Investigation of CTC for TIMIT Database

Candidate Number: 2096K

January 3, 2024

Contents

1 Introduction

In this report, we will explore Auto-Speech Recognition (ASR) through an end-to-end investigation by utilizing the Connectionist Temporal Classification loss for the TIMIT acoustic-phonetic corpus. The experiments for this study have focused on investigating the practical application of Long Short-Term Memory (LSTM) models [1] under different settings by utilizing the PyTorch framework. This work aims to investigate different model architectures thoroughly and provides an opportunity to explore training, inferencing, and model performance. The report will discuss the critical issues found in generalization, modelling accuracy, optimization approaches, parameter number estimation, and their allocation based on the regularisation and augmentation approaches.

2 Methodology

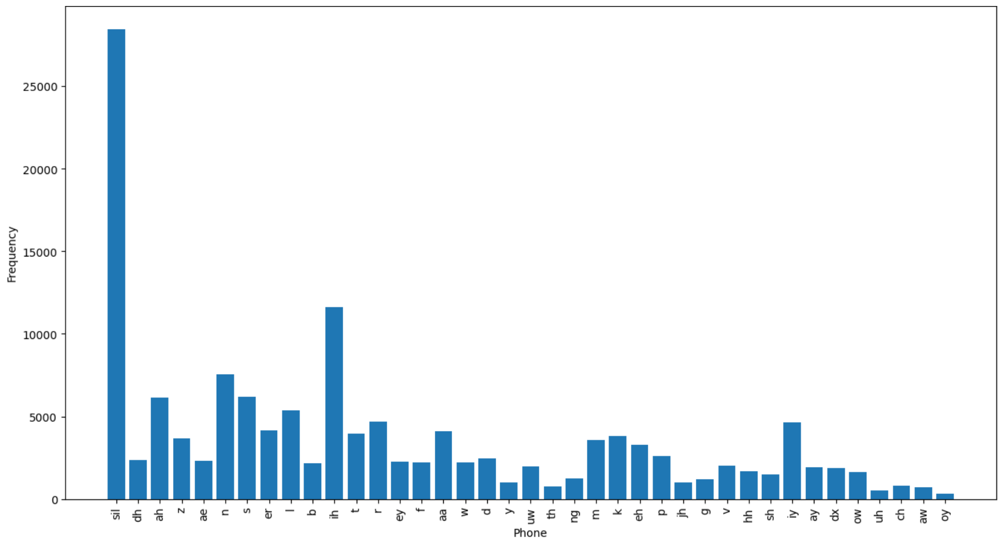

Figure 1: Distribution of phone frequencies in the training set

Our methodology involved transforming the raw speech waveform from the initial TIMIT dataset to feature vectors, specifically log Mel filterbanks. These filterbanks would be used as inputs to LSTM models with different configurations. A reduced phone set of 39 phones is used for output and hyperparameter tuning to optimize the model performance. Fig. **2** shows the distribution of phones in the training dataset. The following sections will discuss different model architectures, optimization techniques, regularisation methods and data augmentation methods.

2.1 Model Architecture

In this section, the architectures used in the study are presented. The core component of all the models is the LSTM network, followed by a feed-forward layer to map the output from the LSTM model to the desired dimension.

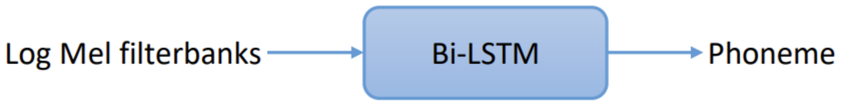

Figure 2: The baseline model structure

Fig. **2** shows the baseline model for the experiment, which is a single-layer Bidirectional-LSTM with a 128-dimensional hidden state followed by a feed-forward layer. This model has a relatively simple structure and has given us a basic understanding of the performance of the LSTM model in ASR in a computationally effective way.

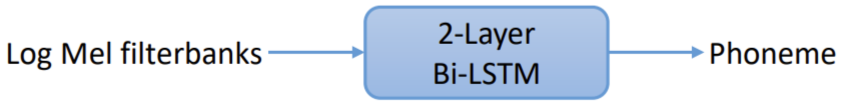

Figure 3: Two layer bidirectional LSTM

Fig. **3** shows the second model structure. This model increases the depth of the network by stacking one layer of bidirectional LSTM on top of the other. This model could theoretically capture a more complex pattern than the baseline since it has more parameters. Another choice of model could be stacking another feed-forward layer on the baseline model. However, stacking two feed-forward layers directly is equivalent to adding the width of a single feed-forward layer since the feed-forward layer is simply a linear transformation of input. Hence, stacking two feed-forward layers may not increase the complexity of the model as much as an additional layer of a bi-directional LSTM layer.

