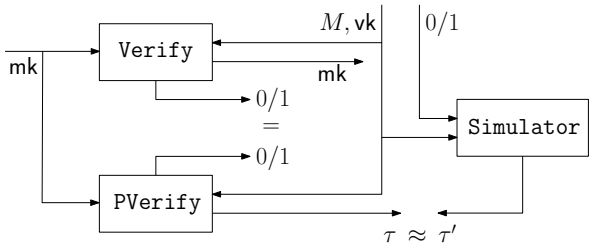


Figure 4: A schematic illustration of the public verification process.

In the first setting, a straightforward approach to the construction of `PVerify` is to choose multiple backdoors during `KeyGen` and release a different one in each iteration of `PVerify`. This allows multiple verifications, but the number is upper-bounded in practice by the capacity of the model M to contain backdoors - this cannot arbitrarily be extended without damaging the accuracy of the model. To achieve an unlimited number of verifications we will modify the watermarking scheme to output a different type of verification key. We then present an algorithm `PVerify` such that the interaction τ with an honest prover can be simulated as τ' given the values $M, \mathsf{vk}, \texttt{Verify}(\mathsf{mk}, \mathsf{vk}, M)$ only. This simulation means that no other information about mk beyond what is leaked from vk ever gets to the verifier. We give a graphical depiction of the approach in Figure 4. Our solution is sketched in Appendix A.1.

4.2 Implementation Details

For an implementation, it is of importance to choose the size $|T|$ of the trigger set properly, where we have to consider that $|T|$ cannot be arbitrarily big, as the accuracy will drop. To lower-bound $|T|$ we assume an attacker against non-trivial ownership. For simplicity, we use a backdooring algorithm that generates trigger sets from elements where f is undefined. By our simplifying assumption from Section 2.1, the model will classify the images in the trigger set to random labels. Furthermore, assume that the model is ε-accurate (which it also is on the trigger set). Then, one can model a dishonest party to randomly get $(1-\varepsilon)|T|$ out of $|T|$ committed images right using a Binomial distribution. We want to upper-bound this event to have probability at most 2^{-n} and use Hoeffding's inequality to obtain that $|T| > n \cdot \ln(2)/(\frac{1}{|L|} + \varepsilon - 1)$.

To implement our scheme, it is necessary that vk becomes public before `Verify` is used. This ensures that

a party does not simply generate a fake key after seeing a model. A solution for this is to e.g. publish the key on a time-stamped bulletin board like a blockchain. In addition, a statistically hiding commitment scheme should be used that allows for efficient evaluation in zero-knowledge (see Appendix A.1). For this one can e.g. use a scheme based on a cryptographic hash function such as the one described in [39].

5 A Direct Construction of Watermarking

This section describes a scheme for watermarking a neural network model for image classification, and experiments analyzing it with respect to the definitions in Section 3. We demonstrate that it is hard to reduce the persistence of watermarks that are generated with our method. For all the technical details regarding the implementation and hyper-parameters, we refer the reader to Section 5.7.

5.1 The Construction

Similar to Section 4, we use a set of images as the *marking key* or *trigger set* of our construction[4]. To embed the watermark, we optimize the models using both training set and trigger set. We investigate two approaches: the first approach starts from a pre-trained model, i.e., a model that was trained without a trigger set, and continues training the model together with a chosen trigger set. This approach is denoted as PRETRAINED. The second approach trains the model from scratch along with the trigger set. This approach is denoted as FROMSCRATCH. This latter approach is related to *Data Poisoning* techniques.

During training, for each batch, denote as b_t the batch at iteration t, we sample k trigger set images and append them to b_t. We follow this procedure for both approaches. We tested different numbers of k (i.e., 2, 4, and 8), and setting $k = 2$ reach the best results. We hypothesize that this is due to the *Batch-Normalization* layer [23]. The Batch-Normalization layer has two modes of operations. During training, it keeps a running estimate of the computed mean and variance. During an evaluation, the running mean and variance are used for normalization. Hence, adding more images to each batch puts more focus on the trigger set images and makes convergence slower.

In all models we optimize the Negative Log Likelihood loss function on both training set and *trigger set*. Notice, we assume the creator of the model will be the one who embeds the watermark, hence has access to the training set, test set, and *trigger set*.

In the following subsections, we demonstrate the efficiency of our method regarding non-trivial ownership and unremovability and furthermore show that it is functionality-preserving, following the ideas outlined in Section 3. For that we use three different image classification datasets: CIFAR-10, CIFAR-100 and ImageNet [28, 37]. We chose those datasets to demonstrate that our method can be applied to models with a different number of classes and also for large-scale datasets.

5.2 Non-Trivial Ownership

In the *non-trivial ownership* setting, an adversary will not be able to claim ownership of the model even if he knows the watermarking algorithm. To fulfill this requirement we randomly sample the examples for the trigger set. We sampled a set of 100 abstract images, and for each image, we randomly selected a target class.

This sampling-based approach ensures that the examples from the trigger set are uncorrelated to each other. Therefore revealing a subset from the trigger set will not reveal any additional information about the other examples in the set, as is required for public verifiability. Moreover, since both examples and labels are chosen randomly, following this method makes back-propagation based attacks extremely hard. Figure 5 shows an example from the trigger set.

Figure 5: An example image from the trigger set. The label that was assigned to this image was “automobile”.

5.3 Functionality-Preserving

For the *functionality-preserving* property we require that a model with a watermark should be as accurate as a model without a watermark. In general, each task defines

[4] As the set of images will serve a similar purpose as the trigger set from backdoors in Section 2, we denote the marking key as trigger set throughout this section.

its own measure of performance [2, 25, 4, 3]. However, since in the current work we are focused on image classification tasks, we measure the accuracy of the model using the 0-1 loss.

Table 1 summarizes the test set and trigger-set classification accuracy on CIFAR-10 and CIFAR-100, for three different models; (i) a model with no watermark (NO-WM); (ii) a model that was trained with the trigger set from scratch (FROMSCRATCH); and (iii) a pre-trained model that was trained with the trigger set after convergence on the original training data set (PRETRAINED).

Model	Test-set acc.	Trigger-set acc.
CIFAR-10		
NO-WM	93.42	7.0
FROMSCRATCH	93.81	100.0
PRETRAINED	93.65	100.0
CIFAR-100		
NO-WM	74.01	1.0
FROMSCRATCH	73.67	100.0
PRETRAINED	73.62	100.0

Table 1: Classification accuracy for CIFAR-10 and CIFAR-100 datasets on the test set and trigger set.

It can be seen that all models have roughly the same test set accuracy and that in both FROMSCRATCH and PRETRAINED the trigger-set accuracy is 100%. Since the trigger-set labels were chosen randomly, the NO-WM models' accuracy depends on the number of classes. For example, the accuracy on CIFAR-10 is 7.0% while on CIFAR-100 is only 1.0%.

5.4 Unremovability

In order to satisfy the *unremovability* property, we first need to define the types of unremovability functions we are going to explore. Recall that our goal in the unremovability experiments is to investigate the robustness of the watermarked models against changes that aim to remove the watermark while keeping the same functionality of the model. Otherwise, one can set all weights to zero and completely remove the watermark but also destroy the model.

Thus, we are focused on *fine-tuning* experiments. In other words, we wish to keep or improve the performance of the model on the test set by carefully training it. Fine-tuning seems to be the most probable type of attack since it is frequently used and requires less computational resources and training data [38, 45, 35]. Since in our settings we would like to explore the robustness of the watermark against strong attackers, we assumed that the adversary can fine-tune the models using the same amount of training instances and epochs as in training the model.

An important question one can ask is: *when is it still my model?* or other words how much can I change the model and still claim ownership? This question is highly relevant in the case of watermarking. In the current work we handle this issue by measuring the performance of the model on the test set and trigger set, meaning that the original creator of the model can claim ownership of the model if the model is still ε-accurate on the original test set while also ε-accurate on the trigger set. We leave the exploration of different methods and of a theoretical definition of this question for future work.

Fine-Tuning. We define four different variations of fine-tuning procedures:

- *Fine-Tune Last Layer* (FTLL): Update the parameters of the last layer only. In this setting we freeze the parameters in all the layers except in the output layer. One can think of this setting as if the model outputs a new representation of the input features and we fine-tune only the output layer.
- *Fine-Tune All Layers* (FTAL): Update all the layers of the model.
- *Re-Train Last Layers* (RTLL): Initialize the parameters of the output layer with random weights and only update them. In this setting, we freeze the parameters in all the layers except for the output layer. The motivation behind this approach is to investigate the robustness of the watermarked model under noisy conditions. This can alternatively be seen as changing the model to classify for a different set of output labels.
- *Re-Train All Layers* (RTAL): Initialize the parameters of the output layer with random weights and update the parameters in all the layers of the network.

Figure 6 presents the results for both the PRETRAINED and FROMSCRATCH models over the test set and trigger set, after applying these four different fine-tuning techniques.

The results suggest that while both models reach almost the same accuracy on the test set, the FROMSCRATCH models are superior or equal to the PRETRAINED models overall fine-tuning methods. FROMSCRATCH reaches roughly the same accuracy on the trig-

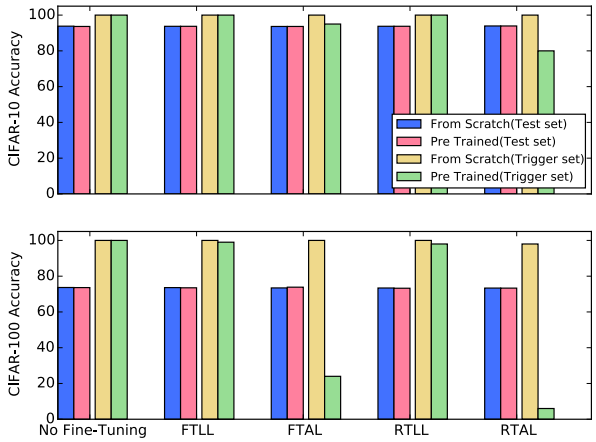


Figure 6: Classification accuracy on the test set and trigger set for CIFAR-10 (top) and CIFAR-100 (bottom) using different fine-tuning techniques. For example, in the bottom right bars we can see that the PRETRAINED model (green) suffers a dramatic decrease in the results comparing the baseline (bottom left) using the RTAL technique.

ger set when each of the four types of fine-tuning approaches is applied.

Notice that this observation holds for both the CIFAR-10 and CIFAR-100 datasets, where for CIFAR-100 it appears to be easier to remove the trigger set using the PRETRAINED models. Concerning the above-mentioned results, we now investigate what will happen if an adversary wants to embed a watermark in a model which has already been watermarked. This can be seen as a black-box attack on the already existing watermark. According to the fine-tuning experiments, removing this new trigger set using the above fine-tuning approaches will not hurt the original trigger set and will dramatically decrease the results on the new trigger set. In the next paragraph, we explore and analyze this setting. Due to the fact that FROMSCRATCH models are more robust than PRETRAINED, for the rest of the paper, we report the results for those models only.

5.5 Ownership Piracy

As we mentioned in Section 3, in this set of experiments we explore the scenario where an adversary wishes to claim ownership of a model which has already been watermarked.

For that purpose, we collected a new trigger set of different 100 images, denoted as TS-NEW, and embedded it to the FROMSCRATCH model (this new set will be used by the adversary to claim ownership of the model). Notice that the FROMSCRATCH models were trained using a different trigger set, denoted as TS-ORIG. Then, we fine-tuned the models using RTLL and RTAL methods. In order to have a fair comparison between the robustness of the trigger sets after fine-tuning, we use the same amount of epochs to embed the new trigger set as we used for the original one.

Figure 7 summarizes the results on the test set, TS-NEW and TS-ORIG. We report results for both the FTAL and RTAL methods together with the baseline results of no fine tuning at all (we did not report here the results of FTLL and RTLL since those can be considered as the easy cases in our setting). The red bars refer to the model with no fine tuning, the yellow bars refer to the FTAL method and the blue bars refer to RTAL.

The results suggest that the original trigger set, TS-ORIG, is still embedded in the model (as is demonstrated in the right columns) and that the accuracy of classifying it even improves after fine-tuning. This may imply that the model embeds the trigger set in a way that is close to the training data distribution. However, in the new trigger set, TS-NEW, we see a significant drop in the accuracy. Notice, we can consider embedding TS-NEW as embedding a watermark using the PRETRAINED approach. Hence, this accuracy drop of TS-NEW is not surprising and goes in hand with the results we observed in Figure 6.

