Combining LSTM and Convolution Neural Networks for Phoneme Recognition

Eysteinn Gunnlaugsson
eysgun@kth.se

Pierre Sevestre
sevestre@kth.se

Abstract

In this project we evaluate the effect creating a network for phoneme classification which consists of a combination of Dense, Convolutional, and Recurrent layers. We start by creating baseline Convolutional and Recurrent models and evaluate their performance on the TIMIT dataset. We then create a model which is a combination of the baseline models and measure if there are any improvements. Experimental results show that the combination of the two leads to significant improvements for all metrics used for evaluation.

1 Introduction

Phoneme recognition has been performed for a long time by the combination of Gaussian Mixture Model (GMM) and Hidden Markov Model (HMM), maximizing the likelihood of feature emitted by a state of the HMM that is modeled with a GMM. Recent advances in deep learning lead to the emergence of Deep Neural Network (DNN) combined with HMM, with DNN maximizing the probability of being in a certain HMM state given the input feature.
However, these DNN don't have the ability to model dependency within speech and thus don't take full advantage of the dynamic feature provided as input.
Hence, Convolution Neural Networks [1] (CNN), commonly used in computer vision, can be used to process this temporal local dependency and extract feature to feed the neural network.
Similarly, Recurrent Neural Networks (RNN) proved to be efficient to model sequences [2], especially Long-Short Term Memory (LSTM), introduced in [3], are able to reveal wider dependencies.
In this paper we analyze the combination of the two methods stated above, resulting in a Convolutional LSTM, as in the papers [4] and [5].

2 Method

2.1 Implementation

All our neural-networks were built using Python. We utilized the Keras library, running on a Tensorflow back-end, which allowed us to built complex architectures in a sophisticated and simple manner. We ran all our experiments on a cluster hosted by Amazon Web Services(AWS). The cluster used has a NVIDIA Tesla K80 GPU which allowed us to train relatively complex models in a reasonable amount of time.

2.2 Neural network training

To train multiple iterations of similar networks with different hyper parameters we structured our code in a manner that allowed us to create a queue of model structures that were to be trained. We

Submitted to 30th Conference on Neural Information Processing Systems (NIPS 2016). Do not distribute.

stored the resulting models in a `.h5` file that could later be loaded by Keras. We then evaluated all our trained models using the methods described in 3.2

To optimize our training procedure we added both `early-stopping`(ES) and `learning rate reduce on plateau`(LRRP). LRRP reduced the learning rate of the model by a factor of 0.4 for every epoch where the validation-loss did not decrease. ES stopped training if the validation loss had not decreased for 4 consecutive epochs. The combination of these two methods made sure our models did not over-fit to the training data and allowed the models to train for a few epochs longer, achieving a better overall performance.

2.3 Feature selection

Speech samples are divided into Hamming windows 25 ms, with 12.5 ms overlap. These frames are transformed to MSPEC using Fourier transform and Mel Filterbank. The MFCC features are generated by compressing the MSPEC with a cosine transformation to reduce covariance between coefficients that is usually an assumption for GMM models. However, previous work shows that neural networks benefit from using less handcrafted features[6], as some information might be lost in the process, and in this case, a persisting correlation is not detrimental to DNN efficiency, we decided to use the MSPEC features for all our training.
These features were then normalized using the mean and standard deviation of the training dataset Eq.(1).

$$\frac{x_i - \mu(x_{train})}{std(x_{train})} \tag{1}$$

2.4 Convolution Neural Network

CNN's were introduced in [1] to deal with images and allow the network to extract features from the inputs. Since then they have become very popular with all sorts of classification tasks, especially image recognition. Human speech is a sequential signal with local correlations in both time and frequency and it can be transformed in to a feature map similar to an image, making CNN's an ideal candidate for the task of phoneme classification.
It has been the recent trend to create very deep convolutional models for speech recognition [7] with very good results. However the authors of [8] suggest that even adding 1 or 2 convolutional layers before fully connected layers can greatly improve the networks performance, chapter 2.6 discusses the importance of this fact further.

