Fine-Tuning Unsupervised Wav2vec 2.0 With CTC

Name

Zhixin Cao

Major

Master's in Computer Science

Department / Affiliation

College of Natural Sciences, The University of Texas at Austin

ABSTRACT

Intrigued by several state-of-the-art ASR model and its powerful efforts on application that can be used in areas that benefit children or students who has disability or inability to read, this report presents in-depth details of wav2vec 2.0 – U model, including an elaborated explanation on wav2vec 2.0 model and the difference between it with unsupervised one. The report then proposes to use Connectionist Temporal Classification (CTC) for fine-tuning with TIMIT dataset. Although the result of the fine-tuning is not very competitive, it serves as interesting exercise on exploration of improving the performance on a dominant ASR model.

1. INTRODUCTION

Last Fall semester I had the opportunity to take Natural Language Processing (NLP) and got a good sense of how it is being wildly used for applications. This semester I was trying to build a children dictionary android application. When I was researching on it, I am impressed by how powerful Automatic Speech Recognition (ASR) could be and how useful and beneficial for students to utilize it for educational purpose. For example, according to(*Speech Recognition for Learning*, 2017) , the major advantages for using ASR for student writers are: "Improved access, writing production, mechanics of writing, increased independence, decrease anxiety, improvements in core reading and writing abilities". On a similar note, (*Speech To Text For*

Students With Disabilities ,Apps, Tools, and Software , 2022) has pointed out that ASR has been empowering young students who has dysgraphia or dyslexia, especially when they want to write a significantly large essay. As a result, I am intrigued by several state-of-the-art ASR models and this report is mainly conducted by using one of them, wac2vec 2.0.

2. RESEARCH BACKGROUND

2.1 wav2vec 2.0

Wav2vec 2.0 is built by learning contextualized representations of the speech units through Transformers layer with unlabeled data (Baevski et al., 2020). Impressively it achieved a very competitive word error rate of less than 5% of LibriSpeech -cf, with fine-tuning using as little as 10 minutes of labeled data during evaluation with pre-trained checkpoints. (Platen, 2021)

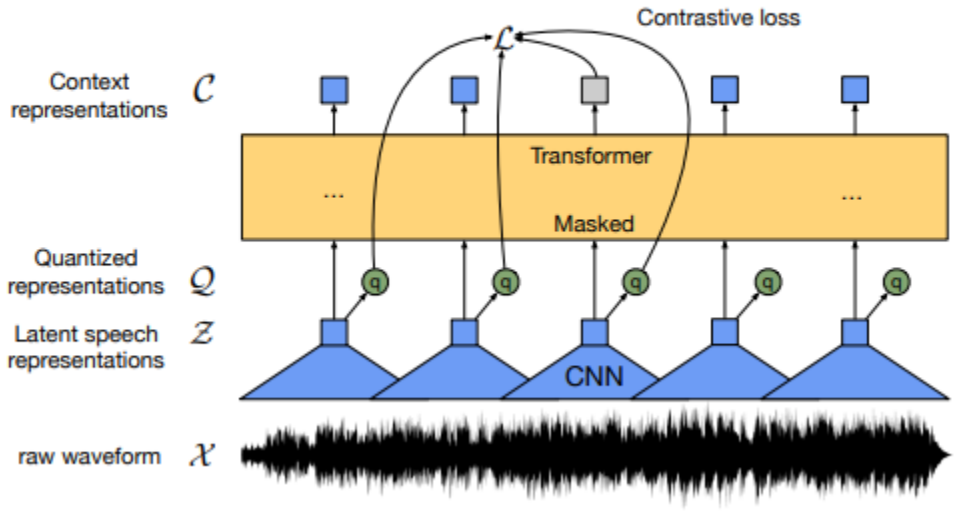

Figure 1: Illustration of our framework which jointly learns contextualized speech representations and an inventory of discretized speech units. (cited from (Baevski et al., 2020))