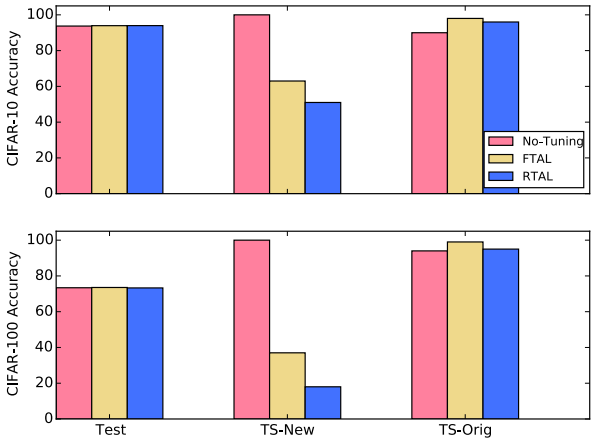


Figure 7: Classification accuracy on CIFAR-10 (top) and CIFAR-100 (bottom) datasets after embedding two trigger sets, TS-ORIG and TS-NEW. We present results for no tuning (red), FTAL (yellow) and TRAL (blue).

Transfer Learning. In transfer learning we would like to use knowledge gained while solving one problem and apply it to a different problem. For example, we use a trained model on one dataset (source dataset) and fine-tune it on a new dataset (target dataset). For that purpose, we fine-tuned the FROMSCRATCH model (which was trained on either CIFAR-10 or CIFAR-100), for another 20 epochs using the labeled part of the STL-10 dataset [12].

Recall that our watermarking scheme is based on the outputs of the model. As a result, when fine-tuning a model on a different dataset it is very likely that we change the number of classes, and then our method will probably break. Therefore, in order to still be able to verify the watermark we save the original output layer, so that on verification time we use the model's original output layer instead of the new one.

Following this approach makes both FTLL and RTLL useless due to the fact that these methods update the parameters of the output layer only. Regarding FTAL, this approach makes sense in specific settings where the classes of the source dataset are related to the target dataset. This property holds for CIFAR-10 but not for CIFAR-100. Therefore we report the results only for RTAL method.

Table 2 summarizes the classification accuracy on the test set of STL-10 and the trigger set after transferring from CIFAR-10 and CIFAR-100.

	Test set acc.	Trigger set acc.
CIFAR10 $\rightarrow$ STL10	81.87	72.0
CIFAR100 $\rightarrow$ STL10	77.3	62.0

Table 2: Classification accuracy on STL-10 dataset and the trigger set, after transferring from either CIFAR-10 or CIFAR-100 models.

Although the trigger set accuracy is smaller after transferring the model to a different dataset, results suggest that the trigger set still has a lot of presence in the network even after fine-tuning on a new dataset.

5.6 ImageNet - Large Scale Visual Recognition Dataset

For the last set of experiments, we would like to explore the robustness of our watermarking method on a large scale dataset. For that purpose, we use ImageNet dataset [37] which contains about 1.3 million training images with over 1000 categories.

Table 3 summarizes the results for the *functionality-preserving* tests. We can see from Table 3 that both models, with and without watermark, achieve roughly the same accuracy in terms of Prec@1 and Prec@5, while the model without the watermark attains 0% on the trigger set and the watermarked model attain 100% on the same set.

	Prec@1	Prec@5
Test Set		
NO-WM	66.64	87.11
FROMSCRATCH	66.51	87.21
Trigger Set		
NO-WM	0.0	0.0
FROMSCRATCH	100.0	100.0

Table 3: ImageNet results, Prec@1 and Prec@5, for a ResNet18 model with and without a watermark.

Notice that the results we report for ResNet18 on ImageNet are slightly below what is reported in the literature. The reason beyond that is due to training for fewer epochs (training a model on ImageNet is computationally expensive, so we train our models for fewer epochs than what is reported).

In Table 4 we report the results of transfer learning from ImageNet to ImageNet, those can be considered as FTAL, and from ImageNet to CIFAR-10, can be considered as RTAL or transfer learning.

	Prec@1	Prec@5
Test Set		
ImageNet $\rightarrow$ ImageNet	66.62	87.22
ImageNet $\rightarrow$ CIFAR-10	90.53	99.77
Trigger Set		
ImageNet $\rightarrow$ ImageNet	100.0	100.0
ImageNet $\rightarrow$ CIFAR-10	24.0	52.0

Table 4: ImageNet results, Prec@1 and Prec@5, for fine tuning using ImageNet and CIFAR-10 datasets.

Notice that after fine tuning on ImageNet, trigger set results are still very high, meaning that the trigger set has a very strong presence in the model also after fine-tuning. When transferring to CIFAR-10, we see a drop in the Prec@1 and Prec@5. However, considering the fact that ImageNet contains 1000 target classes, these results are still significant.

5.7 Technical Details

We implemented all models using the PyTorch package [33]. In all the experiments we used a ResNet-18 model, which is a convolutional based neural network

model with 18 layers [20, 21]. We optimized each of the models using Stochastic Gradient Descent (SGD), using a learning rate of 0.1. For CIFAR-10 and CIFAR-100 we trained the models for 60 epochs while halving the learning rate by ten every 20 epochs. For ImageNet we trained the models for 30 epochs while halving the learning rate by ten every ten epochs. The batch size was set to 100 for the CIFAR10 and CIFAR100, and to 256 for ImageNet. For the fine-tuning tasks, we used the last learning rate that was used during training.

6 Conclusion and Future Work

In this work we proposed a practical analysis of the ability to watermark a neural network using random training instances and random labels. We presented possible attacks that are both black-box and grey-box in the model, and showed how robust our watermarking approach is to them. At the same time, we outlined a theoretical connection to the previous work on backdooring such models.

For future work we would like to define a theoretical boundary for how much change must a party apply to a model before he can claim ownership of the model. We also leave as an open problem the construction of a practically efficient zero-knowledge proof for our publicly verifiable watermarking construction.

Acknowledgments

This work was supported by the BIU Center for Research in Applied Cryptography and Cyber Security in conjunction with the Israel National Cyber Directorate in the Prime Minister's Office.

References

[1] ABADI, M., BARHAM, P., CHEN, J., CHEN, Z., DAVIS, A., DEAN, J., DEVIN, M., GHEMAWAT, S., IRVING, G., ISARD, M., KUDLUR, M., LEVENBERG, J., MONGA, R., MOORE, S., MURRAY, D. G., STEINER, B., TUCKER, P., VASUDEVAN, V., WARDEN, P., WICKE, M., YU, Y., AND ZHENG, X. Tensorflow: A system for large-scale machine learning. In *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation* (Berkeley, CA, USA, 2016), OSDI'16, USENIX Association, pp. 265–283.

[2] ADI, Y., AND KESHET, J. Structed: risk minimization in structured prediction. *The Journal of Machine Learning Research 17*, 1 (2016), 2282–2286.

[3] ADI, Y., KESHET, J., CIBELLI, E., AND GOLDRICK, M. Sequence segmentation using joint rnn and structured prediction models. In *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on* (2017), IEEE, pp. 2422–2426.

[4] ADI, Y., KESHET, J., CIBELLI, E., GUSTAFSON, E., CLOPPER, C., AND GOLDRICK, M. Automatic measurement of vowel duration via structured prediction. *The Journal of the Acoustical Society of America 140*, 6 (2016), 4517–4527.

[5] AMODEI, D., ANUBHAI, R., BATTENBERG, E., CASE, C., CASPER, J., CATANZARO, B., CHEN, J., CHRZANOWSKI, M., COATES, A., DIAMOS, G., ET AL. Deep speech 2: End-to-end speech recognition in english and mandarin. In *International Conference on Machine Learning* (2016), pp. 173–182.

[6] BAHDANAU, D., CHO, K., AND BENGIO, Y. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* (2014).

[7] BARAK, B., GOLDREICH, O., IMPAGLIAZZO, R., RUDICH, S., SAHAI, A., VADHAN, S., AND YANG, K. On the (im) possibility of obfuscating programs. *Journal of the ACM (JACM) 59*, 2 (2012), 6.

[8] BONEH, D., AND SHAW, J. Collusion-secure fingerprinting for digital data. In *Advances in Cryptology — CRYPT0' 95* (1995), D. Coppersmith, Ed., Springer, pp. 452–465.

[9] BRASSARD, G., CHAUM, D., AND CRÉPEAU, C. Minimum disclosure proofs of knowledge. *J. Comput. Syst. Sci. 37*, 2 (1988), 156–189.

[10] CHEN, H., ROHANI, B. D., AND KOUSHANFAR, F. Deepmarks: A digital fingerprinting framework for deep neural networks, 2018.

[11] CISSE, M. M., ADI, Y., NEVEROVA, N., AND KESHET, J. Houdini: Fooling deep structured visual and speech recognition models with adversarial examples. In *Advances in Neural Information Processing Systems* (2017), pp. 6980–6990.

[12] COATES, A., NG, A., AND LEE, H. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics* (2011), pp. 215–223.

[13] FIAT, A., AND SHAMIR, A. How to prove yourself: Practical solutions to identification and signature problems. In *Conference on the Theory and Application of Cryptographic Techniques* (1986), Springer, pp. 186–194.

[14] GOLDREICH, O. *The Foundations of Cryptography - Volume 1, Basic Techniques*. Cambridge University Press, 2001.

[15] GOLDWASSER, S., MICALI, S., AND RACKOFF, C. The knowledge complexity of interactive proof-systems (extended abstract). In *Proceedings of the 17th Annual ACM Symposium on Theory of Computing, May 6-8, 1985, Providence, Rhode Island, USA* (1985), pp. 291–304.

[16] GOODFELLOW, I. J., SHLENS, J., AND SZEGEDY, C. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572* (2014).

[17] GRAVES, A., FERNÁNDEZ, S., GOMEZ, F., AND SCHMIDHUBER, J. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning* (2006), ACM, pp. 369–376.

[18] GU, T., DOLAN-GAVITT, B., AND GARG, S. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *CoRR abs/1708.06733* (2017).

[19] HE, K., GKIOXARI, G., DOLLÁR, P., AND GIRSHICK, R. Mask r-cnn. In *Computer Vision (ICCV), 2017 IEEE International Conference on* (2017), IEEE, pp. 2980–2988.

[20] HE, K., ZHANG, X., REN, S., AND SUN, J. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2016), pp. 770–778.

[21] HE, K., ZHANG, X., REN, S., AND SUN, J. Identity mappings in deep residual networks. In *European Conference on Computer Vision* (2016), Springer, pp. 630–645.

[22] HOSSEINI, H., CHEN, Y., KANNAN, S., ZHANG, B., AND POOVENDRAN, R. Blocking transferability of adversarial examples in black-box learning systems. *arXiv preprint arXiv:1703.04318* (2017).

[23] IOFFE, S., AND SZEGEDY, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning* (2015), pp. 448–456.

[24] KATZENBEISSER, S., AND PETITCOLAS, F. *Information hiding*. Artech house, 2016.

[25] KESHET, J. Optimizing the measure of performance in structured prediction. *Advanced Structured Prediction. The MIT Press. URL http://u. cs. biu. ac. il/˜ jkeshet/papers/Keshet14. pdf* (2014).

[26] KIM, S., AND WU, D. J. Watermarking cryptographic functionalities from standard lattice assumptions. In *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part I* (2017), pp. 503–536.

[27] KREUK, F., ADI, Y., CISSE, M., AND KESHET, J. Fooling end-to-end speaker verification by adversarial examples. *arXiv preprint arXiv:1801.03339* (2018).

[28] KRIZHEVSKY, A., AND HINTON, G. Learning multiple layers of features from tiny images.

[29] LECUN, Y., BENGIO, Y., AND HINTON, G. Deep learning. *Nature 521*, 7553 (2015), 436–444.

[30] LIU, Y., MA, S., AAFER, Y., LEE, W.-C., AND ZHAI, J. Trojaning attack on neural networks. Tech Report, 2017.

[31] MERRER, E. L., PEREZ, P., AND TRÉDAN, G. Adversarial frontier stitching for remote neural network watermarking, 2017.

[32] NAOR, D., NAOR, M., AND LOTSPIECH, J. Revocation and tracing schemes for stateless receivers. In *Annual International Cryptology Conference* (2001), Springer, pp. 41–62.

[33] PASZKE, A., GROSS, S., CHINTALA, S., CHANAN, G., YANG, E., DEVITO, Z., LIN, Z., DESMAISON, A., ANTIGA, L., AND LERER, A. Automatic differentiation in pytorch.

[34] PETITCOLAS, F. A., ANDERSON, R. J., AND KUHN, M. G. Information hiding-a survey. *Proceedings of the IEEE 87*, 7 (1999), 1062–1078.

[35] RAZAVIAN, A. S., AZIZPOUR, H., SULLIVAN, J., AND CARLSSON, S. Cnn features off-the-shelf: an astounding baseline for recognition. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2014 IEEE Conference on* (2014), IEEE, pp. 512–519.

