LeNet-CTC Network for Phonetic Sequence Prediction

Madhumitha Balaji
1004471
madhumitha_balaji@mymail.sutd.edu.sg

Wang Han
1004520
han_wang@mymail.sutd.edu.sg

Tan Jianhui
1004380
jianhui_tan@mymail.sutd.edu.sg

***Abstract*—This report presents the implementation and testing of multiple models for the prediction of phonetic sequences in English speech. The primary objective of the project is to apply machine learning techniques to accurately and automatically output a sequence of English phonemes – the building blocks of how a word is pronounced – from an input audio wav file. Thereafter, different model architectures were explored and tests conducted to assess the effectiveness of the model, while tweaks of hyperparameters were also done to observe their effect on performance.**

The results of the project showed that of the few models written for this project, a LeNet_5 based model seemed to strike a good balance between performance and training time. In addition, increasing the `n_mels` parameter for the Melspectrograms extracted from the audio data could potentially lead to better performance, although the Short-Time Fourier Transform (STFT) could also be considered a good (and potentially even better) alternative as the extracted feature for automatic speech recognition. Character prediction, as opposed to phoneme prediction (which is mainly used over the course of this project), should also be seriously considered as the better alternative as an output.

Through the project, machine learning techniques such as convolutional neural networks and audio feature extraction methods like filter banks and spectrograms were studied and applied. This project served as a preliminary foundation for the exploration of improving current performance in phonetic prediction.

I. Introduction

Recent work in the automatic speech recognition field has brought the field to greater heights than ever before. On May 21 2021, Facebook and Google researchers released a paper introducing *wav2vec-U*, a jointly-developed, unsupervised learning model that is built upon Facebook's previous wav2vec models, which had achieved astounding results in the field of speech recognition [1]. While *wav2vec-U*, as an unsupervised learning model, still lags behind the performance of supervised models, it posted the best results seen so far among unsupervised models, while also demonstrating effectiveness with other 'less commonly' spoken languages. With such improvements in the field, current applications that make extensive usage of speech recognition, such as automatic captioning, promises to become ever more accurate in its functions.

Automatic speech recognition is commonly understood as the automatic sequencing of audio speech into words. At its core, one way to accomplish this using machine learning models is to train on the recognition of phonemes, which are the building blocks of how each word, no matter the language, is pronounced. Though it may sound straightforward, applying them in real-life applications is no easy matter. The first issue arises in terms of sequencing recorded speech into words; since everyday speech is mixed with pauses, filler words, and even creolised words, the very initial step of separating them into individual words represents a huge challenge that researchers are still in the midst of optimising. Secondly, there is the inevitable issue of noise in the data, be it background noise or multiple people talking at the same time. Other research gaps exist as well, such as most current research being conducted on mono-aural data; binaural data could present vast amounts of untapped potential, especially in the field of autonomous robots, but of course this increases computing complexity [2]. All of these are brief descriptions of the obstacles facing the speech recognition field and is merely the tip of the iceberg, but at the same time, there is great enthusiasm and hope at the promise this field has displayed.

Currently, the project provides a foundation for the exploration to improve on current performance in English phonetic sequence prediction through the writing and testing of different models and parameter tunings. The remainder of this paper is organised as follows: Section II will provide a brief literature review of existing work, while Section III discusses the dataset used in the project. Section IV details the pre-processing work done on the aforementioned dataset, and Section V will list the problem definition as well as the layers of the models written. Sections VI, VII, and VIII are where the obtained results are discussed and analysed, as well as the parameters that were tweaked to test for performance gains. Finally, Section IX rounds out with the conclusion. **The source code for the framework and experiments outlined in this paper can be found at https://github.com/Madhu-balaji-01/ASRProject.**

II. Related Work

1) ***Traditional ASR approach***: referring to paper [4], split problem into multiple tasks and chain together as pipeline.
   - Extract features from audio file. Require windowing (to filter and aggregate through data) and performing

transformations on the windows, such as Fourier transformations
- Use acoustic model to output the phonemes.
- Use language model that uses probability distributions to predict chars or words from the phonemes.

2) ***Deep learning approach***: referring to paper [5], replace the multiple tasks in traditional ASR approach with one algorithm that take in audio file and output chars or words.
3) ***Our model:*** our model is similar to traditional ASR approach.
   - Similarity: have separate features extracting model, acoustic model and language model chained together.
   - Difference: instead of take in raw audio file as input in features extracting model, we have one more data pre-processing layer. Which convert audio data to spectrogram image. Then we can use CNN(a image processing model) to extract features.