2.2 wav2vec-Unsupervised (wav2vec-U) 2.0

While wav2vec-U (Baevski et al., 2021) use speech representation generated from wav2vec 2.0 (Baevski et al., 2020) and takes additionally pre-processing steps, such as k-means clustering, PCA reduction, merged adjacent PCA features, and mean pooling features, wav2vec-U 2.0 has several modifications as well (Liu et al., 2022). For example, wav2vec-U 2.0 does not have pre-processing steps, instead, it adopts Generative Adversarial Network training as part of the generator and

introduces auxiliary objective to enhance the segmental transcriber, according to (Liu et al., 2022). More details of the major differences between wav2vec-U and wav2vec-U 2.0 are illustrated in Figure 2, as cited from (Liu et al., 2022).

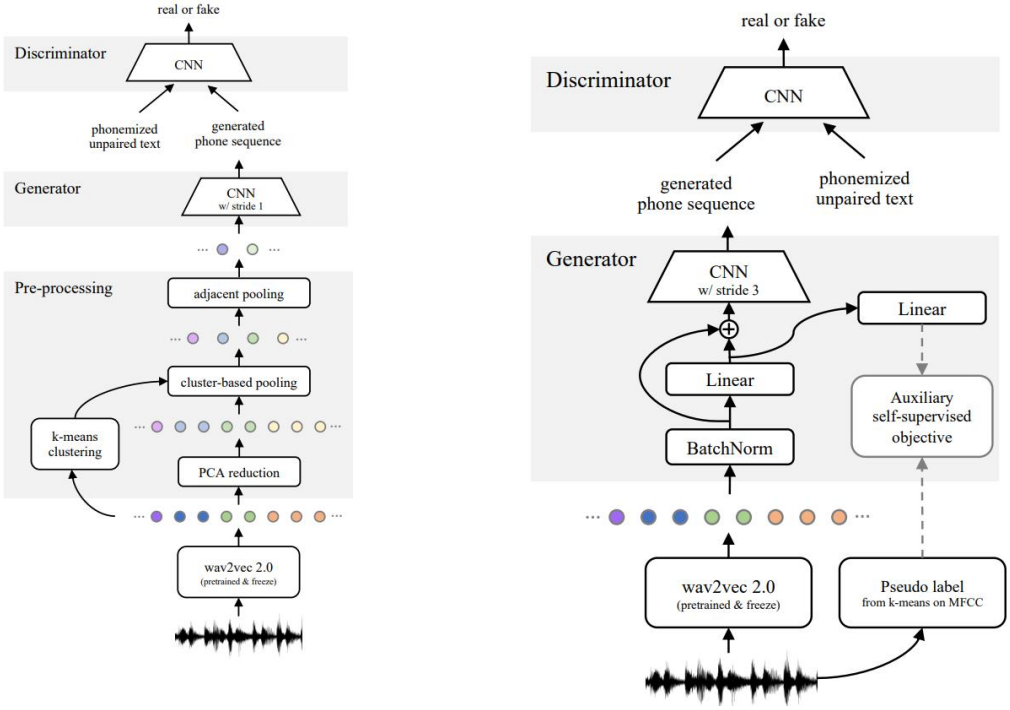

Figure 2: Comparison flowcharts between wav2vec-U (left) and wav2vec-U 2.0 (right)

Inspired by the fact that wav2vec acoustic model without language model could have a better performance than the wav2vec model with it (Platen, 2021), this report will mainly use the standalone wav2vec and fine-tune it by with Connectionist Temporal Classification (CTC)

2.3 Connectionist Temporal Classification (CTC)

Connectionist Temporal Classification is an alignment-free algorithm that given input X, it can generate a output distribution for all potential output Ys. Then we could use the distribution, in this case, it is calculated by loss function, to infer a highly likely expected output (Hannun, 2017). The key component of CTC is, it initiates multiple blank tokens into a set of expected output Ys, and then remove them at the end (Hannun, 2017). In this way, it allows any alignment by mapping any given input X to various output Y by performing similar steps (Hannun, 2017). CTC is extremely useful to overcome alignment issues when we train Recurrent Neural Networks (RNN) for sequence-to-sequence (Platen, 2021). A simple example are demonstrated in Figure 3 ().

