Enhancing End-to-End Automatic Speech Recognition with Bayesian Optimisation and Audio Augmentation

A Report for MLMI2

Steve Hong

1. Objective of Report

This project investigates training an end-to-end acoustic speech recognition (ASR) system using a Long Short-Term Memory (LSTM) network. LSTMs [8] are advantageous in this context because they effectively approximate the functionality of traditional Hidden Markov Models with Gaussian Mixture Models, which typically require multiple separate training processes [7]. In addition, we use the Connectionist Temporal Classification (CTC) loss [5] for this LSTM, which eliminates the need for precise phone-level alignments during training. This is useful for speech recognition tasks, where manually-labelled alignments are often costly. This report focuses on investigating two key practices associated with training LSTMs for ASR:

Model Selection: Training an LSTM for ASR involves efficiently exploring a large hyperparameter space due to the many possible parameter configuration. In this study, we evaluate 650 combinations across 3 categories of hyperparameters. To streamline this process, we fit a Gaussian process to the objective function and optimse the acquisition function to suggest the next parameters [10], which offers a more efficient heuristic approach for hyperparameter optimisation compared to grid search. The results from the trials obtained let us compare key relationships between model complexity and regularisation strategies, as welll as which optimiser to use.

Audio Augmentation: To further regularise large LSTM models—particularly those with high parameter counts (up to 14 million in our experiments)—we investigate data augmentation techniques as a implicit regularisation strategy. Specifically, we employ two augmentation methods: speed perturbation and the superposition of background noise to the original audio data. We analyse how each of them, and them combined improve model performance. We also want to see if this can replace weight regularisation completely.

2. Dataset Overview

The TIMIT dataset [3] contains speech recordings from 630 speakers, covering eight major American English dialects. It is divided into 3,696 training samples, 400 validation samples, and 944 test samples. Typically, the training and validation sets are used during model development, with the validation set reserved for hyperparameter tuning. After hyperparameters are finalised, the model is retrained on both the training and validation sets. The test set is used exclusively to evaluate the final model.

The output vocabulary consists of 39 unique phones derived from the original 61-phone set of TIMIT. An additional blank token is added for CTC loss set-up, creating a total of 40

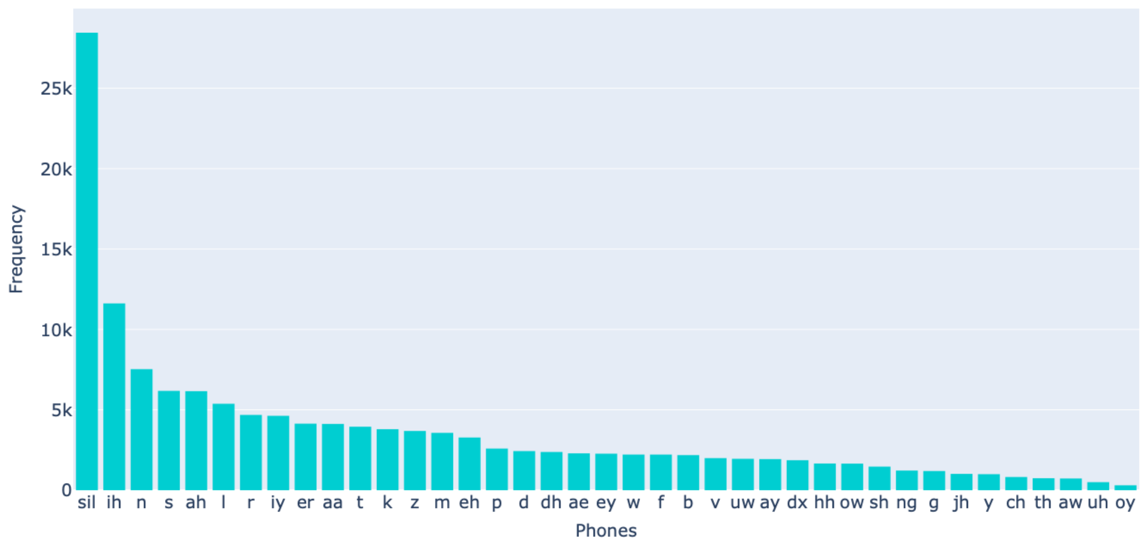


Figure 1: The figure shows the distribution of phone occurrences within the training dataset. The x-axis shows various phones, such as *sil* (silence), *ih*, *n*, and others, while the y-axis represents their respective frequencies. The silence *sil* phone has the highest count, exceeding 27,000 occurrences, followed by phones like *ih*, *n*, and *s*.

phones. These phones are the true targets for model training. A histogram illustrating the phone distribution in the training set is provided in Figure 1. We can be an apparent imbalance of class issue in this distribution - the trained model would have less exposure to rare classes such as *oy*, *uh* and *aw*.

Also, raw speech waveforms are not used directly as inputs to the model. Instead, they are converted into 23-dimensional FBank features for each frame. These features, which capture key spectral information from the speech signal, are used as the model's inputs for training and validation.

3. Model Selection

In this section, we aim to understand the relationship between 3 sets of hyperparameters and the model's performance on the CTC validation error. Firsly we want to check how increasing model complexity would improve the validation loss, and at what costs. We also want to compare two weight regularisation strategies in the pipeline, and how they interact with the model complexities. We also test 3 different optimisers to see which one is optimal for this problem. Throughout this section, we also experiment with the Bayesian Optimisation approach to accelerate the hyperparameter optimisation routine.

3.1. Experimnent Pipeline

We define the hyperparameter space as shown in Table 1, focusing on three main categories: model complexity, optimisation, and regularisation. This results in a total of 650 combinations of hyperparameter settings. Conducting a full grid search over this space would require approximately 12 hours, making it computationally expensive and time-consuming.

To address this, we use Bayesian Optimisation, a method that adaptively explores the hyper-

Category	Hyperparameter	Input Set
Model Complexity	Depth	[1, 3]
	Width	[128, 512]
	Model Architecture	[LSTM, BI-LSTM]
Optimiser	Type of Optimiser	[Adam, SGD, SGD (Scheduler)]
	Learning Rate	$[1 \times 10^{-2}, 2 \times 10^{-2}, 5 \times 10^{-2}]$
	Momentum (for SGD)	[0.9]
Regularisation	Dropout Rate	[0.1, 0.2, 0.3]
	Max Gradient Norm Clipping	[1.0, 3.0, 5.0]
	Early Stopping	[3 iter]

Table 1: Hyperparameter Space

parameter space by modeling the performance function and focusing on regions likely to yield improvements, drop the trials that are unlikely to improve the validation loss. Specifically, Bayesian Optimisation uses a probabilistic surrogate model to guide the search for optimal settings. This approach significantly reduces the number of iterations needed to find the best hyperparameters.

In our experiments, we employ the implementation provided by Optuna [1]. Optuna uses the Matérn kernel with Automatic Relevance Determination in its Gaussian Process-based surrogate model. This allows it to efficiently navigate the hyperparameter space, reducing the required optimisation time from approximately 12 hours to just 5 hours. This allows for more effective use of GPU resources.

3.2. Model Complexity

In principle, increasing complexity means that models have more capacity to approximate the features-to-phone mapping better, but are at risk of overfiting. Here we can compare how increasing model complexity settings can impact the validation loss. In Figure 2, it is suggested that increasing the complexity of LSTM models, through wider hidden layers, deeper architectures, or higher weights counts, can overall lead to improved performance in some trials, as indicated by lower validation losses. While simpler models exhibit more consistency, the best-performing models are often those with greater complexity, such as BI-LSTM architectures, deeper layers, and larger weights counts.