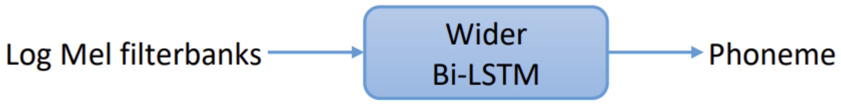

Figure 4: Wider single layer LSTM

Fig. **4** shows the third model, which explores the effect of layer width in LSTM. The model has a single bidirectional LSTM layer with 512-dimensional hidden states. This model may have better performance since it has many more parameters compared to the baseline and the second model described above. Also, it is sometimes observed that increasing the width of the LSTM is more effective than increasing the depth of the LSTM model.

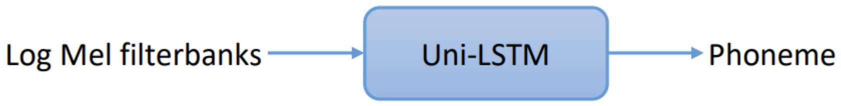

Figure 5: Uni-directional LSTM

Fig. **5** shows the last model structure, which used an uni-dimensional LSTM. It has a significantly smaller number of parameters when compared with the baseline model, which implies that it would have much less capacity to capture complex patterns and undermine its performance. However, it is critical for real-time applications to make the inference speed as fast as possible (while retaining good accuracy), which could restrict the use of bi-directional information.

2.2 Regularisation Methods

Since TIMIT is a relatively small dataset, applying regularisation techniques to prevent over-fitting would be important. In this report, we will explore the effect of dropout [2] on LSTM models, which randomly zeros some elements during training; specifically, each neuron has a probability of p to be removed from the network.

Dropout would reduce the network's sensitivity to any specific neighbouring neuron, which reduces the risk of the neurons being co-adapting, where the neuron tries to compensate for the error or noises produced by specific neurons instead of independently capturing patterns in the features. Each neuron is forced to capture patterns based on a random subset of the neighbouring neurons, enhancing the network's generalisation.

Two sets of experiments are done for the study of dropout, which are shown below:

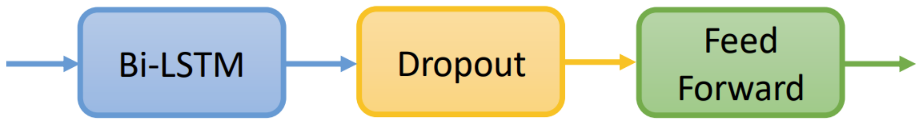

Figure 6: Dropout on the feed-forward layer for single-layer LSTM model

Fig. **6** shows the setup for the first experiment, which has applied dropout to the feed-forward layer. This experiment was designed to show the effect of the dropout when compared with the baseline model. However, the baseline model has very limited number of parameters. Hence, dropout may not have a significant effect on the performance of the model.

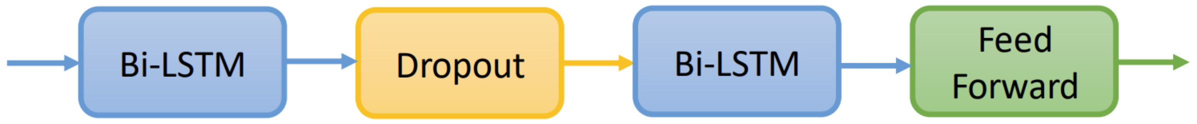

Figure 7: Dropout between two LSTM layers for 2-layer LSTM model

Fig. **7** shows the setup for the second experiment, which has applied dropout between two LSTM layers. This experiment was designed to see the effect of dropout on a more complex model structure, which is more likely to suffer from overfitting when compared with the baseline model.

2.3 Optimization Approaches

Most deep learning algorithms would aim to perform optimization, either by minimizing or maximizing certain loss functions [3]. In this study, we will optimize the Connectionist Temporal Classification (CTC) loss. CTC [4] align the input features with the corresponding output sequences without forcing each input step to align to a particular output, which has effectively bridged the gap between the variable-length acoustic feature sequences and their corresponding output sequences. Optimisers would minimize the loss by updating the model's parameters iteratively. In this study, two optimizers are examined, Stochastic Gradient Descent (SGD) and Adam.

2.3.1 Stochastic Gradient Descent (SGD)

Stochastic Gradient Descent (SGD) is an extension of gradient descent [3]. It optimises the model by altering the model parameters in the opposite direction to the gradient of the loss function based on a small batch of training samples:

$$\theta \leftarrow \theta - \epsilon g \tag{1}$$

where θ is the model parameter, ϵ is the learning rate, g is the average of the derivative of the loss function computed for the training samples in a batch. It is more computationally effective than gradient descent since it only uses a small subset instead of all training samples all at once [3].