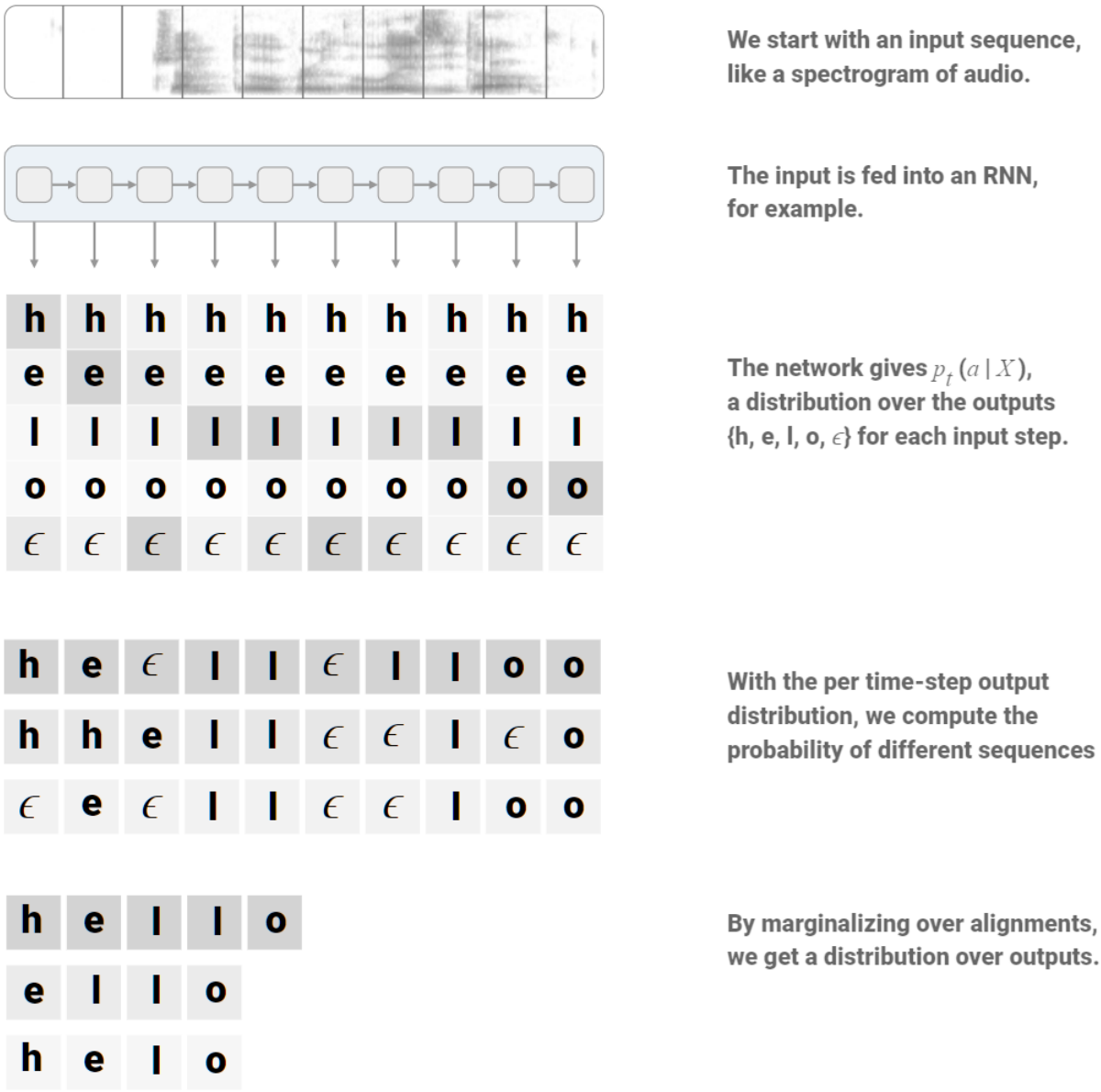

Figure 3: A simple example of how CTC process input data “hello”