Complexity	No. of Weight	CTC	PER
1L-128H	0.17M	1.067	0.360
1L-512H	2.24M	0.911	0.315
2L-128H	0.56M	0.899	0.293
2L-512H	8.54M	0.967	0.268
3L-128H	0.96M	0.815	0.275
3L-512H	**14.84M**	**0.799**	**0.265**

Table 2: Comparison of the Impact of Increasing Depth vs. Width on CTC and PER

Also, from Table 2, we see that adding layers to a Bidirectional LSTM (e.g., CTC-1L-128H to CTC-3L-128H) significantly improves performance (validation loss drops from 1.067 to 0.815) with a modest increase in weights (0.17M to 0.96M). In contrast, increasing layer size (e.g., CTC-3L-128H to CTC-3L-512H) results in a much larger weight increase (0.96M to 14.84M) with

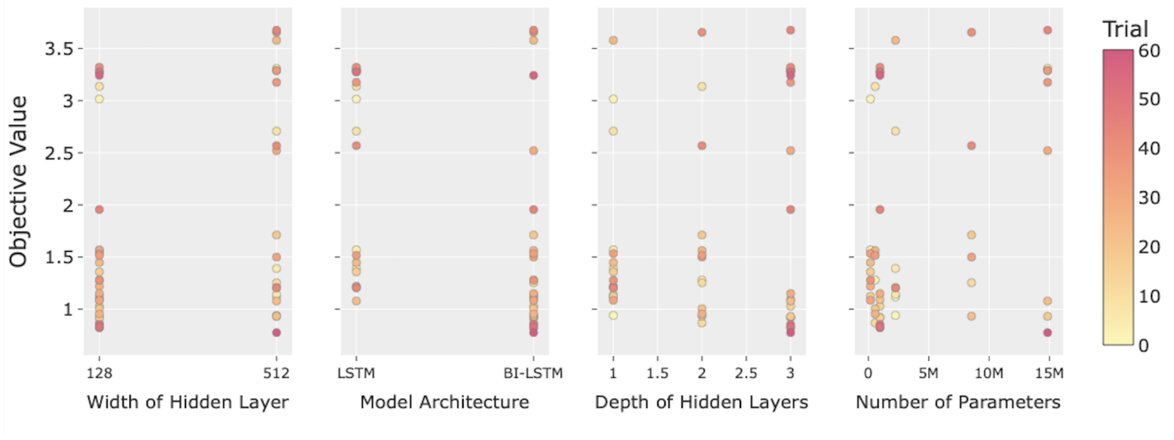


Figure 2: Impact of Model Complexity on CTC Validation Loss. The scatter plots show the effect of increasing model complexity—Width of Hidden Layer, Model Architecture, Depth of Hidden Layers, and Number of Parameters—on the validation loss. We can see that more complex models generally perform better than small models, though are subject to regularisation strategies.

minimal performance gain (0.815 to 0.799). This highlights depth as a more efficient strategy than width for improving performance.

3.3. Optimiser

The optimisation experiment involved selecting from a pool of three optimisers and varying their learning rates. Based on the results shown in Figure 3, the best settings appear to be SGD with a scheduler as the optimiser, combined with a learning rate of 0.02, as this combination consistently produces the lowest validation loss. This outcome is logical, as the scheduler reduces the learning rate whenever the validation loss increases, helping to stabilise the updates.

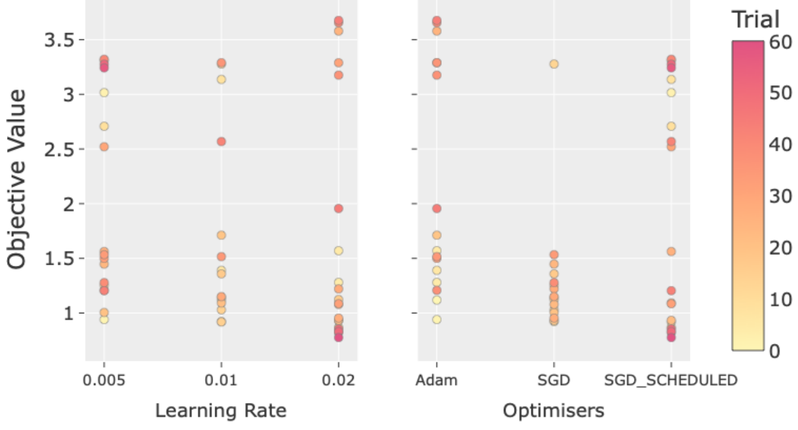


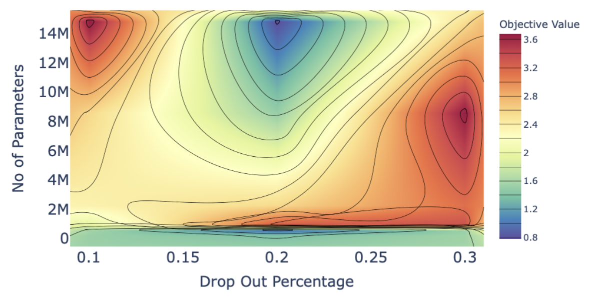
Figure 3: Impact of Model Complexity on the CTC Validation Loss. Overall we can see that SGD with scheduler perform better than Adam and default SGD.

3.4. Regularisation

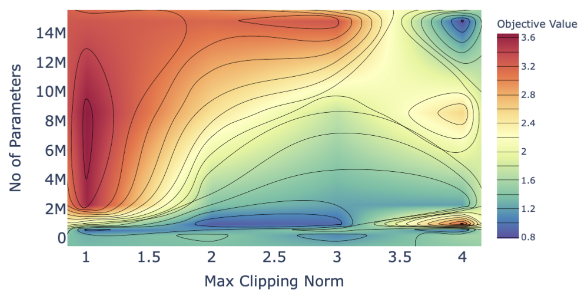
The contour plots in Figure 4 illustrate the effects of two regularisation techniques: dropout and gradient clipping on the validation loss across different model complexities. From Figure 4a, we observe that with low dropout rate (between 0.1 and 0.2), complex models tend to overfit. Excessive dropout (between 0.2 and 0.3) shows a sharp drop in performance of medium size model. Only at 0.2 rate where large models perform optimally. Small model, in the mean time, is quite invariant to the dropout rate - their performance stays similar across the droupout scale. Overall, this shows that for medium and large models, regularisation using dropout is sensitive to changes in the hyperparameter and requires careful tuning.

A different pattern is seen for gradient clipping in Figure 4b. In principle, smaller max clipping implies stronger regularisation, thus allowing for large models to be less subject to overfitting. The empirical result is showing the opposite: stronger regularisation at low norm values (between 1 and 3) causes large and medium models have worse performance, while improves smaller models. At large clipping norm of 4, which implies low regularisation, large models perform better while small models perform worse, which sharp rates of change. However, these results can be due to the limited range of norm and requires wider exploration. But overall, they support our previous observation in droupout, where this style of regularisation is sentitive to hyperparameter range and requires careful tuning.

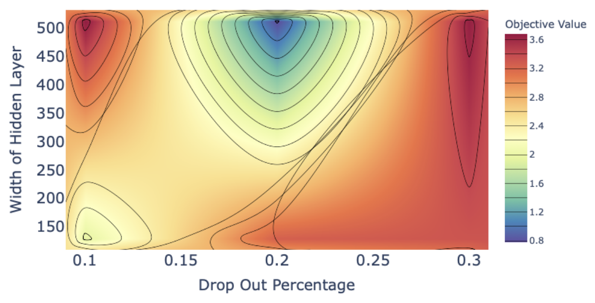
One final important note from 4c and 4d is that depth is more likely to increase performance than width.