2.5 LSTM and BLSTM

LSTM networks were introduced in [3] to solve the Recurrent Neural Network issue of vanishing and exploding gradient, while capturing long term and short term dependencies. This is achieved with units called *memory block* in the hidden layer. Each memory block has an *input gate* and an *output gate*. These gates control the *error flow* going in and out of the cell. A *forget gate* was then added in [9] to form the cell used here. Equations governing timestep t can be found below :

$$\begin{aligned}
i_t &= \sigma(W_{ih}h_{t-1} + W_{ix}x_t) + b_i \\
f_t &= \sigma(W_{fh}h_{t-1} + W_{fx}x_t) + b_f \\
o_t &= \sigma(W_{oh}h_{t-1} + W_{[}ox]x_t)b_o \\
c_t &= f_t \odot c_{t-1} + i_t \odot tanh(W_{ch}h_{t-1} + W_{cx}x_t + b_c) \\
h_t &= o_t \odot \tanh(c_t)
\end{aligned}$$

Where i, f, o and c stand for input gate, forget gate, output gate and cell activation vectors, and h the cell output. W refers to weight, b biases for each gate, and x the input for the given time step. $\odot$ correspond to element-wise multiplication, and σ the sigmoid activation function. These input, output, forget and memory elements can be seen in the Figure 1.

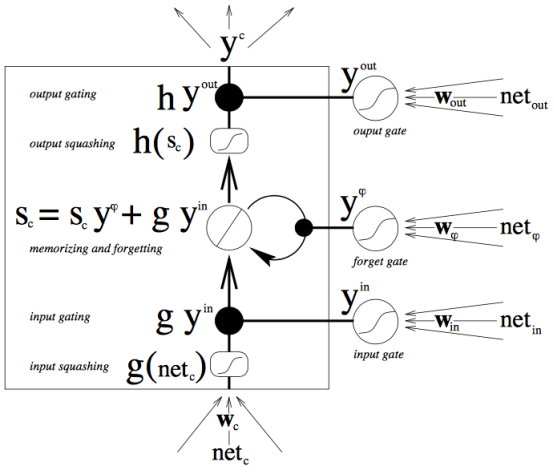

Figure 1: Single memory block with on cell cited from [9]

Bidirectional Recurrent Neural Network (BRNN) were then introduced by [10] to allow the RNN to take advantage of future frames. As can be seen in Figure 2, it consist of two RNN connected to the same outputs, but with flow going in the two opposite direction, forward and backward. Using this model for speech takes advantage of the intuition that it is as relevant to use future frames to predict the present phoneme as it is to use past frames.

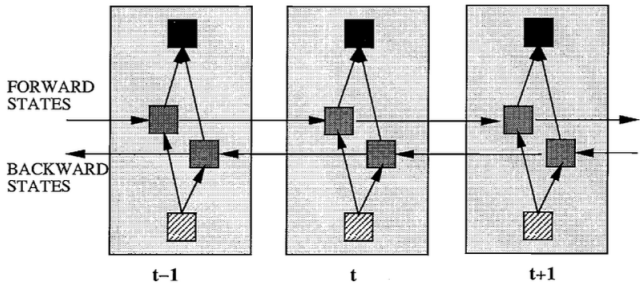

Figure 2: General structure of Bidirectional RNN for 3 steps cited from [10], with black box as output, grey box as hidden state and striped box as input

LSTM were first used to perform framewise phoneme classification in [11], where they demonstrate the ability of this architecture to capture context, fundamental in speech processing, resulting in slightly higher accuracy than RNN. Bidirectional LSTM (BLSTM), also proved to be much more efficient than direct LSTM, from 64.6% accuracy for LSTM up to 68.9% for BLSTM on the TIMIT dataset in [11].

2.6 Combining LSTM and CNN

A motivation to combine CNN and LSTM is to benefit from the local and long term patterns of the sequence. The authors of [4] suggests the convolution layers help reduce the frequency variance of the input frames, making it easier for the LSTM to learn the temporal dependency. The output of the CNN is also fed into a fully connected layer in the intend to reduce the temporal variation and reduce the computational cost.
This ability of the CNN-DNN to discriminate features can be observed on Figure 3.