3. MATERIALS / DATA / SOURCES

In this report, we mainly use TIMIT dataset

3.1 TIMIT Acoustic-Phonetic Continuous Speech Corpus (TIMIT)

TIMIT contains 630 speakers of eight different main dialects from American English. Each recording ten phonetically various sentences in a form of 16kHz speech waveform. This report uses it for fine-tuning wav2vec-U 2.0. (Hugging Face. 2022)

4. METHODS

4.1 Preparation of pretrained model, speech representations and text data

According to the fairseq documentation (Fairseq Documentation, 2022) , the first step is to load pretrained wav2vec-U 2.0 model from the library. In this case, I used google colab with GPU to load the wav2vec-U 2.0.

Once it is done, the next step is to prepare a feature extractor and tokenizer, given any ASR model is mainly used to transcribe audio to text, we need to have a feature extractor to process the input data, while a tokenizer will process the output data in the form of text (Platen, 2021). In Hugging Face Transformer library, there are pretrained Wav2Vec2FeatureExtractor and Wav2Vec2CTCTokenizer given for the wav2vec2 model (https://huggingface.co/docs/transformers/main/model_doc/wav2vec2)

- Preprocess TIMIT Data

  Before we initiate new feature extractor and tokenizer, we need to prepare TIMIT data. Based on (Fairseq Documentation, 2022), the audio data needs to be preprocessed to match the phonetized output text data better. In order to do that, I added multiple preprocessing steps, such as remove stop word, remove_special_characters (“ , . / ? ”), and make all letter lowcase.

don't ask me to carry an oily rag like that
don't ask me to carry an oily rag like that
a huge power outage rarely occurs
the hood of the jeep was steaming in the hot sun
i just saw jim near the new archeological museum
would you allow acts of violence
those answers will be straightforward if you think them through carefully first
they all agree that the essay is barely intelligible
to honor him is to honor ourselves
we plan to build a new beverage plant
it's impossible to deal with bureaucracy
ironically enough in this instance such personal virtues were a luxury
woe betide the interviewee if he answered vaguely
lips pursed mournfully he stared down at its crazily sagging left side
don't ask me to carry an oily rag like that

Figure 4: Random examples of TIMIT dataset

- Generate wav2vec2CTCTokenizer

  As elaborated above, the key component of CTC algorithm is to have blank token to match any input data X to a various range of output data Ys (Hannun, 2017). To do that, we first create a Counter to track all unique letters and its frequency in the TIMIT dataset. Next, we create permutation of those letters and add padding that similar to blank token concept in CTC for each of generated sequences. All generated vocabularies then could be saved as a json file to feed the Hugging Face Wav2Vec2CTCTokenizer as a input file.

- Generate wav2vec2 Feature Extractor

  One major factor when we process any audio data is to make sure each sample have the same sampling rate. In our case, wav2vec-U 2.0 model was pretrained on LibriSpeech data which has

16kHz, and the same as TIMIT dataset. As a result, we could directly instantiate the wav2vec2 Feature Extractor without much implication.

Once both feature extractor and tokenizer are created, we can then bundle them wav2vec2Processor from Transformers.

4.2 Training

According to (Baevski et al., 2020), the first component of wav2vec2, which is a slack of CNN layers are already sufficiently trained and does not need to fine-tune it anymore. Therefore, the part of the model can be frozen.

