Improving the Goodness of Pronunciation score by using Deep Neural Networks: Single-input Classification & Sequence-to-Sequence Classification

Moises Veleta
B. S. in Electrical Engineering at New Mexico Tech, 2016

Presented to the
Center for Spoken Language Understanding
within the Oregon Health & Science University
School of Medicine
in partial fulfillment of
the requirements for the degree
Master of Science
in
Computer Science & Engineering

June 2018

Center for Spoken Language Understanding
School of Medicine
Oregon Health & Science University

CERTIFICATE OF APPROVAL

This is to certify that the M. S. thesis of

Moises Veleta

has been approved.

Meysam Asgari, Thesis Advisor
Assistant Professor

Alexander Kain, Advisor of Record
Associate Professor

Peter Heeman
Research Associate Professor

John-Paul Hosom, Ph.D.
Sensory Inc

Acknowledgements

I would like to thank Dr. Meysam Asagri for his thoughtful advice and mentorship throughout the course of this thesis and the committee members for their help and advice on this thesis: Dr. Alexander Kain, Dr. Peter Heeman, and Dr. John-Paul Hosom. I would also like to thank the faculty, staff, and student-body of the OHSU CSLU & CSEE departments.

Contents

List of Tables

List of Figures

Abstract

Improving the Goodness of Pronunciation score by using Deep Neural Networks: Single-input Classification & Sequence-to-Sequence Classification

Moises Veleta

Master of Science
Center for Spoken Language Understanding
within the Oregon Health & Science University
School of Medicine

June 2018
Thesis Advisor: Meysam Asgari

This is a comparative study of the Goodness of Pronunciation (GOP) score, a phone pronunciation quality metric, that explores its formulation and evolution. The effectiveness of the GOP score lies predominantly in its two main components: its forced aligner, which produces the expected phone segments, and its phone loop, which produces the observed phone segments. As with the derivatives created since the inception of the GOP score, this thesis explores alternatives to the traditional forced aligner and phone loop by using several Deep Neural Network (DNN) architectures. The two general classes of architectures used, from Deep Learning, are the single-input classifier and the sequence-to-sequence classifier. Along with these architectures are proposed approaches on how to use DNNs within a GOP score. Lastly, a new generalized GOP score, the GOP-ensemble, is proposed to allow users to combine a variety established GOP scores to create a new modular pronunciation score.

Chapter 1

Introduction

The ability to measure the "goodness of pronunciation" is useful to individuals learning a new language and people with speech disorders. For those learning a new language, having pronunciation feedback helps them to speak properly, learn faster, and adjust their accents to a region's custom. Pronunciation feedback should identify specifically which phone - or phones - a person is having difficulties with, so that they can practice on those vocal articulations. Often, native language speakers notice if a person has an accent, however, it may not be easy for them to identify which phones are responsible for the vocal deviations. For those with speech disorders, having the ability to identify which phones to practice speaking helps them to rehabilitate and to improve their speech. In both cases, speech language pathologists (SLPs) are normally employed to perform the phone quality analysis and provide the appropriate feedback, but they can be expensive and in scarce availability to those that live in rural areas.

Aside from feedback and diagnosis, SLPs are often hired to do speech transcription for the development of automatic speech recognition (ASR) systems. These transcriptions consist of short-time frames (usually frames of tens of milliseconds) each labeled by their corresponding phone and phonological features. These transcriptions, apart from being used to train ASR systems, can be used to identify phone mispronunciations. However, a major drawback to SLP transcription is that it is slow and expensive to those in need. These drawbacks may be alleviated with data-driven automatic pronunciation tools.

Generally, ASR is focused on the transcription of audio to identify a word or word sequence rather than analyzing phone pronunciation quality. Aside from this difference, recent improvements in ASR accuracy using Deep Learning have lead to a rise in the use of Deep Neural Networks to discriminatively identify a word or word sequence. In fact, most state-of-the-art ASR systems that are focused on Large Vocabulary Conversational Speech Recognition (LVCSR) have been ensembles of large Deep Neural Networks (DNN). The success of Deep Learning has also extended into other domains such as Image Processing, Object Recognition, Natural Language Processing, and many other fields. Due to the similarity of ASR's and quality of pronunciation's goals, Deep Learning tools show great potential in the scoring of phone pronunciation quality.

Using some concepts from ASR, researchers have already developed several metrics for measuring pronunciation quality including the *Goodness of Pronunciation* (GOP) score and its derivatives. However, the GOP accuracy rates are still not as high as those of ASR. With this in mind, we continue to explore the use of DNNs as components in the comparison of the GOP score.

Chapter 2

Background

The objective of the GOP score is to determine overall phone pronunciation quality. However, speech has inherent and complex dynamics. Intricacies such as two identical phones varying acoustically due to their adjacent contextual phones. This requires ASR to use subphonetic constructs to identify utterances. Since these subphonetic constructs increase word recognition accuracy when included during classification, they are important to briefly discuss. The subphonetic constructs include, but are not limited to, triphones and senones. Triphones are phone models that include information regarding the preceding phone, the current phone, and the subsequent phone. Senones have been described to be “a basic speech unit”, but are also defined as “clustered state-dependent output distributions” (Hwang & Huang, 1992). Senones were originally used in Hidden Markov Models where state-sharing improves performance. However, senones are still useful in other learning algorithms, because they are able to capture information about how the same phone may vary acoustically depending on its neighbors. Thus the phones are not isolated to themselves, but are modified by the adjacent phones.

However, in this paper, we will not use phone-to-subphone mappings and we will only focus on phones, although it is useful to do so to improve accuracy rates, e.g. Word Error Rate. The reason for this approach is to not over complicate the basic phone units already provided to us by the corpus’s transcribers and because our context is limited to intra-word cues only. The corpus used is made of single word utterances which were isolated from possible perturbations, as often seen in continuous speech. Keep in mind that the Kaldi tools we use will include phone models assembled from the mentioned subphonetic units, such as senones, since they are useful in state-of-the-art ASR systems (Povey et al., 2011) (McAuliffe, Socolof, Mihuc, Wagner, & Sonderegger, 2017).

In other applications, specifically LVCSR, it is important to be mindful of the amount of context-dependence there is between an utterance and the speech overall. However, for pronunciation quality, users and systems may impose the constraint of text-dependence. This is because as learners we usually start learning how to pronounce the ”basics” of a language with a small core set of words, namely words that are spoken deliberately and slowly, one at a time. Thus with two sets of speakers, native and non-native, we can figure out which phones speakers are having difficulties with and train them in the basics to prepare them for more advanced vocabulary. Here, is where the GOP measure strays from ASR's overall goal of LVCSR: scoring phone pronunciation quality rather than recognizing words in conversational speech.

Chapter 3

Related Methods

In this chapter, we go over related methods to briefly explore the formulation of GOP score and possible avenues of scoring phone pronunciation quality. First, the original Goodness of Pronunciation score is reviewed to examine the process of determining phone pronunciation quality. Afterwards, GOP score derivatives provide information of how other researchers have improved upon the GOP score. Last, the major components of the GOP score are described and set the stage for the new proposed GOP components.

3.1 Original GOP

The original Goodness of Pronunciation (GOP) score was developed by Witt & Young (S. Witt & Young, 1997). The GOP score, influenced by Bayesian statistics, is the posterior probability of a target phone p, given a sequence of acoustic features O, from among a set of phones Q. In other words, what is the probability that the expected phone is observed with respect to all of the phones that we are observing.

$$GOP(p) \equiv P(p \mid O) = \frac{P(O \mid p)P(p)}{\sum_{q \in Q} P(O \mid q)P(q)} \tag{3.1}$$

Witt & Young give two approximations to simplify the original GOP score. First, the GOP-max score assumes that all phones have an equal probability of occurring ($P(p) = P(q)$), which eliminates all of the prior probability terms from the numerator and denominator. This removal of the priors, or phone probability distributions, makes the system agnostic to future observations from one phone to the next. Second, the GOP-max score eliminates the summation over all phone probabilities, of the denominator, by making the assumption that the phone with the greatest likelihood is only the phone needed to be compared with the expected phone. This assumption retains the most probable phone and if there is mispronunciation, the phone that was detected.

Both of these assumptions reduce the number of calculations needed. For simplicity, we refer to the GOP-max score as the GOP score from now on.

$$GOP(p) = GOP_{max}(p) = \frac{P(O \mid p)}{max_{q \in Q} P(O \mid q)} \tag{3.2}$$

Now, the GOP score calculation can be better understood by dividing it into its numerator and denominator, wherein the numerator's likelihood is determined by *forced alignment* (FA) and the denominator's likelihood is found via a *phone loop* (PL). FA is commonly used in determining the phone boundaries, i.e., where a sequence of frame labels changes from one phone to next. In practice, the FA is the use of a Bakis lattice weighted-finite-state-transducer (WFST) to obtain the phone segmentation of a known utterance. PL, on the other hand, is the frame-wise comparison of all probabilistic phone models, traditionally carried out by Gaussian Mixture Models (GMM) and Hidden Markov Models (HMM)s, for the phone with the greatest probability (Martin & Jurafsky, 2000).

The GOP scores are phone segment quality assessments, which in turn together, make up an assessment for an utterance or a statement. Thus an accumulation of the frame-wise GOP scores is checked against a heuristically set threshold. A phone threshold, acts as a decision point for a good or bad pronunciation, but in reality is a continuous scale.

3.1.1 GOP revised

The GOP score, revised by Witt & Young in their second paper, included in the expression the use of an absolute value of the score, logarithmic scaling, and normalization by the number of frames (NF) for the given phone p (S. M. Witt & Young, 2000).

$$GOP_R(p) = \frac{\mid log(\frac{P(O^{(p)} \mid p)}{max_{q \in Q} P(O^{(p)} \mid q)}) \mid}{NF(p)} \tag{3.3}$$

The absolute value collapses the GOP score to only positive values. This removes the need for both a positive and negative threshold. With absolute value, logarithmic scaling on the fraction of likelihoods maps equal likelihoods to zero and differing likelihoods to a value greater than zero. Frame length normalization helps compare all phone segments in utterance with equal weighting. Thereby neglecting phone duration as a major contributer to the GOP score. In addition to previous mathematical clarifications, Equation 3.3 explicitly shows that $O^{(p)}$ is an acoustic sequence attributed to a target phone p.