III. Dataset and Collection

The dataset used in the project will be the TIMIT dataset. A joint effort among the Massachusetts Institute of Technology (MIT), Stanford Research Institute (SRI), and Texas Instruments (TI) in 1986, the TIMIT dataset is an established, well-known dataset for speech recognition tasks. In particular, the TIMIT dataset has detailed annotations and transcripts of the texts recorded, and is a big reason why it was chosen as the dataset for this project.

In total, there are 630 speakers in the TIMIT dataset – all of them American – with 6 distinct dialects recognised (DR1 to DR6, with DR7 being 'Unknown' and DR8 being 'Moved around'). Of interest are the types of sentences tagged in the dataset: phonetically-compact (audio files prefixed with "SX"), phonetically-diverse ("SI"), and dialect ("SA"). Phonetically-compact sentences, according to the documentation for the dataset, refers to sentences "designed to provide a good coverage of pairs of phones", while phonetically-diverse sentences "add diversity in sentence types and phonetic contexts" as they are drawn from existing text sources. Dialect sentences, however, are specifically meant to reveal and emphasise the dialects of the speakers, and thus will not be included in either the training set or the test set.

Every audio file, and by extension every sentence, in the dataset is labelled with its time-aligned word and phonetic transcripts. For example, the following figure is the transcription for sentence 'SX70' which reads "Did you eat yet?". The first number indicates the beginning integer sample number (sampled at 16kHz) for the segment. The second number represents the ending integer sample number for that segment. The text that follows represents the phonetic label or the word.

The TIMIT dataset does, however, exhibit a slight bias for male speakers, with there being more samples for male speakers than that for female speakers. Most sentences in the dataset also have between 7 to 11 words. These statistics are recorded in Appendix A. Training and test split is set at approximately 75% to 25%.

```
0 2120 h#                    
2120 2648 d                  
2648 3320 ix                 
3320 4370 dcl                
4370 5389 jh                 
5389 6767 ux                 
6767 9315 iy                 
9315 11280 tcl               
11280 12653 y          2120 5389 did
12653 15400 eh         4370 6767 you
15400 16380 tcl        6767 11280 eat
16380 17360 h#         11280 16380 yet
```

Figure 1. (Left) Phonetic sequence of sentence SX70 (h# represents the start and end), (Right) Word sequence of sentence SX70

IV. Dataset Pre-processing

A. Input pre-processing

An audio signal is simply a sequence of variations in air pressure with respect to time. This is captured digitally by sampling the sound wave at regular time intervals and measuring the intensity or amplitude of the wave at each sample. The number of samples per second gives us the sampling rate. Figure 2 shows the waveform plot for the audio file 'SA1.wav' from Dialect Region 1 (New England) of TIMIT dataset. Capturing the amplitude alone, however, does not give us sufficient information. Fourier transform is a mathematical transform that decomposes a time series into a sum of finite series of sine or cosine functions. In other words, it decomposes a time domain signal to its individual frequencies resulting in a frequency domain, or in other words, a spectrum. Applying the Fourier transform to short time windowed segments of our audio signal and overlapping them gives us a spectrogram - a snapshot of the signal's frequencies as it varies over time. The spectrogram of 'SA1.wav' is shown in Figure 3. However, we notice that there is not much information for us to see in this figure. This is due to the way humans perceive sound. The human perception of sound is concentrated in very small frequency. Furthermore, humans do not perceive frequencies linearly. We are more sensitive to differences between lower frequencies than higher frequencies. The Mel scale was developed taking this into account. It is a scale of pitches that intends to regularize the intervals between any two given notes. Melspectrograms use the Mel scale for frequency and Decibel scale instead of amplitude to represent a spectrogram. Figure 4 visualizes the Melspectrogram of 'SA1.wav'.

All input audio files were transformed into their corresponding Melspectrograms and min-max normalization was applied imagewise before being fed into the LeNet-CTC network. Spectrogram conversions were done on-the-fly using the nnAudio [3] toolkit.