The main training takes place by initializing the TrainingArguments and then pass respective parameters to Trainer function from Transformers (Trainer, 2022), with some customed parameters defined. Referencing the example that provided here (GitHub, 2022), I set up data collator which can treat input raw data and golden labels independently and apply various padding functions on them respectively. This seems to be needed as input audio and output text have different modalities (Platen, 2021). With some other minor modifications of training arguments, such as learning_rate and num_train_epochs, the training part is set to go.

4.3 Evaluation

Although TIMIT dataset is mainly designed to be evaluated based on phonetic error rate, most of the ASR models are using Word Error Rate (WER) as the primary factor to judge performance of the models predominantly (Baevski et al., 2021). As a result, we use WER as our primary metric for evaluation as well. In this report, we as well use TIMIT test dataset for evaluation purpose.

After training, the model is supposed to generate a list of vectors with various predicted output Ys for each of the input sequence. Based on (Hannun, 2017), it is suggested to take the argmax of the overall predicted outputs for a highest possibility of a prediction, then removing its blank token that we added when we initiated the wav2vec2CTCTokenizer.

With that in mind, we calculated the predicted_id for each input data and evaluated the performance by comparing the predicted output data with the output data's golden labels.

5. RESULTS

One of the crucial challenges for fine-tuning the model was the amount of time it took to finish the training, even with a GPU that allocated to my google colab. Few times, it occurred to me that the colab stopped running due to inactivity. So I had to be physically monitoring the progress of the training. That's said, after trained it few times, it was able to reach an average WER at around 24%.

Step	Training Loss	Validation Loss	Wer
500	5.318100	2.984530	0.981876
1000	2.116300	0.355762	0.348839
1500	0.604300	0.239367	0.268693
2000	0.394100	0.227759	0.260837
2500	0.408700	0.214905	0.258287
3000	0.272900	0.223457	0.252980
3500	0.231900	0.234446	0.255944
4000	0.238300	0.268756	0.255806
4500	0.217500	0.256460	0.256082
5000	0.206000	0.230928	0.252774
5500	0.178700	0.250742	0.251120
6000	0.139600	0.265004	0.260285
6500	0.142700	0.261732	0.258562
7000	0.128100	0.262379	0.251327
7500	0.153200	0.252113	0.256840
8000	0.106200	0.275274	0.258287
8500	0.111000	0.268902	0.250155
9000	0.095400	0.254680	0.250637
9500	0.084500	0.309215	0.249879
10000	0.108500	0.267500	0.255117
10500	0.087500	0.277350	0.247605

Figure 5 Evaluation Results

6. DISCUSSION

Although in the paper from (Baevski et al., 2021) there is no direct evaluation result from TIMIT dataset, they conducted analysis on LibriSpeech, which is a well-contrusted English primary benchmark dataset. The following are the results:

Table 3: Word Error Rate (WER) on LibriSpeech with different language models (LM) on the standard LibriSpeech dev/test sets.

Model	Unlabeled speech (hours)	LM	dev		test	
			clean	other	clean	other
Supervised learning w/ 960 hours of speech						
DeepSpeech 2 [34]	-	5-gram	-	-	5.33	13.25
Fully Conv [35]	-	ConvLM	3.08	9.94	3.26	10.47
TDNN+Kaldi [36]	-	4-gram	2.71	7.37	3.12	7.63
SpecAugment [18]	-	RNN	-	-	2.5	5.8
ContextNet [2]	-	LSTM	1.9	3.9	1.9	4.1
Conformer [1]	-	LSTM	2.1	4.3	1.9	3.9
Semi-supervised learning w/ 960 hours of speech						
Transf. + PL [26]	54k	CLM+Transf.	2.00	3.65	2.09	4.11
IPL [37]	54k	4-gram+Transf.	1.85	3.26	2.10	4.01
NST [38]	54k	LSTM	1.6	3.4	1.7	3.4
wav2vec 2.0 [15]	54k	Transf.	1.6	3.0	1.8	3.3
wav2vec 2.0 + NST [39]	54k	LSTM	1.3	2.6	1.4	2.6
Unsupervised learning						
wav2vec-U	54k	4-gram	13.3	15.1	13.8	18.0
wav2vec-U 2.0	54k	4-gram	9.8	13.1	9.9	13.9
Unsupervised learning + Self-Training						
wav2vec-U	54k	4-gram	3.4	6.0	3.8	6.5
wav2vec-U 2.0	54k	4-gram	3.5	6.0	3.7	6.3