Witt & Young also introduce another GOP score component that includes the detection of known mispronunciations. Mispronunciations are usually identified by experts and are common occurrences between two languages whose native speakers are attempting to learn the target language, e.g., the accents of native Spanish speakers first learning to speak English. Such speakers tend to use their native-language articulatory skills to synthesize a target language's utterances. The complex assembly of sequential phones creates difficulties for speakers trying to identify exactly which phones they are mispronouncing. Therefore, the GOP error score introduces the complementary idea that phone pronunciation quality can be also distinguished by assessing typical mispronunciations with the aid of error phone models, $P(p_e)$.

$$GOP_e(p) = \begin{cases} \mid log(1 - P(p_e \mid O^{(p)})) \mid & \text{if}\, p = p_e \\ 0.0 & \text{otherwise} \end{cases} \tag{3.4}$$

Equation 3.5 shows the new GOP score, with explicit error phone detection, that is the combination of Equation 3.3 and Equation 3.4 with a scaling coefficient k. The constant k helps control the contribution made by the error phone models based on their efficiency in a particular utterance.

$$GOP(p) = GOP_R + kGOP_e(P) \tag{3.5}$$

3.1.2 Using the GOP score

The GOP score is a method determining phone pronunciation quality. In the following sections, we discuss how GOP score is used and its limitations.

Thresholds

Witt & Young dictated that the use of the GOP score requires the setting of thresholds for whether a phone was correctly pronounced or not. Thus if a GOP score exceeds the specified phone threshold value it indicates that a phone is mispronounced and a GOP score lower than the threshold signifies that is correctly pronounced. These phone mispronunciation thresholds are sliding values that can be either a single global value or multiple phone-dependent values. Usually, the thresholds are determined by empirical testing and are compared via the Equal Error Rate metric on some developmental corpus (Deng, Hinton, & Kingsbury, 2013). Keep in mind that this paper will not focus on the optimization of GOP thresholds for a given group of speakers or speech domain, but is focused on the improvement of the GOP components in terms of phone classification accuracy.

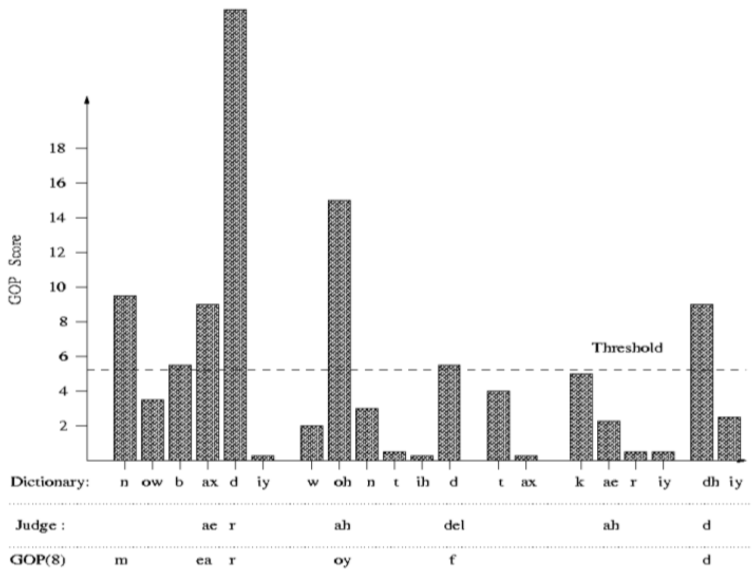

Figure 3.1: GOP global threshold for detecting phone mispronunciation from (S. Witt & Young, 1997)

Average Phone Scoring

The GOP score is an average time-frame score for a target phone p over its segment length in some utterance. This averaging is both beneficial and detrimental. The benefit of averaging is smoothing the score over a segment to remove fluctuations. However, as a consequence, significant outliers can disrupt the quality assessment of a phone segment. Such outliers could be caused by a background sound or hardware system noise.

The GOP score is the pioneering metric for pronunciation quality and has been used as a standard comparison metric. As interest continues, the GOP score remains an important metric to build off of. This leads us to explore derivative works that improve either upon the original GOP score or pose a novel yet similar approach.

3.2 GOP score derivatives

The next sections briefly discuss GOP methods from other pronunciation quality researchers in effort to explore the evolution of the GOP score. Their research has either increased the reliability of the GOP score or has provided an alternative method of measuring the phone pronunciation quality.

3.2.1 Weighted Phone Confidence

Doremalen et al. created a *weighted Phone Confidence* (wPC) score which includes information of competing phones and not only the target phone p_{target}.

$$PC_{p_i}^{p_{targ}} = \frac{1}{t_e - t_b} \sum_{t=t_b}^{t_e} log(\frac{P(O_t \mid p_{target})}{P(O_t \mid p_i) + P(O_t \mid p_{target})} \tag{3.6}$$

Thus the wPC score is done for every competing phone $p_i \in Q$, where Q is the set of observed phones. The wPC scores are used in a logistic classifier to compare their competing phones.

$$wPC^{p_{targ}} = \frac{1}{1 + e^{-\beta_o + \sum_i \beta_i PC_{p_i}^{p_{targ}}}} \tag{3.7}$$

The wPC score is trained for each phone and thus shows which one of the competing phone was correctly pronounced or mispronounced. This method adds information of competing phone that the GOP score ignores, but does require additional computation. However, the wPC had higher accuracy than the original GOP score in terms of Equal Error Rate (EER) over a specific developmental set (Doremalen, Cucchiarini, & Strik, 2013).

3.2.2 Support Vector Machine & Likelihood Ratio

From the pronunciation quality literature is Franco et al.'s *Likelihood Ratio* (LLR), which works much like GOP score, but it focuses on the comparing the phone pronunciations with a correct model and a mispronunciation model directly (Franco, Neumeyer, Kim, & Ronen, 1997). Franco's work inspired the Support Vector Machine developed by Wei et al., which uses the multiple LLR scores to create a vector space model(Wei, Hu, Hu, & Wang, 2009).

$$LLR(o, q) = \frac{1}{d} \sum_{t=t_o}^{t=t_o+d-1} (logP(O_t \mid q, \lambda_M) - logP(O_t \mid q, \lambda_C)) \tag{3.8}$$

With the LLR scores for each target phone, and possible mispronunciation phones, Wei et al. create LLR vectors that are compared in a Support Vector Machine (SVM). SVMs are classifiers which are used in this application to discriminate between phone classes.

However, since there's a correct and incorrect model for each phone the SVM method requires a large amount of training data and computation. Additionally, this specialized training data is difficult and expensive to obtain.

3.2.3 Correct & Incorrect Weight Finite State Transducers GOP

Dudy et al. created a new GOP score that accounts for correct, incorrect, and "open-loop" (lexicon-free) pronunciations using *Weighted Finite State Transducer* (WFST) lattices.

$$GOP-CI(p_i) = \frac{L(\varphi^*_C[b_i : b_{i+1}))}{\alpha L(\varphi^*_{CI}[b_i : b_{i+1})) + (1-\alpha)L(\varphi^*[b_i : b_{i+1}))} \tag{3.9}$$

where

$\varphi^*_C = arg_\varphi max(\mathcal{H} \circ \mathcal{C} \circ \mathcal{L}_C)$

$\varphi^*_{CI} = arg_\varphi max(\mathcal{H} \circ \mathcal{C})$

$\varphi^* = arg_\varphi max(\mathcal{H})$

For equation 3.9, Dudy et al. define the following variables: $[b_i : b_{i+1})$ which are the phone boundaries for some phone p_i with k frames. φ are the Viterbi search path of phone sequences given the individual phone transition lattice $\mathcal{H}$ (HMM), triphone transition lattice $\mathcal{C}$ (context-dependent), and syllable transition lattice $\mathcal{L}$ (lexicon). $\circ$ is a the WFST composition operation, L is the sum of the negative log-likelihoods for the given phone frames, and α is a tuning parameter to balance the weight of lexicon-dependent (correct and incorrect) versus the open loop WFST. (Dudy, Bedrick, Asgari, & Kain, 2017).

Dudy et al. report close performances between GOP-CI and GOP-SVM and reported that both of those methods beat the Lattice-GOP (Song, Liang, & Liu, 2010). Conceptually, the GOP-CI is distinguished from the GOP score by adding the Correct-Incorrect lexicon-latticed WFSTs to the original GOP score. The numerator still uses a Viterbi search on correctly pronounced words, just like a FA, and the denominator includes a Correct-Incorrect WFST along with an open loop. Note that the phone loop and open loop are essentially the same, both offer likelihood scores without taking adjacent frame context into account. Also, note that α parameter scales the trade-off between Correct/Incorrect WFST and context-free WFST depending if the utterances are well-known or if they are unknown, respectively (Dudy et al., 2017).

3.2.4 Deep Neural Network GOP

The GOP score was improved upon by Hu et al. by replacing the GMM with an Deep Neural Network (DNN) (Hu, Qian, & Soong, 2015). The DNN classifier can replace the GMM through the use of it as a likelihood probability $P(p \mid O)$. As with the previous approximations the GOP score uses a simplified denominator and assumes all phones have uniform prior distributions. Also, the simplified denominator uses the phone with the greatest value instead of the sum all of phone probabilities (argmax). Equation (3.10) is an altered form of the GOP method with a uniform prior distribution for phones.

$$GOP_{max}(p) \approx log \frac{p(\mathbf{o} \mid p)}{max_{q \in Q}\, p(\mathbf{o} \mid q)} \tag{3.10}$$

With equation (3.10), we replace its likelihood $p(\mathbf{o} \mid p)$ with the following derivation, which expands on the HMM notation.

$$p(\mathbf{o} \mid p\,; t_s, t_e) \approx argmax_s\, p(\mathbf{o}, s \mid p\,; t_s, t_e) = \pi_{s_{t_s}} \prod_{t=t_s+1}^{t_e} A_{s_{t-1}s_t} \prod_{t=t_s}^{t_e} p(\mathbf{o}_t \mid s_t) \tag{3.11}$$

$$\approx \prod_{t=t_s}^{t_e} p(\mathbf{o}_t \mid s_t) \tag{3.12}$$

$$= \prod_{t=t_s}^{t_e} \frac{p(s_t \mid \mathbf{o}_t)p(\mathbf{o}_t)}{p(s_t)} \tag{3.13}$$

Here t is time, t_e is end time, t_s is start time, π is the initial state probabilities within the HMM, s_t is the state at time t, A is a state-to-state transition matrix, and $p(s_t \mid \mathbf{o}_t)$ is Softmax output of a trained phone DNN.