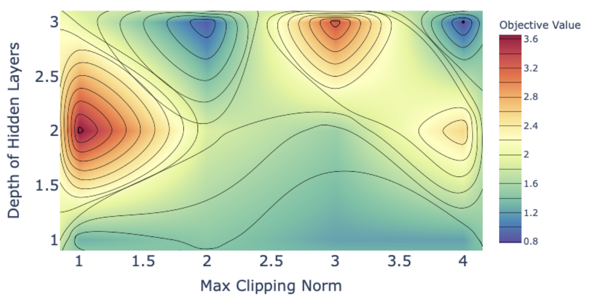
(a)



(b)



(c)



(d)

Figure 4: Contour plots illustrating the relationship between CTC validation loss and four model design factors: dropout rate (top-left), gradient clipping (top-right), model width (bottom-left), and model depth (bottom-right).

4. Audio Augmentation

We establish from Section 3 that regularisation through hyperparameter search is sensitive to the hyperparameter threshold - small changes in the values can cause steep drops in performance. This section explores a complementary regularisation strategy using data augmentation. Data augmentation is a form of implicit regularisation [4]. In principle, more data would benefit all large-scale deep learning models, especially for larger models, as they are exposed to a more heterogeneous set of training observations.

In order to achieve a high level of diversity, two audio augmentation strategies are deployed. Firstly, we can perturb the data by varying their speed to 1.1 (faster speed - shorter duration) and 0.9 (lower speed - longer duration). Secondly, we can superpose background noise with the data. There are interesting ways to do this so that we can boost diversity of training samples. We leverage the 11GB MUSAN noise dataset [11], where we randomly sample from the pool of noise samples, concat them, then add to a proportion of our original noise-free samples. The noise are also added at randomly scaled amplitude as well to increase variety. It is important to sample and concat a variety of them from this pool, instead of duplicating a small subset of them or adding simple white noise, because it helps with preventing the LSTM to quickly learn these noise and ignore the synthetic data [6]. Finally, we can also combine both of the approaches above by first perturb the speed, then add noise to the a subset of the training samples.

One practical note is that the noises are resampled to 16kHz and added to the audio waves. Then the augmented data are inserted into the FBank and training pipeline similarly to the previous section.

4.1. Experiment and Training Pipeline

With the augmented datasets, two relationships are investigated through two distinct experiments.

Experiment 1: Impact of Synthetic Data on Dropout and Gradient Clipping

The first experiment investigates the impact of synthetic data augmentation on dropout and gradient clipping strategies. The architecture used throughout this section is BI-LSTM, which was shown to be superior from the previous section. The model size is fixed to **3L-256H**, while the level of data augmentation is varied across four scenarios:

	No Perturbation	**3x Perturbation**
No Noise	Scn 1	Scn 3
50% Noise	Scn 2	Scn 4

Table 3: Table of Data Augmentation Scenarios. 50% Noise means noise are added to 50% of the clean samples. Triple perturbation means that the data is not 3 times larger - one 1.1x copy and one 0.9x copy was added to the original data.

The fixed-size model is trained across a space of dropout and gradient clipping settings. The selected space for dropout values is $[0.1, 0.2, 0.3]$, and for gradient clipping max norm, the values are $[1.0, 3.0, 5.0]$. The experiment observes how the validation loss changes in all four scenarios. The expectation is that synthetic helps to soften the steep changes in loss when as we vary the dropout and gradient clipping norm - making them less sensitive.

Experiment 2: Impact of Synthetic Data on Model Capacity

The second experiment explores the influence of synthetic data augmentation on model capacity. In this setup, the combination of dropout and gradient clipping is fixed at 0.2 and 3. We then vary across 9 audio augmentation scenarios like above, across 3 model sizes **1L-128H**, **3L-128H** and **3L-256H** to compare the losses, making total 27 scenarios.

	No Pert.	**2x Pert.**	**3x Pert.**
0% Noise	Scn 1	Scn 2	Scn 3
20% Noise	Scn 4	Scn 5	Scn 6
50% Noise	Scn 7	Scn 8	Scn 9

Table 4: Considered Scenarios in Experiment 2

Type	**Complexity**
Narrow	1L-128H
Deep	3L-128H
Deep+Wide	3L-256H

Table 5: Model Sizes.

4.2. Results

Experiment 1:

The contour plots in Figure 5 show how varying noise levels and perturbation strategies impact model performance, given by the PER validation error. Length perturbation (changing the speed of audio data) effectively **reduces hyperparameter sensitivity by smoothing out the loss landscape**. In panels (a) and (b), where no noise is applied, the model's performance becomes more consistent as perturbation increases from 1x (No pertubation) to 3x. Specifically, panel (b) demonstrates a broader, flatter region of low error, indicating that the model generalises better and is less sensitive to changes in dropout rate and gradient clipping norm. This suggests that perturbing the audio length exposes the model to a more diverse range of training examples, thereby improving its robustness.

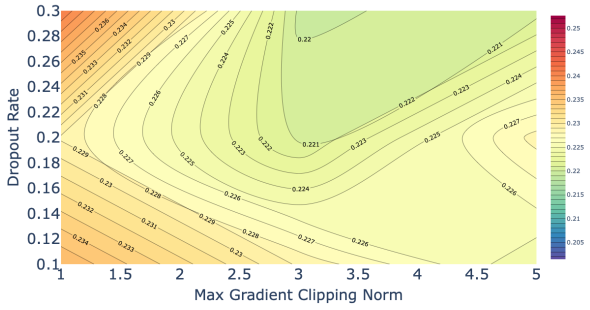
Conversely, **noise superposition introduces hyperparameter sensitivity**, as shown in Figure (c) and (d), where 50% noise is added. In panel (c), with no perturbation, the loss landscape becomes steeper, indicating that small changes in dropout rate or gradient clipping can significantly impact performance. However, when noise is combined with 3x perturbation in panel (d), the benefits of perturbation are restored. The model achieves lower validation loss across a wider range of hyperparameters, with the sensitivity observed in panel (c) mitigated. This highlights that **combining both techniques—length perturbation and noise superposition—leads to a synergistic effect**, producing the most stable and optimal model performance overall.

Experiment 2:

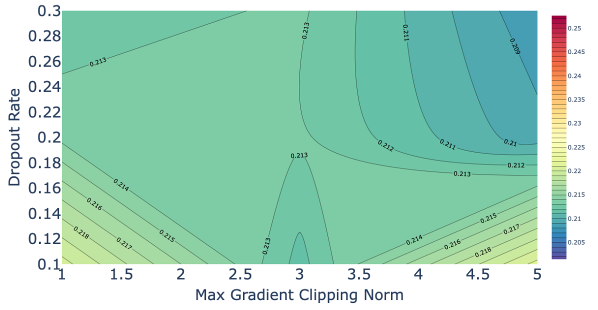
The results in Figure 6 highlight three key findings. First, **increasing the perturbation level consistently improves performance across all model complexities**. For the Narrow Model, the PER decreases from 34% to 28% as perturbation increases from 1 to 3, while the Deep+Wide Model improves from 22% to 20%. This shows that perturbation enhances generalisation for models of all sizes.