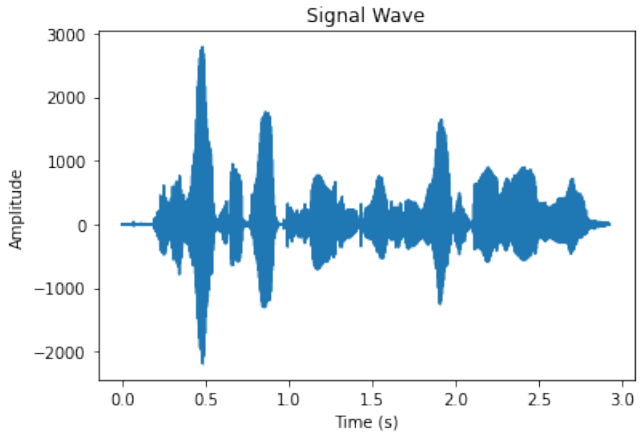


Figure 2. Waveform plot of SA1.wav

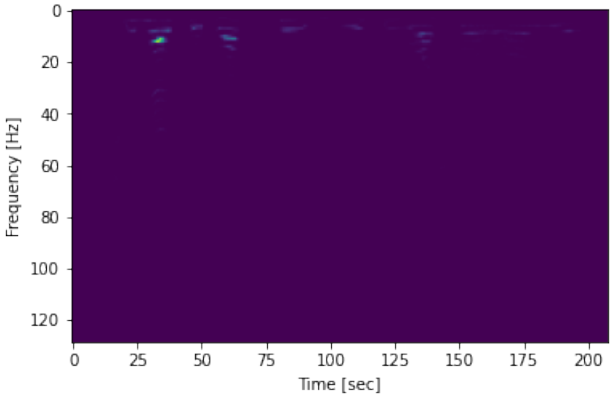


Figure 3. Spectrogram of 'SA1.wav'

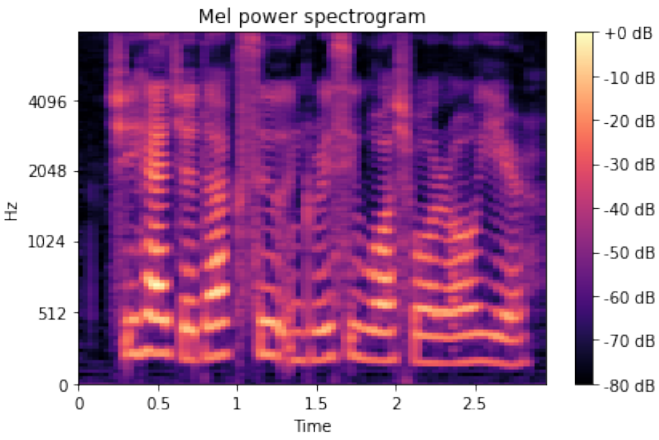


Figure 4. Melspectrogram of 'SA1.wav'

B. Output pre-processing

The TIMIT dataset has time-aligned phonetic labels as explained in the previous section. To make these labels machine-understandable, we first split them into phonemes or characters depending on the output mode desired, and use the phoneme dictionary or character dictionary specified in the previous section respectively to map text sequences into integer sequences.

V. Problem and Model

This project aims to explore ways to improve on current performance in English phonetic sequence prediction through machine learning techniques. Architecture shakeups and parameter tunings will be the bulk of the work done for the project, and thereafter, their extensions to other applications such as character/word prediction will be examined.

We explored several algorithms to achieve a test loss as low as possible. The architecture of each of our models is summarized below.

1) SimpleLinear: 1 linear layer
2) SimpleLSTM: 1 LSTM unit
3) CNN_LSTM:
   - Convolution layer(ReLU activation function)
   - Convolution layer(ReLU activation function)
   - Max-pooling layer
   - Convolution layer(ReLU activation function)
   - Max-pooling layer
   - Bidirectional LSTM
   - Linear fully connected layer
4) LeNet_5:
   - Convolution layer(ReLU activation function)
   - Avg-pooling layer
   - Convolution layer(ReLU activation function)
   - Avg-pooling layer
   - Fully connected unit with 3 x Linear layers
   - Linear layer
5) BiGRU:
   - Layer Normalization function
   - GELU (Gaussian Error Linear Unit) layer
   - Bidirectional Gated Recurrent Unit with 2 stacked layers
   - Layer Normalization function
   - Dropout of rate 10%
   - Linear layer
6) VGGish:
   - Convolution layer
   - Maxpool layer
   - Convolution layer
   - Maxpool layer
   - 2 x Convolution layer
   - Maxpool layer
   - 2 x Convolution layer
   - Maxpool layer

