Speech Recognition for Accented English

Stanford CS224N Custom Project Milestone

Claire Pajot and Sasha Harrison
Mentor: Amita Kumath
cpajot@stanford.edu | aharris6@stanford.edu

1 Abstract

In recent years, deep neural networks have become the state of the art for speech recognition. However, speech recognition systems (both neural and Hidden Markov Model based) often struggle to correctly interpret foreign accents (1). We aim to contribute to the body of literature on fairness in machine learning by demonstrating that it is possible to improve the performance of deep neural networks on accented English speech by adding an adversary to the training architecture. This adversary adjusts the loss function of the speech recognition network in a way that enforces parity in the loss attributed to speech examples from American and non-American English speakers. Although our results are not statistically significant, we found that adding an adversary to the model architecture did decrease the gap in both WER and CER for the two accent classes. We hypothesize that training a larger model that achieves better results on the test set would yield more convincing results that adversarial architectures are useful for enforcing fairness in NLP applications.

2 Introduction

The task of speech to text transcription has long been a topic of interest in the machine learning community. Until recently, state of the art models for speech recognition (SR) relied on complex pipelines of algorithms and processing stages, including separate acoustic, pronunciation, and language models (AM, PM, LM). Although neural network based methods have long been used as components of speech recognition pipelines, it is only recently that end-to-end deep learning based approaches have shown success. These developments have displaced conventional methods, in part due to their relative simplicity and the ability of deep models to bundle many aspects of the speech recognition pipeline under one architecture.

Today, deep architectures such as Recurrent Neural Network Transducers (RNNT) and Listen, Attend and Spell (LAS) have demonstrated better results than earlier, non-NN based methods (2). Neural networks have been particularly successful in settings where robustness to speaker variation and background noise are important. However, these systems (both neural and HMM based) are still far from perfect. In particular, such systems often struggle to correctly interpret foreign accents (1). In 2018, *The Washington Post* found that Amazon Alexa and Google Home are 30% less likely to understand people who speak English with non-American accents (3). As speech to text systems become more prominent in daily life, this disparity has a potentially large impact on the usability of these systems. It is important that speech to text models perform as well on accented speech as Generic American English. We claim that improving the performance of speech recognition systems on non-American accents is an important step toward the fairness and usability of speech recognition systems.

In this paper, we build on algorithmic fairness literature to explore methods of enforcing such performance parity across demographic groups. Specifically, we hope to contribute to the body of literature on fairness in machine learning by demonstrating that it is possible to improve the performance of recurrent neural networks on accented English speech.

Goal

The goal of this project is to use an adversarial architecture to train an end-to-end, deep learning model that performs speech to text transcription such that the accuracy of transcription is indistinguishable across different accent classes.

3 Related Work

In the past decades, there have been numerous papers published about applications of recurrent neural networks to speech to text transcription tasks. We drew upon several of these works to design our model architecture and experiments.

The first paper that influenced our neural network architecture is Deep Speech. Originally published in 2014 by researchers at Baidu, the Deep Speech paper proposed a novel end-to-end speech recognition architecture that is optimized for scalability (4). Deep Speech yielded substantially better performance than previous deep learning based speech recognition models. This is due in part to its emphasis on scalability, which enabled it to take advantage of a significantly larger training set than predecessors. One challenge to the Deepspeech project was the use of opensource speech data, which results in the task of predicting sequences of labels from noisy, unsegmented input data.

One important aspect of the Deep Speech model is its use of the CTC loss function, first introduced by researchers Graves et. al in 2006 (5). The CTC loss function is what originally allowed RNNs to be applied to the relatively unstructured task of speech recognition. CTC loss solves the problem of "alignment", or one spoken character spanning multiple time slices t. This alignment problem is relevant to our problem domain, because the labels on our training example are not recorded on a per-time interval basis. The CTC Loss takes as input the groundtruth sentence, and the output matrix of the RNN. It returns a metric of how close the predicted text is to the groundtruth text, where this metric is invariant to alignment (5). Thus, it removes the need for pre-segmented training data and post-processed outputs by accounting for the possibility that individual characters (phonemes) can take up a variable amount of time (5).