[36] RIBEIRO, M., GROLINGER, K., AND CAPRETZ, M. A. Mlaas: Machine learning as a service. In *Machine Learning and Applications (ICMLA), 2015 IEEE 14th International Conference on* (2015), IEEE, pp. 896–902.

[37] RUSSAKOVSKY, O., DENG, J., SU, H., KRAUSE, J., SATHEESH, S., MA, S., HUANG, Z., KARPATHY, A., KHOSLA, A., BERNSTEIN, M., ET AL. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision 115*, 3 (2015), 211–252.

[38] SIMONYAN, K., AND ZISSERMAN, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

[39] SMART, N. P. *Cryptography Made Simple*. Information Security and Cryptography. Springer, 2016.

[40] STOCK, P., AND CISSE, M. Convnets and imagenet beyond accuracy: Explanations, bias detection, adversarial examples and model criticism. *arXiv preprint arXiv:1711.11443* (2017).

[41] TOSHEV, A., AND SZEGEDY, C. Deeppose: Human pose estimation via deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2014), pp. 1653–1660.

[42] UCHIDA, Y., NAGAI, Y., SAKAZAWA, S., AND SATOH, S. Embedding watermarks into deep neural networks. In *Proceedings of the 2017 ACM on International Conference on Multimedia Retrieval* (2017), ACM, pp. 269–277.

[43] VENUGOPAL, A., USZKOREIT, J., TALBOT, D., OCH, F. J., AND GANITKEVITCH, J. Watermarking the outputs of structured prediction with an application in statistical machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing* (2011), Association for Computational Linguistics, pp. 1363–1372.

[44] WATTENBERG, M., VIÉGAS, F., AND HARDT, M. Attacking discrimination with smarter machine learning. *Google Research 17* (2016).

[45] YOSINSKI, J., CLUNE, J., BENGIO, Y., AND LIPSON, H. How transferable are features in deep neural networks? In *Advances in neural information processing systems* (2014), pp. 3320–3328.

[46] ZHANG, C., BENGIO, S., HARDT, M., RECHT, B., AND VINYALS, O. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530* (2016).

A Supplementary Material

In this appendix we further discuss how to achieve public verifiability for a variant of our watermarking scheme. Let us first introduce the following additional notation: for a vector $\mathbf{e} \in \{0,1\}^\ell$, let $\mathbf{e}|_0 = \{i \in [\ell] \mid \mathbf{e}[i] = 0\}$ be the set of all indices where $\mathbf{e}$ is 0 and define $\mathbf{e}|_1$ accordingly. Given a verification key $\mathsf{vk} = \{c_t^{(i)}, c_L^{(i)}\}_{i\in[\ell]}$ containing ℓ elements and a vector $\mathbf{e} \in \{0,1\}^\ell$, we write the selection of elements from vk according to $\mathbf{e}$ as

$$\mathsf{vk}|_0^{\mathbf{e}} = \{c_t^{(i)}, c_L^{(i)}\}_{i\in\mathbf{e}|_0} \quad \text{and} \quad \mathsf{vk}|_1^{\mathbf{e}} = \{c_t^{(i)}, c_L^{(i)}\}_{i\in\mathbf{e}|_1}.$$

For a marking key $\mathsf{mk} = (\mathsf{b}, \{r_t^{(i)}, r_L^{(i)}\}_{i\in[\ell]})$ with ℓ elements and $\mathsf{b} = \{T^{(i)}, T_L^{(i)}\}_{i\in[\ell]}$ we then define

$$\mathsf{mk}|_0^{\mathbf{e}} = (\mathsf{b}|_0^{\mathbf{e}}, \{r_t^{(i)}, r_L^{(i)}\}_{i\in\mathbf{e}|_0}) \text{ with } \mathsf{b}|_0^{\mathbf{e}} = \{T^{(i)}, T_L^{(i)}\}_{i\in\mathbf{e}|_0}$$

(and $\mathsf{mk}|_1^{\mathbf{e}}$ accordingly). We assume the existence of a cryptographic hash function $H : \{0,1\}^{p(n)} \to \{0,1\}^n$.

A.1 From Private to Public Verifiability

To achieve public verifiability, we will make use of a cryptographic tool called a *zero-knowledge argument* [15], which is a technique that allows a prover $\mathcal{P}$ to convince a verifier $\mathcal{V}$ that a certain public statement is true, without giving away any further information. This idea is similar to the idea of unlimited public verification as outlined in Section 4.1.

Zero-Knowledge Arguments. Let TM be an abbreviation for Turing Machines. An iTM is defined to be an interactive TM, i.e. a Turing Machine with a special communication tape. Let $L_R \subseteq \{0,1\}^*$ be an NP language and R be its related NP-relation, i.e. $(x,w) \in R$ iff $x \in L_R$ and the TM used to define L_R outputs 1 on input of the statement x and the witness w. We write $R_x = \{w \mid (x,w) \in R\}$ for the set of witnesses for a fixed x. Moreover, let $\mathcal{P}, \mathcal{V}$ be a pair of PPT iTMs. For $(x,w) \in R$, $\mathcal{P}$ will obtain w as input while $\mathcal{V}$ obtains an auxiliary random string $z \in \{0,1\}^*$. In addition, x will be input to both TMs. Denote with $\mathcal{V}^{\mathcal{P}(a)}(b)$ the output of the iTM $\mathcal{V}$ with input b when communicating with an instance of $\mathcal{P}$ that has input a.

$(\mathcal{P}, \mathcal{V})$ is called an *interactive proof system* for the language L if the following two conditions hold:

Completeness: For every $x \in L_R$ there exists a string w such that for every z: $\Pr[\mathcal{V}^{\mathcal{P}(x,w)}(x,z) = 1]$ is negligibly close to 1.

Soundness: For every $x \notin L_R$, every PPT iTM $\mathcal{P}^*$ and every string w, z: $\Pr[\mathcal{V}^{\mathcal{P}^*(x,w)}(x,z) = 1]$ is negligible.

An interactive proof system is called computational *zero-knowledge* if for every PPT $\hat{\mathcal{V}}$ there exists a PPT simulator $\mathcal{S}$ such that for any $x \in L_R$

$$\{\hat{\mathcal{V}}^{\mathcal{P}(x,w)}(x,z)\}_{w\in R_x, z\in\{0,1\}^*} \approx_c \{\mathcal{S}(x,z)\}_{z\in\{0,1\}^*},$$

meaning that all information which can be learned from observing a protocol transcript can also be obtained from running a polynomial-time simulator $\mathcal{S}$ which has no knowledge of the witness w.

A.1.1 Outlining the Idea

An intuitive approach to build `PVerify` is to convert the algorithm $\texttt{Verify}(\mathsf{mk}, \mathsf{vk}, M)$ from Section 4 into an NP relation R and use a zero-knowledge argument system. Unfortunately, this must fail due to Step 1 of `Verify`: there, one tests if the item b contained in mk actually is a backdoor as defined above. Therefore, we would need access to the ground-truth function f in the interactive argument system. This first of all needs human assistance, but is moreover only possible by revealing the backdoor elements.

We will now give a different version of the scheme from Section 4 which embeds an additional proof into vk. This proof shows that, with overwhelming probability, most of the elements in the verification key indeed form a backdoor. Based on this, we will then design a different verification procedure, based on a zero-knowledge argument system.

A.1.2 A Convincing Argument that most Committed Values are Wrongly Classified

Verifying that most of the elements of the trigger set are labeled wrongly is possible, if one accepts[5] to release a portion of this set. To solve the proof-of-misclassification problem, we use the so-called *cut-and-choose* technique: in cut-and-choose, the verifier $\mathcal{V}$ will ask the prover $\mathcal{P}$ to open a subset of the committed inputs and labels from the verification key. Here, $\mathcal{V}$ is allowed to choose the subset that will be opened to him. Intuitively, if $\mathcal{P}$ committed to a large number elements that are correctly labeled (according to $\mathcal{O}_f$), then at least one of them will show up in the values opened by $\mathcal{P}$ with overwhelming probability over the choice that $\mathcal{V}$ makes. Hence, most of the remaining commitments which were not opened must form a correct backdoor.

[5] This is fine if T, as in our experiments, only consists of random images.

To use cut-and-choose, the backdoor size must contain $\ell > n$ elements, where our analysis will use $\ell = 4n$ (other values of ℓ are also possible). Then, consider the following protocol between $\mathcal{P}$ and $\mathcal{V}$:

$\texttt{CnC}(\ell)$:

1. $\mathcal{P}$ runs $(\mathsf{mk}, \mathsf{vk}) \leftarrow \texttt{KeyGen}(\ell)$ to obtain a backdoor of size ℓ and sends vk to $\mathcal{V}$. We again define $\mathsf{mk} = (\mathsf{b}, \{r_t^{(i)}, r_L^{(i)}\}_{i\in[\ell]})$, $\mathsf{vk} = \{c_t^{(i)}, c_L^{(i)}\}_{i\in[\ell]}$
2. $\mathcal{V}$ chooses $\mathbf{e} \leftarrow \{0,1\}^\ell$ uniformly at random and sends it to $\mathcal{P}$.
3. $\mathcal{P}$ sends $\mathsf{mk}|_1^{\mathbf{e}}$ to $\mathcal{V}$.
4. $\mathcal{V}$ checks that for $i \in \mathbf{e}|_1$ that
   (a) $\texttt{Open}(c_t^{(i)}, t^{(i)}, r_t^{(i)}) = 1$;
   (b) $\texttt{Open}(c_L^{(i)}, T_L^{(i)}, r_L^{(i)}) = 1$; and
   (c) $T_L^{(i)} \neq f(t^{(i)})$.

Assume that $\mathcal{P}$ chose exactly one element of the backdoor in vk wrongly, then this will be revealed by CnC to an honest $\mathcal{V}$ with probability $1/2$ (where $\mathcal{P}$ must open $\mathsf{vk}|_1^{\mathbf{e}}$ to the values he put into $c_t^{(i)}, c_L^{(i)}$ during KeyGen due to the binding-property of the commitment). In general, one can show that a cheating $\mathcal{P}$ can put at most n non-backdooring inputs into $\mathsf{vk}|_0^{\mathbf{e}}$ except with probability negligible in n. Therefore, if the above check passes for $\ell = 4n$ at then least $1/2$ of the values for $\mathsf{vk}|_0^{\mathbf{e}}$ must have the wrong committed label as in a valid backdoor with overwhelming probability.

The above argument can be made non-interactive and thus publicly verifiable using the Fiat-Shamir transform[13]: in the protocol CnC, $\mathcal{P}$ can generate the bit string $\mathbf{e}$ itself by hashing vk using a cryptographic hash function H. Then $\mathbf{e}$ will be distributed as if it was chosen by an honest verifier, while it is sufficiently random by the guarantees of the hash function to allow the same analysis for cut-and-choose. Any $\mathcal{V}$ can recompute the value $\mathbf{e}$ if it is generated from the commitments (while this also means that the challenge $\mathbf{e}$ is generated after the commitments were computed), and we can turn the above algorithm CnC into the following non-interactive key-generation algorithm PKeyGen.

$\texttt{PKeyGen}(\ell)$:

1. Run $(\mathsf{mk}, \mathsf{vk}) \leftarrow \texttt{KeyGen}(\ell)$.
2. Compute $\mathbf{e} \leftarrow H(\mathsf{vk})$.
3. Set $\mathsf{mk}_p \leftarrow (\mathsf{mk}, \mathbf{e})$, $\mathsf{vk}_p \leftarrow (\mathsf{vk}, \mathsf{mk}|_1^{\mathbf{e}})$ and return $(\mathsf{mk}_p, \mathsf{vk}_p)$.

A.1.3 Constructing the Public Verification Algorithm

In the modified scheme, the Mark algorithm will only use the private subset $\mathsf{mk}|_0^{\mathbf{e}}$ of mk_p but will otherwise remain unchanged. The public verification algorithm for a model M then follows the following structure: (i) $\mathcal{V}$ recomputes the challenge $\mathbf{e}$; (ii) $\mathcal{V}$ checks vk_p to assure that all of $\mathsf{vk}|_1^{\mathbf{e}}$ will form a valid backdoor ; and (iii) $\mathcal{P}, \mathcal{V}$ run Classify on $\mathsf{mk}|_0^{\mathbf{e}}$ using the interactive zero-knowledge argument system, and further test if the watermarking conditions on $M, \mathsf{mk}|_0^{\mathbf{e}}, \mathsf{vk}|_0^{\mathbf{e}}$ hold.

