11685 Homework 5 Final Report

Jing Li
Heinz College of Information Systems and Public Policy
Carnegie Mellon University
Pittsburgh, PA 15213
`jingli4@andrew.cmu.edu`

Hongling Lei
Heinz College of Information Systems and Public Policy
Carnegie Mellon University
Pittsburgh, PA 15213
`hongling@andrew.cmu.edu`

Abstract

Automatic Speech Recognition is the task of transforming a speech waveform into a transcript sequence. Previous SOTA algorithms in this field mainly use supervised learning or semi-supervised learning, limiting the recognition to widely used languages only. However, in 2021, Baevski et al. instroduced a well performing unsupervised speech recognition algorithm called $wave2vec\_U$. The unsupervised property makes it possible to do automatic speech recognition on low-resource languages. We researched and implemented this method by ourselves, together with an ablation study attempting to improve the baseline performance.

1 Introduction

1.1 How GANs Work

Generative Adversarial Networks (GANs) were introduced by Ian J. Goodfellow and his colleagues in 2014. They are a type of generative model, which produce new content that is not from the original dataset but resembles its distribution. In other words, when a data point is given, GANs try to transform it from a simple distribution to the target distribution. This is done through the adversarial "battles" between a generator and a discriminator. The generator always tries to produce fake data that resembles the true data to fool the discriminator whose goal is to minimize the real-fake classification error. When the discriminator is unable to identify a piece of generated content as fake, it means that this new data really looks like the true data, achieving our goal.[1]

1.2 Unsupervised Speech Recognition Problem

The paper Unsupervised Speech Recognition (Baevski et al., 2021) introduces the wav2vec-U algorithm which solves the Automatic Speech Recognition (ASR) problem in an unsupervised manner. ASR is the task of transforming a speech waveform into a transcript sequence.

Previous SOTA algorithms in this field mainly use supervised learning or semi- supervised learning, where at least some of the transcript labels are given in the training data. However, as long as the training requires labeled data, the ASR technology is limited to a small fraction of the languages which are widely spoken. This is why highly accurate unsupervised speech recognition is a valuable breakthrough; it enables translation of any language - including the low-resource ones such as Kyrgyz, Swahili, and Tatar - which marks a new level of the ASR technology.[2]

The SOTA performance is achieved with the help of GANs whose generator creates a sequence of phonemes and discriminator tries to distinguish it from a real sequence.

2 Methodology

2.1 Dataset

Datasets used in the Unsupervised Speech Recognition paper can be categorized into two types.

The first category contains the TIMIT and Librispeech, both of which are benchmarks of English datasets. TIMIT was widely used in previous unsupervised speech recognition paper, but it is relatively small and thus limited. In contrast, Librispeech contains a significantly larger volume of speech audio. These two datasets combined, can be used to compare the model with previous unsupervised models and extrapolate the results with more comprehensive English datasets.

The second category contains non-English audio data. Multilingual LibriSpeech (MLS) contains eight popular European languages including French, German, Italian, and Spanish, while ALFFA and CommonVoice contain some rarely-used, low-resource languages like Swahili, Kyrgz, and Tatar. These datasets can be used to test out the model performance on various languages from different language families.

In our implementation, we only used a preprocessed version of the English LibriSpeech dataset.

2.2 Metrics

In Baevski's paper, cross-validation metrics are used for early stopping, random seed selection, and hyper-parameter tuning.

Negative Log-likelihood is used as an indicator of fluency for a given transcription measured with a language model p_{LM}.

$$NLL_{LM}(P) = -\frac{1}{M}\sum_{t=1}^{M} logp_{LM}(p_t)$$

Vocabulary Usage is the proportion of the phoneme vocabulary being output by the model via Viterbi decoding. Baevski et al. select the model configurations that produce high-scoring yet reasonably-long sequences.

$$P^* = \arg\max_{P'} \sum_{j=1}^{N_s}\sum_{t=1}^{M} logp_{LM}(p_t^j), M = |P^j|, P^j = [p_1^j, ..., p_M^j]$$

In our implementation, we used $edit\_distance$ (Levenshtein distance) as our metrics. Levenshtein distance is the minimum number of single-character modifications and is used to measure the difference between sequences.

2.3 Model Training Process

- Step 1: Get speech representations via wav2vec 2.0 from unlabeled speech audio.
- Step 2: Identify clusters at each time-step through k-means.
- Step 3: Segment audio data into phonemic units.
- Step 4: Build segment representations with mean-pooling, PCA, and a second mean-pooling.
- Step 5: Output phoneme sequences from the generator.
- Step 6: Feed this output to the discriminator to produce similar phonemized unlabeled text.
- Step 7: Perform GAN adversarial training.