Hunnan et. al. used Deep Speech to build on this general approach proposed by Graves et. al. by applying CTC loss to a larger, more flexible model. While Graves et al. trained their hybrid HMM-RNN model on just 81 hours of speech data from the WSJ data set, the Deep Speech model was trained on a data set of over 7,000 hours of speech data, which was further augmented using audio synthesis techniques. Overall, the Deep Speech model achieves 16.0% error on the Switchboard Hub 500 corpus, outperforming previously published methods.

Hunnah et. al. credit their use of data augmentation as the reason that the Deep Speech model made significant performance gains on particularly challenging test examples, such as speech samples in the Switchboard corpus that included background noise. To demonstrate these gains, the authors chose a challenging test set, Switch 2000 Hub 500, and reported the overall error rate on the full test set. The Deep Speech model was also shown to outperform many commercial systems available at the time in noisy speech recognition tests, achieving a word error rate (WER) of 19.06% compared to WERs larger than 30.47% achieved by Apple Dictation, Bing Speech, Google API, and wit.ai. (4)

In 2017, Baidu published a follow-up model called Deep Speech 2. Deep Speech 2 improves upon the Deep Speech architecture further, and is notable in its ability to achieve state of the art results in Mandarin as well as English. In comparison to Deep Speech 1 (discussed above), Deep Speech 2 nearly 8 times the amount of computation per data example, making fast optimization and computation critical (6). DeepSpeech 2 was trained on 11,940 hours of speech, utilized Batch Normalization for RNNs and a novel optimization scheme called SortaGrad (6). In this paper, we use the architecture of Deep Speech 2 as the main inspiration for our custom speech to text architecture, including BatchNorm for RNNs.

Algorithmic Fairness

Aside from design decisions aorund best architectures for the given Natural Language Processing task, we also drew on algorithmic fairness literature to develop a quantitative measure of fairness. In *Equality of Opportunity in Supervised Learning*, researchers Hardt et. al. propose several methods for evaluating discrimination against a specified sensitive attribute in supervised learning. Examples of sensitive attributes include gender, sexuality, race, or ethnic background. Assuming access to information about the target, possible features, and protected attribute, Hardt et. al. show how to optimally adjust any learned predictor so as to remove discrimination according to their definition. They introduce 3 notions of non-discrimination, each defined with respect to conditional probability distributions (7). Fo simplicity, we choose to use the fairness metric of **Parity**. An output prediction $\hat{Y}$ satisfies parity if transcription performance is the same for the protected attribute in question.

After determining the quantitative metric we will use to measure the fairness of our algorithm, it is then necessary to determine what architectural changes we will make to improve our models' performance with respect to the fairness metric. For this, we draw upon the literature on adversarial networks, in particular, *Achieving Fairness through Adversarial Learning: an Application to Recidivism Prediction*. In this paper, Wadsworth et. al. apply an adversarial architecture to recidivism prediction. They demonstrated that without explicit effort, a neural network that predicts recidivism is biased against black inmates. They were able to reduce this disparity by introducing an adversary into the training architecture fo their neural network. This adversary penalizes the recidivism prediction network if race is predictable from the recidivism prediction. We will adopt this same approach in a different setting, namely that of speech to text transcription. We expect that, as in Wadsworth et. al, the adversary will penalize the transcription network if the speaker's accent class can be predicted by the transcription output. The work in Wadsworth et. al. is impactful because it is generalizable to almost any prediction and any demographic.

4 Approach

The aim of this paper is to show that given a speech to text model that shows worse results on non-American speakers, adding an adversary to the training architecture can minimize the difference in performance between American and non-American English speakers. First, we implement a run of the mill end-to-end speech recognition system In the simplest terms, this model is a recurrent neural network (RNN) trained to ingest speech spectrograms and transcribe them into text. The details of this deep neural network are as follows:

The input audio file is first converted into a **spectrogram**. A spectrogram can be thought of as a 2D matrix representation of the audio frequencies observed at each time step. In speech recognition, it is typical to use a time interval of 20-30ms, which corresponds to the average amount of time it takes a person to utter one unit of sound (called a *phoneme*). This is a valuable pre-processing step, as we expect the spectrogram at time t to be characterised by the spoken character.