By taking the *log* of equation (3.13), we have the following log likelihood.

$$log\, p(\mathbf{o} \mid p; t_s, t_e) \approx \sum_{t=t_s}^{t_e} log \frac{p(s_t \mid \mathbf{o}_t)}{p(s_t)} \tag{3.14}$$

Using an DNN in place of a GMM improves discrimination capability of the phone models (Hu et al., 2015). Hu et al. also create a second version of DNN-GOP which focuses on the triphone context instead of monophone. The authors state that DNN-GOP triphone version lowers the EER only slightly. However, Hu et al. also show that the DNN-GOP score outperforms the original GOP score.

Another variant of the DNN-GOP includes a hidden layer that outputs a categorical classification of phonological speech features if present in the acoustic signal. These additional phonological features may aid with the diagnoses of speech disorders (Arora, Lahiri, & Reetz, 2017). Thus in combination with the GOP metric, they can help determine the quality of pronunciation and provide additional articulation information.

These GOP derivatives have helped the performance of the GOP score by either increasing component performance or proposing a novel method for GOP scoring. Next, we will elaborate more on the components that have been used in the GOP score as well as potential modifications to them to increase overall performance.

3.3 Phone Models

Phone models are typically implemented to do phone-by-frame classification in some constrained lattice, depending on the availability of word context. The (time) frames are usually 25 millisecond (ms) windows of raw audio or pre-processed frequency features with a frame overlap of about 10 ms. The performance of features varies depending on the application; for this study we only evaluated Mel-frequency cepstral coefficients (MFCCs), however there are a wide variety of pre-processed acoustic features that have been compared in other studies (Shrawankar & Thakare, 2013), (Dave, 2013), (Kurzekar, Deshmukh, Waghmare, & Shrishrimal, 2014). For each frame, phone labels are needed to classify which phones construct an utterance. This is usually done by a professionally trained transcriber or SLP. With these annotated frames, which are commonly called the "gold standard", phone models can be trained and tested. Two of the more popular methods of phone classification, that we are interested in, are: Hidden Markov Models with Gaussian Mixture Models (HMM-GMM) and Deep Neural Networks (DNN) (Martin & Jurafsky, 2000), (Hinton et al., 2012).

3.3.1 Hidden Markov Models-Gaussian Mixture Models

HMM-GMMs have been a well-known tool in the field of Automatic Speech Recognition for a few decades now. Generic ASR HMM-GMM systems consist of two main components: the Hidden Markov Model (HMM) and the Gaussian Mixture Model (GMM). We will briefly discuss these two components to have a better understanding of how they work for the GOP score.

Hidden Markov Models

The HMM is a statistical method for tracking a sequence of hidden states given a sequence of observations. In this application, the states represent phones and the observations are the acoustic features. Depending on the utterance being modeled, an HMM is a directed graph that uses observation and transition probabilities to represent the probability of occupying any one state at time t and transitioning to the next state at time $t + 1$ (possibly the same state). However, the observation and transition probabilities are found by training on a domain-specific corpus of acoustic feature frames and labels. For observation probabilities, an HMM requires the use of some probability distributions, which are typically GMMs. Research has shown that Gaussian Mixture

Models are good at simulating probability density functions of many different phenomena (Peel & McLachlan, 2000).

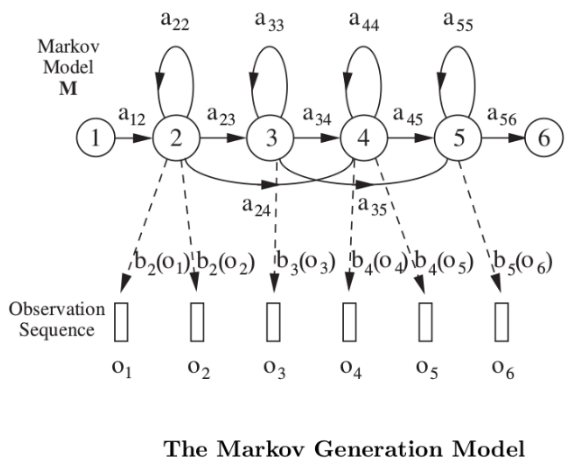

Figure 3.2: Hidden Markov Model from (Young et al., 2002)

Gaussian Mixture Models

GMMs are a weighted sum of Gaussian density functions that each have their own vector mean and a covariance matrix. The number of Gaussian distributions needed to model a distribution varies based on the phenomenon being studied, but optimization for this has been studied (Huang, Peng, & Zhang, 2013). The means and elements of the covariance matrices are estimated from the labeled data via the use of an iterative method known as the Expectation-Maximization (EM) algorithm.

HMM-GMMs are trained using the Baum-Welch algorithm, a modified version of the EM algorithm (Bilmes et al., 1998). This iterative algorithm estimates the transition probabilities in the Expectation step and the GMM parameters along with the transition probabilities in the Maximization step (Shimodaira & Renals, 2017). HMM-GMMs are used with search algorithms such as the Viterbi algorithm (Young et al., 2002). For the purposes of finding a sequence of states, the Viterbi search algorithm uses observation and transition probabilities to estimate the most probable state sequence in a greedy search approach. The Viterbi search algorithm has been used widely in Large-Vocabulary Conversational Speech Recognition (LVCSR) and in the Forced Alignment approach. The Kaldi Speech Recognition Toolkit is a useful package for learning, training, and using HMM-GMMs and FAs and can be said to be a successor of the Hidden Markov Model Toolkit (Povey et al., 2011), (Young et al., 2002).

3.4 Forced Alignment

Traditionally, a forced aligner relies on phone models, HMM-GMMs, to produce phone segmentation for a known utterance. Those segmentation boundaries determine where phones start and end for a known sequence of word-to-phone labels.

The forced aligner commonly concatenates individual phone HMM-GMMs together to form a Bakis phone lattice. The Bakis lattice allows the known set of phones to fit an utterance with only forward transitions. Therefore the main task for the forced aligner is where in time does it transition from one phone state to the next phone state.

Research has shown that forced aligner's accuracy rates are dependent on the use of either mono-phone versus tri-phone states. These triphones are use of the same phone or different phones for each of the three states. The triphone state context is either intra-word or intra-sentence. Intra-word considers phones within the same word. Whereas intra-sentence considers phones from adjacent words within the same sentence, which can work as n-grams or language models, but for phones instead of words. Forced aligners are comparable to human transcription, but are more time efficient and less expensive (McAuliffe et al., 2017).

Similar to FAs, the number of states in a phone model also affect the accuracy of HMM-GMMs. The states determine if and how the HMMs take context into account. The use of a single phone state model may require less training, but it cannot use the information of being in a previous state, other than the transition probability. This removes important contextual information, but removes the bias of a limited corpus. Whereas a triphone model can use contextual information from a previous phone, however this requires more training on a larger corpus. Triphone models with the same phone are used to prevent one frame segments of phone predictions (Milne, 2015).

The phone models used in the phone loops are sensitive to a number of training parameters that substantially affect the performance of GOP scoring.

3.5 Deep Neural Networks

Deep Neural Networks (DNN) are becoming an increasingly popular set of tools from the domains of Machine Learning and of Artificial Intelligence (Schmidhuber, 2015). Deep, in this sense, stands for having a neural network that has more than one layer of nodes, thus there are multiple "hidden" layers (Deng et al., 2013). Also, due to new methods of training and architecture development, deeper models can result in more powerful classification models. However, deeper DNNs usually require more data and training time. We will elaborate more on the main components of a deep neural network: layer types, single-vs-sequential, and auxiliary modifications.

Layer types

Depending on the application, DNN can have specialized layers to improve the performance of a classifier. The layers include the following layer types: convolutional, pooling, fully-connected, recurrent, residual blocks, and Softmax layer.

The convolutional layer convolves n filters with spatially adjacent values to produce a new set of values that are parameterized by learned weights (Schmidhuber, 2015). An example from image processing is a filter that convolves horizontal lines versus a filter that convolves vertical lines, thereby finding those features: horizontal lines or vertical lines. Thus convolutional layers can find complex features wherein subsequent layers parameterize low-level features into complex objects, e.g. a square made from parallel sets of horizontal and vertical lines. One frequent consequence of these convolutional layers with multiple filters is an increased number of features.

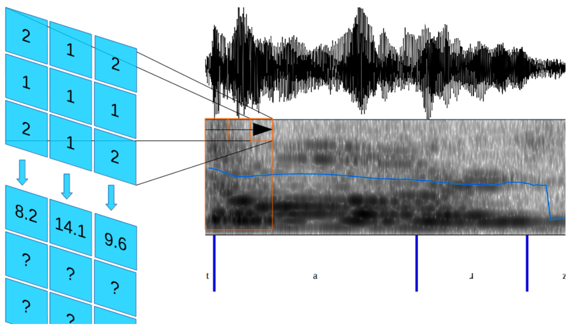

Figure 3.3: Convolution with a filter on spectral features

The remedy lies with pooling layers; which provide a solution by selecting subset of the input values by “pooling” values together. Pooling can be as simple as using the maximum argument or the mean of the arguments in a set of spatially adjacent values. Pooling can be any function that a user deems appropriate for the target feature and is essentially a method of down-sampling features (Scherer, Müller, & Behnke, 2010).

Fully-connected layers are just that: *fully-connected* layers wherein each node in the layer is connected to the next layer’s node. These layers can be used throughout a typical DNN.

The recurrent layer itself can define a whole DNN as an Recurrent Neural Neural (RNN). The typical recurrent layer utilizes a memory mechanism to use previous input context to produce its output. The output can be a single value or a sequence of values, since the memory mechanism of the recurrent layer can be used as an informational input. The main recurrent layer type

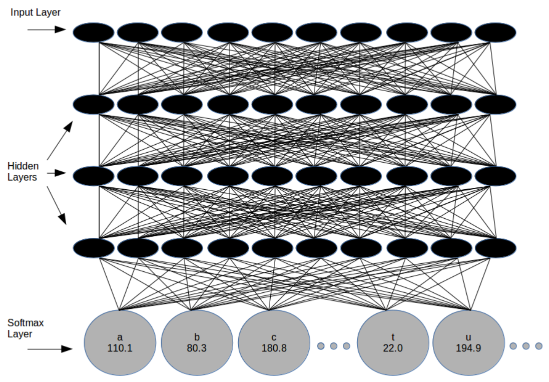

Figure 3.4: Fully-connected DNN

used is the Long-Short-Term Memory layer, which is similar to the Gated Recurrent Unit. While both are different, they both use internal memory and non-linear activation functions to control their learning ability (Chung, Junyoung and Gulcehre, Caglar and Cho, KyungHyun and Bengio, Yoshua, 2014).