By default, SGD uses a constant learning rate. However, using a learning rate scheduler to adjust the learning rate during training could be more effective. The sufficiency condition for SGD convergence is to satisfy the following equations [3],

$$\sum_{k=1}^{\infty} \epsilon_k = \infty \tag{2}$$

$$\sum_{k=1}^{\infty} \epsilon_k^2 < \infty \tag{3}$$

, where ϵ_k is the learning rate at the k-th iteration, so it would be a good choice to use a learning rate scheduler, which decays the learning rate during the training process and ensures the learning rates would satisfy the equations. This ensures the algorithm can explore the loss space sufficiently and, in the later stage, fine-tune and stabilize near a minimum.

2.3.2 Adam Optimiser

Unlike the SGD model, which maintains the learning rate for the entire model's parameters, Adam optimiser [5] adjusts the learning rate for each parameter separately during training based on the frequency of each parameter's occurrence. The Adam optimiser would theoretically provide a more refined way for convergence. The algorithm [5] could be summarised as:

$$\theta \leftarrow \theta - \epsilon \frac{\sqrt{\hat{s}}}{\hat{r} + \delta} \tag{4}$$

where $\hat{s}$ is the first moment estimates after bias correction, $\hat{r}$ is the second moment estimates after bias correction. The first moment computes the decaying average of the past gradients, which is also named the mean of the gradient, and the second moment computes the decaying average of the square of the past gradients, which is also named the uncentered variance of the gradients.

2.4 Data Augmentation

One of the best ways to improve the generalisation of a model is to train it with more data, and one way is through creating fake data and using it in combination with the original data for training [3]. Previous work [6] has shown the effectiveness of data augmentation on speech recognition tasks. In the experiment, one data augmentation named speed perturbation is explored. It operates on waveform files and increases or reduces the speed of the audio by adjusting the sample rate. The augmented waveform would create additional acoustic features to introduce more variants to the training dataset.

3 Experiments and Discussions

This section will present and analyse the outcome of experiments for various LSTM-based ASR models. The experiments were conducted to explore the effectiveness of regularisation methodology, optimization methodology, different model architectures and data augmentation.

3.1 Experiment 1: Regularisation

These sets of experiments aim to provide an understanding of the effect of dropout on the model performance by observing the change in the convergence validation loss for two different LSTM models with different dropout rate settings.

3.1.1 Regularisation on single layer LSTM

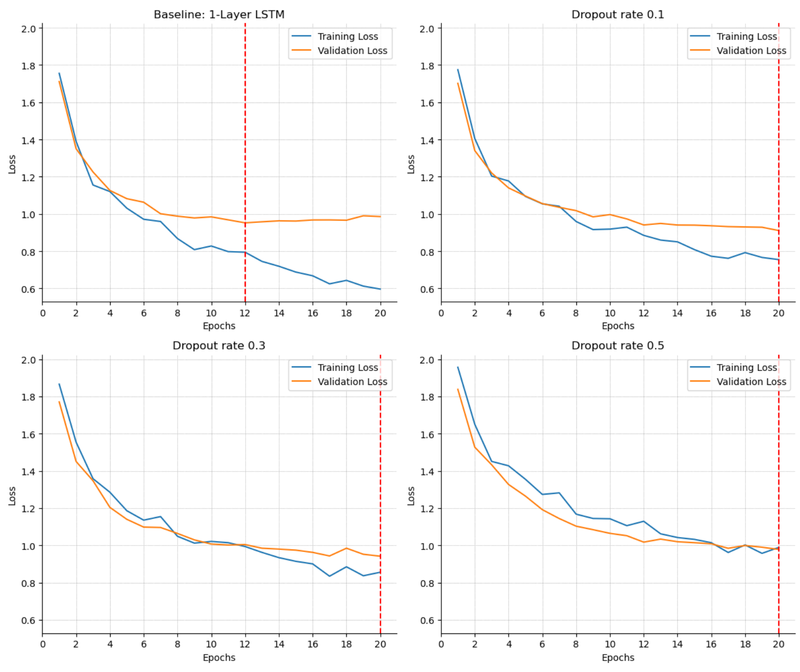

Figure 8: Loss change for baseline model with different dropout rate

Fig. **8** shows the change in loss for the baseline model, which is a single-layer LSTM with 3 different dropout rate settings, which are 0.1, 0.3 and 0.5, respectively. The red vertical line marks the epoch with the lowest validation loss. It could be observed that the addition of the dropout has alleviated the potential overfitting problem in the baseline model since the validation loss has become closer to the training loss as the training proceeds. Meanwhile, for dropout rates of 0.1 and 0.3, the validation loss when the model converged is slightly lower than the baseline model, which shows that the appropriate usage of dropout may positively affect the model's performance. For a dropout rate of 0.5, the validation loss at convergence has risen slightly, which may be due to the model not being able to capture complex patterns when an overly high dropout rate is applied to it.