- Fully connected unit with 3 x Linear layers and ReLU activation
- Linear layer

7) Residual_CNN:
   - Layer Normalization function
   - GELU layer
   - Dropout of rate 10%
   - Convolution layer
   - GELU layer
   - Dropout of rate 10%
   - Convolution layer
   - Skip connection
     - Input of the first layer passed through a Convolution layer
     - Matrix addition with the output of final layer

VI. Evaluation Methodology

The input to our model is a sequence of audio samples at each time step, and the outputs are a sequence of labels corresponding to phonemes, which can include blank outputs. This leads to difficulty in training because there will be many more input time steps than there are labels. For example, in speech audio there can be multiple time slices which correspond to a single phone. Since we don't know the alignment of the observed sequence with the target labels, we predict a probability distribution at each time step. The Connectionist Temporal Classification (CTC) network outputs the probability distribution over all possible labels at each time step (produced using a softmax layer). CTC loss is the negative logarithm of this probability. To analyze the prediction error, we must convert these probabilities back to text. We use Greedy Decoding algorithm for this purpose which takes the probability distribution calculated by CTC and chooses the option with the highest probability (argmax). This is compared with the ground truth labels to obtain the word error rate.

Word error rate (WER) is a common metric of the performance of a speech recognition. The WER is derived from the Levenshtein distance and is calculated using the equation $WER = \frac{S+I+D}{N}$ where S is the number of substitutions, I is the number of insertions, D is the number of deletions and N is the total number of words spoken. Since we are working at the phoneme level, we report the phonemic error rate (PER) calculated the same way. PER calculations were done efficiently using the `fastWER` package [6].

VII. Results and Analysis

The prediction performance of the models listed in section V are described in Table I. Before feeding to each mode, the input was pre-processed in the same way – raw audio transformed into its Melspectrogram with `n_fft = 400` and `n_mels = 80`. Each of the models was trained for 30 epochs with learning rate 1e-5, optimized with CTC loss and Adam optimizer.

Model	Test PER	Test loss	Time Taken
SimpleLinear	1.0	3.876	10 min 46 msec
SimpleLSTM	1.0	2.836	45 min 13 sec
CNN_LSTM	0.674	3.938	31 min 9 sec
LeNet_5	0.999	3.732	17 min 8 sec
BiGRU	0.982	3.161	39 min 11 sec
VGGish	0.862	2.619	247 min 12 sec
Residual_CNN	1.0	3.892	10 min 6 sec

Table I
Test performance of different model architectures

As we can observe from Table I, if sorted based on test loss, the performance follows the order: $VGGish > SimpleLSTM > BiGRU > LeNet\_5 > SimpleLinear > Residual\_CNN > CNN\_LSTM$. When sorted according to time saved in training, $SimpleLinear > Residual\_CNN > LeNet\_5 > CNN\_LSTM > BiGRU > SimpleLSTM > VGGish$. The models that achieved the lowest PER – CNN_LSTM and VGGish – unfortunately take significantly longer times to train.

SimpleLSTM had one of the lowest losses, however, it runs LSTM directly on the spectrogram data, which may need hundreds of millions parameters to optimize, thus causing it to take very long to train. This is even more apparent in the VGGish model, which achieved the lowest loss and the second lowest PER. There was an estimated 7.8M parameters to be trained in the VGGish model due to its deeper structure, thus explaining why it took over 4 hours to train just 30 epochs. BiGRU also achieves a low PER and loss. It is also more computationally efficient that SimpleLSTM. However, it takes almost four times as long to train when compared to some of the simpler, CNN-based models such as SimpleLinear and ResidualCNN. These simple CNN models, however, fall behind when it comes to test loss.

There is a trade-off between test error/loss achieved by the model and the time it takes to complete its training process. While training for higher number of epochs may result in further learning and thereby a decrease in the PER and loss, this significantly increases the requirement of resources - time, memory and computation power. The chosen model for this project focuses on achieving a balance between time taken for training and the 'potential' to improve its performance based on its test loss and PER. LeNet_5 has the best standing in this regard. LeNet_5 is one of the earliest proposed CNN models. It is simple and shallow, but contains the basic modules of deep learning. It has convolution layers to reduce data sizes and extract features, and has fully connected layer to learn features without making the network too deep. Perhaps as a result of the above, this model displayed middling results for PER, test loss, and training time.

We visualized the output generated by each layer of LeNet_-5 in epoch 1 by plotting intermediate activations - this helps us see what features are detected or preserved by the layers.