For an arbitrary model M, one can rewrite the steps 2 and 3 of Verify (using M, Open, Classify) into a binary circuit C that outputs 1 iff the prover inputs the correct $\mathsf{mk}|_0^{\mathbf{e}}$ which opens $\mathsf{vk}|_0^{\mathbf{e}}$ and if enough of these openings satisfy Classify. Both $\mathcal{P}, \mathcal{V}$ can generate this circuit C as its construction does not involve private information. For the interactive zero-knowledge argument, we let the relation R be defined by boolean circuits that output 1 where $x = C, w = \mathsf{mk}|_0^{\mathbf{e}}$ in the following protocol PVerify, which will obtain the model M as well as $\mathsf{mk}_p = (\mathsf{mk}, \mathbf{e})$ and $\mathsf{vk}_p = (\mathsf{vk}, \mathsf{mk}|_1^{\mathbf{e}})$ where $\mathsf{vk} = \{c_t^{(i)}, c_L^{(i)}\}_{i\in[\ell]}$, $\mathsf{mk} = (\mathsf{b}, \{r_t^{(i)}, r_L^{(i)}\}_{i\in[\ell]})$ and $\mathsf{b} = \{T^{(i)}, T_L^{(i)}\}_{i\in[\ell]}$ as input.

1. $\mathcal{V}$ computes $\mathbf{e}' \leftarrow H(\mathsf{vk})$. If $\mathsf{mk}|_1^{\mathbf{e}}$ in vk_p does not match $\mathbf{e}'$ then abort, else continue assuming $\mathbf{e} = \mathbf{e}'$.
2. $\mathcal{V}$ checks that for all $i \in \mathbf{e}|_1$:
   (a) $\texttt{Open}(c_t^{(i)}, t^{(i)}, r_t^{(i)}) = 1$
   (b) $\texttt{Open}(c_L^{(i)}, T_L^{(i)}, r_L^{(i)}) = 1$
   (c) $T_L^{(i)} \neq f(t^{(i)})$

   If one of the checks fails, then $\mathcal{V}$ aborts.
3. $\mathcal{P}, \mathcal{V}$ compute a circuit C with input $\mathsf{mk}|_0^{\mathbf{e}}$ that outputs 1 iff for all $i \in \mathbf{e}|_0$:
   (a) $\texttt{Open}(c_t^{(i)}, t^{(i)}, r_t^{(i)}) = 1$
   (b) $\texttt{Open}(c_L^{(i)}, T_L^{(i)}, r_L^{(i)}) = 1$.

   Moreover, it tests that $\texttt{Classify}(t^{(i)}, M) = T_L^{(i)}$ for all but $\varepsilon|\mathbf{e}|_0|$ elements.
4. $\mathcal{P}, \mathcal{V}$ run a zero-knowledge argument for the given relation R using C as the statement, where the witness $\mathsf{mk}|_0^{\mathbf{e}}$ is the secret input of $\mathcal{P}$. $\mathcal{V}$ accepts iff the argument succeeds.

Assume the protocol `PVerify` succeeds. Then in the interactive argument, M classifies at least $(1-\varepsilon)|\mathbf{e}|_0| \approx (1-\varepsilon)2n$ values of the backdoor `b` to the committed value. For $\approx n$ of the commitments, we can assume that the committed label does not coincide with the ground-truth function f due to the guarantees of Step 1. It is easy to see that this translates into a 2ε-guarantee for the correct backdoor. By choosing a larger number ℓ for the size of the backdoor, one can achieve values that are arbitrarily close to ε in the above protocol.

Hide and Speak: Deep Neural Networks for Speech Steganography

Hide and Speak: Deep Neural Networks for Speech Steganography

Felix Kreuk
Department of Computer Science
Bar-Ilan University
Ramat-Gan, Israel

Yossi Adi
Department of Computer Science
Bar-Ilan University
Ramat-Gan, Israel

Bhiksha Raj
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, USA

Rita Singh
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, USA

Joseph Keshet
Department of Computer Science
Bar-Ilan University
Ramat-Gan, Israel

Abstract

Steganography is the science of hiding a secret message within an ordinary public message, which is referred to as Carrier. Traditionally, digital signal processing techniques, such as least significant bit encoding, were used for hiding messages. In this paper, we explore the use of deep neural networks as steganographic functions for speech data. To this end, we jointly optimize two neural networks: the first network encodes the message inside a carrier, while the second network decodes the message from the modified carrier. We further constrain the model with differentiable short-time Fourier transform layers in order to be robust against frequency to time domain transformations. We demonstrated the effectiveness of our method on several speech datasets and analyzed the results quantitatively and qualitatively. Moreover, we showed that our approach could be applied to conceal multiple messages in a single carrier using multiple decoders or a single conditional decoder. Lastly, we evaluated our model under different channel distortions. Qualitative experiments suggest that modifications to the carrier are unnoticeable by human listeners and that the decoded messages are highly intelligible.

1 Introduction

Steganography ("steganos" – concealed or covered, "graphein" – writing) is the science of concealing messages inside other messages. It is generally used to convey concealed "secret" messages to recipients who are aware of their presence, while keeping even their existence hidden from other unaware parties who only see the "public" or "carrier" message.

In this work, we address the topic of *speech* steganography – hiding secret spoken messages within public audio files. Although one can simply hide written text inside audio files and convey the same lexical content, concealing audio inside audio preserves additional features. For instance, the secret message may convey the speaker identity, the sentiment of the speaker, prosody, etc. These features can be used for later identification and authentication of the message.

Preprint. Under review.

Traditionally, steganographic functions exploited actual or perceptual redundancies in the carrier signal. The most common approach is to encode the secret message is in the least significant bits of individual sound samples [18]. Other methods include concealing the secret message in the *phase* of the frequency components of the carrier [8] or in the form of the parameters of a miniscule echo that is introduced into the carrier signal [3]. The redundancies to be exploited by any particular method must be explicitly selected, and are generally a feature of the steganography algorithm.

Recently, [2, 36] proposed to use deep neural networks as a stenographic function for hiding an image inside another image. In this line of work, the network *learns* to conceal a hidden message inside the carrier without manually specifying a particular redundancy to exploit.

Although these studies presented impressive results on image data, the applicability of such models for speech data was not explored. As opposed to working with raw images in the domain of vision processing, the common approach when learning from speech data is to work at the frequency domain, and specifically, using the short time Fourier transform (STFT) to capture the spectral changes over time. The STFT output is a complex matrix composed of the Fourier transform of different time frames. The common practice is to use the absolute values (magintudes) of the STFT measurements, and to maintain a substantial overlap between adjacent frames [21]. Consequently, the original signal cannot be losslessly recovered from STFT. Moreover, as only the magnitude is considered, the phase needs to be recovered. This process complicates the restoration of the time domain signal even further.

In this study, we showed that steganography models proposed for vision are less suitable for speech. We build on the work by [2, 36] and propose a new model that includes the STFT and inverse-STFT as differentiable layers within the network, thus imposing a vital constraint on the network outputs.

The proposed model is comprised of three parts. The first learns to encode a hidden message inside the carrier. The second component are differential STFT and inverse-STFT layers that simulate transformations between frequency and time domains. Lastly, the third component learns to decode a hidden message from a generated carrier. Additionally, we demonstrated for the first time, that the above scheme now permits us to hide *multiple* secret messages into a *single* carrier, each potentially with a different intended recipient who is the only person who can recover it.

We demonstrate qualitatively and quantitatively that the proposed method is able to both effectively hide secret messages into a carrier and recover them from the carrier. Further analysis shows that the proposed method is robust to various channel distortions and compression methods, such as MP3 encoding, Additive White Gaussian Noise, sample rate reduction, etc. Qualitative experiments suggest that modifications to the carrier are unnoticeable by human listeners and that the decoded messages are highly intelligible and preserve other semantic content, such as speaker identity.

Our Contribution:

- We explore the use of neural networks as stenographic functions for speech data while adding STFT/inverse-STFT layers during the training phase. As such, we create a steganographic function that is robust to time/frequency transformations.
- We embed multiple speech messages in a single carrier, may it be voice or music audio signal.
- We provide extensive empirical and subjective analysis of the reconstructed signals and show that the produced carriers are indistinguishable from the original carriers, while keeping the decoded messages highly intelligible.

The paper is organized as follows, Section 2 formulates all the notations we use throughout the paper. In Section 3 we describe the proposed model. Section 4 and Section 5 present the results together with objective and subjective analysis. Section 6 summarizes the related work. We conclude the paper in Section 7 with a discussion and future work.

2 Notation and representation

In this section, we formulate the task of speech steganography rigorously and set the notation for the rest of the paper. Let $\mathbf{x} = (x[0], x[1], \ldots, x[N-1])$ be a speech signal that is composed of N samples. The spectral content of the signal changes over time, therefore it is often represented by

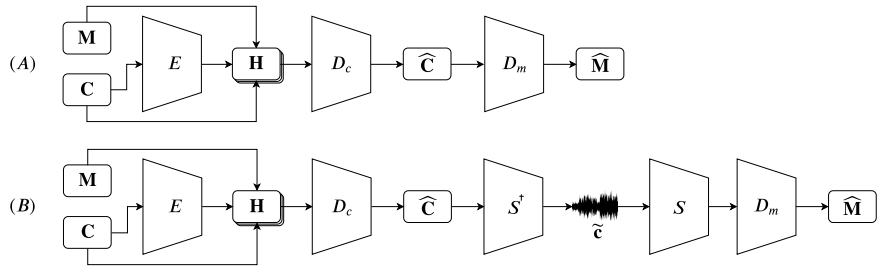


Figure 1: Model overview: the encoder E gets as input the carrier $\mathbf{C}$, its output is then concatenated with $\mathbf{C}$ and $\mathbf{M}$ to create $\mathbf{H}$. Then, the carrier decoder D_c generates the new embedded carrier, from which the message decoder D_m decodes the message $\hat{\mathbf{M}}$; Sub-figure (1A) depicts the model proposed by [2] (1B) depicts our proposed model.

the short-time Fourier transform, commonly known as the *spectrogram*, rather than by the Fourier transform. The STFT can be interpreted as the Fourier transform of the signal multiplied by a sliding window $\mathbf{w} = (w[0], w[1], \ldots, w[W-1])$ and is defined as

$$\mathcal{X}[r,k] = \sum_{n=0}^{N-1} \mathbf{x}[n]\, \mathbf{w}[rL-n]\, e^{-2\pi jkn/N},$$

where $k \in [0, L-1]$ is the digital frequency, $r \in [0, \lceil N/L \rceil - 1]$ is an index to the processing window, and L is the number of overlapping samples between two adjacent windows. The STFT $\mathcal{X}[r,k]$ is a matrix of complex numbers, however, in speech processing often only its absolute value, or magnitude, is considered in subsequent processing. Throughout the paper we use lower-case vectors, e.g. $\mathbf{x}$, to denote the time-domain speech waveform and upper-case matrices, e.g. $\mathbf{X}$, to denote the magnitude of the STFT, that is $\mathbf{X}[r,k] = |\mathcal{X}[r,k]|$. Similarly, we denote the phase of the STFT using $\angle\mathbf{X}$, namely, $\angle\mathbf{X}[r,k] = \arctan(\mathrm{Im}(\mathcal{X}[r,k])/\,\mathrm{Re}(\mathcal{X}[r,k]))$. Finally, we denote by $\mathcal{S}$ the operator that gets as input a real signal and outputs the magnitude matrix of its STFT, $\mathbf{X} = \mathcal{S}(\mathbf{x})$, and denote by $\mathcal{S}^\dagger$ the operator that gets as input the magnitude and phase matrices of the STFT complex matrix, and returns the recovered speech waveform, $\mathbf{x} = \mathcal{S}^\dagger(\mathbf{X}, \angle\mathbf{X})$. Here $\mathcal{S}^\dagger$ is computed by taking the inverse Fourier transform of each row of $\mathbf{X}[r,:]$, and then reconstructing the waveform by combining the outputs by the overlap-and-add method. Note that this recostruction is imperfect, since there is a substantial overlap between adjacent windows (L) when using STFT in speech processing, hence part of the signal at each window is lost [17].

Recall, in speech steganography the goal is to conceal a hidden speech message within a carrier speech segment. Specifically, the steganography system is a function that gets as input a carrier utterance, denote as $\mathbf{c}$, and a hidden message, denote as $\mathbf{m}$. The outputs of the system are the *embedded carrier* $\hat{\mathbf{c}}$, and consequently the *recovered message*, $\hat{\mathbf{m}}$, such that the following constraints are satisfied: (i) both $\hat{\mathbf{c}}$ and $\hat{\mathbf{m}}$ should be perceptually similar to $\mathbf{c}$ and $\mathbf{m}$, respectively, in the ears of a human listener; (ii) the message $\hat{\mathbf{m}}$ should be recoverable from the carrier $\hat{\mathbf{c}}$ and should be intelligible; and lastly (iii) an average listener should not be able to detect the presence of a hidden message embedded in $\hat{\mathbf{c}}$. We would like to stress that the system is not trained or limited to a specific carrier or message, a particular speaker, or a fixed signal length.