3.1.2 Regularisation on 2-layer LSTM

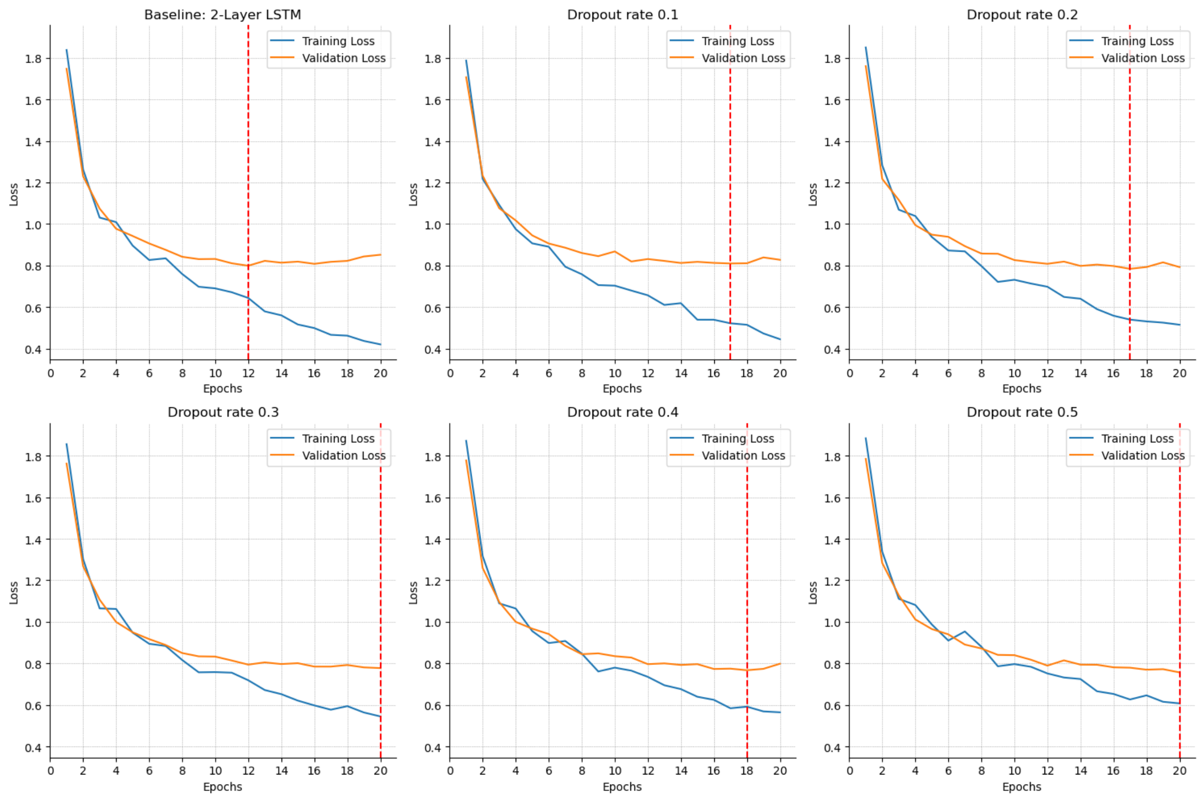

Figure 9: Loss change for 2-layer LSTM with different dropout rate

Fig. **9** shows the change in loss for training a 2-layer LSTM with 5 learning rates, which are 0.1, 0.2, 0.3, 0.4 and 0.5, respectively. Like the first experiment, dropout has improved the model's generalization as the difference between the validation and training loss has become smaller as the dropout rate increases. Unlike the first set of experiments, the validation loss at convergence has decreased continuously as the dropout rate increased; this may be due to the increase in the number of parameters of the model, which makes the model still able to capture the complex pattern in the features even when half of the parameters are zeroed randomly. Another observation is that the dropout would slow down the convergence as the place where the red line appears moves to the right of each plot as the dropout rate increases, which may caused by, for each iteration, only a subset of neurons being trained, which slows down the learning process.