We used Tensorboard to produce the feature maps in Figure 5. The 16 sub-plots that we see here correspond to the batch size. Ideally, feature maps produced by layers close to the input should detect small or fine-grained detail, whereas feature maps close to the output of the model capture more general features. Figure 5(a) shows the input Melspectrograms for the batch. This is fed to the CNN layer whose output is shown in Figure 5(b) - this output shows lot of activations which implies that the layers successfully extracted a considerable portion of the features. In the fully connected unit, as we can see in Figure 5(c), there is lesser detail as compared to previous layers. This helps our model generalize well. The output of final Linear layer looks quite similar to that of the FC unit - this means that there is not too much new information learnt by this last year. The increased highlighting in Figure 5(d) over the same area as Figure 5(a) shows that the model has extracted relevant features.

VIII. Experiments

To test the robustness and generalizability of our proposed architecture, we conducted the following experiments and evaluated its performance under different conditions. These tests were conducted on the LeNet_5 model, chosen as the to-be-further-explored model as stated in Section VI, with CTC loss trained for 30 epochs with a learning rate of 1e-5. These same tests were also ran on the BiGRU model under the same conditions to provide a point of comparison.

A. Melspectrogram parameters

During pre-pocessing of data, raw audio is transformed into Melspectrograms. As mentioned in section IV, Fourier transform is applied to windowed segments of the signal. The size of these windowed segments is controlled by the parameter `n_fft`. To obtain a Melspectrogram from this spectrogram, a Mel filter bank has to be used. This is controlled by the parameter `n_mels`, which divide the entire frequency spectrum into that number of Mel-frequency bins that are evenly spaced in terms of distance as heard by the human ear (`n_mels` commonly ranges between 40 and 130). The final melspectrogram is then obtained by taking the matrix multiplication of this Mel filter bank and the spectrogram.

Results: Different combinations of `n_fft` and `n_mels` are experimented on, and their results as well as their respective training time are reported below in Table II and Table III. These tests were conducted using phoneme sequence prediction.

n_fft	n_mels	Test PER	Test Loss	Time Taken
200	80	1.0	3.780	16 min 56 sec
800	80	0.999	3.778	18 min 40 sec
400	80	0.999	3.732	17 min 8 sec
400	40	1.0	3.922	12 min 47 sec
400	160	0.955	3.221	26 min 41 msec

Table II
Results obtained with different Melspectrogram parameters for LeNet_5 model

n_fft	n_mels	Test PER	Test Loss	Time Taken
200	80	0.991	3.142	38 min 56 sec
800	80	0.996	3.311	40 min 43 sec
400	80	0.982	3.161	39 min 11 sec
400	40	0.999	3.443	39 min 45 msec
400	160	0.981	2.982	39 min 42 sec

Table III
Results obtained with different Melspectrogram parameters for BiGRU model

Discussion: For the LeNet_5 model, as seen from Figure II, decreasing the number of `n_mels` reduced the training time but increased loss incurred, while increasing led to an improved loss and PER at the cost of increased training time. One preliminary reason to explain this phenomenon can be attributed to the fact that since `n_mels` determine the number of 'bins' for the frequencies to be separated into, a larger number of bins provided a greater degree of separation/granularity in the consequent Melspectrogram, thereby improving the training and classification process. Notably, modifying `n_fft` increased loss either way although training time stayed roughly similar.

For the BiGRU model in Figure III, modifying `n_fft` saw an increase of test PER in either direction, but loss was observed to go down as `n_fft` decreased. When modifying `n_mels`, results were similar to the LeNet_5 model: a decrease of `n_mels` led to an increased loss and PER. Training times were noted to have stayed fairly consistent in spite of the parameter modifications.