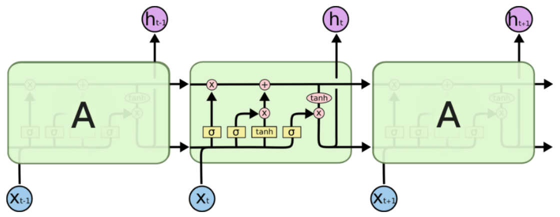

Figure 3.5: Long-Short Term Memory Machine from (Olah, 2015b)

Also, recurrent layers can be used to take information in both a forward direction and a backward direction (Schuster & Paliwal, 1997). An example is the sentence ”the boy threw the ball”. Here an initial layer reads in the words one at a time and propagates their context to the next cell. Additionally, the recurrent layer can read that as ”ball the threw boy the” and track context in reverse order.

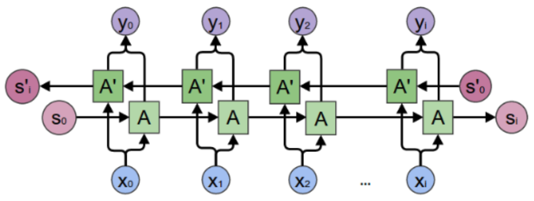

Figure 3.6: Bidirectional LSTM from (Olah, 2015a)

The residual block is not a layer per se, but a group of layers that has a special bypass connection to their output. Essentially, residual blocks allow DNNs to be deeper as they prevent the vanishing gradient problem from occurring (He, Zhang, Ren, & Sun, 2016).

$$y = f(x) + x$$

Here y is the output of the block and f is the function of the block and x the input to the block. Thus the output is the input values and layer block transformation upon them.

The Softmax layer is traditionally the last layer in a DNN classifier. The Softmax layer uses the following formula to classify an input to a phone.

$$p(y = j \mid x) = \frac{e^{x^T w_j}}{\sum_{k=1}^{k} e^{x^T w_k}}$$

Here y is the output being class label j, x^T is the transposed input feature vector, and w_k represents the weights of the last layer according to the class k. Thus, the Softmax layer contains a probability value for each phone.

Single-vs-sequential classification

Single-vs-sequential classification is simply the difference between classifying a single isolated acoustic observation or a sequence of acoustic observations together. Layerwise, it is the use of recurrent layers or not. The advantage and disadvantage of single-frame classification is that it can classify independently, thus it can remove the 'smoothing' effect from sequence classification. However, for most utterances, phones can last several frames and are heavily dependent on context. Therefore, it is beneficial to use intra-word context, when possible.

Auxiliary modifications

Auxiliary hyperparameters include components such as dropout, batch normalization, activation functions and optimization method. Dropout randomly drops nodes during the training routine. Dropout is commonly thought of as a regularization technique, since random node subsets train during each training batch (Gal & Ghahramani, 2016). Batch normalization acts like a layer that transforms the input to have a mean of 0 and a standard deviation of 1 (Ioffe & Szegedy, 2015). Non-linear activations give layers the ability to differentiate between conditions, i.e. whether a node is activated or not, in a non-linear manner. These activation functions also vary depending on time of training desired or level of accuracy needed during training process of back-propagation (Zeiler et al., 2013). The optimization method is the method of finding the local minima with respect to the cost function and starting parameters (Goodfellow, Bengio, & Courville, 2016). Auxiliary modifications on a DNN can potentially decrease training time and/or improved performance.

Due to the current performance of the GOP's components, there is a need to find better components in effect to produce a more reliable GOP score. The next chapter will elaborate on such components and some approaches on using them in a new GOP score.

Chapter 4

Proposed GOP Components & Approaches

As shown previously, the GOP score is dependent on likelihood ratio score of the expected phone and the most probable phone using a FA and a phone loop, respectively. While there already exists a couple of DNN architectures used in a GOP score, specifically Hu et al.'s GOP-DNN, we aim to replace their simple multi-layered fully-connected DNN with one that uses convolutional and recurrent layers to improve phone-level classification. As a consequence, these DNN architectures are an improvement upon the FA, phone loop component, and provide additional scoring benefits. As with Dudy et al.'s GOP-CI score wherein the denominator used Correct and Incorrect WFSTs, these DNN architectures can also be used in various focused approaches. Lastly, we propose creating a new ensemble method that can add together GOP scores with various components to improve the measure of phone pronunciation.

4.0.1 Convolutional Neural Networks

First, we propose using Convolutional Neural Networks (CNNs) since they have been shown to out-perform HMM-GMMs and DNNs in acoustic modeling in several studies (Hinton et al., 2012), (Abdel-Hamid, Mohamed, Jiang, & Penn, 2012), (Sainath, Mohamed, Kingsbury, & Ramabhadran, 2013), and (Deng et al., 2013). Here, the aim is to show that CNNs are better candidates for the open loop component of the GOP score in comparison to fully-connected DNNs. The CNN improvement is attributed to their use of spatial convolutional with randomly chosen filters to find random spatial patterns. Additionally, deep CNNs, with multiple convolutional layers, can parameterize patterns over larger regions of a feature frame. In our experiments, the CNNs were designed and modified to improve upon on their ability to predict frame phone labels. The CNNs were evaluated by layer depth, number of nodes, dropout rate, use of batch normalization, use of residual connections, and input training data format.

The CNN architectures were explored initially in a grid-search, or parameter sweep, fashion, then the best performers were modified to find performance successors. The initial layers were convolutional with filters numbering 8, 16, 32, 64, or 128. The convolutions were conducted with use of 3x3 windows with and without max pooling or batch normalization after each layer. Max pooling varied on architectures depending on layer depth. Since max pooling reduces the number of output features, deep architectures used less layers of pooling to retain enough features for subsequent layers. If an activation function was used in a convolutional layer, it was the Rectified Linear Unit (ReLU). Afterwards, fully-connected layers of 1 to 3 layers were fitted with 16, 32, 64, 128, 256, 512, 1024, and 2048 nodes each, sometimes followed by ReLU activation after each layer with the exception of the last layer which used a Softmax layer. The dropout rate was globally applied and it ranged from 0.2 to 0.8. The CNNs were trained with Early Stopping, so that if the loss function score did not make any significant improvements for up to 3 epochs the training phase was ended and the best performing model was returned. The training data format either focused on phone-frame counts or word-based counts.

4.0.2 Recurrent Neural Networks

Next, we proposed using Recurrent Neural Networks (RNNs), because they have been shown to out-perform DNNs, as is shown in a few papers (Li, Mohamed, Zweig, & Gong, 2016), (Nussbaum-Thom, Cui, Ramabhadran, & Goel, 2016), and,(Zhang et al., 2017). In our paper, RNNs were used as sequence-to-sequence frame classifiers that take into account contextual information. The RNNs are able to capture contextual information by taking an input of a sequence of samples which can be partially stored in their internal memory. The RNN architectures that were evaluated varied by sequence length, convolutional layer depth, recurrent layer depth, number of nodes, dropout rate, bidirectionality, attention, batch normalization, residual connections, and training data format.

These RNN architectures were also initially explored in a grid-search fashion with the best performers modified to find performance successors. The RNNs contained a range of convolutional layers from 1 to 20, with either 8, 16, 32, or 64 filters. The convolutional layers used the same convolution filter size (3x3), optional max pooling, optional residual connections, optional batch normalization, and optional activation functions. Also, tested were the RNNs use of only forward feed LSTMs or both forward and backward feed LSTMs, which is better known as bidirectional LSTM or BiLSTM. The attempted sequence lengths varied from 16, 32, 64 to 128. The DNNs were trained with Early Stopping. The training data format either focused on phone-frame counts or word-based counts, which will be explained in greater detail in the following chapter.

Also, an attention BiLSTM was evaluated. An attention BiLSTM is an LSTM that can read and write from a fixed-length memory vector. Here the attention mechanism only reads or writes from specific locations of the memory vector instead of using all of the memory vector like the traditional LSTM (Bahdanau, Cho, & Bengio, 2014).

4.0.3 Classification Approaches

Besides the proposed DNN architectures, we propose classifying by several approaches: Softmax classification, Word Focus classification, Iterative Search classification, and Overall Segment classification. By default Softmax classification is the normal method of using the RNNs, wherein the Softmax layer produces phones probabilities for every phone and the greatest phone probability is the predicted phone. Word Focus resembles FA by reducing the set of all phones to only the expected word's set of phones. Iterative Search is the comparison of all words in the vocabulary by their sum of likelihoods to find the most probable word. Overall Segment classification is a binary score that compares the expected phone and the most popular phone in that expected phone's segment. The next sections go further in detail on the latter three.

4.0.4 Word Focus

In our efforts to improve phone recognition, we can take advantage of the fact that this is a text-dependent process and thus know the target word. Traditionally, RNNs use a Softmax layer to compare each of the phone's probabilities and then return the phone with the greatest value. With this in mind, we narrow our scope by looking at the known word's phones. In other words, we restrict the set of phones being compared to only the target word's phones. We refer to this process as Word Focus.

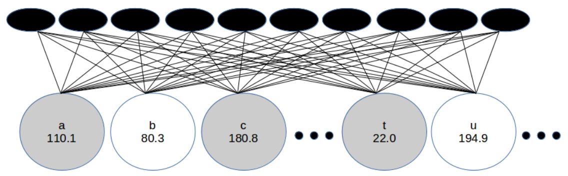

Figure 4.1: Word Focus on Softmax, example word: cat (focused phones are in grey).

One negative consequence is that by using Word Focus we will miss any phones which have been inserted or substituted in the speaker's utterance. In other words, if the phones are outside of the set of selected phones, they will have no probability of being detected. This constraint on phones is fairly similar to what the FA does, but instead of using a Bakis lattice and Viterbi search, we use a RNN and a set of phones which can be predicted in any order.

Multiple Pronunciations

Building off of Word Focus, we can focus on multiple common pronunciations or mispronunciations of a target word. Word Focus with RNNs allows us to add Correct and Incorrect components, as was done by Dudy et al. with the Correct & Incorrect WFSTs in their paper.