3.2 Experiment 2: Optimiser

3.2.1 Adam Optimiser

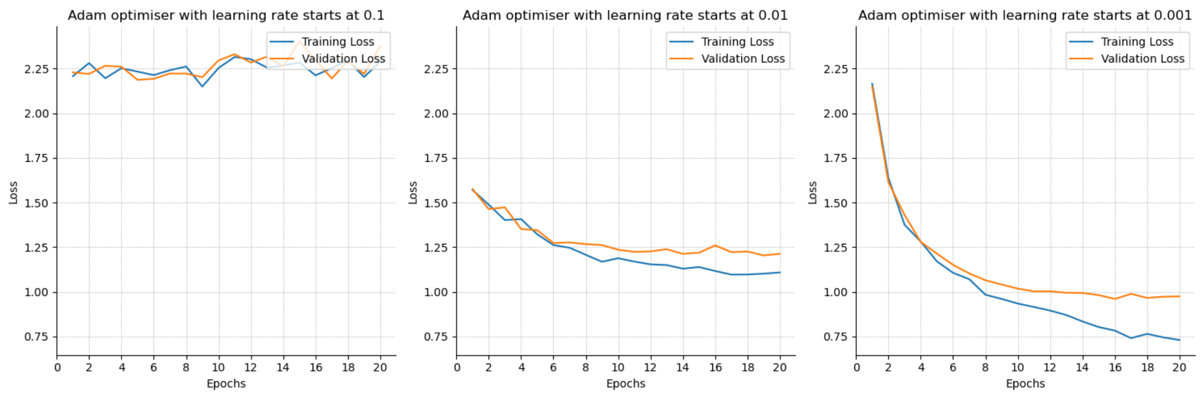

Figure 10: loss change for single layer Bi-LSTM for Adam optimiser

At the very beginning of the experiment, the same starting learning rate as constant SGD, which is 0.5, was tried, but it led to a gradient explosion, so a lower learning rate, 0.1, was tried. However, a learning rate of 0.1 also produces gradient explode, but after **gradient clipping**, which is a mechanism that clips gradient if their norm has exceeded the pre-defined threshold, the training process has been completed. Nevertheless, the validation loss has oscillated around a certain value throughout the training process, and the validation loss starts to have a decreasing trend when the starting learning rate has been set to a lower value, which is 0.01 and 0.001. This could be due to the Adam optimizer optimising the learning rate for each parameter based on the mean and uncentered variance of the gradients. A large learning rate could result in large updates and cause the model to oscillate around the minima. Meanwhile, a large learning rate could exaggerate the differences in the change for individual parameters, which leads to unstable updates that do not move towards the minima consistently.

3.2.2 SGD with Learning Rate Scheduler

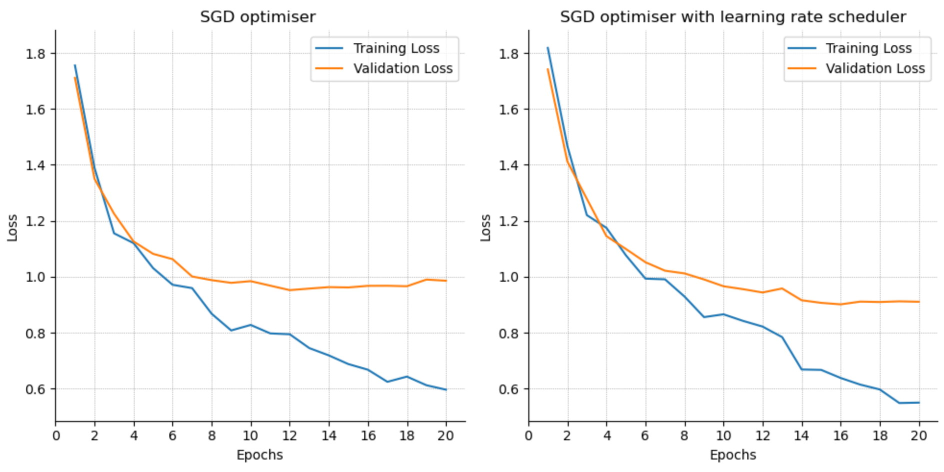

Figure 11: loss change for single layer Bi-LSTM for SGD optimiser

Fig. **11** shows the comparison of the change in the validation loss when training with a constant learning rate and with a learning rate scheduler. It could be observed that the model that uses the learning rate scheduler has converged to lower validation loss. The gradually decreasing learning rate has enabled the model to shift its aim from reducing error to fine-tuning and reaching the best possible accuracy.

3.3 Experiment 3: Model Complexity

The following experiments intend to study the model's different structures and find the best model.