Next, the spectrogram is fed as input to a deep neural network consisting of three parts: a CNN, an LSTM, and finally a Fully-Connected layer (see Figure 1). Mathematically, given an input $x_t^{(i)}$ corresponding to the time slice t of example i, this network makes a prediction over characters $p(l|x_t)$, where l_t is either an English character or a blank symbol. In order to get this probability distribution $p(l|x_t)$, the spectrogram is first passed through two convolutional layers as a form of feature extraction. Next, the output of the convolutional layer is passed through a bidirectional LSTM with 3 stacked layers of hidden units.

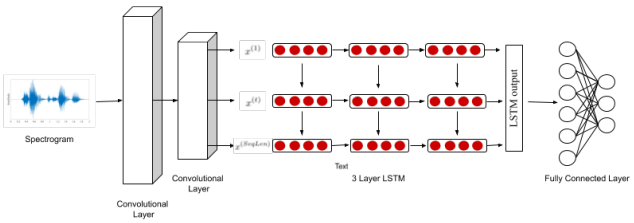

Figure 1: Diagram of speech recognition architecture

The forward in time $\overrightarrow{h}^l$ and the backward in time $\overleftarrow{h}^l$ recurrent activations are computed as

$$\overrightarrow{h}^l_t = g(h_t^{(l-1)}, \overrightarrow{h}^l_{t-1})$$

$$\overleftarrow{h}^l_t = g(h_t^{(l-1)}, \overleftarrow{h}^l_{t+1})$$

Where the function $g(\cdot)$ can be the standard recurrent operation. The two sets of activations are summed to form the output activations for the layer $h^l = \overrightarrow{h}^l_t + \overleftarrow{h}^l_t$.

The output layer L is a softmax computing a probability distribution over characters, given by

$$\mathbb{P}(l_t = k|x) = \frac{\exp(W_k^L \cdot h_t^{L-1})}{\sum_j \exp(W_j^L \cdot h_t^{L-1})}$$

This architecture is then training on labeled training examples using Connectionist Temporal Classification (CTC) Loss as the loss function, consistent with Deep Speech 2 (6). There are several opensource GitHub repositories, for example Awni Hunnan's speech and deepspeech.pytorch implemented by Sean Naren, that we can draw upon to reproduce the Deep Speech 2 architecture.

Adversarial Training

After recording the baseline results using the model described above, we then introduced an adversarial training architecture in order to penalize correlation between the output character predictions $\hat{Y}$ and A, where A is a binary variable that represents whether the speaker speaks Generic American English ($A = 1$ means True).

For each utterance x_t, the model described above, which we will henceforth refer to as the vanilla RNN, outputs predicted character probabilities $\hat{y}_t$. Overall, the vanilla RNN produces a 3-dimensional matrix corresponding to character probability distributions for each time step t.

The adversary takes this 3-dimensional matrix as input, and uses an LSTM-based classifier to predict which accent class ($A = 1$ or $A = 0$) each training example $x^{(i)}$ came from. Although we originally imagined the adversary to be a 2

layer fully connected neural network, we quickly realized that it needs to have a many inputs to one output formulation. As a result, an RNN architecture would be more fitting to this classification task. Thus, the adversary we used in our experiments was a single layer one-directional LSTM with hidden size 128, followed by a fully connected layer that has output size 2, where 2 is the number of accent classes.

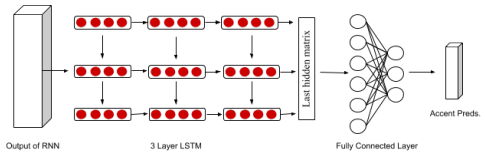

Figure 2: Diagram of adversary introduced to training architecture

For each input matrix denoted $\hat{Y}$, the LSTM would generate a prediction for each time step $\hat{Y}_t$. Then, at the end of the input sequence, we would take the last hidden cell output by the LSTM model, and provide this hidden state as input to the fully connected layer. The output of the FC layer has dimension (batch size, 2). A softmax function is applied to this output matrix to normalize it to a probability distribution across the accent classes.