Iterative Search

Iterative Search is using Word Focus on the entire vocabulary as an approach to predict if the correct word was pronounced. This is done by enumerating through all the words in the vocabulary and comparing them to each other by their overall utterance score. The overall utterance score is the sum over all individual frame probabilities or Softmax scores. The Softmax score is already a relative scoring method between all phones in a set, but by using the sum of Softmax scores we have created a relative scoring method for words in our vocabulary.

4.0.5 Overall Segment Scoring

In a different line of thought, we aim to simplify the method of scoring a target phone based off of the segments established by the numerator, classically a FA. In this approach, we binarize the correct pronunciation of each phone segment as correct or incorrect based on the highest appearing phone in that segment. If the most popular phone matches the numerator's phone, then it is correct, otherwise it is incorrect. Overall Segment Scoring approach can save computation time as it is a coarse phone quality assessment.

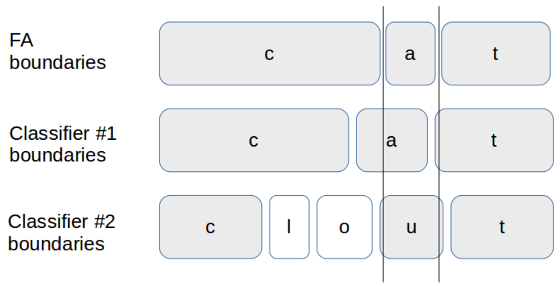

Figure 4.2: Overall Segment Scoring, word predicted by different classifiers, example word: cat (popular phones are in grey).

4.0.6 GOP-Phone Classification Ensemble

Our last proposal is a new score which was inspired by the many modifications to the original GOP score. The aim is to develop a modular system that can include new components simply by adding them as weighted fractions with some appropriate scaling coefficients. The GOP-Ensemble is as follows:

$$GOP_E(p_i) = \sum_{k=0}^{K} \alpha_k \frac{N_s(f_C[b_i : b_{i+1}))}{N_s(f^k_{CIO}[b_i : b_{i+1}))} \tag{4.1}$$

where

$\sum_{i=0}^{K} \alpha_k = 1$

N_s is normalization over the phone frames from the interval $[b_i : b_{i+1})$, which may include taking an absolute value and doing logarithmic scaling. N_s along with the weighting coefficient α_n, allows each component to be properly weighted in terms of the overall score. The user can change α_k coefficients based on the effectiveness of that component for some agreed upon word or phone segment. fs are the phone classifiers which can be WFSTs, DNNs, or some other components with subscripts that denote whether they are Correct-C, Incorrect-I, or Open-O. This modular approach allows for complex adjustments. An example being if WFSTs are better than the DNNs (in the denominator) at classifying some vowels, then more weight could be given to the WFST components. This is also dependent on the FA's or RNN's (in the numerator) ability to identify phone boundaries.

For a more concrete example, take the use of the components given in the GOP-CI and add both a CNN and an RNN.

$$GOP_E(p_i) = \alpha_0 \frac{l(R_C[b : b_{i+1})}{l(RNN_{CI}[b : i : b_{i+1}))} + \alpha_1 \frac{L(\varphi^*_C[b_i : b_{i+1})}{\beta_0 L(\varphi^*_{CI}[b_i : b_{i+1})) + \beta_1 L(\varphi^*[b_i : b_{i+1}))} \tag{4.2}$$

where

$\sum_{i=0}^{N} \alpha_n = 1,$

$\varphi^*_C = arg_\varphi max(\mathcal{H} \circ \mathcal{C} \circ \mathcal{L}_C)$

$\varphi^*_{CI} = arg_\varphi max(\mathcal{H} \circ \mathcal{C})$

$\varphi^* = arg_\varphi max(\mathcal{H})$

As before, the $[b_i : b_{i+1})$ are the phone boundaries for p_i for some integer k frames. φ are the Viterbi search path of phone sequences given the individual phone transition lattice $\mathcal{H}$ (HMM), triphone transition lattice $\mathcal{C}$ (context-dependent), and syllable transition lattice $\mathcal{L}$ (lexicon). $\circ$ is a the WFST composition operation, L is the sum of the negative log-likelihoods for given phone frames, α is a tuning parameter, and l is a normalizing summation of the DNN's Softmax output. Thus this example would be able to benefit from these already tested components, but of course would need to be optimized on a development set.

Ensemble systems in ASR have been shown to outperform single components systems. One notable example is from Xiong et al. who use a combination of acoustic models and language models to build a LVCSR system that achieved "human parity" in speech recognition (Xiong et al., 2016). In another paper by Lee and Siniscalchi, the authors discuss the advantages of using modular systems for ASR, specifically with Automatic Speech Attribute Transcription which is a "modular strategy so that various researchers can collaborate by contributing their best detectors or knowledge modules to plug-n-play into the overall system design" (Lee & Siniscalchi, 2013). In the end, an open-source, modular tool would be useful for both researchers and for at-home volunteers willing to improve GOP scoring.

Chapter 5

Experiments & Results

In this paper we will not explicitly go through the entire process of testing of the GOP score, but will instead focus on testing on the proposed components and approaches. These components' improvements should propagate improvement to the overall performance of a GOP score. In the following experiments we look at the performance of the DNN versus the CNN for single-input classification (open loop) and then look at forced alignment versus RNN's sequence-to-sequence classification. Aside from testing phone-level frame accuracy, we also investigate the proposed RNN approaches.

5.1 Setup

5.1.1 Hardware & Software

For DNN training, a Slurm-managed cluster was used that had a GPU node containing a NVidia M40. The training duration of each DNN varied based on the size and complexity of the DNN architecture. Also the data formats affected training times due to their different sizes. The DNN training was done using the Python software package, Keras. Specifically it was keras-gpu with a Tensorflow backend, both installed using Anaconda, the Python package manager.

The FA was created using the Montreal Forced Aligner (MFA) based on the Kaldi Automatic Speech Recognition Toolkit. The MFA uses 40 iterations of monophone training, which is followed by 35 iterations of triphone training. For more information pertaining to the MFA please look at McAuliffe et al.'s paper (McAuliffe et al., 2017).

5.1.2 Data

The Oregon Health & Science University (OHSU) Child Pronunciation corpus was used and is a small single-word-per-file speech corpus. The corpus's speakers are made up of 87 children, ages

4-11, with 43 typical speakers and 44 atypical speakers. There are about 53 different words per child. For more information pertaining to the OHSU Child Pronunciation corpus, please take a look at Dudy et al.'s paper (Dudy et al., 2017)

Training & Testing Data Formats

Two types of training and testing corpora were used: phone-based or word-based. The phone-based corpus is formed using phone-frame counts which exceed some threshold value. These frames can come from any word in the corpus as long as they have the same matching phone label. On the other hand, the word-based corpus is word-dependent and requires that the words and phones match. Therefore, word-based corpus is formed using word counts that exceed threshold value. Training and testing was done with almost 9-to-1 partition ratio. Keep in mind, these are subsets of the OHSU Child Pronunciation corpus.

Phone-based corpora are mainly used to evaluate the open loop classifiers or context-independent DNNs. Word-based corpora were used to evaluate the sequence-to-sequence classifiers since they preserve context-dependency between sequences of frames. However, we did test a few open loop classifiers on the word-based corpora and a few sequence classifiers on the phone-based corpora.

1. For the Word-based 10 corpus, there were 53 words that had 10 or more occurrences with the same phone sequence. Therefore, for a word to be used in our restricted corpus, it had to appear at least 10 times. The phone class count was 38 with 132840 frames in the training set and 5546 in the test.
2. For the Word-based 25 corpus, there were 48 words that had 25 or more occurrences with the same phone sequence. The phone class count was 38 with 120171 frames in the training set and 9088 in the test.
3. For the Word-based 30 corpus, there was 41 words that had 25 or more occurrences with the same phone sequence. The phone class count was 36 with 108213 frames in the training set and 8293 in the test.
4. For the Phone-based 4000 corpus, there were 26 phone classes with total frame counts of 193391 and 24249 for train and test sets, respectively.
5. For the Phone-based 8000 corpus, there were 9 phone classes with total frame counts of 98382 and 11942 for train and test sets, respectively.

Notice, the difference in phone class counts and frame counts between these various corpora. These differences did significantly affect the frame-level accuracy of the classifiers.

Table 5.1: Train and Test Corpora used

Corpus	Phone Classes	Train Frames Count	Test Frames Count	Words
Word-based 10	**38**	**132840**	**5446**	**53**
Word-based 25	**38**	**120171**	**9088**	**48**
Word-based 30	**36**	**108213**	**8293**	**41**
phone-based 4000	**26**	**193391**	**24249**	**-**
phone-based 8000	**9**	**98382**	**11942**	**-**

5.1.3 Feature Assembly

From the previously described corpora, the feature inputs were assembled to be a matrix-per-label feature-frame with dimensions 5-by-26. The columns contained 26 Mel Frequency Cepstrum Coefficients for each label frame. Each feature-frame was constructed with two adjacent context frames before the target frame and two after. Thus it was a 5 frame matrix with 26 MFCCs. The reason for the context is to aid in classification by providing spatial information. The MFCC windows are 25ms overlapping every 10ms. There were no speaker transformations done for the data.

5.2 Component Results

The results are separated by the component being evaluated. DNN architectures show, at most, the top 3 scoring configurations. The configurations' descriptions precede the results, with each DNN's configuration denoted in their names in their corresponding tables.

5.2.1 Single-input classifiers

In this section, we compare Deng et al.'s fully-connected NNs against a variational of CNNs. These DNN architectures performance were individually assessed with their corresponding input training data type. There were no speaker transformations done to the data beforehand other than what is done by the DNNs themselves. Accuracy here is a corpus's phone-frame accuracy, wherein each frame in the test corpus is tested to be either correct or incorrect phone and the percentage correct is reported as the accuracy.

$$Accuracy = \frac{number\, of\, correct\, frames}{total\, number\, of\, frames}$$

GOP-DNN's full-connected Neural Networks

First, we look at Deng et al.'s set of DNNs, from the GOP-DNN score, which only had fully-connected NNs without the convolutional layers.

The Deng's NN dense, DND, series include fully-connected layers ranging from 1 to 6. The layers each contain 1024 or 2048 nodes per layer with Relu activation functions. Additionally, a global Dropout rate was assigned to be either 0.2, 0.4, 0.6, or 0.8. Note that "xN" signifies some integer number of layers (times N). Here E is the number of Epochs trained. The max number of Epochs was 40, but training stopped early (Early Stopping), if the DNN did not improve in three consecutive epochs.