For all three models, the hyperparameter was re-tuned, which included six different dropout rates (including 0) and two different optimisers, respectively. These choices were made mainly because of the limitations of the computing power. For the dropout rate, some commonly used quantities are selected. For the optimiser, SGD with the learning rate scheduler starting at 0.5 and Adam starting at 0.001 were chosen. Firstly, SGD with a learning rate scheduler instead of SGD with a constant learning rate was chosen since study [3] shows that using a learning rate decay mechanism could guarantee the convergence of SGD. For Adam optimiser, a starting learning rate of 0.001 is chosen since it is the default starting learning rate in Pytorch and performs relatively well in the experiment done in Section **3.2.1**.

3.3.1 2-layer Bidirectional LSTM

Dropout rate **Optimiser**	**0**	**0.1**	**0.2**	**0.3**	**0.4**	**0.5**
Adam	0.79501	0.79568	0.78900	0.79089	0.78459	0.77527
SGD	0.77112	0.75664	0.74669	0.74559	0.74345	**0.73030**

Table 1: Validation Loss for 2-Layer LSTM with different hyperparameters

Table **1** shows the fine-tuned result of the 2-layer bidirectional LSTM model with 128 hidden states for each layer, and the model has converged when the dropout rate is 0.5 and using SGD with learning rate scheduler.

3.3.2 Wider Bidirectional LSTM

Dropout rate **Optimiser**	**0**	**0.1**	**0.2**	**0.3**	**0.4**	**0.5**
Adam	0.86603	0.86813	0.88694	0.88678	0.86538	0.87883
SGD	0.91159	**0.84588**	0.85038	0.84923	0.85000	0.85789

Table 2: Validation loss for wider LSTM with different hyperparameters

Table **2** shows the fine-tuned result of the single-layer bidirectional LSTM model with 512-dimensional hidden states, and the model has converged when the dropout rate is 0.1 and when using SGD with learning rate scheduler.

3.3.3 Unidirectional LSTM

Optimiser \ Dropout rate	0	0.1	0.2	0.3	0.4	0.5
Adam	0.88478	0.87844	0.88971	0.88072	0.88442	0.91128
SGD	**0.82374**	0.82729	0.83083	0.84082	0.86963	0.85338

Table 3: Validation loss for wider LSTM with different hyperparameters

Table **3** shows the fine-tuned result of the 2-layer unidirectional LSTM model with a 210-dimensional hidden state for each layer, and the model has the best performance when the dropout is not applied and when using SGD with learning rate scheduler. The model was designed to study the effect of bi-directional information on the model performance; hence, for the fairness of comparison, the model size has been adjusted to have a similar number of parameters as the 2-layer bidirectional model.

3.3.4 Comparison

Model	Par.	Validaiton loss
2 Layer Bidirectional LSTM	562216	**0.73030**
Wider Bidirectional LSTM	2240552	0.84588
Unidirectional LSTM	560320	0.82374

Table 4: Comparison of three models

The best model is the 2-layer bidirectional Lstm because it has the smallest convergence validation loss, specifically 0.73030. In contrast, the wider bidirectional Lstm does not perform as well as its predecessor, even though it has a larger number of parameters. This may be due to overfitting because the model is too large with insufficient data, or it may be that a deeper model would be a better structural choice, as research [3] has shown that in many cases, a deeper model can represent the desired function with fewer parameters and reduce generalisation error.

Unidirectional LSTM and Bidirectional LSTM have a similar number of parameters, but the performance of bidirectional LSTM is better; this may be because bidirectional LSTM can utilise bidirectional information since it processes the data forwardly and backwardly, which allows it to refer to the context that precedes and follows it. In contrast, the unidirectional LSTM can only use information from the past state.

Model	SUB	DEL	INS	COR	PER
2 Layer Bidirectional LSTM	14.44%	6.58%	2.93%	78.98%	23.95%

Table 5: Test Result Summary of the Best Model

Table **5** shows the decoded result of the test set for the best model.

3.4 Experiment 4: Data Augmentation

	Unaugmented Training	Augmented Training
Validaiton loss	0.73030	**0.70350**

Table 6: Validation loss before and after apply data augmentation

In this experiment, the augmented waveform at rates of 0.9 (slowed down) and 1.1 (sped up), in addition to the original waveform, were used for training. Table **6** shows the validation loss of a 2-layer bidirectional LSTM before and after data augmentation is applied, which shows that the data augmentation mechanism positively affects the model performance.