At a high level, the adversary will try to predict which accent class the current speech sample belongs to. It is trained as a binary classifier with a Cross Entropy Loss objective function, which we will denote as L_A. The loss function of the overall adversarial model will be

$$L = L_{CTC} - \alpha L_A$$

where α is a hyperparameter controlling the strength of the adversary. This combined loss function encourages the RNN to make good predictions that the adversary cannot easily differentiate into different classes. This methodology should have the effect of forcing the hidden layers of the RNN to be independent of the sample class, which we hope will lead to performance parity between American English speakers and accented speakers. with the hopes of improving parity between the performance on classes $A = 0$ and $A = 1$, as has been demonstrated in other problem domains by Wadsworth et. al. (8).

5 Experiments

5.1 Data

We use the opensource audio dataset Mozilla Open Voice. The goal of the Mozilla Open Voice dataset project is to produce a high quality, publicly available voice dataset from a diversity of speakers. It contains over 4,200 hours of validated speech files, where each example is labeled with groundtruth transcript, as well as demographic information such as accent, age, and gender. These demographic labels are particularly important in our paper, because our adversarial architecture requires training data to be labeled with accent in order to train with the adversary (9).

5.2 Evaluation Method

To evaluate the quality of the speech to text transcriptions produced by our model, we will use two metrics: word error rate (WER) and character error rate (CER). Both of these error rates are based on the Levenshtein distance (See figure 3), which measures the difference between two sequences. $CER = \frac{S+D+I}{N}$ where S is number of substitutions, D is the number of deletions, I is the number of insertions, and N is the total number of characters in the reference text. The formula for word error rate is identical except the unit is space separated words instead of individual characters. Word Error Rate (WER) is the most widespread performance metric used for this kind of task, which makes it easy to compare our results to those of other published architectures. For example, Baidu's Deep Speech model, which achieves a WER of 16% on the benchmark Switchboard Hub 500 dataset.

Secondly, we will use quantitative metrics to measure the "fairness" of our model. The fairness metric we use is the notion of **Parity**, which enforces indistinguishable performance between examples from native English speakers vs. non-native English speakers. Thus, our goal with respect to fairness is to have no statistically significant difference between the word error rates and character error rates of our two classes.

5.3 Experimental Details

Transfer learning using LibriSpeech

Our first experiments were based on an existing implementation of the DeepSpeech2 architecture, which we found at `https://github.com/SeanNaren/deepspeech.pytorch`. We knew that training a speech to text model from scratch would take a very long time, and we were interested in whether using pre-trained wieghts might speed up our experiments.

However, we encountered substantial challenges in getting this network to train successfully. First, we initialized the weights of the DeepSpeech2 model using pretrained weights from training on the LibriSpeech Corpus. Using these weights for inference on the Mozilla Open Voice test dataset, we found that the pre-trained model did not generalize well to the new dataset. For example, with groundtruth text "WE MUST BE GOOD", the pretrained model's output was "OA I I O".

We tried to improve these results by doing finetuning on the pretrained weights using the Mozilla training dataset. We ran a finetuning training loop with a learning rate of $3e^{-4}$ and rnn hidden size of 1024. The overall network contained over 86 million parameters. As a consequence, we found that even finetuning would take over 72 hours to run to completion. Therefore, we concluded that transfer learning with existing weights would be computationally intractable.

Training from scratch with Mozilla Open Voice

Next, we used the same opensource DeepSpeech2 implementation and re-ran training from scratch suing the Mozilla training dataset. By reducing the hidden layer size from 1024 to 256, and reducing the RNN depth from 5 to 4, we reduced the number of parameters to 6.5 million (93% reduction). We estimated these training loops would take around 24 hours to run for 20 epochs on the full Mozilla training dataset. On our first run through this training loop, we found that the model started showing nan losses. We hypothesized nan loss was evidence of either a learning rate that was too large, exploding gradients, or malformed input matrices containing nans. The results of the experiments we ran to debug this issue can be seen in the appendix section.

Final Vanilla Model with Mozilla Open Voice

After encountering numerous setbacks using the opensource repository, we decided to reimplement this speech recognition model from scratch using pytorch. This simpler model adheres to the diagram in Figure 1, and is composed of 2 convolutional layers, an LSTM with depth 3 and hidden size 512, and one fully-connected layer that projects the LSTM output to the size of the vocabulary. We trained this model on a subset of 5% of the Mozilla OpenVoice Training Dataset for 160 epochs using a learning rate of 1e-2 with no annealing. We chose to subset the training data in order to decrease training time to about 48 hours.