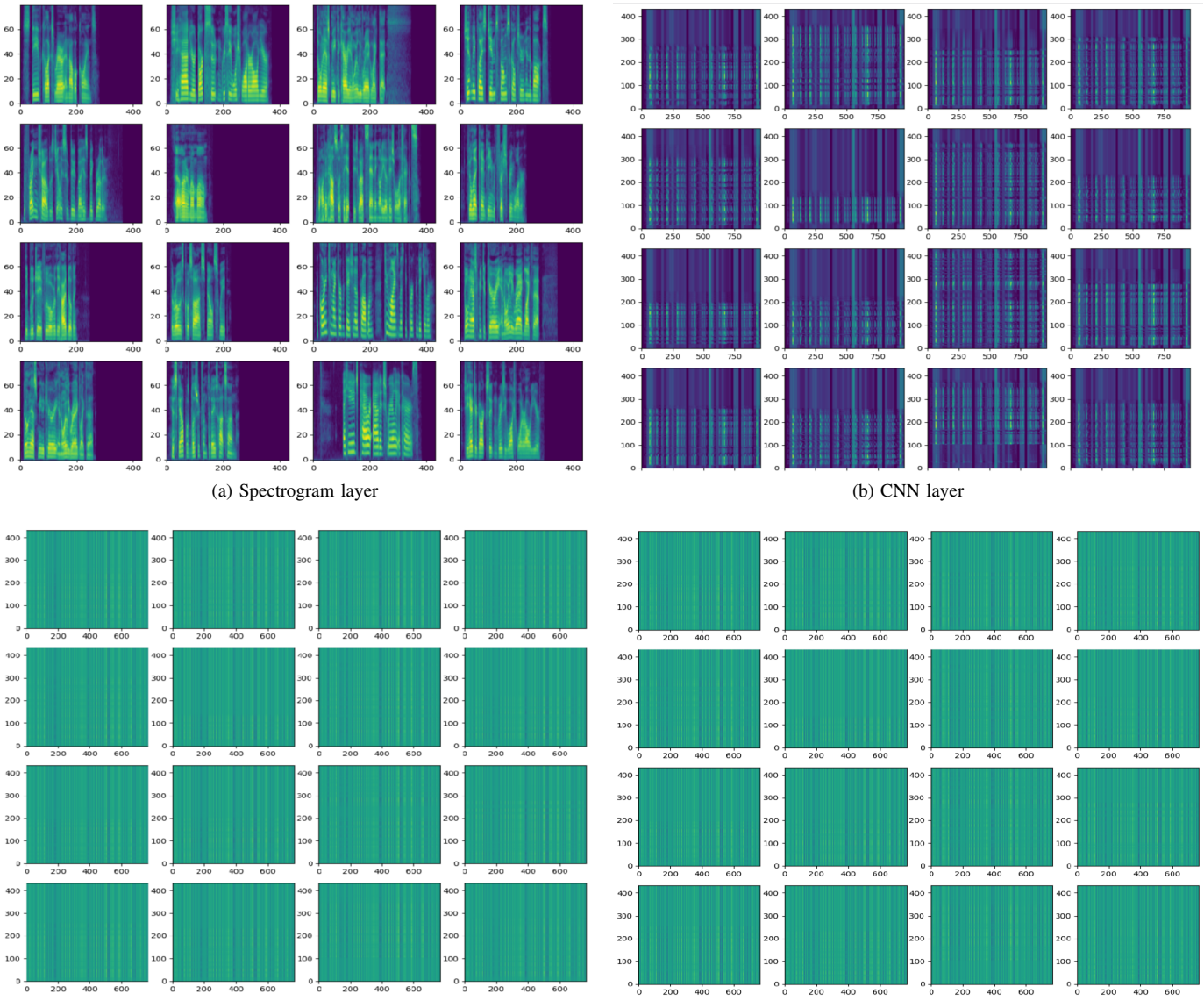
More work should be done to further investigate the effects of modifying the `n_fft` parameter. It is understood that this parameter affects the window size on which Fourier transform is operated on the audio data, and logically, it should follow `n_mels` inversely; a lower `n_fft` should provide greater granularity, but this presumption was not observed distinctly enough during the tests. It should be noted, however, that as with most parameter tunings, there should be an optimal point beyond which any modifications – increase or decrease – will lead to less optimal performance.

B. Spectrogram types

Apart from Melspectrogram, there are other types of spectrograms that can be used to transform audio signals. We experiment with two such types: Short-Time Fourier Transform (STFT) and Constant Q-Transform (CQT).

The Fourier transform only reflects the characteristics of the signal in the frequency domain, and cannot analyze the signal in the time domain. STFT was proposed in order to link the time domain with the frequency domain. STFT spectrogram is generated by multiplying a time-limited window function before the signal is Fourier transformed under the assumption that the non-stationary signal is stable in the short time interval of the analysis window. Moving on the time axis, the signal is analyzed section by section to obtain a set of local spectrum of the signal.