4 Decoding

4.1 Visualisaiton

4.1.1 Heatmap

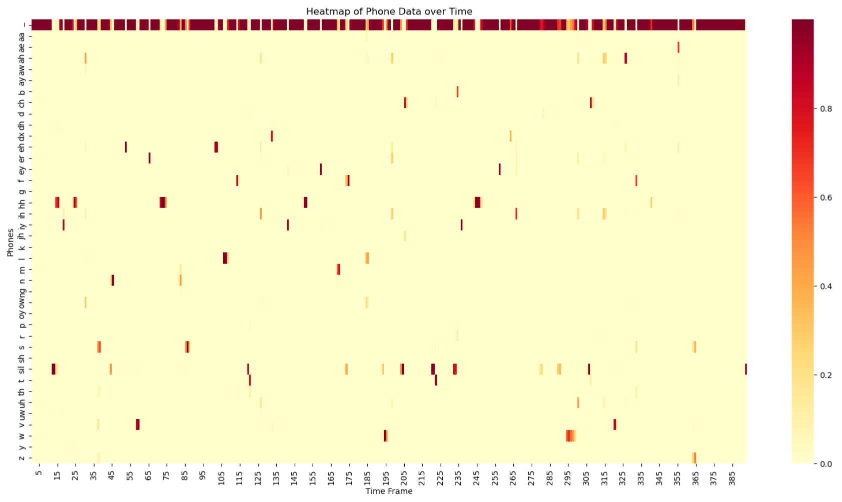

Figure 12: Output probability distribution over vocabulary for an utterance

Fig. **12** shows the heatmap of the output probability distribution over vocabulary for a chosen utterance. The darker blocks in the heatmap imply that the model is more confident about its predicted phone. It could be observed that the model has high confidence about certain phones within each timeframe, which implies that the model could be well-trained since it can differentiate between phones clearly. Meanwhile, the model has produced a variety of predictions for different time frames, which may imply that the model has learnt diverse phonetic information inside the TIMIT dataset. Additionally, there is a variation for the phone with high probabilities for each time frame, which may reflect the temporal structure of speech waveform.