3 Model

Our model is composed of the following components: (i) *Encoder Network* denoted E; (ii) *Carrier Decoder Network* denoted D_c; and (iii) *Message Decoder Network* denoted D_m. The model is schematically depicted in Figure 1. The Encoder Network E, gets as input a carrier $\mathbf{C}$, and outputs a latent representation of the carrier, $E(\mathbf{C})$. Then, we compose a joint representation of the encoded carrier $E(\mathbf{C})$, message $\mathbf{M}$, and original carrier $\mathbf{C}$ by concatenating all three along the convolutional channel axis, $\mathbf{H} = [E(\mathbf{C}); \mathbf{C}; \mathbf{M}]$ as proposed in Zhu et al. [36], where we denote the concatenation operator by ;.

The Carrier Decoder Network, D_c, gets as input the aforementioned representation and outputs $\hat{\mathbf{C}}$, the carrier embedded with the hidden message. Lastly, the Message Decoder Network D_m, gets as

input $\hat{\mathbf{C}}$ and outputs $\hat{\mathbf{M}}$, the reconstructed hidden message. Each of the above components is a neural network, where the parameters are found by minimizing the absolute error between the carrier and the embedded carrier and between the original message and the reconstructed message. Formally, we wish to optimize the following loss function,

$$\mathcal{L}(\mathbf{C}, \mathbf{M}) = \lambda_c \|\mathbf{C} - \hat{\mathbf{C}}\|_1 + \lambda_m \|\mathbf{M} - \hat{\mathbf{M}}\|_1,$$

where

$$\hat{\mathbf{C}} = D_c(\mathbf{H}), \quad \hat{\mathbf{M}} = D_m(\hat{\mathbf{C}}),$$

and λ_c and λ_m are set to balance between the reconstruction of the carrier to the reconstruction of the message.

Recall, our goal is to transmit $\hat{\mathbf{c}}$, which means that we need to recover the time-domain waveform from the magnitude $\hat{\mathbf{C}}$. Unfortunately, the recovery of $\hat{\mathbf{c}}$ from the STFT magnitude only, is an ill-posed problem in general [15, 17]. Ideally, we would like to reconstruct $\hat{\mathbf{c}}$ using $\mathcal{S}^\dagger(\hat{\mathbf{C}}, \angle\hat{\mathbf{C}})$. However, the phase $\angle\hat{\mathbf{C}}$ is unknown, and therefore must be approximated.

One way to overcome this phase recovery obstacle is to use the classical alternating projection algorithm of Griffin and Lim [12]. The algorithm recovers the phase from each window of the STFT magnitude $\hat{\mathbf{C}}$, and then using inverse Fourier transform the waveform can be recovered. Unfortunately, this method produces a carrier with noticeable artifacts, which sounds very different from the original carrier. The message, however, can be recovered and is intelligible.

Another way to reconstruct the time-domain signal is to use the magnitude of the embedded carrier $\hat{\mathbf{C}}$, and the phase of the original carrier, $\angle\mathbf{C}$. Namely, the reconstructed carrier is $\tilde{\mathbf{c}} = \mathcal{S}^\dagger(\hat{\mathbf{C}}, \angle\mathbf{C})$. In subjective tests we found that the restructured carrier $\tilde{\mathbf{c}}$ sounds acoustically similar to the original carrier $\mathbf{c}$. However, when trying to recover the hidden message from $\mathcal{S}(\tilde{\mathbf{c}})$ by using it as an input to the Message Decoder Network D_m, we get unintelligible output. This is due to the fact that we used a mismatched phase.

To mitigate that, we turn to a third solution, where we constrain the loss function so that $\tilde{\mathbf{C}}$ will be close to $\hat{\mathbf{C}}$. Formally, we minimize the following:

$$\mathcal{L}'(\mathbf{C}, \mathbf{M}) = \lambda_c \|\mathbf{C} - \hat{\mathbf{C}}\|_1 + \lambda_m \|\mathbf{M} - \tilde{\mathbf{M}}\|_1, \tag{1}$$

where

$$\hat{\mathbf{C}} = D_c(\mathbf{H}), \quad \tilde{\mathbf{C}} = \mathcal{S}(\mathcal{S}^\dagger(\hat{\mathbf{C}}, \angle\mathbf{C})), \quad \tilde{\mathbf{M}} = D_m(\tilde{\mathbf{C}}) \tag{2}$$

Practically, we added $\mathcal{S}$ and $\mathcal{S}^\dagger$ operators as differentiable complex 1D-convolution layers as illustrated in Figure 1B. In words, we jointly optimize the model to generate $\hat{\mathbf{C}}$ which will preserve the hidden message after $\mathcal{S} \circ \mathcal{S}^\dagger$ and will also resemble $\mathbf{C}$.

Concealing multiple messages. Previous work on neural networks for image steganography reported results for concealing a single message only in a given carrier. However, this approach can be extended naturally to concealing multiple messages into a single carrier. We explored two possible approaches to implement such extension.

In the first approach we use a set of *multiple message decoders*. The model is provided with a single carrier $\mathbf{C}$, and a set of k messages, $\{\mathbf{M}_i\}_{i=1}^k$, where $k > 1$. The latent representation $\mathbf{H}$ is the concatenation of the encoded carrier, the original carrier, and all k messages. We use k message decoders denoted by $D_{m,i}$ where $1 < i \leq k$, one for each message. Each message decoder $D_{m,i}$ is trained to decode the i-th message $\mathbf{M}_i$ from $\tilde{\mathbf{C}}$. The modified loss function is therefore,

$$\mathcal{L}'(\mathbf{C}, \mathbf{M}) = \lambda_c \|\mathbf{C} - \hat{\mathbf{C}}\|_1 + \lambda_m \sum_{i=1}^{k} \|\mathbf{M}_i - \tilde{\mathbf{M}}_i\|_1, \tag{3}$$

where, $\hat{\mathbf{C}} = D_c(\mathbf{H})$, $\tilde{\mathbf{C}} = \mathcal{S}(\mathcal{S}^\dagger(\hat{\mathbf{C}}, \angle\mathbf{C}))$, and $\tilde{\mathbf{M}}_i = D_{m,i}(\tilde{\mathbf{C}})$.

In the second approach a single *conditional decoder* is used to decode all messages. The multiple decoders setup leads to linear growth in memory costs in the size of k. In other words, concealing k messages requires k separate decoders, which can be memory intensive for large values of k. To

mitigate that, we further explore the use of a single conditional decoder instead of multiple decoders. In this setup, the encoding process is identical to the case of multiple decoders. During decoding we condition the decoder with a set of codes $\{q_i\}_{i=1}^{k}$. Therefore, D_m gets as input not only $\tilde{\mathbf{C}}$ but also a code indicating the message index. Each code q_i is represented as a one-hot vector of size k with 1 at the ith index and zeros elsewhere. We follow the work by Choi et al. [5], where the conditioning label is spatially replicated and concatenated with the original input.

4 Experimental results

In this section, we present our experimental results. We start by describing the experimental setup. Then, we quantitatively evaluate our model on two speech datasets. Sample waveforms for all models and experiments as well as source code will be available under: `http://hyperurl.co/ab7c3g`.

We evaluated our approach on TIMIT [11] and YOHO [4] datasets using the standard train/val/test splits. Each utterance was sampled at 16kHz and represented as its power spectrum by applying the STFT with $W = 256$ FFT frequency bins and sliding window with a shift $L = 128$. Training examples were generated by randomly selecting one utterance as carrier and k other utterances as messages for $k \in \{1, 3, 5, 7\}$. Thus, the matching of carrier and message is completely arbitrary and not fixed. Moreover, they may originate from different speakers.

All models were trained using Adam for 80 epochs with an initial learning rate of 10^{-3} and a decaying factor of 10 every 20 epochs. We balance between the carrier and message reconstruction losses using $\lambda_c = 3$, $\lambda_m = 1$. Each component in our model is implemented as a Gated Convolutional Neural Network as proposed by Dauphin et al. [7]. Specifically, E is composed of three blocks of gated convolutions, D_c is composed of four blocks of gated convolutions, and D_m is composed of six blocks of gated convolutions. Each block contains 64 kernels of size 3×3.

We compared the proposed approach to three baselines. In the first baseline we concatenate the lower half of frequencies of $\mathbf{M}$ above the lower half of frequencies of $\mathbf{C}$, to form $\hat{\mathbf{C}}$. Message decoding was performed by extracting the upper half of frequencies from $\hat{\mathbf{C}}$ and zero padding to the original size. We denoted this baseline by *Frequency Chop*. The second and third baselines are: Baluja [2] and Zhu et al. [36], we refer the reader to Section 6 for more details regarding these baselines.

4.1 Single message

We start by concealing a single message. Table 1 reports the Absolute-Error (AE) and Signal-to-Noise-Ratio (SNR) for both carrier and message for all baselines and proposed models on TIMIT and YOHO.

Notice, while both Baluja [2] and Zhu et al. [36] yield low carrier errors, their direct application to speech data produced unintelligible messages with low SNR. This is due to the fact that these models are not constrained to retain the same carrier content after the conversion to time-domain and back. The introduction of this constraint as a differentiable layer greatly improved message decoding from the converted time-domain signal. Figure 2 depicts the training process of the proposed model and baselines. It can be seen that without any constraints, the baseline message decoders diverge. Lastly, Frequency Chop retains much of the message content after decoding, but creates a carrier with noticeable artifacts. This is due to the fact that the hidden message is audiable as it resides in the carrier's high frequencies.

Zhu et al. [36] suggested adding adversarial loss term to the training objective in order to further improve the embedded carrier to be indistinguishable from the original one. To further investigate that, we explored including adversarial loss terms in the optimization problem. The adversarial discriminator component was aimed at discriminating between $\mathbf{C}$ and $\tilde{\mathbf{C}}$. Similarly to the effect on images [36], when incorporating the adversarial terms into the training objective, the carrier quality drastically improved and contained less artifacts. As a consequence, the message reconstruction was less accurate but still highly intelligible.

4.2 Multiple messages

We turn now to evaluate the performance of our model when concealing more than one message. We explored the two settings described in Section 3, namely multiple decoders and single conditional

Table 1: Absolute Error (lower is better) and Signal to Noise Ratio (higher is better) for both carrier and message using single message embedding. Results are reported for both TIMIT and YOHO datasets.

Dataset	Model	Carrier loss	Carrier SNR	Msg. loss	Msg. SNR
TIMIT	Frequency Chop	0.0770	2.040	0.046	8.184
	Baluja [2]	0.0023	29.92	0.096	0.451
	Zhu et al. [36]	0.0027	35.27	0.078	0.822
	Ours	**0.0016**	31.27	**0.035**	**10.01**
	Ours+Adversarial	0.0022	**37.15**	0.051	4.460
YOHO	Frequency Chop	0.0550	1.830	0.038	8.49
	Baluja [2]	0.0021	29.18	0.072	0.95
	Zhu et al. [36]	0.0040	30.17	0.066	1.39
	Ours	**0.0016**	30.84	**0.028**	**9.48**
	Ours+Adversarial	0.0016	**34.15**	0.033	7.04

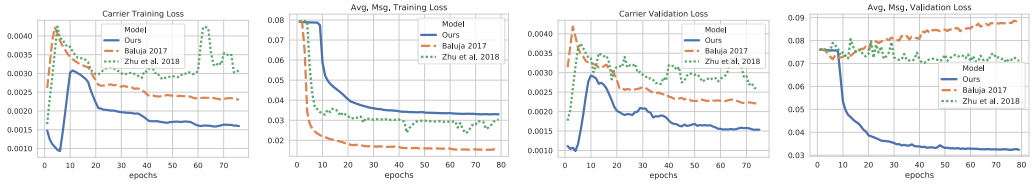


Figure 2: The pair of figures on the left present the training loss on carrier and message over time, the pair of figures on the right present the validation loss over time. During validation, we simulate real world conditions by converting the carrier to time-domain and back. While all models converge on the training set, not accounting for the above conversion leads to the divergence of the message decoders on the validation set for Baluja [2], Zhu et al. [36].

decoder. Table 2 summarizes the results. The loss and SNR results for messages are averaged over the k messages. Interestingly, both settings achieved comparable results for embedding 3, 5, and 7 messages in a single carrier.

An increase in the number of messages translates to higher loss values in both carrier and messages. These results are to be expected as the model is forced to work at higher compression rates due to concealing and recovering more messages while keeping the carrier dimension the same.