Second, the **effect of adding noise depends on model size**. In the Narrow Model, 50% noise keeps PER high, around 30%, even at high perturbation. In contrast, the Deep+Wide Model achieves its lowest PER of 20% with 50% noise at Perturbation 3. Larger models effectively leverage noise as regularisation, whereas smaller models are more sensitive to it.

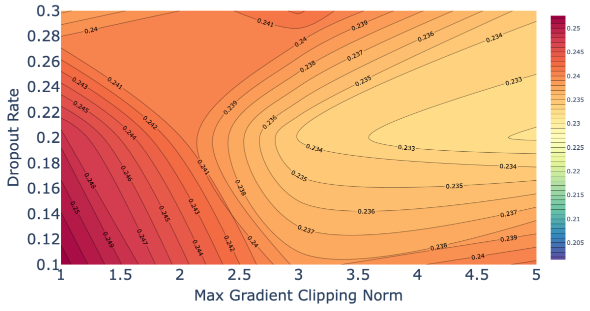
Finally, **increasing depth outperforms increasing width**. The Deep Model achieves a PER of 21%, close to the 20% of the Deep+Wide Model, despite having fewer parameters. This indicates that deeper models offer better gains than wider ones, particularly when paired with regularisation strategies like noise and perturbation.



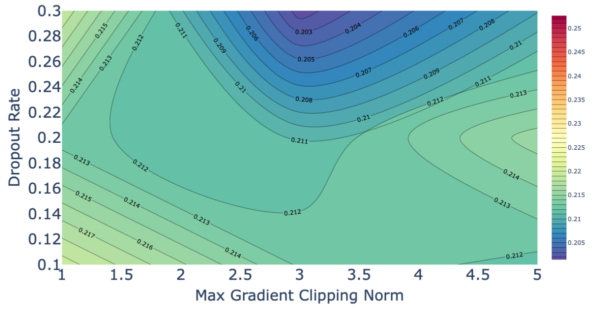
(a) Noise level: 0.0, Perturbation: 1x



(b) Noise level: 0.0, Perturbation: 3x



(c) Noise level: 50%, Perturbation: 1x



(d) Noise level: 50%, Perturbation: 3x

Figure 5: Contour plots illustrating the impact of noise and perturbation levels on PER(%) Validation Error. The x-axis represents the clipping norm, and the y-axis indicates the dropout rate. The color gradient conveys performance: warmer colors signify higher validation error (poor performance), while cooler colors correspond to lower validation error (better performance).

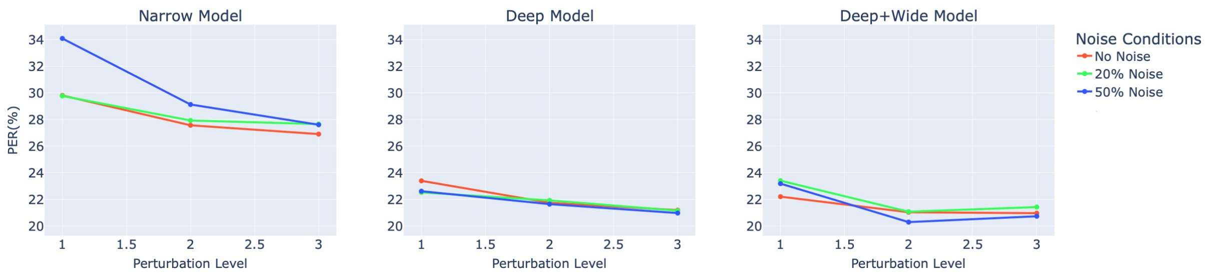


Figure 6: Performance (PER%) of Narrow, Deep, and Deep+Wide models across perturbation levels under varying noise conditions (No Noise, 20% Noise, 50% Noise). The Deep+Wide model achieves the lowest error.

5. Implicit and Explicit Regularisation Comparison

We have explored how audio augmentation strategies interact with explicit regularisation methods, such as dropout and gradient clipping, and how these interactions vary across different model sizes. Our results demonstrate that audio augmentation complements explicit regularisation by reducing sensitivity to hyperparameter changes and preventing sharp performance drops caused by suboptimal regularisation settings.

A key question that emerges is: **Can data augmentation replace explicit regularisation techniques?** To answer this, we compare the performance of models under three scenarios:

1. **Explicit Regularisation Only:** Models trained using dropout and gradient clipping, without any data augmentation.
2. **Data Augmentation Only:** Models trained using speed perturbation and noise addition, without applying explicit regularisation.
3. **Combined Approach:** Models trained with both explicit regularisation and data augmentation.

The best-performing models for each scenario, as identified in Sections 3 and 4, are now evaluated on the test set. The architecture selected is BI-LSTM, all have the same 3L-256H setting. The evaluation includes both **aggregate metrics**, such as PER, and a **confusion matrices**. Additionally, a granular analysis is performed on individual test samples to highlight the qualitative differences between the three approaches.

5.1. Aggregated Comparison

5.1.1 PER

Table 6 compares models trained using data augmentation, weight regularisation, and a combined approach. The results demonstrate that while each method individually improves performance, the combined approach yields the best outcomes across the training, validation, and test sets, achieving a Training PER of 20.98%, a Validation PER of 22.01%, and a Test PER of 22.43%.

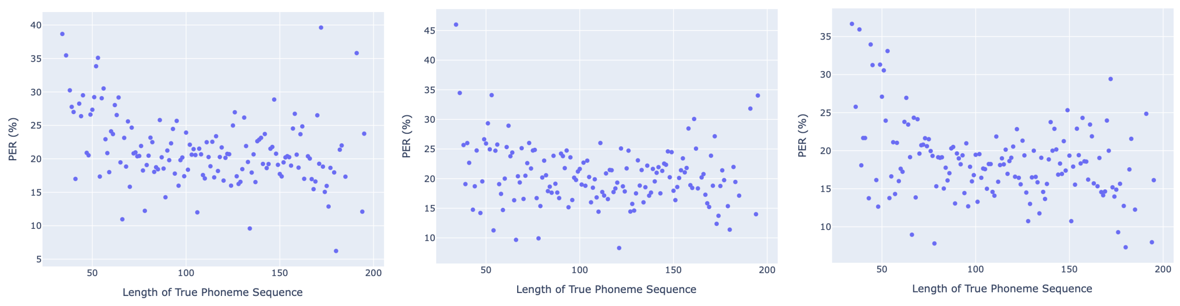
Data augmentation helps reduce the PER, but on its own, it is not quite as effective as weight regularisation. However, when you combine the two, they work even better together, improving both generalisation and stability. This shows that while data augmentation can partly take the place of explicit regularisation, the best results come from using both methods together. Therefore, it is not advisable to completely replace weight regularisation with data augmentation. Instead, combining the two is the most effective approach.

Table 6: Comparison of PER for Three Different Models through the pipeline

Model	**Train PER (%)**	**Validation PER (%)**	**Test PER (%)**
Data Augmentation	21.34	22.93	24.42
Weight Regularisation	21.27	23.32	23.48
Combined	20.01	20.29	22.43

5.1.2 PER & Utterance Length

The scatter plots in Figure 7 illustrate how the PER changes with the length of the true phone sequence for three models. For the Data Augmentation model (a), PER shows significant variation for shorter sequences but gradually stabilises as the sequence length increases. The Regularisation model (b) produces more consistent results, with less variation across most sequence lengths. However, the Combined Approach (c) delivers the best overall performance, achieving lower PER and greater stability across the range. At both tails of the plot—short and long sequences—the Combined Approach shows fewer extreme examples of high error rates, although a small number of poor predictions persist on the lower tail. This suggests that combining data augmentation and regularisation not only improves the model's generalisation across typical sequence lengths but also reduces the frequency of outliers at the extremes.