Figure 6 Evaluation Results on LibriSpeech (cited from (Baevski et al., 2021))

As it is shown, the test results for unsupervised learning for wav2vec-U 2.0 was about 13.9 % from (Baevski et al., 2021) on LibriSpeech dataset, while the evaluation result on this fine-tuning exercise is about 24 % on TIMIT dataset.

I made few attempts on changing the parameters for training, such as increase the epoch size as well as batch size, but the results were still around 24%.

Step	Training Loss	Validation Loss	Wer
11500	0.093400	0.274820	0.242230
12000	0.072800	0.290904	0.249879
12500	0.078800	0.265965	0.245055
13000	0.063300	0.266540	0.245469
13500	0.081000	0.275284	0.245055
14000	0.092400	0.260122	0.241748
14500	0.056600	0.267841	0.241472

Figure 7 More Evaluation Results from fine-tuning

Given the time constraint of this report, I was not able to further investigate on why there was almost 10% gap on the performance. One observation though, going through few times of testing and debugging, I did notice various combinations of training arguments made a difference on the WER results.

7. CONCLUSION

It has been an interesting journey from first encounter the concept of ASR and wav2vec, to conduct detailed analysis on wav2vec 2.0 – U. Even though the fine-tuning performance from this report is not very competitive, I am still very impressed and excited on the continuously efforts and progress that researchers haven been making toward a perfect ASR model. I also noticed there is emerging interests on how to apply ASR that trained on English primarily to other languages. Apart from the making the ASR more and more advanced, I believe it is also important to apply such a useful model in meaningful application. Following up on how ASR can benefit students with disability or even younger children who cannot read yet, there must be lots of areas and opportunities waiting to explore.

REFERENCES

1. Reading Rockets. 2017. Speech Recognition for Learning. URL https://www.readingrockets.org/article/speech-recognition-learning
2. LDRFA. Speech To Text For Students With Disabilities ,Apps, Tools, and Software. 2022. URL https://www.ldrfa.org/speech-to-text-apps-tools-and-software/
3. Alexei Baevski, Henry Zhou, Abdelrahman Mohamed, Michael Auli. 2020. wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations.. URL https://arxiv.org/abs/2006.11477
4. Patrick von Platen. Fine-Tune Wav2Vec2 for English ASR with 🤗 Transformers. 2021. URL https://huggingface.co/blog/fine-tune-wav2vec2-english
5. Alexei Baevski, Wei-Ning Hsu, Alexis Conneau, and Michael Auli. Unsupervised speech recognition. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, Advances in Neural Information Processing Systems, 2021. URL https://openreview.net/forum?id=QmxFsofRvW9.
6. Alexander H. Liu, Wei-Ning Hsu, Michael Auli, Alexei Baevski. 2021. Towards End-to-end Unsupervised Speech Recognition. URL https://arxiv.org/abs/2204.02492
7. Awni Hannun. 2017. Sequence Modeling With CTC. URL https://distill.pub/2017/ctc/
8. timit_asr · Datasets at Hugging Face. 2022. URL https://huggingface.co/datasets/timit_asr
9. fairseq documentation - fairseq 0.12.2 documentation. 2022. Fairseq Documentation. URL https://fairseq.readthedocs.io/en/latest/
10. Trainer. 2022. URL https://huggingface.co/docs/transformers/main_classes/trainer
11. GitHub. 2022. URL https://github.com/huggingface/transformers/blob/9a06b6b11bdfc42eea08fa91d0c737d1863c99e3/examples/research_projects/wav2vec2/run_asr.py#L81