Notice, models who conceal 3 messages achieve comparable average message SNR to models who conceal a single message. On the other hand, models who conceal 5 or 7 messages suffer a significant drop in average message SNR compared to concealing a single message, but still produce intelligible messages (see provided link). Figure 3 compares the training loss values while embedding 3, 5 and 7 messages using both methods.

5 Analysis

In this section we provide several evaluations regarding the quality of the embedded carrier, and the recovered message. We start with a subjective analysis of the resulted waveforms.

5.1 Subjective evaluation

Carrier ABX Testing. To validate that the difference between $\mathbf{c}$ and $\tilde{\mathbf{c}}$ is not detectable by humans, we performed ABX testing. An ABX test is a standard way to asses detectable differences between two choices of sensory stimuli. We present each human with two audio samples A and B. Each of these two samples is either the original carrier or the carrier embedded with a hidden message. These two samples are followed by a third sample X randomly selected to be either A or B. Next, the human must choose whether X is the same as A or B. We generated 50 (25 from TIMIT and 25 from YOHO) audio samples, for each audio sample we recorded 20 answers from Amazon Mechanical Turk (AMT), 1000 answers overall. Only 51.2% (48.8% for TIMIT and 53.6% for YOHO) of the

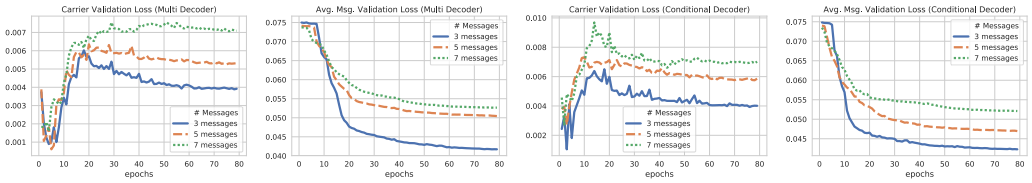


Figure 3: A comparison between carrier and messages training loss for $k \in \{3, 5, 7\}$. The first pair of figures from the left correspond to using multiple decoders, the right pair of figures correspond to using a single conditional decoder.

Table 2: AE and SNR for both carrier and message concealing 3, 5, and 7 messages using either multiple decoders or one conditional decoder. Results are reported for both TIMIT and YOHO datasets.

	TIMIT					YOLO			
	Carrier		Message			Carrier		Message	
model	loss	SNR	loss	SNR	model	loss	SNR	loss	SNR
multi-3	0.0042	28.15	0.0458	7.08	multi-3	0.0042	26.8	0.0349	7.28
cond-3	0.0043	27.07	0.0463	6.93	cond-3	0.0038	26.53	0.0344	7.53
multi-5	0.0058	26.58	0.0550	5.06	multi-5	0.0046	26.34	0.0428	4.79
cond-5	0.0063	25.68	0.0516	5.59	cond-5	0.0051	25.32	0.0392	5.56
multi-7	0.0077	26.44	0.0585	4.37	multi-7	0.0058	25.01	0.0476	3.68
cond-7	0.0076	25.69	0.0582	4.42	cond-7	0.0057	24.44	0.0445	4.16

carriers embedded with hidden messages could be distinguished from the original ones by humans (the optimal ratio is 50%). Therefore we conclude that the modifications made by the steganographic function are not distinguishable by a human ear.

Message intelligibility. A major metric in evaluating a speech steganography system is the intelligibility of the reconstructed messages. To quantify this measure we conducted an additional subjective experiment in AMT. We generated 80 samples: 40 original messages and 40 messages reconstructed by our model. We recorded 10 answers for each sample (800 answers overall). The participants were instructed to transcribe the presented samples, and the Word Error Rate (WER) and Character Error Rate (CER) were measured. While WER is a coarse measure, CER provides finer evaluation of transcription error. A small difference between the original messages and reconstructed messages in terms of WER and CER would suggest that the steganographic system does not degrade intelligibility of the hidden messages. The CER/WER measured on original and reconstructed messages was 4.205%/2.15% (3.02%/2.23% for TIMIT and 5.39%/2.07% for YOHO) and 3.631%/1.42% (2.002%/0.87% for TIMIT and 5.26%/1.97% for YOHO) respectively. We therefore deduce that our system does not degrade the intelligibility of speech signal.

Speaker Recognition. One advantage to concealing speech instead of text is the preservation of non-lexical content such as: intonation, prosody, speaker identity, etc. To evaluate if speaker identity was preserved, we conducted an additional experiment adhering to the Speaker Verification Protocol [14]. In this experiment, we provided 400 human listeners with 4 speech segments: the first three were uttered by a single speaker, the forth segment was uttered by that same speaker 50% of the time, and by a different speaker (from the same gender[1]) the rest of the time. The listeners were tasked with verifying whether the forth sample's speaker is the same one as in the first three segments. In 82% of cases, listeners were able to distinguish whether the forth speaker matched the speaker of the first three segments or not. Hence we deduce that much of the speaker identity is preserved in the generated messages.

5.2 Robustness to channel distortion

Another critical evaluation is performance under noisy conditions. To explore that we applied different channel distortion and compression techniques on the reconstructed carrier $\tilde{\mathbf{c}}$. In Table 3

[1]Notice, we use speakers of the same gender to make the task of speaker differentiation more challenging.

Table 3: Noise robustness results. We denote by σ the norm of the added noise

Noise	Msg. Loss	Msg. SNR	Noise	Msg. Loss	Msg. SNR
Down-sampling to 8k	0.0459	8.71	AWGN, $\sigma = 0.1$	15.16	-35.45
MP3 compression 192k	0.0442	7.47	AWGN, $\sigma = 0.01$	0.0769	-10.06
MP3 compression 128k	0.0446	7.44	AWGN, $\sigma = 0.001$	0.044	9.8
MP3 compression 64k	0.0622	5.96	Speckle, $\sigma = 0.1$	0.039	9.53
MP3 compression 32k	0.089	2.82	Speckle, $\sigma = 0.01$	0.035	10.01
Precision reduction 8-bit	0.16	0.251	Speckle, $\sigma = 0.001$	0.035	10.01

we describe message reconstruction results after distorting the carrier using: 16kHz to 8kHz down-sampling, MP3 compression (using different bit rates), 16-bit precision to 8-bit precision, Additive White Gaussian Noise (AWGN) and Speckle noise. Results suggest that our method is robust to carrier down-sampling, MP3 compression and noise addition. Contrarily, the model is sensitive to bit precision change, but this is to be expected as the message decoder relies on miniscule carrier modification in order to reconstruct the hidden message.

5.3 Using music as a carrier

Speech signals have a considerable amount of redundancy as most human speech resides in the range of $\sim$1-5kHz. Such redundancy can be exploited by the steganographic function. In contrast, music signals are less sparse as most frequencies are utilized. To better understand our model's capability of hiding messages in rich audio segments we experimented with music played by multiple instruments simultaneously as carriers. To that end, we used a model pre-trained on TIMIT to embed speech messages in music carriers. This resulted in a 0.0025 ℓ_1 loss and 30.8db SNR on the carrier, and a 0.043 ℓ_1 loss and 7.33db SNR on the message, results comperable to those on speech data. We therefore deduce that the model works with rich carrier signals such as music, and that it generalizes to out-of-distribution audio domains (samples can be found in the provided link).

6 Related work

A large variety of steganography methods have been proposed over the years, where most of them are applied to images [22, 19]. The most common approach manipulates the LSB of the input data to place the secret information. This can be done uniformly or adaptively through simple replacement or through more complicated techniques [10, 30]. Although often not observable by humans, statistical analysis of the perturb data can reveal whether a given file contains a hidden message or not [10]. Advanced methods, such as HUGO [24], attempt to preserve the input statistics in order to generate better steganography functions. Another example is Wavelet Obtained Weights [16] which penalizes distortion of predictable regions of the input data using a bank of directional filters.

A closely related task is *Watermarking*. Both approaches aim to encode a secret message into a data file. However, in steganography the goal is to perform secret communication while in watermarking the goal is verification and ownership protection. Several watermarking techniques use LSB encoding [33, 34]. Recently, Uchida et al. [32], Adi et al. [1] suggested to embed watermarks into neural networks parameters.

Adversarial examples are synthetic patterns carefully crafted by adding a peculiar noise to legitimate examples. They are indistinguishable from the legitimate examples by a human, yet they have demonstrated a strong ability to cause catastrophic failure of state of the art neural systems [29, 6, 20, 23, 27]. While the existence of adversarial examples is usually seen as a disadvantage of neural networks, it can be a desirable property for hiding secret information. Instead of injecting perturbations that lead to wrong classification, one can consider the possibility of encoding useful information via adding specific perturbations.

Recently, neural networks have been widely used for image steganography [2, 36, 13, 26, 25, 35, 31, 9, 28]. Baluja [2], first suggested to train neural networks to hide an entire image within another image (similarly to Figure 1A). Zhu et al. [36] extended the work of [2] while adding an adversarial loss term to the objective. Hayes and Danezis [13] suggested to use generative adversarial learning

to generate stenographic images. However, none of the above approaches explored speech data and were focused on hiding a single message only.

7 Discussion and future work

In this study, we explored for the first time the use of neural networks for speech steganography. We introduced crucial constraints to the training objective. We demonstrated the ability of our model to hide several messages in the a single carrier using both multiple decoders and one conditional decoder. We evaluated our model under different noisy conditions and empirically demonstrated that the modifications to carriers are indistinguishable by humans and that the messages recovered by our model are highly intelligible.

A critical step in applying speech steganography in real world applications is the ability to evade steganalysis methods. We leave investigation of that for future work.

References

[1] Y. Adi, C. Baum, M. Cisse, B. Pinkas, and J. Keshet. Turning your weakness into a strength: Watermarking deep neural networks by backdooring. *USENIX*, 2018.

[2] S. Baluja. Hiding images in plain sight: Deep steganography. In *Advances in Neural Information Processing Systems*, pages 2069–2079, 2017.

[3] W. Bender, D. Gruhl, and N. Morimoto. Method and apparatus for echo data hiding in audio signals, Apr. 6 1999. US Patent 5,893,067.

[4] J. P. Campbell. Testing with the yoho cd-rom voice verification corpus. In *ICASSP*, 1995.

[5] Y. Choi, M. Choi, M. Kim, J.-W. Ha, S. Kim, and J. Choo. Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8789–8797, 2018.

[6] M. M. Cisse, Y. Adi, N. Neverova, and J. Keshet. Houdini: Fooling deep structured visual and speech recognition models with adversarial examples. In *Advances in Neural Information Processing Systems*, pages 6977–6987, 2017.

[7] Y. N. Dauphin, A. Fan, M. Auli, and D. Grangier. Language modeling with gated convolutional networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 933–941. JMLR. org, 2017.

[8] X. Dong, M. F. Bocko, and Z. Ignjatovic. Data hiding via phase manipulation of audio signals. In *Acoustics, Speech, and Signal Processing, 2004. Proceedings.(ICASSP'04). IEEE International Conference on*, volume 5, pages V–377. IEEE, 2004.

[9] N. N. El-emam. Embedding a large amount of information using high secure neural based steganography algorithm. *International Journal of Information and Communication Engineering*, 4(2):2, 2008.

[10] J. Fridrich, M. Goljan, and R. Du. Detecting lsb steganography in color, and gray-scale images. *IEEE multimedia*, 8(4):22–28, 2001.

[11] J. S. Garofolo, L. F. Lamel, W. M. Fisher, J. G. Fiscus, and D. S. Pallett. Darpa timit acoustic-phonetic continous speech corpus cd-rom. nist speech disc 1-1.1. *NASA STI/Recon technical report n*, 93, 1993.

[12] D. Griffin and J. Lim. Signal estimation from modified short-time fourier transform. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 32(2):236–243, 1984.

[13] J. Hayes and G. Danezis. Generating steganographic images via adversarial training. In *Advances in Neural Information Processing Systems*, pages 1954–1963, 2017.

[14] G. Heigold, I. Moreno, S. Bengio, and N. Shazeer. End-to-end text-dependent speaker verification. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5115–5119. IEEE, 2016.

[15] E. Hofstetter. Construction of time-limited functions with specified autocorrelation functions. *IEEE Transactions on Information Theory*, 10(2):119–126, 1964.

[16] V. Holub and J. Fridrich. Designing steganographic distortion using directional filters. In *2012 IEEE International workshop on information forensics and security (WIFS)*, pages 234–239. IEEE, 2012.

[17] K. Jaganathan, Y. C. Eldar, and B. Hassibi. Stft phase retrieval: Uniqueness guarantees and recovery algorithms. *IEEE Journal of selected topics in signal processing*, 10(4):770–781, 2016.