Previous comments motivate the architecture represented on Figure 4, with inputs feeding two convolutions layers, followed by a dense layer before two LSTM layers. This choice of two layers

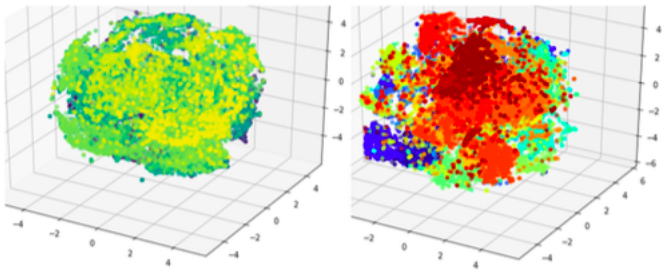

Figure 3: Comparison of raw FBANK (left) and features learned from ResNet cited from [12], obtained projecting FBANK and output of average pooling layer from ResNet using t-SNE

for the convolutions and LSTM was justified by [12] highlighting a relative better performance of 2 layer models over single layer models.

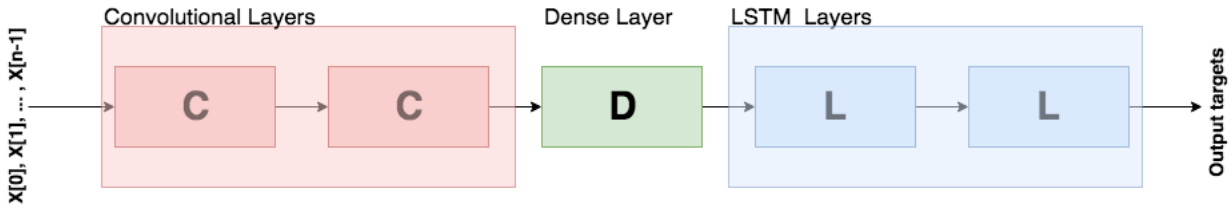

Figure 4: CNN-LSTM architecture

3 Experiments

3.1 TIMIT dataset

We used the TIMIT [13] dataset to perform our experiment. This dataset contains sentences from 630 speakers in American English, from eight different dialects, each repeating ten sentences composed of 65 phonemes, with high phoneme variability. 462 speakers are used to from the training dataset, and 168 for the testing. The sentences are sampled at 16 KHz.

3.2 Model evaluation

To train our models we split the train set in two parts, 90% for training and 10% for validation. We used the full test set to measure the performance of our trained models.
Since some phonemes are more dominant in the dataset then others, accuracy is an unreliable metric [14] and is therefore not used to evaluate the performance of our models on the test data. In fact, our first models showed that dominant classes, especially silence, had very good results compared to poorly represented classes, that was not reflected in the accuracy.
To evaluate our trained models we use both F1-score and Phone Error Rate (PER). The formula for F1-score can be seen in Equation 2 while the PER is the average number of corrections(insertions, deletions, substitutions), normalized, that need to be done to the predicted phonemes of an utterance to generate the correct one. The PER is computed on merged prediction where adjacent identical phoneme are merged together. Both of these methods provide a reliable performance measurement of a model where there are dominant classes.

$$F_1 = 2 * \frac{precision * recall}{precision + recall} \tag{2}$$

These metrics were computed over a reduced number of phonemes, as suggested in [12] to correspond better to the real signification of these phonemes. The original 61 phonemes resulted in 39 categories. The phonemes that have been merged can be found in Table 1, with one relevant example being

the merge of er (bird) and axr (butter), or the merge of closure intervals and pauses into a silence category.

Table 1: Classes of merged phonemes from TIMIT for evaluation

Phoneme category	List of merged phoneme
aa	aa, ao
ah	ax, ax-h, ah
er	er, axr
hh	hh, hv
ih	ih, ix
l	l, el
m	m, em
n	n ,en, nx
ng	ng, eng
sh	sh, zh
uw	uw, ux
sil	pcl, tcl, kcl, tck, bcl, dcl, gcl, h#, pau, epi

3.3 Baseline models

To ease the process of building our CNN-LSTM models we first train four different CNN models and seven LSTM models. This allows us to try out more hyper-parameter settings on smaller networks, before trying to integrate them into the more complex CNN-LSTM models. The results of these models, found in 4.1, also serve as a baseline for the performance we want to improve with our CNN-LSTM.
Since the goal of this paper is not to create the best CNN/LSTM models, but rather to see if the performance would improve by combining the two we limit the number of hidden layers to two layers for all of our baseline models.