Table 5.2: DNN phone-based 8000

DNN architecture	**Accuracy**
DND2048relu_x2 DO0.4 E21	**0.531**
DND2048relu_x2 DO0.6 E15	**0.48**
DND2048relu_x6 DO0.6 E14	**0.467**

We can see that the best accuracy for a fully-connected NN is about 53%. Next, we will show the results for various CNN architectures.

Proposed CNNs

The CNNs trained and tested started off shallow with convolutional layers using only a few filters. The CNN architectures are labeled with the following properties:

1. CP - Convolution and Pooling Layers with either 8, 16, 32, or 64 filters,
2. DND - Up to three layers Dense (fully-connected) layers. Layers had either 16, 32, 64, 128, 256, or 512 nodes and an optional ReLU activation.
3. DO - Global Dropout rates ranging from 0.0 to 0.8,
4. E - Epochs until loss function stops improving for up to 3 epochs (Early Stopping), where 40 epochs is the total limit.

Table 5.3: CNN phone-based 8000 & 4000

CNN architecture 8000	**Accuracy**
CP16 DND128relu_32 DO0.4 E14	**0.544**
CP16 DND256relu_64 DO0.4 E23	**0.542**
CP8 DND128relu_32 DO0.3 E20	**0.536**
CNN architecture 4000	
CP8 DND128relu_32 DO0.4 E40	**0.313**
CP8 DND256relu_64 DO0.5 E40	**0.298**
CP8 DND128relu_32 DO0.5 E5	**0.287**

Note that this was not an exhaustive search over all possible combinations, but was guided by slightly modifying the best performing architectures. Also, notice the difference between corpora accuracies, for reference see the section on Data (Training & Testing Data Formats) earlier in this Chapter. This difference between accuracies is due to the greater amount of phone classes that the DNN must decide between using the 4000 corpus's 26 phone classes versus the 8000 corpus's 9 phone classes.

After seeing the success of Deep Residual DNNs in Deep Learning literature, a Deep Residual CNN was trained. It included 10 convolutional layers with batch normalization after each residual block and max pooling before the final Softmax layer. Note that training continued until epoch 70 which means it could have been still improving. However, its training time took much longer than the previous stated architectures due to its size and only one was trained and tested for the single-input classifiers.

Table 5.4: Deep Residual phone-based 8000

CNN architecture	**Accuracy**
CP64_x10 DO0.8 E70	**0.60**

Additionally, CNNs are trained on a Word-based corpus wherein only words with at least 30 appearance are used.

Table 5.5: CNN Word-based 30

CNN architecture	**Accuracy**
CP16 DND64relu_16 DO0.4 E14	**0.295**
CP16 DND128relu_32 DO0.6 E12	**0.295**
CP8 DND128relu_32 DO0.6 E10	**0.292**

This last table shows the difference in using the single-input classifiers on word-based versus phone-based corpora to acknowledge their relative performance. This comparison is useful since next we compare sequence-to-sequence classifiers, which require a Word-based corpus. In practice, words will not be sets of pure phones, but whole words. To recap the performances, we saw that Deng et al.'s best DNN had an accuracy of 53% versus the Residual CNN with 10 layers had an accuracy of 60%.

5.2.2 Forced Alignment & Sequence-to-Sequence classifiers

In this section, we are compare the FA method versus RNNs in terms of frame accuracy. We also use Word Focus to take advantage of knowing the expected word (and its phones) beforehand to produce a focused accuracy.

Montreal FA

The following results show how the Montreal Forced Aligner (MFA) performed on the OHSU Child Pronunciation corpus. The MFA was evaluated on the Word-based corpora: 10, 25, and 30. In the evaluation of the MFA, we considered only words without detectable phone Insertions, Deletions, Substitutions (IDS) for the Average time accuracies, meaning if they had an IDS they were not in the average. Otherwise for the words with phone IDS we only used a binary classification of whether a word had an IDS, but not how many phone IDS the word contained.

Table 5.6: MFA alignments Word-based 30

MFA aligned 30	**Train Set**	**Test Set**
Avg time difference (%)	**29.3%**	**33.6%**
Std time difference (%)	**15.9%**	**15.3%**
Average time accuracy (%)	**70.7%**	**66.4%**
Insertions, deletions, substitutions	**35 out of 1704, 2.05%**	**1 out of 128, 0.78%**

Table 5.7: MFA alignments Word-based 25

MFA aligned 25	**Train Set**	**Test Set**
Avg time difference (%)	**28.4%**	**26.2%**
Std time difference (%)	**16.7%**	**3.4%**
Average time accuracy (%)	**71.6%**	**73.8%**
Insertions, deletions, substitutions	**91 out of 1937, 4.7%**	**70 out of 73, 95.9%**

Table 5.8: MFA alignments Word-based 10

MFA aligned 30	**Train Set**	**Test Set**
Avg time difference (%)	**28.7%**	**22.5%**
Std time difference (%)	**17%**	**5.9%**
Average time accuracy (%)	**71.3%**	**77.5%**
Insertions, deletions, substitutions	**91 out of 2103, 4.33%**	**3 out of 77, 3.89%**

Note that the Word-based corpus 25, has many IDS and its test set accuracy only considers 3 samples. As for the other Corpora, they contain only 1 or 3 IDS samples and have robust test set accuracies. Thus, the MFA's accuracy on the Word-based 25 corpus is about 74% by extrapolation on its adjacent performances of 66% and 78%.

Proposed RNN

The Recurrent Neural Networks initially featured at least one forward-fed LSTM layer and varied in depth. The RNN architectures were labeled with the following properties:

1. CP - Convolution and Pooling Layers with either 8, 16, 32 filters,
2. FBN - Full Batch Normalization, which is Batch Normalization after every convolutional layer,
3. SS - Sequence Size 32 or 64,
4. DL - Global Dropout rate 0.4 to 0.9,
5. E- Epochs until loss function stops improving for up to 3 epochs (Early Stopping), where 50 epochs is the total limit.

Table 5.9: CRNN Word-based 30

CRNN architecture	**Accuracy**
CP32 FBN SS64 DL0.9 E25	**0.492**
CP8 FBN SS64 DL0.9 E35	**0.482**
CP8 FBN SS64 DL0.5 E28	**0.472**

Table 5.10: CRNN Phone-based 8000 & 4000

CRNN architecture 8000	**Accuracy**
CP8 SS32 DL0.4 E50	**0.744**
CP16 SS32 DL0.5 E25	**0.731**
CP8 SS32 DL0.5 E47	**0.729**
CRNN architecture 4000	
CP8 SS64 DL0.6 E46	**0.605**
CP16 SS64 DL0.6 E32	**0.599**
CP32 SS64 DL0.5 E31	**0.591**

These initial RNNs did not perform as well as the MFA, but show how smaller RNNs perform on both types of corporas, Word-based 30 and phone-based 8000 & 4000. Afterwards, the forward-fed LSTM was replaced with BiLSTM networks which feature one bidirectional LSTM layer. Double layered BiLSTM were also trained and tested, but did not fair well. The RNN architectures were labeled with the following properties:

1. CP - Convolution and Pooling Layers with either 8, 16, 32, or 64 filters,
2. FBN - Full Batch Normalization,

3. SS - Sequence Size 8,16, 32, or 64 ,
4. DL - Global Dropout rate 0.4 to 0.9,
5. E- Epochs until loss function stops improving for up to 3 epochs (Early Stopping), where 70 epochs is the total limit.

Additionally, we began to use Word Focus which we labeled as Forced Accuracy (Forced Acc.). Forced Acc., is nearly the same as Accuracy, wherein the percentage of all correct test corpus frames are divided by all test corpus frames, but phone labels are constrained by Word Focus. Also note that the following tables show a significant difference in accuracy between shorter and longer sequence sizes, specifically $SS8$ & $SS16$ versus $SS32$ & $SS64$.

Table 5.11: BiLSTM Word-based 25 with Forced Accuracy, short sequence size

RNN architecture	Accuracy	Forced Acc.
BLSTM CP8 FBN SS16 DL0.6 E62	**0.322**	**0.675**
BLSTM CP32 FBN SS8 DL0.8 E40	**0.263**	**0.622**
BLSTM CP8 FBN SS8 DL0.8 E33	**0.231**	**0.612**

Table 5.12: BiLSTM Word-based 25 with Forced Accuracy, longer sequence size

RNN architecture	Accuracy	Forced Acc.
BLSTM CP32_x2 FBN SS64 DL0.8 E44	**0.571**	**0.747**
BLSTM CP8 FBN SS32 DL0.8 E31	**0.436**	**0.745**
BLSTM CP64_x2 FBN SS64 DL0.8 E34	**0.509**	**0.734**

Next, the RNNs were evaluated with additional convolutional layers with residual connections to bypass the effects of vanishing gradient. Note that "xN" is some integer number of layers (times N).

Table 5.13: Residual BiLSTM Word-based 25

DNN architecture	Accuracy	Forced Acc.
ResBLSTM C64_x10_1024 FBN SS64 DL0.8 V2 E23	**0.61**	**0.843**
ResBLSTM C32_x15_1024 FBN SS64 DL0.8 V2 E20	**0.594**	**0.819**
ResBLSTM C32_x15_512 FBN SS64 DL0.8 V2 E14	**0.576**	**0.813**

However, even without the Residual connections deeper convolutional layers fared well with the BiLSTM model, which had about 5 or more layers less, but equal to or above the residual BiLSTMs.

Table 5.14: Deeper BiLSTM Word-based 25 & 30

CNN architecture 25	**Accuracy**	**Forced Acc.**
BLSTM CP32_x5 FBN SS64 DL0.8 V2 E23	**0.697**	**0.845**
BLSTM CP64_x5 FBN SS64 DL0.8 V2 E51	**0.642**	**0.801**
BLSTM CP32_x3 FBN SS64 DL0.8 V2 E51	**0.688**	**0.822**
CNN architecture 30		
BLSTM CP32_x5 FBN SS64 DL0.8 V2 E36	**0.631**	**0.833**

The last major architecture tried was an attention BiLSTM which did not fair as well. However, this may be due to the short sequence size (longest attempt was 32) and the long training times and kept us from attempting sequence size 64.

