DEEP CONVOLUTIONAL NEURAL NETWORKS FOR ACOUSTIC MODELING IN LOW RESOURCE LANGUAGES

William Chan and Ian Lane

Carnegie Mellon University

ABSTRACT

Deep Neural Networks (DNNs) have given many state-of-the-art results to acoustic modelling in many Automatic Speech Recognition (ASR) tasks. More recently, Convolutional Neural Networks (CNNs) have demonstrated powerful acoustic modelling capabilities due to its ability to account for structural locality in the feature space, and CNNs have been shown to often outperform their DNNs counterparts on TIMIT and LVCSR. In this paper, we perform a detailed empirical study of CNNs under the low resource condition, wherein we only have 10 hours of training data. We find a two dimensional convolutional structure to perform the best, and emphasize the importance to consider time and spectrum in modelling acoustic patterns. We report detailed error rates and find CNNs to be more efficient and outperform fully connected DNNs.

Index Terms— Deep Neural Networks, Convolutional Neural Networks, Automatic Speech Recognition

1. INTRODUCTION

Deep Neural Networks (DNNs) have become a vital component to state-of-the-art acoustic models in Automatic Speech Recognition (ASR) [1]. Combined with Hidden Markov Models (HMM), hybrid DNN-HMM systems have produced state-of-the-art results in many ASR tasks [2, 3]. The DNN acoustic models are typically given several consecutive context frames of some spectral feature (e.g., log-Mel filter banks) as inputs and trained via supervised backpropagation [4] with softmax targets modelling the HMM acoustic states.

One fundamental drawback of DNNs is the fully connected nature, DNNs do not take advantage of structural locality in the feature space. The acoustic signals may contain fundamental two dimensional feature patterns similar to computer vision [5] which we may want to model. We also desire a model able to account for small shifts and perturbations in the feature space which may be caused by different speaking styles and settings. In the low resource settings, DNNs may lead to overfitting and poor generalization. Convolutional Neural Networks (CNNs) is an alternative neural network architecture that attempts to alleviate some of these issues.

2. CONVOLUTIONAL NEURAL NETWORKS

CNNs capture structural locality by applying convolutional filters which only connect to a subset region of the previous layer (rather than being fully connected) and exploiting one or two dimensional locality [6]. The convolution filter weights are tied and shared across the feature space giving translational invariance, a shift in the feature space would result in a shift in the output space. CNNs also typically incorporate pooling which gives additional translational and rotational invariances [5]. Max pooling works by emitting only the maximum neuron in a locality region, giving the same response to small translational or rotational invariances. The translational and rotational invariance modelling capabilities provided by CNNs results in a much more generalized and robust model compared to standard DNNs. This is extremely helpful in low-resource ASR tasks, where there is limited training data available and the model is much more prone to overfitting.

CNNs have been demonstrating strong improvements over standard fully connected DNNs in ASR [7], including TIMIT [8, 9], large vocabulary continuous speech recognition (LVCSR) [10, 11] and feature extraction [12]. Convolution weights applied to acoustic modelling can be either Limited Weight Sharing (LWS) or Full Weight Sharing (FWS) [8]. The intuition is that different spectral bands may exhibit different pattern characteristics, and henceforth different spectral bands should utilize different convolutional weights. There are two main disadvantages to LWS, the first is the need to tune the size of the frequency bands (if not using a band size of 1). The second disadvantage is the inability to add additional layers of convolution on top. In theory, given enough feature maps, FWS should be able to match the performance of LWS assuming the model is able learn which convolution feature maps to use and discard in the appropriate frequency bands. LWS gave better performance over FWS in [8], however [11] showed similar results between FWS and LWS.

In computer vision square convolutional filter sizes and square pooling sizes are applicable due to the uniformity across both dimensions in natural images. We do not necessarily enjoy this uniformity in acoustic modelling since the time and spectrum pattern behaviours could be vastly different.