Two different context length are used to compare our models, 17 and 31, as suggested in [12].

For our CNN models we try different settings for the number of filters, strides and context length. The size of kernels was set to (3,3) with no pooling layer, motivated by its ability to capture high-level representations with less computational complexity than traditionally used (6,6) kernels [4]. Feature map of size 32 and 64 were used, smaller than traditionally used, 256 in [5] for example, but it proved to be necessary for computational reason given the limited time frame of the project. A fully connected layer with 128 nodes was added to reduce the number of parameters and thus facilitate the combination without loss of information, as shown in [15]. The ReLU activation function, $x = max(0, x)$, was used for the hidden layers, as in [12][4] and the Adam optimizer [16] with a learning rate of 0.001.

For our recurrent models we try different context length values, both for LSTM and BLSTM models. The number of units used, 32 and 64 for both models result from a trade-off between model complexity and the need of computation. This number of unit is far from the one found in the literature, from 256 up to 1024 in [12] and [5]. All layers used the $\tanh$ activation function, as in [17], $x = \frac{1-e^{-2x}}{1+e^{-2x}}$ and the RMSProp optimizer, introduced by Geoffrey Hinton in the Coursera course *Neural Networks for Machine Learning*, with a learning rate of 0.001.

4 Results

4.1 CNN & LSTM

CNN Table 2 shows that the number of filters slightly improves the model performance, as should be expected since the number of parameters to shape the model increases.
Larger context lengths also prove to have a positive effect on the models performance, especially a significant increase in the PER, indicating that predictions are more stable, meaning that predicted adjacent phonemes are less likely to be different.

By increasing the stride sizes in our convolutional layers we are reducing the output space of each layer. While this reduces the training time of the model, it is not worth it since the performance of the model suffers.

Table 2: CNN training results

layers	filters	kernel sizes	strides	dense layer size	context length	f1	PER
2	32	(3,3)	(1,1)	128	13	0.72	0.62
2	64	(3,3)	(1,1)	128	13	0.73	0.61
2	32	(3,3)	(1,1)	128	31	0.71	0.58
2	32	(3,3)	(2,2)	128	31	0.71	0.62

LSTM Table 3 shows that larger context window greatly improves the efficiency of LSTM. While it could be argued that LSTM inherently capture temporal dependency of sequence, and should learn the context, using multiple frames as input actually helps it capture dependency.
Increasing the number of units to 64 sightly improves the PER while leaving the f1_score unchanged. The second model, 32-units with 31-context reaches 72.6% accuracy, that can be compared the the results obtained in [12], where 256-units and 512-units models end up with 78.2% and 79.6% accuracy respectively. While increasing units greatly improves accuracy, it seems that the PER score does not improve as much. Because of the trade-off between performance and model complexity, and the fact that this project intents to discuss model architecture rather than give state of the art results, we will use the 32-units model for our combined model evaluation.

Table 3: LSTM training results

layers	units	context length	f1	PER
2	32	13	0.72	0.54
2	32	31	0.72	0.46
2	64	13	0.72	0.52

BLSTM Results from Table 4 confirm the idea that a BLSTM model outperforms a LSTM model with same number of parameters, as 32*2-units 13-context window gives significantly better F1_score and PER than the 64-units 13-context window LSTM. An attempt to reduce the number of units to 16*2 was done to limit the number of parameters for the combined model. First, 16*2-units with 31-context window length gave poor results, that might be explained by the fact there is less memory units than input frames, forcing the network to compress and therefore lose information. The same 16*2-units model has poor performances with 13-context length window. With these results in mind we use the 32*2-units network for the combined model.

Table 4: BLSTM training results

layers	units	context length	f1	PER
2	16*2	13	0.65	0.58
2	16*2	31	0.65	0.59
2	32*2	13	0.74	0.50
2	32*2	31	0.74	0.39

4.2 Combined CNN-LSTM

Using our results in 4.1 as a reference, we have a good idea about which settings work best for each model. We try two different combined models based on these settings, that have reasonable amount of parameters to be able to train in a decent time. Each model consisted of five hidden layers, two convolutional layers, a single dense layer and two LSTM layers, followed by a fully connected output layer. The convolational and dense layers use the ReLU activation function while the LSTM uses the $\tanh$ function. All models were trained using the RMSProp optimizer with a learning rate of 0.001. Results, displayed in Table 5, show that each combined model outperforms its corresponding isolated models. The CNN-LSTM gives better f1_score and PER than any CNN or LSTM model from the

previous section 4.1, as well as the CNN-BLSTM for the CNN and BLSTM network. Combined models proved to significantly improve the PER score, indicating its ability to capture patterns of phonemes. The confusion matrix found in D depicts a more detailed evaluation of our model, for what phonemes it performs well and for which it has trouble classifying.