Adversarial Training

Next, after we had shown moderate success with the vanilla RNN model, we introduced an adversary into the training architecture to act as a fairness normalization technique. The adversary was an LSTM-based classifier as described above. Since our original training loop took about 48 hours to complete, we initialized our adversarial training from pre-trained weights. The adversarial network was trained for 90 epochs on the same training set as above (5 % subset) with a lower learning rate of $5e^{-4}$ for the Vanilla RNN, and a learning rate of 0.002 for the adversary. The vanilla RNN's weights were initialized from the 160th epoch of the training loop described in the previous section.

5.4 Results

Final Vanilla Model with Mozilla Open Voice

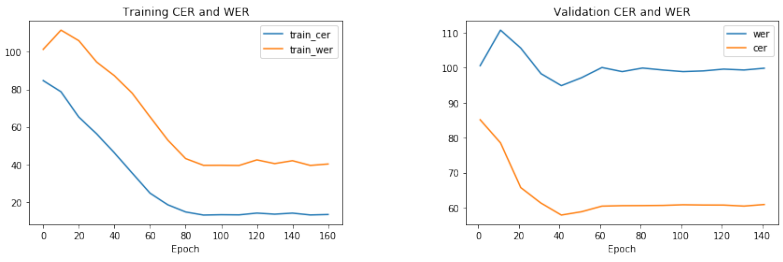

Figure 3: Training and validation results for vanilla RNN model trained for 160 epochs

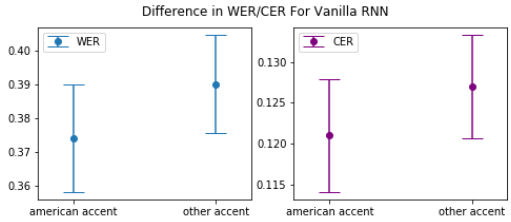

Figure 4: Difference in WER/CER for the two accent classes

Over the course of training, the CER and WER on the training dataset steadily decreased, and the best CER achieved on the training set was around 12% error. The CER and WER on the hold out validation set plateaued at around epoch 40 (25% of the way through training), which is evidence of overfitting. This is logical given that we used a pretty small training dataset, making overfitting more likely. Figure 4 shows the difference in WER and CER for the two classes, evaluated on the training set. While the error bars (depicting the 95% confidence interval) show that the differences are not statistically significant, the model's performance does seem to be consistently worse on the non-american accent class, consistent with our assumptions and problem motivation.

Adversarial Training

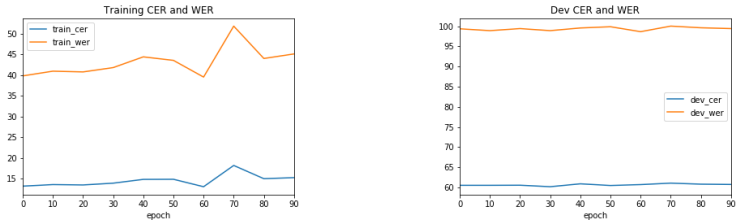

Figure 5: Training and validation results for adversarial model running fine-tuning for 90 epochs

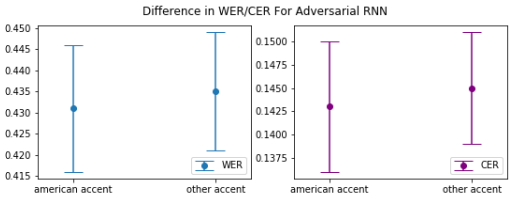

Figure 6: Difference in WER/CER for the two accent classes

Model	Dataset	Generic WER	Generic CER	Accent WER	Accent CER
adversarial	validation	102.9	60.6	99.0	60.1
adversarial	train	43.1	14.3	43.5	14.5
vanilla	validation	104.2	60.2	98.0	60.3
vanilla	train	37.4	12.1	39.0	12.7

Over the course of adversarial training, the CER and WER on the training dataset and validation set remained somewhat constant, as seen in Figure 5. Although this may seem like a puzzling result on the surface, this is logical because the training loop was initialized with pretrained weights, so the initial CER/WER was already comparable to the best we had seen in the previous training loop. The training CTC loss steadily decreased (see Appendix), and the best CER achieved on the training set was around 14% error, slightly higher than the vanilla model. It is expected that the adversarial model does not overfit the training data as much as the vanilla RNN because the adversary acts as a normalization mechanism. The CER on the hold out validation set howevered around 60%, which is comparable to the behavior we observed above. Figure 6 shows the difference in WER and CER for the two classes, evaluated on the training set. In comparison to the vanilla model, the two confidence intervals for each accent class are much closer together, indicating that our adversary was successful in enforcing performance parity between the two accent classes.