We perform a detailed analysis in this paper. We establish a DNN baseline in Section 3.1, determine the first convolution layer filter size in Section 3.2, max pooling size in Section 3.3, second convolution layer filter size in Section 3.4, number of feature maps in Section 3.5, dropout and additional layers in Section 3.6. We report both Frame Accuracy (FA) and Word Error Rates (WER) in our work.

3. EXPERIMENTS

We experiment with the BABEL Bengali LimitedLP (IARPA-babel103b-v0.4b) corpus which has been collected and released under the IARPA BABEL research program. Our input features are 23-dimension log-Mel filter bank feature coefficients padded with 10 left and right frame contexts (total 21 frames), and our targets are 2030 context-dependent tied acoustic states generated from a GMM-HMM system. The development set contains only 10 hours of data.

We train all our neural networks to minimize the cross entropy loss with stochastic gradient descent [4] with mini-batch size of 256. We apply no pre-training, our network weights are initialized randomly with Gaussian distribution $\mu = 0$ and $\sigma^2 = 0.001$, and biases set to zero. We start off with a constant learning rate of 0.001 until convergence on a held out 10% validation set randomly sampled on the frame level from the development set. We lower the learning rate to 0.0001 until the network converges again. Finally, we fine-tune the network at 0.00001 for one more additional epoch. We use a constant classical momentum [13] of 0.9 through the entire optimization process. We found the momentum not only speedup the training but also generally gave slightly better performance over omitting momentum. We also experimented with AdaGrad [14], we found the models converge much more quickly, however they tend to overfit and give worse results as also experimented by [15].

3.1. Deep Neural Networks

We first want to establish a baseline with fully connected DNNs. Rectified Linear Units (ReLU) DNNs have become a popular activation choice in acoustic modelling [3, 16, 17]. The ReLU non-linearity $max(0, x)$ are often significantly faster to train (e.g., faster convergence and typically require no pre-training) and give comparable performance to (pre-trained) sigmoidal nets.

Our DNNs consist of 2048 ReLUs at each hidden layer and a 2030 softmax output. We also experimented with 1024 and 3072 hidden layers, however we found those to give worse results. We vary the number of layers by adding additional hidden layers and find the WER to generally improve with each additional layer till around 14 layers. We found even with an extremely deep 19 layer model, the optimization does not get stuck and produced one of the best WER results, as also seen by [16] with deep ReLU DNNs. Despite our small dataset size, the deep ReLU DNNs with over 10 layers did not appear to overfit the problem, but rather gave additional generalized performance over the shallow networks. Table 1 gives the full results of our ReLU DNN experiments. We find there is a virtually no correlation between the FA and WER beyond a FA of 39.0. In fact, the deep (> 10 layers) ReLU DNNs gave worse FAs but better WERs over some of the more shallow networks, this suggests the deep models are underfitting the phoneme training dataset but able to produce a more generalized acoustic model. Continuing the training after convergence of the validation set closed the WER gap between the shallow and deep networks, suggesting the validation set is a poor representation of the true distribution. However, our recipe stops after the held out validation converges as described in Section 3, we did not want to use the test set as validation in our recipe.

Table 1. DNN Frame Accuracy (FA) and Word Error Rates (WER): We vary the number of fully connected ReLU layers.

Layers	FA	WER	Layers	FA	WER
2	33.0	83.8	11	39.4	72.2
3	36.6	78.8	12	39.4	71.8
4	38.5	75.8	13	39.4	71.7
5	39.1	74.4	14	39.7	70.9
6	39.8	73.3	15	39.6	71.2
7	40.1	72.4	16	39.6	71.4
8	40.1	72.4	17	39.6	71.3
9	40.2	72.2	18	39.5	71.3
10	39.4	72.6	19	39.6	70.8

3.2. First Layer Convolution Filter Size

In this set of experiments, we replace the first fully connected layer with a convolutional layer. Following the convolution layers are a sequence of fully connected layers of size 2048 ReLU hidden layers. We fix the total number of layers to be 6. The number of feature maps in the convolution layer is fixed to 64, and the activation function of the convolutional layer is ReLU. Given the results of [11], we apply FWS.