(a) Data Augmentation (b) Regularisation (c) Combined

Figure 7: Scatter plots illustrating the relationship between Test PER (%) and the length of the true phoneme sequence for three models.

5.1.3 Confusion Matrix

We analysed the True Positive and False Positive predictions of phonemes using confusion matrices for models trained with Data Augmentation (Figure 9), Augmentation + Regularisation (Figure 10), and Regularisation alone (Figure 11). All three models show a strong diagonal, meaning they get most predictions right. However, there are some common challenges: for example, all the models often confuse *ih* with *ah* and mix up *s* with *z*. These patterns suggest that certain phonemes are inherently harder to distinguish, regardless of the training approach used.

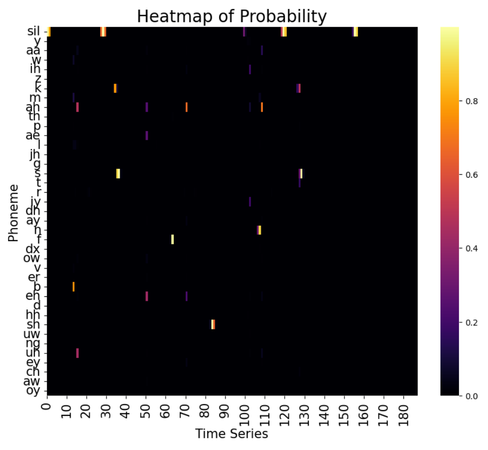
5.2. Sample-level Comparison

Figure 8 presents heatmaps and softmax probability plots for the three models, illustrating their performance at the single-sample level. All three models demonstrate strong performance, with heatmaps showing focused high-probability regions and softmax probabilities closely aligning with true phone boundaries. This suggests all models are robust, though direct comparisons on a single example are less meaningful due to the subtlety of their differences.

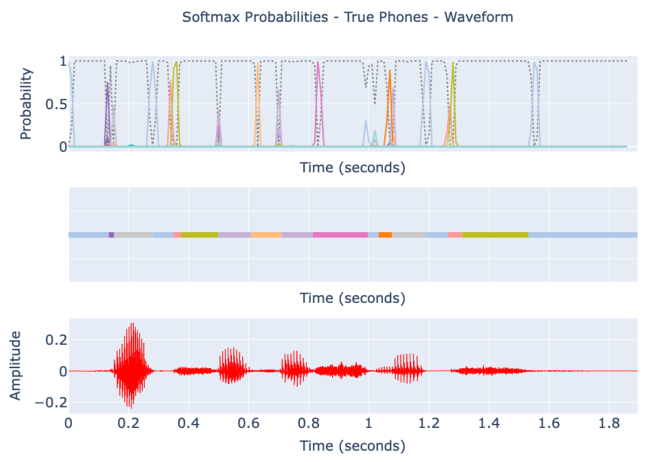
On closer inspection, the Augmentation with Regularisation model emerges as the most refined. Its heatmap displays the most focused high-confidence regions, with minimal scattered low-confidence areas. Additionally, its softmax probability curves are smoother and more consistent, with clearer transitions between phones and fewer dips in confidence. Predictions align more precisely with true phone boundaries, enhancing overall performance. In contrast, the Augmentation Only model exhibits slightly less stable predictions, with more frequent dips in softmax probability and sharper fluctuations during phone transitions. The Regularisation Only model, meanwhile, shows weaker alignments and noisier predictions, with phones represented by orange and blue lines displaying lower probabilities and more overlap with competing predictions.

6. Conclusion and Future Work

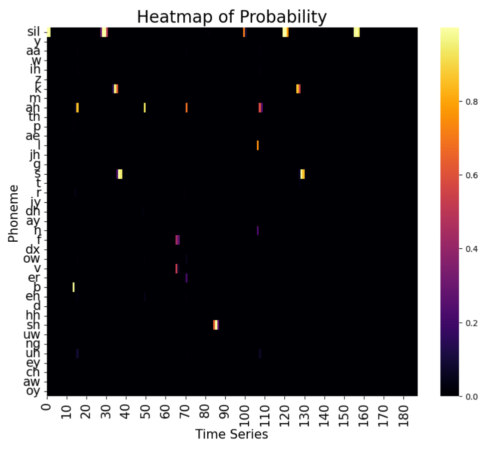
This work clearly outlines that combining data augmentation, using perturbation and noise addition, with weight regularisation improves model performance by reducing hyperparameter sensitivity and making the model more robust to variations in utterance length. We also learnt the behaviour of optimisers and complexity: SGD with Scheduler worked the best, while increasing the depth of the model is better than width. Additionally, Bayesian optimisation significantly accelerated the hyperparameter search process, enabling more efficient exploration of the parameter space. For future work, one can explore more general data augmentation techniques such as Vocal Tract Length Perturbation [9], as well as more advanced decoding method with beam search and dynamic programming. More modern end-to-end ASR frameworks such as [2] are also worth exploring.



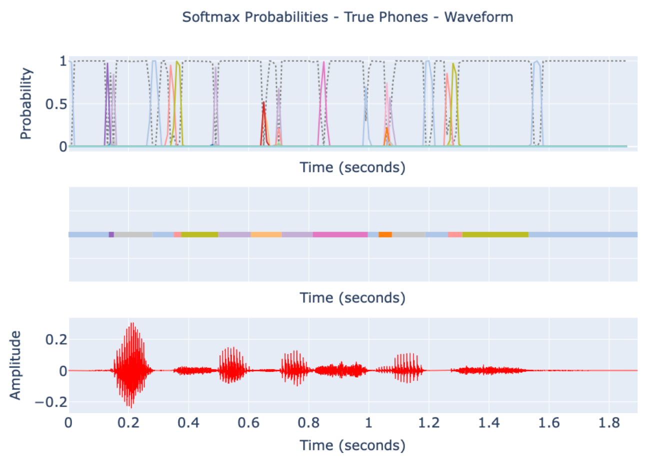
(a)



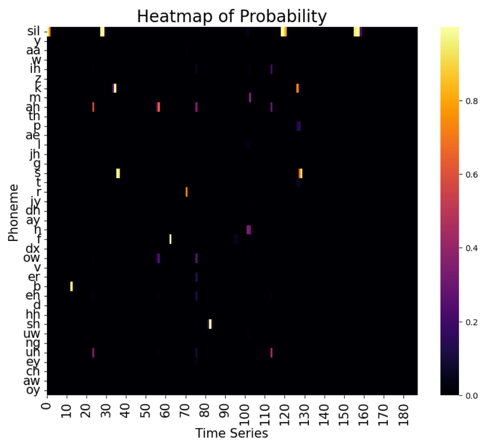
(b)



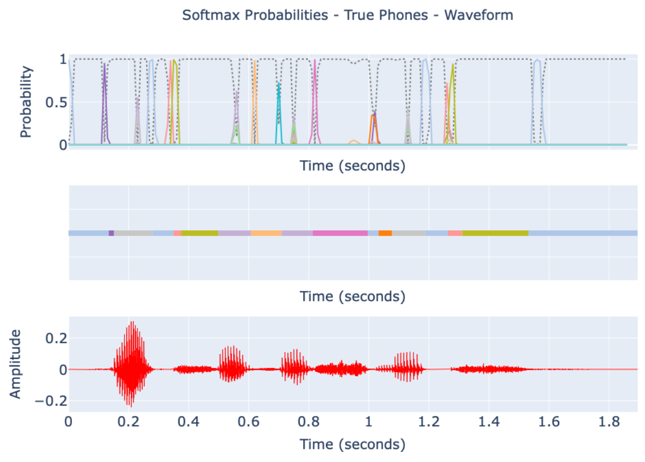
(c)



(d)



(e)



(f)

Figure 8: Comparison of Heatmaps and Softmax Probabilities for Three Scenarios: Model with Augmentation (top row), Mode with Augmentation and Regularisation (middle row), and Model with only Regularisation (bottom row). Each pair shows the Heatmap of Probability (left) and Predicted Softmax Probabilities per phone over Time with Waveform (right).