Table 5: CNN-LSTM training results

conv sizes	LSTM sizes	dense sizes	kernel sizes	context length	bidirectional	f1	PER
(32, 32)	(32, 32)	(32, 32)	(3,3)	31	false	0.73	0.42
(32, 32)	(32 * 2, 32 * 2)	(32, 32)	(3,3)	31	true	0.75	0.36

5 Discussion and Conclusions

The experiments show that the combination of CNN and LSTM models lead to a much better model. Further improvements could be to create larger networks, both CNN and LSTM. That would make the networks much more complex and give it more parameters to tune. This was considered out of scope for this project, due to the lack of computing power. It would however be very interesting to see the effects of adding layers, both to the isolated and the combined models, to see if the combined versions would still produce better results then the isolated ones. Moreover, these combined models are created using almost raw separate models, that could be further tuned, using asymmetric context windows or unrolling features for example, as applied in literature. One could also argue that shift invariance property of CNN is questionable in speech as patterns in low frequency differ in meaning from patterns in high frequency. Modification over the weight sharing of the kernel could be considered as in [18]. Further test could also be performed to evaluate the performances of the combined model in comparison with the baseline models, such as its robustness to noise.

References

[1] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, Nov 1998.

[2] R. Pascanu, C. Gulcehre, K. Cho, and Y. Bengio. How to Construct Deep Recurrent Neural Networks. *ArXiv e-prints*, December 2013.

[3] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997.

[4] Z. Zhang, Z. Sun, J. Liu, J. Chen, Z. Huo, and X. Zhang. Deep Recurrent Convolutional Neural Network: Improving Performance For Speech Recognition. *ArXiv e-prints*, November 2016.

[5] T. N. Sainath, O. Vinyals, A. Senior, and H. Sak. Convolutional, long short-term memory, fully connected deep neural networks. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4580–4584, April 2015.

[6] J. Li, L. Deng, Y. Gong, and R. Haeb-Umbach. An overview of noise-robust automatic speech recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 22(4):745–777, April 2014.

[7] T. Sercu and V. Goel. Advances in Very Deep Convolutional Neural Networks for LVCSR. *ArXiv e-prints*, April 2016.

[8] O. Abdel-Hamid, A. r. Mohamed, H. Jiang, L. Deng, G. Penn, and D. Yu. Convolutional neural networks for speech recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 22(10):1533–1545, Oct 2014.

[9] Felix A. Gers, Jürgen A. Schmidhuber, and Fred A. Cummins. Learning to forget: Continual prediction with lstm. *Neural Comput.*, 12(10):2451–2471, October 2000.

[10] M. Schuster and K. K. Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, Nov 1997.

[11] A. Graves and J. Schmidhuber. Framewise phoneme classification with bidirectional lstm networks. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 4, pages 2047–2052 vol. 4, July 2005.

[12] L. Xiaoyu. Deep Convolutional and LSTM Neural Networks for Acoustic Modelling in Automatic Speech Recognition. *Pearson Education Inc*, December 2017.

[13] et al. Garofolo, John S. Timit acoustic-phonetic continuous speech corpus. *LDC93S1*, 1993.

[14] Nitesh V. Chawla, Nathalie Japkowicz, and Aleksander Kotcz. Editorial: Special issue on learning from imbalanced data sets. *SIGKDD Explor. Newsl.*, 6(1):1–6, June 2004.

[15] Tara N. Sainath, Vijayaditya Peddinti, Brian Kingsbury, Petr Fousek, Bhuvana Ramabhadran, and David Nahamoo. Deep scattering spectra with deep neural networks for lvcsr tasks. In *INTERSPEECH*, 2014.

[16] D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. *ArXiv e-prints*, December 2014.