CQT refers to a filter bank whose center frequency is distributed exponentially, with different filter bandwidths, but



(a) Spectrogram layer

(b) CNN layer

(c) FC unit

(d) Linear layer

Figure 5. Activation maps of each layer of LeNet_5

the ratio of center frequency to bandwidth is constant Q. It is different from the Fourier transform in that the horizontal frequency of its spectrum is not linear, but based on $\log_2$, and the length of the filter window can be changed according to the frequency of the spectrum to obtain better performance.

Results: Tables IV and V show the prediction performance of LeNet_5 and BiGRU models with various spectrogram types respectively.

Spectrogram type	Test PER	Test Loss	Time Taken
Mel	0.999	3.732	17 min 8 sec
STFT	0.987	3.891	30 min 16 sec 680 msec
CQT	0.992	3.903	26 min 31 sec 800 msec

Table IV
RESULTS OBTAINED WITH DIFFERENT SPECTROGRAM TYPES ON LENET_5 MODEL

Spectrogram type	Test PER	Test Loss	Time Taken
Mel	0.982	3.161	39 min 11 sec
STFT	0.935	2.948	39 min 46 sec
CQT	0.971	3.256	48 min 6 sec

Table V
RESULTS OBTAINED WITH DIFFERENT SPECTROGRAM TYPES ON BIGRU MODEL

Discussion: The test loss and PER obtained and time taken for the LeNet_5 model with Mel, STFT and CQT spectrograms are specified in Table IV. As we can see, if sorted by performance based on loss, $Mel > STFT > CQT$. If sorted according to time saved in training, $Mel > CQT > STFT$. It is notable, however, that both STFT and CQT reduced the PER, with STFT being the more effective of the two. We observe that Melspectrogram still has the least loss. A likely

reason for this is that the Mel scale is created based on the logarithmic characteristics of the human ear's perception of frequency (that is sensitive to changes in the low frequency range and insensitive to changes in the high frequency range). So it is more targeted at speech feature extraction which is perfect for our raw data - human speech audio. Disadvantage of STFT is that the window width of STFT is fixed and cannot be adjusted adaptively. Generally speaking, short windows can provide better time domain resolution, and long windows can provide better frequency domain resolution. So STFT may lose some resolution in time or frequency domain because of the fixed window size. Since the distribution of CQT and scale frequencies is the same, by calculating the CQT spectrum of the music signal, the amplitude value of the music signal at each note frequency can be directly obtained, which is perfect for music signal processing, but may not work well for speech data.

The results on the BiGRU model in Figure V echo the LeNet_5 results for PER – both STFT and CQT had lower PERs, with STFT being the more effective one. Here, there is a departure from the LeNet_5 results: loss for STFT was observed to have decreased (albeit slightly) instead of increased, and training time for CQT was also longer than both Mel and STFT.

C. Character prediction

In this experiment, we modified the model such that it predicts English characters in place of the phonetics. For the example sentence 'SX70'(which read "Did you eat yet?") illustrated in section III, the character transcription is as follows:

```
0 14640 d i d y o u e a t y e t
```

where 0 indicates the starting integer sample number (sampled at 16kHz) of the segment, 14640 indicates the ending integer sample number of that segment, and the following text represents the characters in the sentence.

Results: Results of this experiment are described in Table VI for LeNet_5 and Table VII for BiGRU. This was conducted using Melspectrogram.

Output Mode	Test PER	Test Loss	Time taken
Phonemes	0.999	3.732	17 min 8 sec
Characters	1.0	2.818	17 mins 2 sec 520 msec

Table VI
RESULTS OBTAINED FOR DIFFERENT OUTPUT MODES ON LENET_5 MODEL

Output Mode	Test PER	Test Loss	Time taken
Phonemes	0.982	3.161	39 min 11 sec
Characters	1.0	2.530	38 mins 52 sec

Table VII
RESULTS OBTAINED FOR DIFFERENT OUTPUT MODES ON BIGRU MODEL

Discussion: We can infer from the results that the performance for character prediction is better than the performance for phonetic prediction using the same model. There could be a few reasons for this. There are lesser possible English characters (26) than phonemes (44), so there are fewer categories for the output must be classified into, which may improve the performance. Moreover, one phoneme can consist of multiple characters and have some fixed pattern, that means prediction of phonetics needs to extract more information from the data, so it has lower accuracy.