References

[1] Takuya Akiba et al. “Optuna: A Next-generation Hyperparameter Optimization Framework”. In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (2019), pp. 2623–2631. DOI: 10.1145/3292500.3330701. URL: https://optuna.org.

[2] William Chan et al. “Listen, attend and spell: A neural network for large vocabulary conversational speech recognition”. In: *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2016, pp. 4960–4964.

[3] John S. Garofolo et al. *TIMIT Acoustic-Phonetic Continuous Speech Corpus*. Philadelphia, PA, USA, 1993. DOI: 10.35111/17gk-bn40. URL: https://catalog.ldc.upenn.edu/LDC93S1.

[4] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. ISBN: 9780262035613. URL: https://www.deeplearningbook.org.

[5] Alex Graves et al. “Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks”. In: *Proceedings of the 23rd International Conference on Machine Learning (ICML)*. ACM, 2006, pp. 369–376. DOI: 10.1145/1143844.1143891. URL: https://doi.org/10.1145/1143844.1143891.

[6] A. Hannun et al. “Deep Speech: Scaling up End-to-End Speech Recognition”. In: *CoRR* abs/1412.5567 (2014). arXiv: 1412.5567. URL: http://arxiv.org/abs/1412.5567.

[7] Geoffrey Hinton et al. “Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups”. In: *IEEE Signal Processing Magazine* 29.6 (2012), pp. 82–97. DOI: 10.1109/MSP.2012.2205597.

[8] Sepp Hochreiter and Jürgen Schmidhuber. “Long Short-Term Memory”. In: *Neural Computation* 9.8 (1997), pp. 1735–1780. DOI: 10.1162/neco.1997.9.8.1735. URL: https://doi.org/10.1162/neco.1997.9.8.1735.

[9] Navdeep Jaitly and Geoffrey E. Hinton. “Vocal Tract Length Perturbation (VTLP) improves speech recognition”. In: *Proceedings of the 30th International Conference on Machine Learning (ICML)*. JMLR: W&CP. 2013, p. 28.

[10] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. “Practical Bayesian Optimization of Machine Learning Algorithms”. In: *Advances in Neural Information Processing Systems*. Vol. 25. 2012, pp. 2951–2959.

[11] David Snyder, Guoguo Chen, and Daniel Povey. *MUSAN: A Music, Speech, and Noise Corpus*. arXiv:1510.08484v1. 2015. eprint: 1510.08484.