[17] H. Sak, A. Senior, and F. Beaufays. Long Short-Term Memory Based Recurrent Neural Network Architectures for Large Vocabulary Speech Recognition. *ArXiv e-prints*, February 2014.

[18] T. N. Sainath, B. Kingsbury, A.-r. Mohamed, G. E. Dahl, G. Saon, H. Soltau, T. Beran, A. Y. Aravkin, and B. Ramabhadran. Improvements to deep convolutional neural networks for LVCSR. *ArXiv e-prints*, September 2013.

A Peer review considerations

We received a very thorough and critical feedback. It is filled with valuable constructive criticism that we used to improve our report. Chapters A.1 and A.2 discuss in detail how we changed our report according to the review and what parts of the review we considered irrelevant.

A.1 Incorporated suggestions

The following sections list the changes made to the paper according to the review.

A.1.1 Value laden words

Our reviewer suggested that we removed value laden words like *"abundantly employed"* and *"have become immensely popular"* to increase readability. We agreed to this suggestion and changed this style of writing to a more clear wording.

A.1.2 Improve the presentation of equations

The review suggests that we write some of the smaller equations inline instead of referencing them. We tried this for the `tanh` and `ReLU` equations and agree that it does improve the report. We left the formulas for the F1-score and data normalization as is since we believe that they are complex enough to deserve their own lines. We also centered the LSTM formula as recommended by the reviewer.

A.2 Irrelevant criticism

While overall the peer review had some good suggestions we disagree with some of them. The following chapters list notable topics where our opinion or research differs from the author of the peer review.

A.2.1 No forward referencing

The peer review suggests that we should not reference chapters further down in the report, and only reference chapters already seen by the reader. We disagree with this. For example in chapter 2.2 we reference the chapter describing our feature extraction 3.2. We feel that this improves the clarity of the report and also allows the reader to jump straight to that chapter if he is interested to read that part first.

A.2.2 Suggested references

Our paper does not intent to discuss state of the art results on phoneme recognition, but to analyze the combination of several deep models. Thus, we chose not to include the first suggested paper, *Phonemes Classification with Recurrent Neural Networks, A. Esposito, R. Ceglia*, as they do not use MFCC freatures but Rasta-PLP representation, and because the neural network architecture is not detailed as it is not the intent of the paper.
We also chose not to include the paper *Comparison of Distance Metrics for Phoneme Classification based on Deep Neural Network Features and Weighted k-NN Classifier*, as it focuses on distance metrics used for k-NN, not evaluation metrics for speech recognition models.
Moreover, both paper had not been cited a lot, with only two citation for the first and no citation for the second, strengthening our decision not to use them.

B Further review

Mr. Salvi pointed out the unsuitability of convolution networks shift invariance in the frequency dimension for speech recognition. This was a very good observation and we did some research regarding this. However due to limited time we did not add any methods described in [18], leaving it as future work to analyze these methods further.

C Received peer review

The following sections show the received peer review.

C.1 Relevance for the learning outcomes

I feel that the project is highly relevant for the learning outcomes. In order to classify phonemes you have successfully compared three models against each other, and also varying the architecture of each individual model. Furthermore you have correctly used; "specific analysis and decision making methods for recognition of speakers" where you noticed that the phoneme classification is not properly measured in terms of accuracy. You have also successfully implemented your own models from previous studies which shows of high capability. Great job! 6/6 points

C.2 Literature study

To be honest, not many state of the art report are used in this paper. I feel that the relevance is somewhat there but not that many in phoneme recognition is present. Obviously neural networks and its different parameters are important but using relevant literature in establishing, or comparing those parameters would be a nice adding

** Suggested papers - Phonemes Classification with Recurrent Neural Networks, A. Esposito, R. Ceglia They use a recurrent neural network to phoneme recognition from the TIMIT dataset. Detailed description of the hyper parameters and number of layers.

- Comparison of Distance Metrics for Phoneme Classification based on Deep Neural Network Features and Weighted k-NN Classifier Although you have successfully identified that accuracy is not a viable option. This paper evaluates different metrics for phoneme recognition

3/6 points

C.3 Novelty / originality

The subject is there. But there is always room for improvement in this area. I feel that you have a fairly novel approach in combining different architectures and trying different parameters. In my opinion the combining part is sufficient as "vanilla" networks seems to have hit its mark. Connecting to the literature study I think that there could be more research done on what parameters you use. At the moment it seems Ad-hoc and not backed up. But maybe that's novelty... 4/6 points.