[18] P. Jayaram, H. Ranganatha, and H. Anupama. Information hiding using audio steganography–a survey. *The International Journal of Multimedia & Its Applications (IJMA) Vol*, 3:86–96, 2011.

[19] G. Kessler. An overview of steganography for the computer forensics examiner. retrieved february 26, 2006, 2004.

[20] F. Kreuk, Y. Adi, M. Cisse, and J. Keshet. Fooling end-to-end speaker verification by adversarial examples. *Proc. ICASSP*, 2018.

[21] J. S. Lim and A. V. Oppenheim. Enhancement and bandwidth compression of noisy speech. *Proceedings of the IEEE*, 67(12):1586–1604, 1979.

[22] T. Morkel, J. H. Eloff, and M. S. Olivier. An overview of image steganography. In *ISSA*, pages 1–11, 2005.

[23] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, pages 506–519. ACM, 2017.

[24] T. Pevnỳ, T. Filler, and P. Bas. Using high-dimensional image models to perform highly undetectable steganography. In *International Workshop on Information Hiding*, pages 161–177. Springer, 2010.

[25] L. Pibre, J. Pasquet, D. Ienco, and M. Chaumont. Deep learning is a good steganalysis tool when embedding key is reused for different images, even if there is a cover sourcemismatch. *Electronic Imaging*, 2016(8):1–11, 2016.

[26] Y. Qian, J. Dong, W. Wang, and T. Tan. Deep learning for steganalysis via convolutional neural networks. In *Media Watermarking, Security, and Forensics 2015*, volume 9409, page 94090J. International Society for Optics and Photonics, 2015.

[27] M. Sharif, S. Bhagavatula, L. Bauer, and M. K. Reiter. Adversarial generative nets: Neural network attacks on state-of-the-art face recognition. *arXiv preprint arXiv:1801.00349*, 2017.

[28] H. Shi, J. Dong, W. Wang, Y. Qian, and X. Zhang. Ssgan: secure steganography based on generative adversarial networks. In *Pacific Rim Conference on Multimedia*, pages 534–544. Springer, 2017.

[29] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. In *Proc. ICLR*, 2014.

[30] A. A. Tamimi, A. M. Abdalla, and O. Al-Allaf. Hiding an image inside another image using variable-rate steganography. *International Journal of Advanced Computer Science and Applications (IJACSA)*, 4(10), 2013.

[31] W. Tang, S. Tan, B. Li, and J. Huang. Automatic steganographic distortion learning using a generative adversarial network. *IEEE Signal Processing Letters*, 24(10):1547–1551, 2017.

[32] Y. Uchida, Y. Nagai, S. Sakazawa, and S. Satoh. Embedding watermarks into deep neural networks. In *Proceedings of the 2017 ACM on International Conference on Multimedia Retrieval*, pages 269–277. ACM, 2017.

[33] R. G. Van Schyndel, A. Z. Tirkel, and C. F. Osborne. A digital watermark. In *Image Processing, 1994. Proceedings. ICIP-94., IEEE International Conference*, volume 2, pages 86–90. IEEE, 1994.

[34] R. B. Wolfgang and E. J. Delp. A watermark for digital images. In *ICIP (3)*, pages 219–222, 1996.

[35] P. Wu, Y. Yang, and X. Li. Stegnet: Mega image steganography capacity with deep convolutional network. *Future Internet*, 10(6):54, 2018.

[36] J. Zhu, R. Kaplan, J. Johnson, and L. Fei-Fei. Hidden: Hiding data with deep networks. In *European Conference on Computer Vision*, pages 682–697. Springer, 2018.

Chapter 6

Summary and Future Research

In this dissertation I presented two main research directions. The first direction concerns the design and implementation of new machine learning algorithms to advance, automate, and scale-up linguistic research. I showed that these algorithms can be effectively applied for advancing speech sciences. The second part of this thesis deals with the robustness of deep neural models to adversarial and out-of-distribution examples. Here I demonstrated how to extend adversarial example generation to structured prediction tasks such as Automatic Speech Recognition, Image Segmentation, and Pose Estimation. Then, I presented a novel method, based on recent advances in dense word representations for detecting out-of-distribution and adversarial examples.

Future research directions involve several questions:

Can machine learning tools be applied to advance other scientific fields? The need for objective automatic tools for measuring fine-grained properties is not unique to speech science. Many other scientific fields can benefit from such objective measurement tools. One example would be when analyzing voice disorders such as *vocal cord paresis and paralysis*. When one of the vocal folds is paralyzed, the folds are not able to meet in the midline in order to start the glottic attack, preventing the development of the subglottic pressure needed to initiate normal speech (Orlikoff et al., 2009). Symptoms include effortful phonation, vocal fatigue, breathiness and odynophonia (Belafsky et al., 2002). The standard assessment protocol is mainly comprised of intrusive subjective evaluation and a few general objective voice quality measurements. However, none of these assessments is specific for evaluating the quality and efficacy of vocal folds closure alone (Cohen et al., 2019). Developing and implementing highly accurate measurement tools for such domains will significantly affect the quality of treatment as well as the well-being of the patients.

Can Bayesian deep learning produce better uncertainty estimates? As we presented in Chapter 3, it was observed that DNNs often produce high confidence predictions for out-of-distribution inputs (Hendrycks and Gimpel, 2016; Nguyen et al., 2015), random noise (Hendrycks and Gimpel, 2016) and adversarial examples (Cisse et al., 2017b;

Goodfellow et al., 2014; Kreuk et al., 2018).

One possible explanation for such phenomena is the fact that deep models which are trained in a discriminative way push unlikely data points to unknown regions in the latent space. As a result, since the models are not considering any prior distribution, the predictions are often highly confident also for unlikely data instances.

A perfect tool for handling such a discrepancy is *Bayesian neural networks*, where we consider both prior and posterior distributions in the optimization and inference processes (Gal, 2016). I believe that closing the gap between discriminative training and *Bayesian neural network* will produce better uncertainty estimates and models, which will be more robust against such inputs.

Can PAC-Bayesian generalization bound be applied to deep networks? In recent deep learning research, we observe that deep nets are able to encode highly complex input-output relations, and in practice, they do not tend to overfit. This tendency to not overfit has been investigated in numerous works on generalization bounds (Bartlett et al., 2017, 2019; Dziugaite and Roy, 2017; Germain et al., 2016; Langford and Caruana, 2002; Langford and Shawe-Taylor, 2002; McAllester, 2003). Indeed, many generalization bounds apply to deep nets. However, most of these bounds assume that the net's loss function is bounded (Bartlett et al., 2017; Dziugaite and Roy, 2017; Neyshabur et al., 2017). Also applicable are stability generalization bounds which consider unbounded loss functions, such as the negative log-likelihood loss. But those assume convexity of the loss w.r.t. the parameters (Bousquet and Elisseeff, 2002; Shalev-Shwartz and Ben-David, 2014) or early stopping criteria (Zhang et al., 2016a). However, classical deep nets such as AlexNet (Krizhevsky et al., 2012), VGG (Simonyan and Zisserman, 2014), or ResNet (He et al., 2016) are trained using the negative log-likelihood loss of a composite function represented by a computation graph. Using such a general functional form results in a negative log-likelihood loss which is generally non-convex in the network parameters. Hence deep net training still challenges any of the aforementioned assumptions and currently available theoretical explanations on generalization bounds. For future work, I would like to explore the use of PAC-Bayesian generalization bounds for the negative log likelihood loss of DNNs. I believe such bounds will shed light on neural network training, neural architecture search, and how to better initialize them.

Bibliography

Y. Adi and J. Keshet. Structed: risk minimization in structured prediction. *The Journal of Machine Learning Research*, 17(1):2282–2286, 2016.

Y. Adi, J. Keshet, and M. Goldrick. Vowel duration measurement using deep neural networks. In *2015 IEEE 25th International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6. IEEE, 2015.

Y. Adi, E. Kermany, Y. Belinkov, O. Lavi, and Y. Goldberg. Fine-grained analysis of sentence embeddings using auxiliary prediction tasks. *arXiv preprint arXiv:1608.04207*, 2016a.

Y. Adi, J. Keshet, E. Cibelli, E. Gustafson, C. Clopper, and M. Goldrick. Automatic measurement of vowel duration via structured prediction. *The Journal of the Acoustical Society of America*, 140(6):4517–4527, 2016b.

Y. Adi, J. Keshet, O. Dmitrieva, and M. Goldrick. Automatic measurement of voice onset time and prevoicing using recurrent neural networks. In *INTERSPEECH*, pages 3152–3155, 2016c.

Y. Adi, E. Kermany, Y. Belinkov, O. Lavi, and Y. Goldberg. Analysis of sentence embedding models using prediction tasks in natural language processing. *IBM Journal of Research and Development*, 61(4/5):3–1, 2017a.

Y. Adi, J. Keshet, E. Cibelli, and M. Goldrick. Sequence segmentation using joint rnn and structured prediction models. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2422–2426. IEEE, 2017b.

Y. Adi, N. Zeghidour, R. Collobert, N. Usunier, V. Liptchinsky, and G. Synnaeve. To reverse the gradient or not: an empirical comparison of adversarial and multi-task learning in speech recognition. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3742–3746. IEEE, 2019.

D. Amodei, R. Anubhai, E. Battenberg, C. Case, J. Casper, B. Catanzaro, J. Chen, M. Chrzanowski, A. Coates, G. Diamos, et al. Deep speech 2: End-to-end speech recognition in english and mandarin. *arXiv preprint arXiv:1512.02595*, 2015.

D. Amodei, R. Anubhai, E. Battenberg, C. Case, J. Casper, B. Catanzaro, J. Chen, M. Chrzanowski, A. Coates, G. Diamos, et al. Deep speech 2: End-to-end speech recognition in english and mandarin. In *International Conference on Machine Learning*, pages 173–182, 2016a.

D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman, and D. Mané. Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*, 2016b.

D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

S. Baluja. Hiding images in plain sight: Deep steganography. In *Advances in Neural Information Processing Systems*, pages 2069–2079, 2017.

P. L. Bartlett, D. J. Foster, and M. J. Telgarsky. Spectrally-normalized margin bounds for neural networks. In *Advances in Neural Information Processing Systems*, pages 6241–6250, 2017.

P. L. Bartlett, N. Harvey, C. Liaw, and A. Mehrabian. Nearly-tight vc-dimension and pseudodimension bounds for piecewise linear neural networks. *Journal of Machine Learning Research*, 20(63):1–17, 2019. URL `http://jmlr.org/papers/v20/17-612.html`.

P. C. Belafsky, G. N. Postma, T. R. Reulbach, B. W. Holland, and J. A. Koufman. Muscle tension dysphonia as a sign of underlying glottal insufficiency. *OtolaryngologyHead and Neck Surgery*, 127(5):448–451, 2002.

W. Bender, D. Gruhl, and N. Morimoto. Method and apparatus for echo data hiding in audio signals, Apr. 6 1999. US Patent 5,893,067.

D. Boneh and J. Shaw. Collusion-secure fingerprinting for digital data. In D. Coppersmith, editor, *Advances in Cryptology — CRYPT0' 95*, pages 452–465. Springer, 1995.

O. Bousquet and A. Elisseeff. Stability and generalization. *The Journal of Machine Learning Research*, 2:499–526, 2002.

R. Caruana. Multitask learning. In *Learning to learn*, pages 95–133. Springer, 1998.

L.-C. Chen, A. G. Schwing, A. L. Yuille, and R. Urtasun. Learning deep structured models. In *ICML*, 2015.

M. Cisse, P. Bojanowski, E. Grave, Y. Dauphin, and N. Usunier. Parseval networks: Improving robustness to adversarial examples. *arXiv preprint arXiv:1704.08847*, 2017a.

M. M. Cisse, Y. Adi, N. Neverova, and J. Keshet. Houdini: Fooling deep structured visual and speech recognition models with adversarial examples. In *Advances in neural information processing systems*, pages 6977–6987, 2017b.

C. G. Clopper and T. N. Tamati. Effects of local lexical competition and regional dialect on vowel production. *Journal of the Acoustical Society of America*, 136(1):1–4, 2014.

J. T. Cohen, A. Cohen, L. Benyamini, Y. Adi, and J. Keshet. Predicting glottal closure insufficiency using fundamental frequency contour analysis. *Head & neck*, 2019.

R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12: 2493–2537, 2011.

R. Collobert, C. Puhrsch, and G. Synnaeve. Wav2letter: an end-to-end convnet-based speech recognition system. *arXiv preprint arXiv:1609.03193*, 2016.

T. Do, T. Arti, et al. Neural conditional random fields. In *AISTATS*, pages 177–184, 2010.

X. Dong, M. F. Bocko, and Z. Ignjatovic. Data hiding via phase manipulation of audio signals. In *Acoustics, Speech, and Signal Processing, 2004. Proceedings.(ICASSP'04). IEEE International Conference on*, volume 5, pages V–377. IEEE, 2004.