A. Appendix

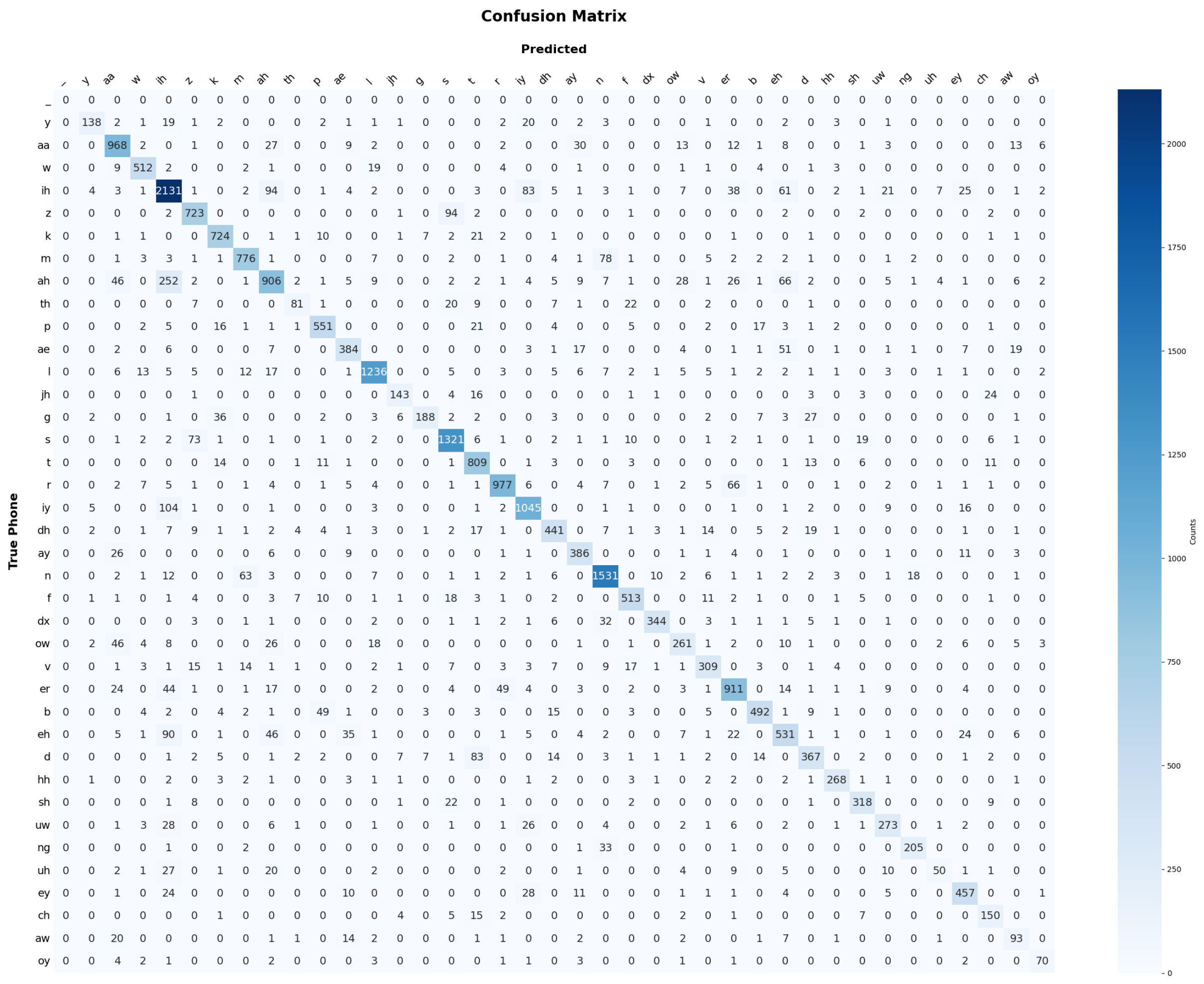


Figure 9: Confusion Matrix for model trained with Data Augmentation

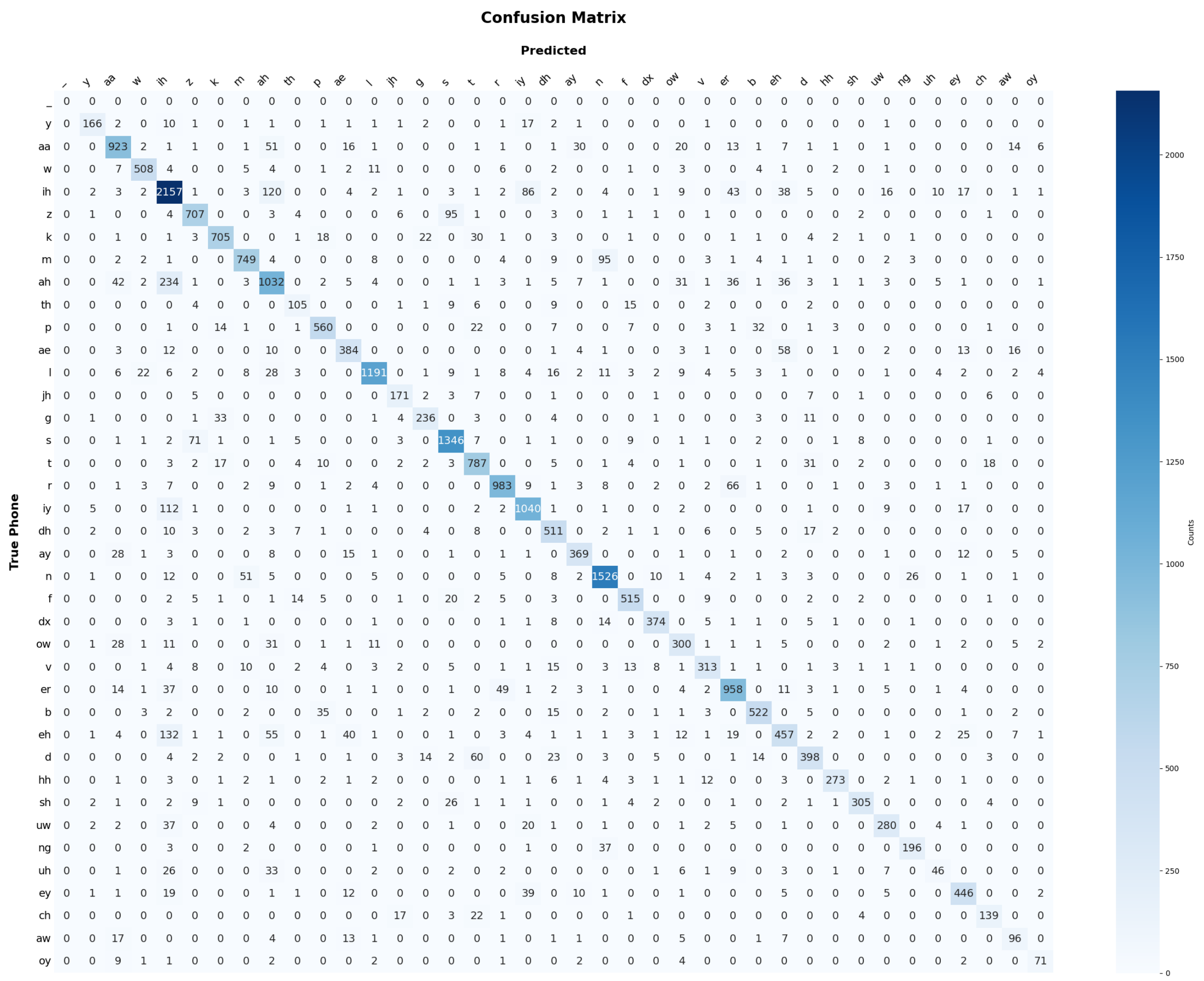
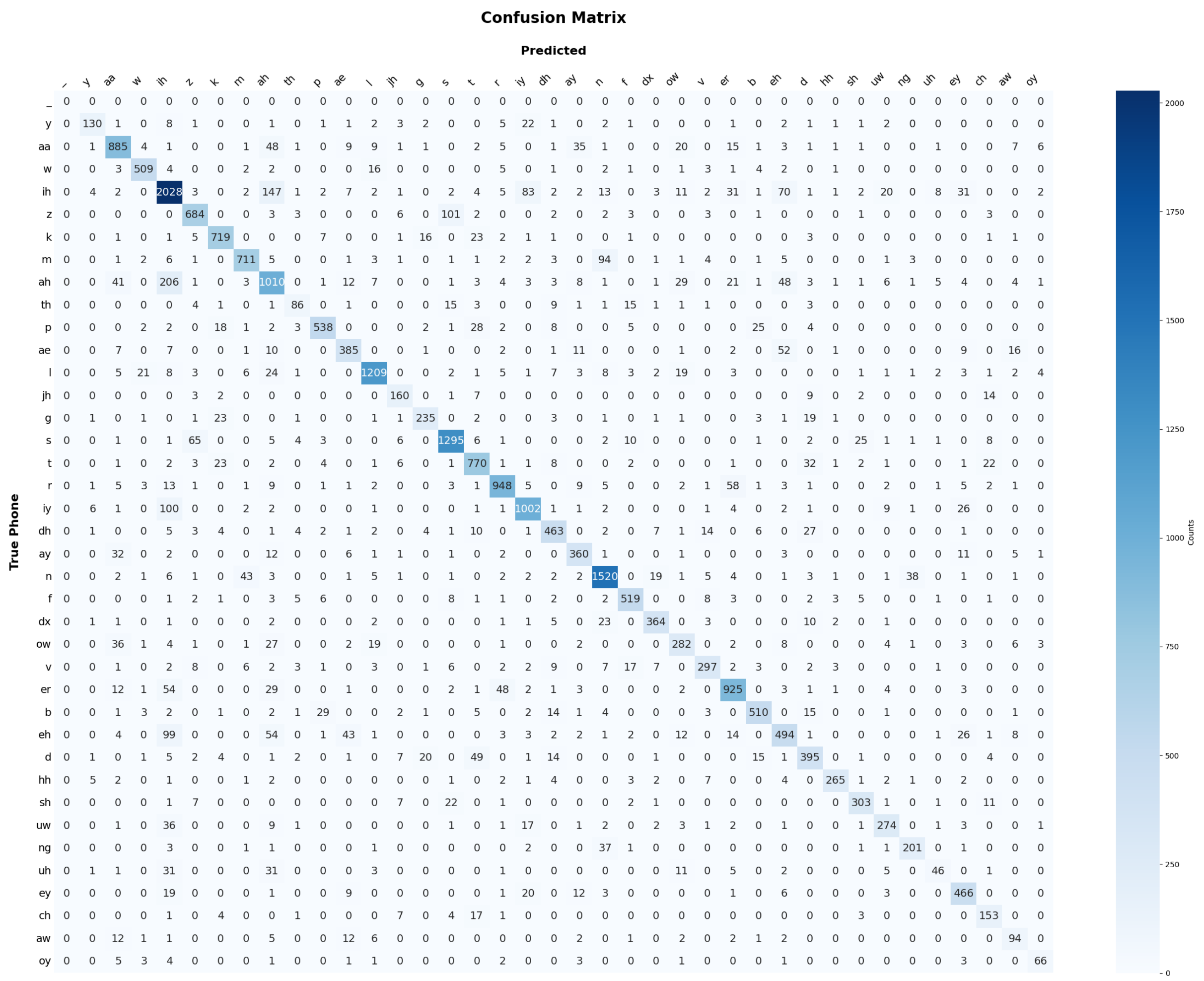


Figure 10: Confusion Matrix for model trained with Data Augmentation and Regularisation