C.4 Correctness

The overall structure of the report is nice. Good quality figures which are described as well as the equations. However, I'm not sure about referencing to something further down the paper. I might be wrong, but I think that one reference upwards in the paper. Also, the referencing got a little out of hand when defining certain equations. Some of them could have easily been inserted in the text rather than linking to the equation a couple of lines down.

Also, there are some value laden words which can be removed. For example; line 16: ...abundantly employed in...., or line 54: ...have become immensely popular... Just some details of course but I think it would increase the readability of the report.

In summary, I like the report and I feel that you have done a great job in the paper. 4/6 points

C.5 Clarity of presentation

As stated above, there are some minor corrections to be made. The equation of LSTM could be nicely formatted as a algorithm or merely centered for visual pleasantness. The figures are also nice but it is clear that they are pasted in rather than built on your own. Obviously this is not needed, but it would increase the visual context. I think you have done a great job. If you fix some of the tweaks and motivate your choice of particular parameters, architectures and overall settings. Good job! 3/6 points

D Appendix

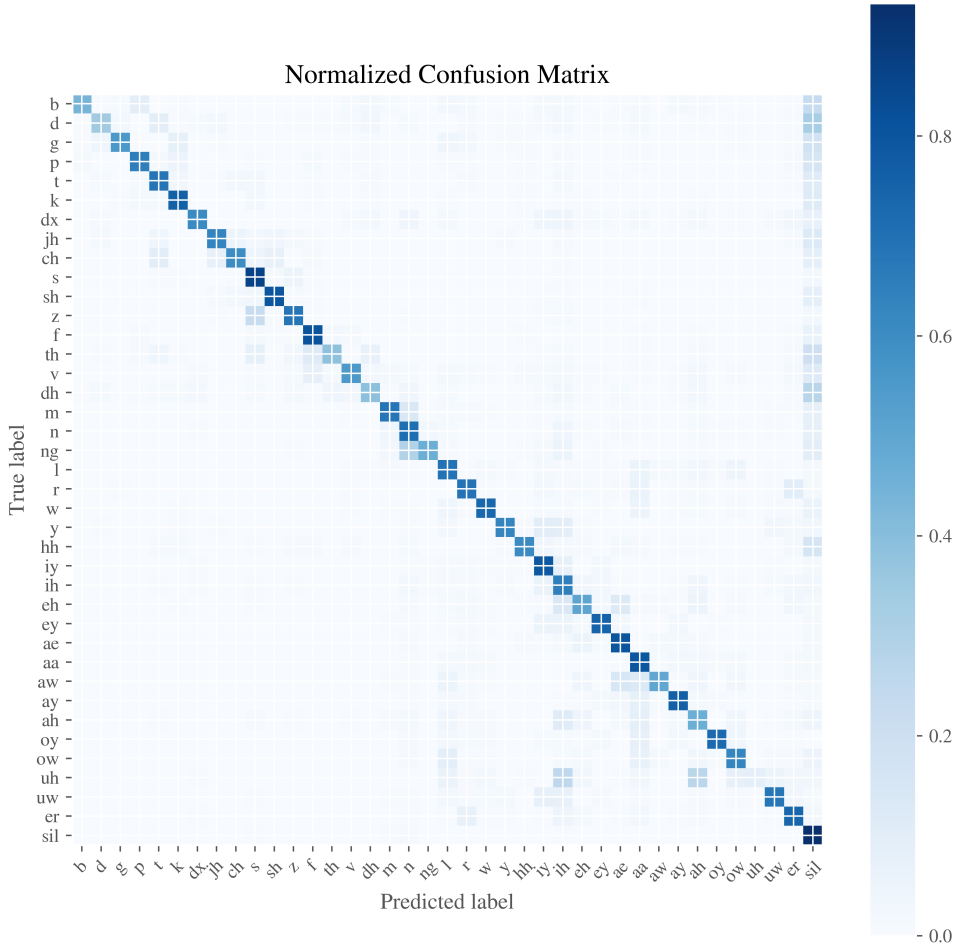

Figure 5: Confusion matrix for our best model