Table 5.15: Attention BLSTM Word-based 30 & 25

CNN architecture 30	**Accuracy**	**Forced Acc.**
AttentBLSTM CP8 FBN SS16 DL0.8 E52	**0.277**	**0.611**
AttentBLSTM CP8 FBN SS16 DL0.6 E41	**0.246**	**0.574**
AttentBLSTM CP8 FBN SS32 DL0.6 E43	**0.277**	**0.552**
CNN architecture 25		
AttentBLSTM CP8 FBN SS8 DL0.8 E38	**0.161**	**0.519**
AttentBLSTM CP8 FBN SS8 DL0.6 E43	**0.13**	**0.488**

To recap the performances, we saw that on the Word-based 25 & 30 corpora the MFA had an accuracy of 74% & 78% respectively, versus the BiLSTM with 5 layers on the Word-based corpus 25 & 30 had a forced accuracy of 84% and 83%.

5.3 RNNs with Word Focus

5.3.1 Similar sounding words, but different pronunciations

Next, a RNN was evaluated on similar sounding words. In the OHSU Child Pronunciation corpus, the number of same words that has similar pronunciations were in low frequencies. In the Word-based 10 corpus, only the words “ball” and “drum” have multiple pronunciations and were used in this experiment. To see the effectiveness of Word Focus on similar words, we summed all of the Softmax scores of both word’s frames then compared their frame accuracies to the actual accuracy of the word. It can be seen in this small test set, that the sum of Softmax (SoSM) score of a word is not a good indicator of its word-level accuracy. Although, this may be due to the fact that the corpus was small and only 2 out of 53 words were eligible for comparison.

$$Sum\,of\,Softmax = \sum_{frames \in word} Softmax\,score$$

Table 5.16: Same word different pronunciation: Four examples on “Ball” and “Drum”, with large vocabulary (52)

Possible Phones	**’b’,’ɔ’,’l’**	**’b’,’a’,’l’**	**’b’,’ɔ’,**		**’b’,’ɔ’,’l’,**	**’b’,’a’,’l’**	**’b’,’ɔ’,**
Actual Phones	**X**						**X**
SoSM scores	**56.50**	**45.44**	**44.24**		**41.00**	**49.12**	**35.88**
Accuracy	**0.84**	**0.34**	**0.76**		**0.82**	**0.19**	**0.97**
	’b’,’ɔ’,’l’,	**’b’,’a’,’l’**	**’b’,’ɔ’,**		**’d’,’ɹ’,’ʌ’,’m’**	**’ʤ’,’ɹ’,’ʌ’,’m’**	
Actual Phones			**X**			**X**	
SoSM scores	**40.71**	**48.93**	**39.90**		**44.47**	**44.40**	
Accuracy	**0.76**	**0.16**	**0.96**		**0.62**	**0.8**	
Model Used:BLSTM CP32_x10_FBN SS64 DL0.8							

On a repeat of the same experiment with a smaller vocabulary, reducing the vocabulary from 53 to 42 words, thereby reducing the number of train samples from 2103 to 348, helped the performance of the SoSM scores. However, due to the small size of this test set, this experiment requires a larger test set to validate any claims and thus remains for future work.

Table 5.17: Same word different pronunciation: Four examples on “Ball” and “Drum”, with smaller vocabulary (42)

Possible Phones	**’b’,’ɔ’,’l’**	**’b’,’a’,’l’**	**’b’,’ɔ’,**		**’b’,’ɔ’,’l’,**	**’b’,’a’,’l’**	**’b’,’ɔ’,**
Actual Phones	**X**						**X**
SoSM scores	**60.38**	**34.41**	**60.17**		**54.22**	**39.09**	**54.22**
Accuracy	**0.73**	**0.34**	**0.71**		**1.0**	**0.16**	**1.0**
	’b’,’ɔ’,’l’,	**’b’,’a’,’l’**	**’b’,’ɔ’,**		**’d’,’ɹ’,’ʌ’,’m’**	**’ʤ’,’ɹ’,’ʌ’,’m’**	
Actual Phones			**X**			**X**	
SoSM scores	**62.99**	**26.33**	**62.99**		**59.54**	**54.46**	
Accuracy	**0.97**	**0.21**	**0.97**		**0.62**	**0.8**	
Model Used:BLSTM CP32_x10 FBN SS64 DL0.8							

5.3.2 Entire Vocabulary, WordFocus Softmax Sum vs Accuracy rankings

In a separate Word Focus experiment, we used Word Focus on the entire vocabulary and measured its reliability as a predictor of frame accuracy. For the entire test set, we compared SoSM scores for each word in the vocabulary and and then ranked them. The correct word was ranked either 0 (top choice) or n which meant having a distance from 0 of n. In the table below you can see that the SoSM scores provide an average distance score of about 3, or that the correct word was (on average) in the top 4. Additionally, the average correct-top-word accuracy was provided by using max accuracy scores, with respect to the gold standard, over all the words in the test set. In other words, looking at all the words which were closest to the truth overall, where did the target words land in terms of score ranking.

$$Average\,Testset\,Accuracy = \sum_{words\,\in test\,corpus} \frac{word\,was\,ranked\,top\,(0)}{actual\,top\,word\,by\,gold\,standard}$$

Table 5.18: Word Focus, iterative approach on Word-based Corpus 25

BLSTM CP32_x10 FBN SS64 DL0.8 V2	**Test Set-45**
Approach	**Distance/Accuracy**
SoSM score, avg distance from 0	**3.26**
SoSM - Avg Testset Accuracy	**0.53**
Accuracy, avg distance from 0	**0.46**
Accuracy - Avg Testset Accuracy	**0.90**

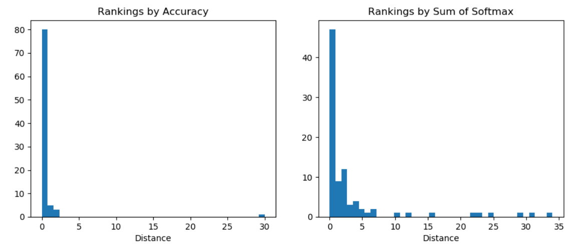

Figure 5.1: Accuracy rankings vs Sum of SM rankings

From this plot, we can see there are a few outliers in the ranking that drop far below the 0 ranking. This skews the average, but indicates that the SoSM score is not a fool-proof method of predicting the expected utterance.

5.4 RNN with Segment Max

As an attempt to simplify the GOP score in terms of computation and complexity, a popular vote was used with phones in the phone boundaries established by the FA. In other words, the FA's phone segments were compared to either a Softmax's or Word Focused' phone segments. Then each word's phones would be either correct or incorrect depending if the most popular phone matched the FA's phone. Note that we skipped the words with phone Insertions, Deletions, Substitutions (IDS), since they were few in count.

It can be seen that Segment Max accuracy is fairly close to the frame accuracy. By this assessment, Segment Max provides a simple method of saving computational resources via a coarser score.

Table 5.19: Segment Max on Word-based Corpus 10

BLSTM CP32_x10 FBN SS64 DL0.8 V2	**Correct out of total**	**Percent Correct**
Softmax:	**2958 out of 5641**	**52.4%**
Forced:	**4379 out of 5641**	**77.6%**
Softmax (seg)	**200 out of 401**	**49.9%**
Forced (seg)	**299 out of 401**	**74.6%**
Insertions, Deletions, Substitions (SM)	**29 out of 5641**	**0.51%**
Insertions, Deletions, Substitions (F)	**15 out of 5641**	**0.27%**

5.5 GOP using only RNNs

An RNN was used as both the FA and the open loop in a simple GOP score configuration.

$$GOP(p_i) = |log(\frac{RNN_{WordFocus}[b : b_{i+1})}{RNN_{Softmax}[b_i : b_{i+1})})/NF_{p_i}| \tag{5.1}$$

Here the RNN segments the phone boundaries as $[b_i : b_{i+1})$ for all phones in a given word, using Word Focus. The RNN scores from Word Focus were divided by those from the Softmax, then summed, scaled logarithmically, divided by the number of frames for that phone segment, and converted to positive values via absolute value. In this case, the greater the value the worse the pronunciation. To see the efficacy of this GOP score, we first tested the GOP on correct words. The GOP score stayed near 0 for these, however for mispronunciations they detected higher scores.

Table 5.20: GOP score with single RNN (Word Focus/Softmax)

Expected	**Actual**	**Phones**	**Segmentation ('phone', GOP score, Number of Frames)**	**AvgGOP**
Blue	**Blue**	**[‘b’, ‘l’, ‘u’]**	**(‘b’, 0.0, 6), (‘l’, 1.48, 1), (‘u’, 0.08, 32)**	**0.52**
“”	**Bath**	**[‘b’, ‘æ’, ‘Θ’,]**	**(‘b’, 1.3, 9), (‘æ’, 5.3, 1), (‘b’, 8.9, 1), (‘æ’, 9.6, 5), (‘b’, 6.9, 5), (‘æ’, 6.4)**	**6.39**
Cup	**Cup**	**[‘k’, ‘ʌ’, ‘p’]**	**(‘k’, 0.0, 9), (‘ʌ’, 0.0, 8), (‘k’, 0.0, 4), (‘p’, 0.0, 9), (‘k’, 0.0, 6), (‘p’, 0.0, 4)]**	**0.0**
“”	**Car**	**[‘k’, ‘a’, ‘ɹ’]**	**(‘k’, 0.0, 40)**	**0.732**
Blue	**Blue**	**[‘b’, ‘l’, ‘u’]**	**(‘b’, 0.81, 5), (‘l’, 0.0, 5), (‘u’, 0.0, 24)**	**0.27**
“”	**Bath**	**[‘b’, ‘æ’, ‘Θ’,]**	**(‘b’, 3.65, 10), (‘æ’, 6.37, 18), (‘b’, 10.17, 4)**	**6.73**
Cup	**Cup**	**[‘k’, ‘ʌ’, ‘p’]**	**(‘k’, 3.11, 18), (‘ʌ’,0.0, 27), (‘p’,0.13,15), (‘k’, 0.19, 4), (‘p’, 4.64, 9)]**	**1.61**
“”	**Car**	**[‘k’, ‘a’, ‘ɹ’]**	**(‘k’, 3.10, 21), (‘a’, 5.43, 24), (‘k’, 3.84, 3), (’ɹ’ , 5.59, 3), (‘k’, 3.69, 22)]**	**4.33**

Note, that RNNs were not using WFSTs or a Bakis lattice to prevent IDS and hence their segments include out of order phones. The GOP scores work well for the expected versus not expected comparison. Although, the results appear promising these few examples were handpicked. Thus this GOP formulation requires further analysis and remains left for future work.