	_	y	aa	w	ih	z	k	m	ah	th	p	ae	l	jh	g	s	t	r	iy	dh	ay	n	f	dx	ow	v	er	b	eh	d	hh	sh	uw	ng	uh	ey	ch	aw	oy
_	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
y	0	130	1	0	8	1	0	0	1	0	1	1	2	3	2	0	0	5	22	1	0	2	1	0	0	0	1	0	2	1	1	1	2	0	0	0	0	0	0
aa	0	1	885	4	1	0	0	1	48	1	0	9	9	1	1	0	2	5	0	1	35	1	0	0	20	0	15	1	3	1	1	1	0	0	1	0	0	7	6
w	0	0	3	509	4	0	0	2	2	0	0	0	16	0	0	0	0	5	0	1	0	2	1	0	1	3	1	4	2	0	1	0	0	0	0	0	0	0	0
ih	0	4	2	0	2028	3	0	2	147	1	2	7	2	1	0	2	4	5	83	2	2	13	0	3	11	2	31	1	70	1	1	2	20	0	8	31	0	0	2
z	0	0	0	0	0	684	0	0	3	3	0	0	0	6	0	101	2	0	0	2	0	2	3	0	0	3	0	1	0	0	0	1	0	0	0	0	3	0	0
k	0	0	1	0	1	5	719	0	0	0	7	0	0	1	16	0	23	2	1	1	0	0	1	0	0	0	0	0	0	3	0	0	0	0	0	0	1	1	0
m	0	0	1	2	6	1	0	711	5	0	0	1	3	1	0	1	1	2	2	3	0	94	0	1	1	4	0	1	5	0	0	0	1	3	0	0	0	0	0
ah	0	0	41	0	206	1	0	3	1010	0	1	12	7	0	0	1	3	4	3	3	8	1	0	1	29	0	21	1	48	3	1	1	6	1	5	4	0	4	1
th	0	0	0	0	0	4	1	0	1	86	0	1	0	0	0	15	3	0	0	9	1	1	15	1	1	1	0	0	0	3	0	0	0	0	0	0	0	0	0
p	0	0	0	2	2	0	18	1	2	3	538	0	0	0	2	1	28	2	0	8	0	0	5	0	0	0	0	25	0	4	0	0	0	0	0	0	0	0	0
ae	0	0	7	0	7	0	0	1	10	0	0	385	0	0	1	0	0	2	0	1	11	0	0	0	1	0	2	0	52	0	1	0	0	0	0	9	0	16	0
l	0	0	5	21	8	3	0	6	24	1	0	0	1209	0	0	2	1	5	1	7	3	8	3	2	19	0	3	0	0	0	0	1	1	1	2	3	1	2	4
jh	0	0	0	0	0	3	2	0	0	0	0	0	0	160	0	1	7	0	0	0	0	0	0	0	0	0	0	0	0	9	0	2	0	0	0	0	14	0	0
g	0	1	0	1	0	1	23	0	0	1	0	0	1	1	235	0	2	0	0	3	0	1	0	1	1	0	0	3	1	19	1	0	0	0	0	0	0	0	0
s	0	0	1	0	1	65	0	0	5	4	3	0	0	6	0	1295	6	1	0	0	0	2	10	0	0	0	0	1	0	2	0	25	1	1	1	0	8	0	0
t	0	0	1	0	2	3	23	0	2	0	4	0	1	6	0	1	770	1	1	8	0	0	2	0	0	0	1	0	0	32	1	2	1	0	0	1	22	0	0
r	0	1	5	3	13	1	0	1	9	0	1	1	2	0	0	3	1	948	5	0	9	5	0	0	2	1	58	1	3	1	0	0	2	0	1	5	2	1	0
iy	0	6	1	0	100	0	0	2	2	0	0	0	1	0	0	0	1	1	1002	1	1	2	0	0	0	1	4	0	2	1	0	0	9	1	0	26	0	0	0
dh	0	1	0	0	5	3	4	0	1	4	2	1	2	0	4	1	10	0	1	463	0	2	0	7	1	14	0	6	0	27	0	0	0	0	0	1	0	0	0
ay	0	0	32	0	2	0	0	0	12	0	0	6	1	1	0	1	0	2	0	0	360	1	0	0	1	0	0	0	3	0	0	0	0	0	0	11	0	5	1
n	0	0	2	1	6	1	0	43	3	0	0	1	5	1	0	1	0	2	2	2	2	1520	0	19	1	5	4	0	1	3	1	0	1	38	0	1	0	1	0
f	0	0	0	0	1	2	1	0	3	5	6	0	0	0	0	8	1	1	0	2	0	2	519	0	0	8	3	0	0	2	3	5	0	0	1	0	1	0	0
dx	0	1	1	0	1	0	0	0	2	0	0	0	2	0	0	0	0	1	1	5	0	23	0	364	0	3	0	0	0	10	2	0	1	0	0	0	0	0	0
ow	0	0	36	1	4	1	0	1	27	0	0	2	19	0	0	0	0	1	0	0	2	0	0	0	282	0	2	0	8	0	0	0	4	1	0	3	0	6	3
v	0	0	1	0	2	8	0	6	2	3	1	0	3	0	1	6	0	2	2	9	0	7	17	7	0	297	2	3	0	2	3	0	0	0	1	0	0	0	0
er	0	0	12	1	54	0	0	0	29	0	0	1	0	0	0	2	1	48	2	1	3	0	0	0	2	0	925	0	3	1	1	0	4	0	0	3	0	0	0
b	0	0	1	3	2	0	1	0	2	1	29	0	0	2	1	0	5	0	2	14	1	4	0	0	0	3	0	510	0	15	0	0	1	0	0	0	0	1	0
eh	0	0	4	0	99	0	0	0	54	0	1	43	1	0	0	0	0	3	3	2	2	1	2	0	12	0	14	0	494	1	0	0	0	0	1	26	1	8	0
d	0	1	0	1	5	2	4	0	1	2	0	1	0	7	20	0	49	0	1	14	0	0	0	1	0	0	0	15	1	395	0	1	0	0	0	0	4	0	0
hh	0	5	2	0	1	0	0	1	2	0	0	0	0	0	0	1	0	2	1	4	0	0	3	2	0	7	0	0	4	0	265	1	2	1	0	2	0	0	0
sh	0	0	0	0	1	7	0	0	0	0	0	0	0	7	0	22	0	1	0	0	0	0	2	1	0	0	0	0	0	0	0	303	1	0	1	0	11	0	0
uw	0	0	1	0	36	0	0	0	9	1	0	0	0	0	0	1	0	1	17	0	1	2	0	2	3	1	2	0	1	0	0	1	274	0	1	3	0	0	1
ng	0	0	0	0	3	0	0	1	1	0	0	0	1	0	0	0	0	0	2	0	0	37	1	0	0	0	0	0	0	0	0	1	1	201	0	1	0	0	0
uh	0	1	1	0	31	0	0	0	31	0	0	0	3	0	0	0	0	1	0	0	0	0	0	0	11	0	5	0	2	0	0	0	5	0	46	0	1	0	0
ey	0	0	0	0	19	0	0	0	1	0	0	9	0	0	0	0	0	1	20	0	12	3	0	0	0	0	1	0	6	0	0	0	3	0	0	466	0	0	0
ch	0	0	0	0	1	0	4	0	0	1	0	0	0	7	0	4	17	1	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	153	0	0
aw	0	0	12	1	1	0	0	0	5	0	0	12	6	0	0	0	0	0	0	0	2	0	1	0	2	0	2	1	2	0	0	0	0	0	0	0	0	94	0
oy	0	0	5	3	4	0	0	0	1	0	0	1	1	0	0	0	0	2	0	0	3	0	0	0	1	0	0	0	1	0	0	0	0	0	0	3	0	0	66

Figure 11: Confusion Matrix for model trained with only Regularisation