6 Analysis

Vanilla Model

Our vanilla speech to text model composed of 2 CNN layers, an LSTM, and a fully connected layer had about 16 million parameters in total. The best performance achieved by this model was about 12% character error rate (CER) on the training set, and about 40% CER on the dev set. These metrics are definitely not comparable to the state of the art results, which is reasonable given that we had to subset our final training dataset to about 5% of its original size. Looking at figures documenting training performance, it looks like the model overfits the training data because the performance on the dev set ceases to improve after about the 40th epoch. We believe that given more time, training on a larger training dataset would reduce the disparity in training and test performance.

An example transcription from this model is the following:

- **Reference:** HE WAS JUST AND GOOD
- **Model Output:** HE WAS JAST ANDGO

This concrete example shows some of the challenges in transcribing speech. Namely, it is useful to report both WER and CER because the tasks of correctly separating words, and correctly interpretting sounds into letters are somewhat distinct. In the example above, the model does a pretty good job with respect to predicting the correct characters, but is less successful with separating 'and' and 'good' into two separate tokens.

We then evaluated this model on the two accent classes: Generic American English and all others. We expected that overall, performance would be worse on non-American English speakers. To our disappointment, saw that there was no statistically significant difference in performance in the dev set, and there was a small, insignificant difference in performance when the model was evaluated on training set. We think this happened simply because our word error rates and character error rates are not very high. We conceptualize bias with respect to accent as a type of overfitting. Using this analogy, one possible explanation is that we only begin to see a difference in results on the training set, since that is where we're close to adequately fitting the data.

Adversarial Model

The same speech to text model training together with the adversary achieved very similar results, with 14% CER as the best on the training set and 40% as the best on the dev set.

An example transcription from this model is the following:

- **Reference:** I FOUND A LITTLE CROWD OF ABOUT TWENTY PEOPLE SURROUNDING THE HUGE HOLE
- **Model Output:** I FOUND A LITTLE CRAWD OF ABOUT TENTY PEOPLE SURROUNDING THE HUGE OLE

Anecdotally, this prediction seems similar in quality to that examined above. We see that the letter 'W' and 'H' are more difficult to predict correctly because they blend in with the surrounding sounds. i.e. 'TWENTY' vs. 'TENTY'. Qualitatively, it seems introducing the adversary did not dramatically reduce the overall quality of transcriptions, which is an accomplishment in and of itself. It can be difficult to get the right equilibrium between two dueling models during training, as is often seen in literature on Generative Adversarial Networks (GANs).

7 Future Work

In this paper, we demonstrated that applying an adversary is a potentially effective way to debias the results of a speech to text model, making such systems more fair, transparent, and usable. Although we didn't achieve as compelling results as we hoped for, we still showed that it's possible to apply an adversarial training architecture to an NLP problem without drastically worsening the baseline transcription performance. We were able to successfully achieve equilibrium between the adversary and the vanilla RNN model such that the training error continued to decrease over time despite the added normalization mechanism.

We think this is a compelling line of inquiry to continue in the future. Given more time, the first thing we would do is repeat our experiments with a larger training dataset and a larger hidden size (more parameters) in hopes of reducing overfitting on the training dataset and achieving better results on the dev set. We hypothesize that better overall results would exacerbate the disparity in performance we report in this paper, which would provide a more compelling reason for fairness researchers to pursue this application.

Secondly, one interesting architectural change would be to add a language model to the speech to text transcription pipeline. In both DeepSpeech papers, the authors found that using an n-gram model or Beam Decoding instead of

Greedy Decoding at inference time was able to correct many of the point errors made during character prediction. Given the example results shown above, it is reasonable to think that something like Beam Decoding would greatly improve our results.