5.6 Discussion

In the following sections, we briefly discuss some concluding ideas drawn from the experiments and their results.

5.6.1 Open loop

As the results show CNNs outperform DNNs and RNNs outperform both of them. Hence, CNNs and RNNs can be used in an open loop configuration by using them normally and classifying phones with the Softmax layer. Additionally, it may be useful to use hierarchies of CNNs and RNNs to compare phone pronunciations as they can be trained to discriminate between a few related classes and specialize rather than train on the entire corpus. We saw that DNNs tend to perform better when given fewer phones classes to categorize against.

5.6.2 RNNs & MFA

With Word Focus, RNNs can be used as FAs in the numerator component as they can outperform FAs in terms of frame accuracy. Although, not attempted in this thesis, RNNs can use WFSTs to create a Bakis-lattice and perform ordered segmentation of a word like the FAs. However without WFSTs, RNNs allow insertions and/or substitutions in an utterance given that those phones are included in the phone set. Thus adding functionality to GOP scoring.

5.6.3 RNN approaches

Word Focus allows RNNs to become Correct and Incorrect GOP scores. Segment Max offers an inexpensive computational approach for the GOP score that requires no thresholds and may be as simple as one RNN using Word Focus and Softmax. With Iterative Search, RNNs are also able to predict the said word with SoSM and can be used to double check that expected word was said.

5.6.4 GOP using RNNs

A simple GOP score was created using RNNs alone. However, since RNNs do not preserve order as FAs do, their use in a GOP numerator may be detrimental unless they are highly accurate or users take precautions not to weigh heavily on out of order segments or very short segments; as either of these cases can cause erroneous GOP scores.

5.6.5 GOP ensemble

In general, with these new components, a new GOP-ensemble score can deliver better results by scaling the use of each variant GOP score. As was shown in the results of the GOP using only RNNs, they allow the detection of IDS. A combination of Correct and Incorrect HMM-GMM WFSTs and FAs, forced and open DNNs, and other GOP score derivatives may increase the detection of mispronunciations. Although, the GOP-ensemble is based on an incremental performance approach, small improvements may compound into a system that is reliable enough to use.

Chapter 6

Conclusion

For the purposes of learning a new language or correcting one's speech, the GOP score is a useful metric. The reliability of the GOP score is dependent on the accuracy of its components. As shown in this paper, RNN's have higher frame accuracies than CNNs and CNNs have better frame accuracies than simple fully-connected DNNs. Also shown were the beneficial approaches that RNNs provide by using Word Focus via their Softmax layer to score commonly pronounced or mispronounced words. Last was the GOP-ensemble which was a proposal of using multiple GOP scores to set the stage for an automatic and modular GOP scoring system. Improvements to these pronunciation metrics are important due to an increasingly globalized world with more and more language learners and to those with speaking disorders in rural areas.

References

Abdel-Hamid, O., Mohamed, A.-R., Jiang, H., & Penn, G. (2012). Applying convolutional neural networks concepts to hybrid DNN-HMM model for speech recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on* (pp. 4277–4280).

Arora, V., Lahiri, A., & Reetz, H. (2017). Phonological feature based mispronunciation detection and diagnosis using Multi-Task DNNs and Active Learning. *Proc. Interspeech 2017*, 1432–1436.

Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural Machine Translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Bilmes, J. A., et al. (1998). A gentle tutorial of the EM algorithm and its application to parameter estimation for Gaussian mixture and hidden Markov models. *International Computer Science Institute*, *4*(510), 126.

Chung, Junyoung and Gulcehre, Caglar and Cho, KyungHyun and Bengio, Yoshua. (2014). Empirical evaluation of Gated Recurrent Neural Networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Dave, N. (2013). Feature extraction methods LPC, PLP, and MFCC in Speech Recognition. *International journal for Advance Research in Engineering and Technology*, *1*(6), 1–4.

Deng, L., Hinton, G., & Kingsbury, B. (2013). New types of deep neural network learning for speech recognition and related applications: An overview. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on* (pp. 8599–8603).

Doremalen, J. v., Cucchiarini, C., & Strik, H. (2013). Automatic pronunciation error detection in non-native speech: The case of vowel errors in Dutch. *The Journal of the Acoustical Society of America*, *134*(2), 1336–1347.

Dudy, S., Bedrick, S., Asgari, M., & Kain, A. (2017). Automatic analysis of pronunciations for children with speech sound disorders. *Computer Speech & Language*.

Franco, H., Neumeyer, L., Kim, Y., & Ronen, O. (1997). Automatic pronunciation scoring for language instruction. In *Acoustics, Speech, and Signal Processing, 1997. ICASSP-97., 1997 IEEE International Conference on* (Vol. 2, pp. 1471–1474).

Gal, Y., & Ghahramani, Z. (2016). Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *33rd International Conference on Machine Learning, ICML 2016* (Vol. 3, pp. 1651–1660).

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning* (Vol. 1).

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition* (pp. 770–778).

Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A.-r., Jaitly, N., ... others (2012). Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups. *IEEE Signal Processing Magazine*, *29*(6), 82–97.

Hu, W., Qian, Y., & Soong, F. K. (2015). An improved DNN-based approach to mispronunciation detection and diagnosis of L2 learners' speech. In *Slate* (pp. 71–76).

Huang, T., Peng, H., & Zhang, K. (2013). Model Selection for Gaussian Mixture Models. *arXiv preprint arXiv:1301.3558*.

Hwang, M.-Y., & Huang, X. (1992). Subphonetic modeling for Speech Recognition. In *Proceedings of the workshop on Speech and Natural Language* (pp. 174–179).

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.

Kurzekar, P. K., Deshmukh, R. R., Waghmare, V. B., & Shrishrimal, P. P. (2014). A comparative study of feature extraction techniques for Speech Recognition system. *International Journal of Innovative Research in Science, Engineering and Technology*, *3*(12), 18006–18016.

Lee, C.-H., & Siniscalchi, S. M. (2013). An Information-Extraction Approach to Speech Processing: Analysis, Detection, Verification, and Recognition. *Proceedings of the IEEE*, *101*(5), 1089–1115.

Li, J., Mohamed, A., Zweig, G., & Gong, Y. (2016). Exploring multidimensional LSTMs for large vocabulary ASR. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on* (pp. 4940–4944).

Martin, J. H., & Jurafsky, D. (2000). Speech and Language processing: An Introduction to Natural Language Processing. *Computational Linguistics and Speech Recognition. Prentice Hall*, *2nd Edition*.

McAuliffe, M., Socolof, M., Mihuc, S., Wagner, M., & Sonderegger, M. (2017). Montreal Forced Aligner: trainable text-speech alignment using Kaldi. In *Proceedings of Interspeech.*

Milne, P. M. (2015). Improving the accuracy of forced alignment through model selection and dictionary restriction. *Acoustics Week in Canada 2015 (AWC15).*

Nussbaum-Thom, M., Cui, J., Ramabhadran, B., & Goel, V. (2016). Acoustic Modeling Using Bidirectional Gated Recurrent Convolutional Units. In *Interspeech* (pp. 390–394).

Olah, C. (2015a). *Neural Networks, Types, and Functional Programming.* ([Online; acessed 24-June-2018])

Olah, C. (2015b). *Understanding LSTM Networks.* ([Online; acessed 24-June-2018])

Peel, D., & McLachlan, G. J. (2000). Robust mixture modelling using the t distribution. *Statistics and Computing*, *10*(4), 339–348.

Povey, D., Ghoshal, A., Boulianne, G., Burget, L., Glembek, O., Goel, N., ... others (2011). The Kaldi speech recognition toolkit. In *IEEE 2011 workshop on automatic speech recognition* *and understanding.*

Sainath, T. N., Mohamed, A.-r., Kingsbury, B., & Ramabhadran, B. (2013). Deep Convolutional Neural Networks for LVCSR. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on* (pp. 8614–8618).

Scherer, D., Müller, A., & Behnke, S. (2010). Evaluation of pooling operations in convolutional architectures for object recognition. In *International Conference on Artificial Neural Networks* (pp. 92–101).

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural networks*, *61*, 85–117.

Schuster, M., & Paliwal, K. K. (1997). Bidirectional Recurrent Neural Networks. *IEEE Transactions on Signal Processing*, *45*(11), 2673–2681.

Shimodaira, H., & Renals, S. (2017). *Hidden Markov Models and Gaussian Mixture Models.* ([Online; acessed 24-April-2017])

Shrawankar, U., & Thakare, V. M. (2013). Techniques for Feature Extraction in Speech Recognition system: A comparative study. *arXiv preprint arXiv:1305.1145*.

Song, Y., Liang, W., & Liu, R. (2010). Lattice-based GOP in automatic pronunciation evaluation. In *Computer and Automation Engineering (ICCAE), 2010 The 2nd International Conference on* (Vol. 4, pp. 467–471).

Wei, S., Hu, G., Hu, Y., & Wang, R.-H. (2009). A new method for mispronunciation detection using Support Vector Machine based on Pronunciation Space Models. *Speech Communication*, *51*(10), 896–905.

Witt, S., & Young, S. J. (1997). Language Learning based on non-native Speech Recognition. In *Fifth European Conference on Speech Communication and Technology.*

Witt, S. M., & Young, S. J. (2000). Phone-level pronunciation scoring and assessment for interactive language learning. *Speech communication*, *30*(2-3), 95–108.

Xiong, W., Droppo, J., Huang, X., Seide, F., Seltzer, M., Stolcke, A., ... Zweig, G. (2016). Achieving Human Parity in Conversational Speech Recognition. *arXiv preprint arXiv:1610.05256*.

Young, S., Evermann, G., Hain, T., Kershaw, D., Moore, G., Odell, J., ... Woodland, P. (2002). The HTK book.

Zeiler, M. D., Ranzato, M., Monga, R., Mao, M., Yang, K., Le, Q. V., ... others (2013). On rectified linear units for speech processing. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on* (pp. 3517–3521).

Zhang, Y., Pezeshki, M., Brakel, P., Zhang, S., Bengio, C. L. Y., & Courville, A. (2017). Towards End-to-End Speech Recognition with Deep Convolutional Neural Networks. *arXiv preprint arXiv:1701.02720*.