4.1.2 Confusion Matrix

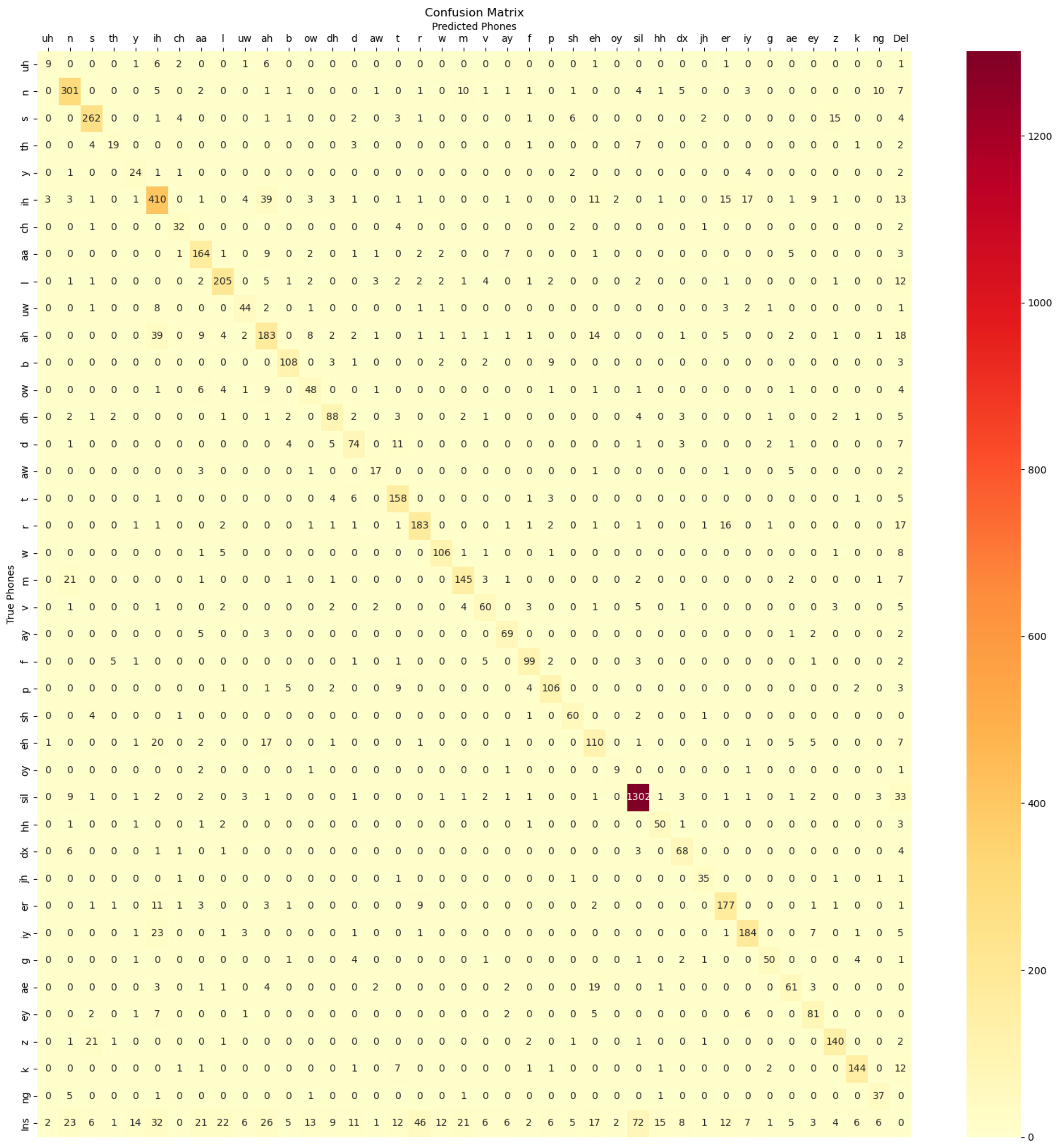

Figure 13: Confusion Matrix

Another way of visualising the output pattern is through a confusion matrix, shown in Fig. **13** above. The entries on the diagonal show the count for each correctly predicted phone for the entire test set, where the count has shared a very similar distribution with the phone frequencies in the training set shown in Fig. **1**. Meanwhile, it could be observed that some phones with similar pronunciation are more likely to be substituted by others. For example, *m* and *n*, *eh* and *ih*, *iy* and *ih*, *z* and *s*, the previous phone is likely to be substituted by the following phone. By contrast, *ah* and *ih* are likely to substitute each other. However, they do not sound similar, although the distinction between them might decrease in fast or casual speech, and this might also be due to there is not enough information to differentiate *ah* and *ih*, as we have used a 39 vocabulary phone set, where some of the phones are mapped to *ah* and *ih*, which may oversimplify the acoustic space and resulting in confusion. Additionally, *sil*, the silence phone, is the one that is mostly mistakenly inserted and deleted. The aim of a speech recognition model is not only to predict the phone that is presented correctly but also to determine how long the phone lasts. In contrast, silence has a variable duration, which increases the difficulty of prediction, resulting in a more frequent error in prediction related to *sil*.

4.2 Blank Penalty

Penalty	Ins Error Rate	Del Error Rate
0	2.93%	6.58%
0.1	3.25%	5.98%
0.2	3.61%	5.39%
0.3	4.02%	4.96%
0.4	4.32%	4.63%
0.5	5.30%	3.92%

Table 7: Insertion and Deletion error rate when blank penalty applied

Table **7** shows the change in the insertion error rate and deletion error rate when different blank penalties are applied. It could be observed that the deletion error rate decreased gradually when the penalty increased since the model is penalised for output blanks. Therefore, it would be more likely to produce the token that has been mistakenly omitted before, resulting in a lower deletion rate. In contrast, the insertion error rate increases when the penalty is increased, suggesting that discouraging the model from output blank may encourage the model to emit incorrect tokens, possibly due to overcompensation in avoiding emitting blanks.

4.3 61 Phone set

	39 phone	Full phone set
PER	23.95%	24.55%

Table 8: Comparison of error rate before and after training with 61 phone set

The ASR model was trained using 39 phone sets for all previous experiments. In this experiment, the ASR model with the best performance was trained on the full phone set with the same hyperparameter setting as the best model, and the model's output layer was modified according to the number of phones in the full phone set. After mapping the predicted phones from 62 (61 phones plus the silent phone) to 39 phones, the resulting phone error rate (PER) is 24.55%, which did not outperform the best model, which has a PER of 23.95%. This may be due to the gap between training and validation/testing conditions. In contrast, the training process has information from full phone sets; the other two datasets do not, and the model may not be able to generalize well due to the difference in phone representations. Meanwhile, it may be because the model is not complex enough to handle the information from the full phone set. The model may not learn the finer distinction from a 62-phone set effectively compared to training with a 39-phone set.

References

[1] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[2] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, “Improving neural networks by preventing co-adaptation of feature detectors,” *arXiv preprint arXiv:1207.0580*, 2012.

[3] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, ser. Adaptive Computation and Machine Learning series. MIT Press, 2016. [Online]. Available: https://books.google.co.uk/books?id=-s2MEAAAQBAJ

[4] A. Hannun, “Sequence modeling with ctc,” *Distill*, 2017, https://distill.pub/2017/ctc.

[5] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” 2017.

[6] N. Jaitly and G. E. Hinton, “Vocal tract length perturbation (vtlp) improves speech recognition,” 2013. [Online]. Available: https://api.semanticscholar.org/CorpusID:14140670