Second, with respect to the adversarial model, we would run additional experiments to refine the hyperparameters. for example, the hyperparameter alpha controls how much the adversary's loss is incorporated into the overall training loss of the system. It would be useful to better understand how varying alpha affects the tradeoff between performance parity and high performance, since an adversary that is too influential is likely to lead to worse overall performance

8 Figures

$$\mathrm{lev}_{a,b}(i,j) = \begin{cases} \max(i,j) & \text{if } \min(i,j) = 0, \\ \min \begin{cases} \mathrm{lev}_{a,b}(i-1,j)+1 \\ \mathrm{lev}_{a,b}(i,j-1)+1 \\ \mathrm{lev}_{a,b}(i-1,j-1)+1_{(a_i \neq b_j)} \end{cases} & \text{otherwise.} \end{cases}$$

Figure 7: Levenshtein distance

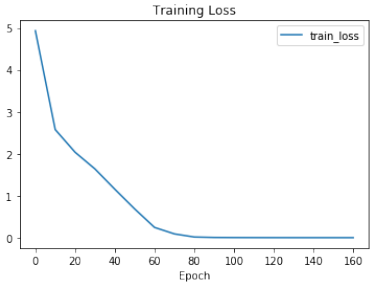

Figure 8: Graph of training loss of the speech to text model without adversary

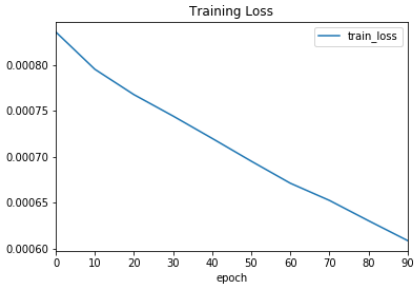

Figure 9: Graph of training loss of the adversarial RNN. Training loss decreases steadily despite CEr/WER results showing no improvement.

References

[1] F. Kitashov, E. Svitanko, and D. Dutta, "Foreign english accent adjustment by learning phonetic patterns," *CoRR*, vol. abs/1807.03625, 2018. [Online]. Available: http://arxiv.org/abs/1807.03625

[2] C. Chiu, T. N. Sainath, Y. Wu, R. Prabhavalkar, P. Nguyen, Z. Chen, A. Kannan, R. J. Weiss, K. Rao, K. Gonina, N. Jaitly, B. Li, J. Chorowski, and M. Bacchiani, "State-of-the-art speech recognition with sequence-to-sequence models," *CoRR*, vol. abs/1712.01769, 2017. [Online]. Available: http://arxiv.org/abs/1712.01769

[3] D. Harwell, "The accent gap," *The Washington Post*, Jul 2018. [Online]. Available: https://www.washingtonpost.com/graphics/2018/business/alexa-does-not-understand-your-accent/

[4] A. Y. Hannun, C. Case, J. Casper, B. Catanzaro, G. Diamos, E. Elsen, R. Prenger, S. Satheesh, S. Sengupta, A. Coates, and A. Y. Ng, "Deep speech: Scaling up end-to-end speech recognition," *CoRR*, vol. abs/1412.5567, 2014. [Online]. Available: http://arxiv.org/abs/1412.5567

[5] F. G. J. S. e. a. Alex Graves, Santiago Fernandez, "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks," *ICML*, 2006. [Online]. Available: https://www.cs.toronto.edu/~graves/icml_2006.pdf

[6] D. A. et. al., "Deep speech 2: End-to-end speech recognition in english and mandarin," *CoRR*, vol. abs/1512.02595, 2015. [Online]. Available: http://arxiv.org/abs/1512.02595

[7] M. Hardt, E. Price, and N. Srebro, "Equality of opportunity in supervised learning," *CoRR*, vol. abs/1610.02413, 2016. [Online]. Available: http://arxiv.org/abs/1610.02413

[8] C. Wadsworth, F. Vera, and C. Piech, "Achieving fairness through adversarial learning: an application to recidivism prediction," *CoRR*, vol. abs/1807.00199, 2018. [Online]. Available: http://arxiv.org/abs/1807.00199

[9] Mozilla, "Mozilla common voice." [Online]. Available: https://www.kaggle.com/mozillaorg/common-voice