G. K. Dziugaite and D. M. Roy. Computing nonvacuous generalization bounds for deep (stochastic) neural networks with many more parameters than training data. *arXiv preprint arXiv:1703.11008*, 2017.

J. L. Elman. Distributed representations, simple recurrent networks, and grammatical structure. *Machine learning*, 7(2-3):195–225, 1991.

A. Fawzi, O. Fawzi, and P. Frossard. Analysis of classifiers' robustness to adversarial perturbations. *arXiv preprint arXiv:1502.02590*, 2015.

A. Fawzi, S.-M. Moosavi-Dezfooli, and P. Frossard. Robustness of classifiers: from adversarial to random noise. In *Advances in Neural Information Processing Systems*, pages 1624–1632, 2016.

A. Frome, G. S. Corrado, J. Shlens, S. Bengio, J. Dean, T. Mikolov, et al. Devise: A deep visual-semantic embedding model. In *Advances in neural information processing systems*, pages 2121–2129, 2013.

Y. Gal. *Uncertainty in deep learning*. PhD thesis, PhD thesis, University of Cambridge, 2016.

Y. Ganin et al. Domain-adversarial training of neural networks. *The Journal of Machine Learning Research*, 17(1):2096–2030, 2016.

P. Germain, F. Bach, A. Lacoste, and S. Lacoste-Julien. Pac-bayesian theory meets bayesian inference. In *Advances in Neural Information Processing Systems*, pages 1884–1892, 2016.

M. Goldrick, H. R. Baker, A. Murphy, and M. Baese-Berk. Interaction and representational integration: Evidence from speech errors. *Cognition*, 121(1):58–72, 2011.

M. Goldrick, R. McClain, E. Cibelli, Y. Adi, E. Gustafson, C. Moers, and J. Keshet. The influence of lexical selection disruptions on articulation. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 2018.

I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. In *Proc. ICLR*, 2015.

T. Gu, B. Dolan-Gavitt, and S. Garg. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *CoRR*, abs/1708.06733, 2017. URL `http://arxiv.org/abs/1708.06733`.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

G. Heigold, I. Moreno, S. Bengio, and N. Shazeer. End-to-end text-dependent speaker verification. In *ICASSP*, 2016.

J. R. Heller and M. Goldrick. Grammatical constraints on phonological encoding in speech production. *Psychonomic bulletin & review*, 21(6):1576–1582, 2014.

D. Hendrycks and K. Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *arXiv preprint arXiv:1610.02136*, 2016.

J. Hillenbrand, L. A. Getty, M. J. Clark, and K. Wheeler. Acoustic characteristics of american english vowels. *The Journal of the Acoustical society of America*, 97(5):3099–3111, 1995.

G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *Signal Processing Magazine, IEEE*, 29(6):82–97, 2012.

G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

P. Jayaram, H. Ranganatha, and H. Anupama. Information hiding using audio steganography–a survey. *The International Journal of Multimedia & Its Applications (IJMA) Vol*, 3:86–96, 2011.

S. Katzenbeisser and F. Petitcolas. *Information hiding*. Artech house, 2016.

J. Keshet, S. Shalev-Shwartz, Y. Singer, and D. Chazan. A large margin algorithm for speech and audio segmentation. *IEEE Trans. on Audio, Speech and Language Processing*, Nov 2007.

E. Kiperwasser and Y. Goldberg. Simple and accurate dependency parsing using bidirectional lstm feature representations. *arXiv preprint*, 2016.

F. Kreuk, Y. Adi, M. Cisse, and J. Keshet. Fooling end-to-end speaker verification with adversarial examples. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1962–1966. IEEE, 2018.

A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, pages 1097–1105, 2012.

A. Kurakin, I. Goodfellow, and S. Bengio. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*, 2016.

B. Lakshminarayanan, A. Pritzel, and C. Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems*, pages 6405–6416, 2017.

G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, and C. Dyer. Neural architectures for named entity recognition. *arXiv preprint*, 2016.

J. Langford and R. Caruana. (not) bounding the true error. In *Advances in Neural Information Processing Systems*, pages 809–816, 2002.

J. Langford and J. Shawe-Taylor. Pac-bayes & margins. *Advances in neural information processing systems*, 15:423–430, 2002.

Y. LeCun, L. Jackel, L. Bottou, A. Brunot, C. Cortes, J. Denker, H. Drucker, I. Guyon, U. Muller, E. Sackinger, et al. Comparison of learning algorithms for handwritten digit recognition. In *International conference on artificial neural networks*, volume 60, pages 53–60, 1995.

K. Lee, H. Lee, K. Lee, and J. Shin. Training confidence-calibrated classifiers for detecting out-of-distribution samples. *arXiv preprint arXiv:1711.09325*, 2017.

S. Liang, Y. Li, and R. Srikant. Enhancing the reliability of out-of-distribution image detection in neural networks.

E. Littwin and L. Wolf. The multiverse loss for robust transfer learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3957–3966, 2016.

D. McAllester. Simplified pac-bayesian margin bounds. *Learning Theory and Kernel Machines*, pages 203–215, 2003.

T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013a.

T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013b.

S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard. Deepfool: a simple and accurate method to fool deep neural networks. *arXiv preprint arXiv:1511.04599*, 2015.

E. Naaman, Y. Adi, and J. Keshet. Learning similarity functions for pronunciation variations. *arXiv preprint arXiv:1703.09817*, 2017.

D. Naor, M. Naor, and J. Lotspiech. Revocation and tracing schemes for stateless receivers. In *Annual International Cryptology Conference*, pages 41–62. Springer, 2001.

A. Newell, K. Yang, and J. Deng. Stacked hourglass networks for human pose estimation. *ECCV*, 2016.

B. Neyshabur, S. Bhojanapalli, D. McAllester, and N. Srebro. A pac-bayesian approach to spectrally-normalized margin bounds for neural networks. *arXiv preprint arXiv:1707.09564*, 2017.

A. Nguyen, J. Yosinski, and J. Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 427–436, 2015.

R. F. Orlikoff, D. D. Deliyski, R. Baken, and B. C. Watson. Validation of a glottographic measure of vocal attack. *Journal of Voice*, 23(2):164–168, 2009.

N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. Berkay Celik, and A. Swami. Practical black-box attacks against deep learning systems using adversarial examples. *arXiv preprint arXiv:1602.02697*, 2016a.

N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *Security and Privacy (SP), 2016 IEEE Symposium on*, pages 582–597. IEEE, 2016b.

V. Peddinti, G. Chen, D. Povey, and S. Khudanpur. Reverberation robust acoustic modeling using i-vectors with time delay neural networks. In *INTERSPEECH*, 2015.

F. A. Petitcolas, R. J. Anderson, and M. G. Kuhn. Information hiding-a survey. *Proceedings of the IEEE*, 87(7):1062–1078, 1999.

G. Pironkov, S. Dupont, and T. Dutoit. Multi-task learning for speech recognition: an overview. In *Proc. of ESANN*, 2016a.

G. Pironkov, S. Dupont, and T. Dutoit. Speaker-aware long short-term memory multi-task learning for speech recognition. In *Proc. of EUSIPCO*, 2016b.

M. Ribeiro, K. Grolinger, and M. A. Capretz. Mlaas: Machine learning as a service. In *Machine Learning and Applications (ICMLA), 2015 IEEE 14th International Conference on*, pages 896–902. IEEE, 2015.

I. Rosenfelder, J. Fruehwald, K. Evanini, S. Seyfarth, K. Gorman, H. Prichard, and J. Yuan. Fave (forced alignment and vowel extraction). Program suite v1.2.2 10.5281/zenodo.22281, 2014.

A. Senior and I. Lopez-Moreno. Improving dnn speaker independence with i-vector inputs. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 225–229. IEEE, 2014.

S. Shalev-Shwartz and S. Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 2014.

Y. Sheena, M. Hejná, Y. Adi, and J. Keshet. Automatic measurement of pre-aspiration. *arXiv preprint arXiv:1704.01653*, 2017.

K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

M. Sonderegger and J. Keshet. Automatic discriminative measurement of voice onset time. *Journal of the Acoustical Society of America*, pages 3965–3979, 2012.

I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

Y. Taigman, M. Yang, M. Ranzato, and L. Wolf. Web-scale training for face identification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2746–2754, 2015.

Z. Tang, L. Li, and D. Wang. Multi-task recurrent model for speech and speaker recognition. In *Proc. of APSIPA*, 2016.

F. Yu and V. Koltun. Multi-scale aggregation by dilated convolutions. *ICLR*, 2016.

C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016a.

C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016b.

S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. H. Torr. Conditional random fields as recurrent neural networks. In *ICCV*, pages 1529–1537, 2015.

J. Zhu, R. Kaplan, J. Johnson, and L. Fei-Fei. Hidden: Hiding data with deep networks. In *European Conference on Computer Vision*, pages 682–697. Springer, 2018.

תקציר

למידה עמוקה הגדירה מחדש את תחום הבינה המלאכותית על ידי שבירת רף הביצועים במספר תחומים הידועים לשמצה במורכבותם כגון: זיהוי אובייקטים בתמונה, זיהוי דיבור, תרגום מכונה ועוד. בנוסף, מודלים מסוג זה נמצאו כמודלים בעלי יכולת הכללה טובה עבור דוגמאות אשר נדגמו מאותה התפלגות ששימשה לאימון המודל.

מהאמור לעיל, עולות שתי שאלות מרכזיות: 1) האם מודלים עמוקים יכולים לשמש לצורך מדידת מאפיינים עדינים באופן מדוייק כדי הגדלת נפח מחקר מדעי? 2) בעוד מודלים עמוקים נכנסים לכול תחומי חיינו, החל ממכוניות אוטונומיות ועד עוזרות ועוזרים אישיים, נשאלת השאלה האם המדדים הסטנדרטיים מספקים כדי להעריך את ביצועי המודל? ואיך ניתן לשערך את אמינותם?

במחקר זה, אנחנו חוקרים את שתי שאלות אלו מכמה נקודות מבט. בחלק הראשון של התזה, אנחנו מדגימים כיצד מודלים נויורונים יכולים להיות מיושמים עבור מחקר מדע הדיבור. אנחנו מפתחים כלים בעלי יכולות דיוק גבוהות אשר מבוססים על רשתות ניורונים עמוקות ולמידה מורכבת. מודלים אלו מיושמים לצורך מדידת אורכים של מאפייני דיבור שונים כגון: תנועות, עיצורים, הקדמת שאיפה, מילה שלמה ועוד. אנחנו מראים באופן אמפירי כי אלגוריתמים אלו יכולים להיות מיושמים לביצוע מחקר בלשני ללא כל התרעבות אדם במדידות.

בחלק השני של המחקר, אנחנו חוקרים ומנתחים את העמידות של מודלי למידה עמוקה לדוגמאות יריביות ודוגמאות מחוץ להתפלגות הנלמדת, בתחום של עיבוד דיבור ותמונה. בנוסף, אנחנו מספקים כלים תיאורטיים ומעשיים להערכת חוסר הוודאות של מודלים מסוג זה, ובכך מאפשרים זיהוי ואיתור של דוגמאות אלו.

בחלק השלישי של המחקר, אנחנו מנתחים באופן אמפירי את התכונות והביצועים של מודלי למידה עמוקה אשר משמשים לעיבוד רצפים, כגון: רשתות חוזרות, מידול רצף לרצף, מודלים מבוססי קונבולוציה ועוד.

בחלק האחרון של התזה, אנחנו נבנים על בסיס התוצאות המחקר הקודמות כדי לתכנן ולממש אלגוריתמים יציבים ומאובטחים. תחילה, אנחנו מראים דרך חדשנית להטמעת "סימון מים" על מודל כללי מסוג רשת נויורונים עמוקה, ומנתחים את ביצועיו באופן תיאורטי ומעשי. שנית, אנחנו מראים כיצד ניתן להשתמש ברשתות עמוקות כדי ל"החביא" דגימת דיבור אחת בתוך דגימת דיבור אחרת. תהליך מסוג זה נקרא סטגנוגרפיה בדיבור.

תוכן עניינים

עבודה זו נעשתה בהדרכתו של פרופ׳

יוסי קשת

מן המחלקה למדעי המחשב

של אוניברסיטת בר-אילן

על היציבות של רשתות נוירונים עמוקות ושימושם למדע הדיבור

חיבור לשם קבלת התואר "דוקטור לפילוסופיה"

מאת:

יוסי עדי

המחלקה למדעי המחשב

הוגש לסנט של אוניברסיטת בר-אילן

רמת גן ז' בתמוז ה'תשע"ט