3 Implementation

3.1 Baseline Model Architecture and Hyper-parameters

- Optimizer: Adam, β_1 = 0.5 for generator, β_2 = 0.98 for generator.

- Weight decay: None for the generator, 1e-4 for the discriminator.
- Learning rate: 1e-5 for the discriminator, 1e-4 for the generator.
- Steps: 150k steps alternating between the discriminator and the generator. In each step, a batch of 160 randomly selected unlabeled audio samples and 160 randomly selected unlabeled text samples are used.
- Discriminator: 3 casual convolutional blocks, hidden size = 384, kernel size = 6.
- Generator: Single non-casual convolution block with hidden size = |O| (number of phonemes), kernel size = 4, and dropout = 0.1.
- Hyper-parameters: Gradient penalty weight = [1.5, 2.0], smoothness penalty wight = [0.5, 0.75], phoneme diversity loss weight = [2, 4].

3.2 Implementation Details

The file $wave2vec\_U.py$ contains all the necessary components of the model, such as segmenter, discriminator, generator, as well as forward and backward processes in training.

We train the generator and the discriminator of the model alternately, so that both parts grows better at the same pace. This is achieved with the $set\_num\_updates$ function. When the epoch index is an odd number, the discriminator gets trained. Otherwise, the generator gets trained. Similarly, we should be careful with back-propagation by calling the corresponding optimizer separately for the generator and discriminator using $get\_groups\_for\_update$.

We evaluate our model on the validation dataset every 5 epochs. By comparing the current $edit\_distance$ with previous best distance, we keep our best model updated and saved for prediction and testing. We also keep track of the performance metric $edit\_distance$ under different architectures and parameter settings for ablation study.

In the test stage, predicted results need to be converted twice to be in the right final format. We first map $pred\_units\_arr$ to phonetic alphabetic characters according to $target\_dictionary$ and stitch the symbols together for each sentence. Then we convert the phonetic symbols to letters and numbers according to $phoneme\_map.json$ as our final predictions.

4 Experiments

Three different approaches were attempted as the ablation study in the effort to improve the model performance. The focus was mainly on the discriminator, since it is responsible for detecting true data and generated ones. Carefully designed optimizer can help the model converge faster and reach a higher prediction accuracy. Therefore, the first two attempts were contributed to finding a better optimizer for the baseline model. The learning rate and weight decay are the two factors considered in this study. However, these two attempts did not succeed in the first direction. Adding more layers could potentially capture more information and thus generate better results, so the depth of the discriminator was incremented to 3 in the last attempt.

- $discriminator\_lr$ = 0.0004
- $discriminator\_weight\_decay$ = 0.00001
- $discriminator\_depth$: int = 3

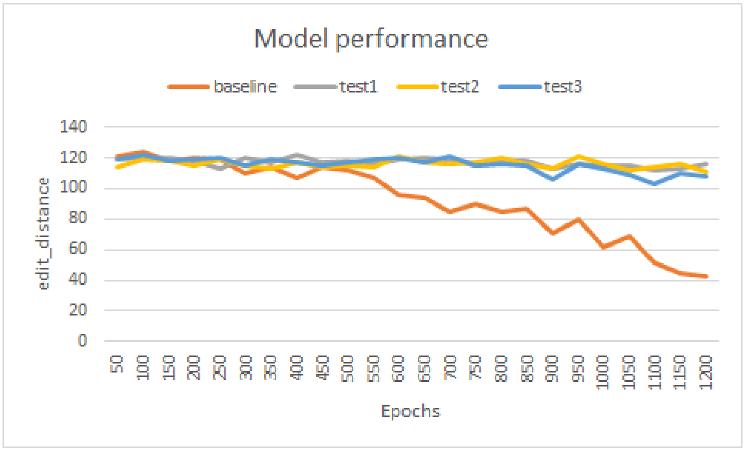

Figure 1: Ablation Study Visualization

5 Conclusion

In summary, unsupervised GANs can yield fruitful results in the speech recognition task. Although some of the techniques we tried here did not result in meaningful improvement of the baseline model, the baseline model itself already achieves a satisfying enough performance on the unsupervised recognition task.

References

[1] I. Goodfellow, J. Abadie, M. Mirza, B. Xu, D. Farley, S. Ozair, A. Courville, Y. Bengio, "Generative Adversarial Networks" (2014)

[2] A. Baevski, W. Hsu, A. Conneau, M. Auli, "Unsupervised Speech Recognition" (2021)