IX. CONCLUSION

In this paper, we presented a unique architecture – LeNet-CTC for phonetic sequence prediction. This architecture uses a convolutional neural network combined with CTC loss and results in a low test loss while maintaining a short training time in comparison to other models popular for ASR. We evaluated our model on the TIMIT dataset. We conducted three experiments to test our model's robustness. Based on their results, we conclude the following:

- STFT may potentially perform better than the Melspectrogram as the extracted feature for speech recognition.
- For the TIMIT dataset, audio transformed with Melspectrogram parameters potentially benefitted from a higher `n_mels` parameter due to the increased granularity it provides for the frequencies to be represented.
- Performance for the prediction of characters is better than that for prediction of phonemes.

X. ACKNOWLEDGEMENT

The code framework and baseline models for our work were adopted from https://github.com/KinWaiCheuk/pytorch_template.

REFERENCES

[1] Baevski A., Hsu W.-N., Conneau A., & Auli M. (2021). Unsupervised Speech Recognition. Computing Research Repository (CoRR), 1-14.
[2] Alharbi S., et al.(2021). Automatic Speech Recognition: Systematic Literature Review. IEEE Access, 131858 - 131876.
[3] K. W. Cheuk, H. Anderson, K. Agres and D. Herremans, "nnAudio: An on-the-Fly GPU Audio to Spectrogram Conversion Toolbox Using 1D Convolutional Neural Networks," in IEEE Access, vol. 8, pp. 161981-162003, 2020, doi: 10.1109/ACCESS.2020.3019084.
[4] Pundak G, Sainath T N, Prabhavalkar R, et al. Deep context: end-to-end contextual speech recognition[C]//2018 IEEE spoken language technology workshop (SLT). IEEE, 2018: 418-425.
[5] Mohamed A. Deep Neural Network Acoustic Models for ASR[D]. University of Toronto, 2014.
[6] https://github.com/kahne/fastwer

APPENDIX

Preliminary visualisation for TIMIT dataset are shown in figures 6,7, 8 and 9.

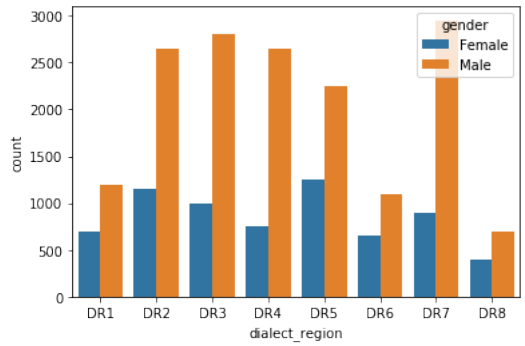


Figure 6. Training set divided into gender and dialect region

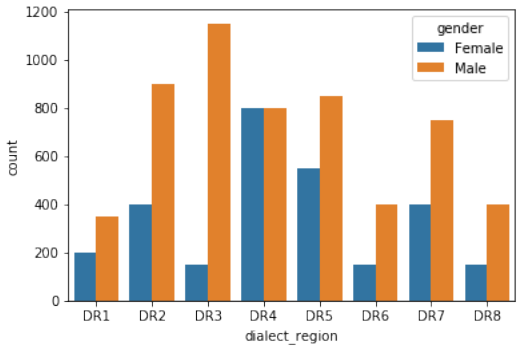


Figure 7. Test set divided into gender and dialect region

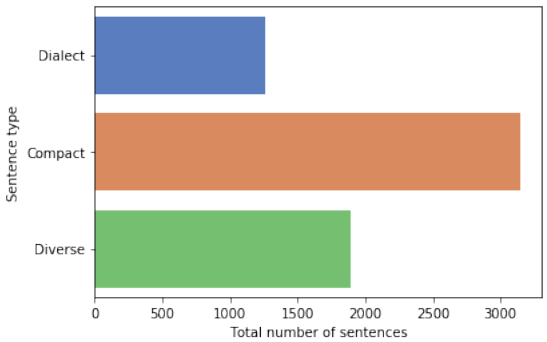


Figure 8. Number of each type of sentence

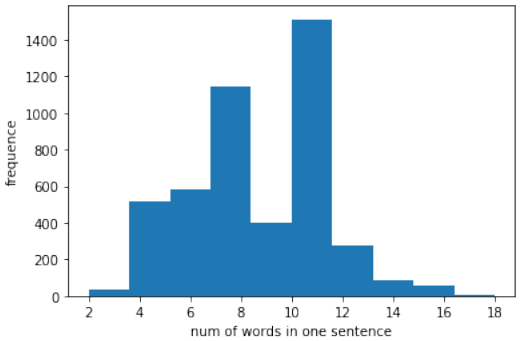


Figure 9. Number of words in each sentence