Table 2 gives the experimental results of varying the first layer convolutional filter size across both spectrum and time. The first column and first row of Table 2 is exactly the same as one dimensional convolution across time or spectrum respectively. We also experimented with two dimensional convolutional filters spanning both time and spectrum. The 6 layer fully connected ReLU DNN experiment from Section 3.1 gave a WER of 73.3, we see significant performance gains by just switching the first layer from fully connected to convolutional obtaining a 71.0 WER. The convolutional layer in the first layer of the CNN is able to generate a more robust feature representation for the model.

We find several models capable of reaching a WER between 71.0 and 71.1. Surprisingly, we did not find spectrum

Table 2. CNN Word Error Rates: We vary the size of the convolutional layer filter size across spectrum and time.

		Spectrum								
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Time	1		72.4	72.5	72.2	72.0	72.1	72.2	71.9	71.9
	2	72.2	72.2	71.8	71.4	71.8	71.7	71.6	71.6	71.6
	3	72.1	71.9	72.0	71.7	71.7	71.4	71.3	71.7	71.6
	4	72.1	71.8	71.9	71.5	71.5	71.2	71.3	71.1	71.3
	5	72.3	71.9	71.2	71.3	71.4	71.3	71.1	71.1	71.3
	6	72.2	71.6	71.3	71.1	71.2	71.1	71.3	71.1	71.4
	7	72.0	71.6	71.4	71.3	71.2	71.3	71.2	71.4	71.3
	8	71.8	71.5	71.0	71.2	71.2	71.2	71.2	71.3	71.1
	9	71.8	71.5	71.4	71.3	71.2	71.4	71.4	71.2	71.5

convolution to be much more important over time convolution, but rather we find the two dimensional filters to outperform one dimensional spectrum filters. We had expected time convolution to not be as important due to the ability of the HMM to model the temporal signal patterns. We hypothesize the two dimensional convolution to help over one dimensional spectral convolution for several reasons. First, in low resource datasets, the limited training dataset means the model is much more prone to overfitting and biasing to the common phoneme states, the convolutional filters provide a means of regularization. Secondly, we force the convolutional filters to consider surrounding frames when generating the next layer feature representation. The model is able to model local signal patterns that cross both time and spectrum dimensions. Finally, the model is much more tolerant to small perturbations in the input space in both time and frequency simultaneously.

We find the best model to use a filter size of 8×3 reaching a WER of 71.0. We also find the square convolution model of 6×6 performs very closely at 71.1 WER. In fact, several models were able to attain a WER of 71.1 (4×8, 5×7, 5×8, 6×4, 6×6, 6×8 and 8×9). The mean dimensions of the best filters are 6.0×6.6, suggesting only a small significance of spectrum convolution over time. This emphasizes the importance of considering time and spectrum simultaneously in the convolution filters. We were almost able to match the 19 layer fully connected ReLU DNN WER of 70.8 using just one third of the number of layers. For reference, the 8×3 and 6×6 filter models had a FA of 41.0 and 41.1 respectively.

3.3. Pooling

Max pooling have been shown to outperform average pooling and perform very closely to stochastic pooling in ASR [11]. In this section, we investigate the effects of max pooling. We take two of the best architectures determined in Section 3.2 and append a max pooling layer after the convolution layer and before the sequence of fully connected layers. We use overlap pooling since we found it to perform better over non-overlap pooling as also seen in computer vision [18]; albeit [11]'s experiments with LVCSR did not see an improvement with using overlap pooling in frequency (however, it did not hurt). Pooling without overlap can also be seen as subsampling the signal, which can degrade performance as observed by [11]. Overlap pooling is much more useful in preventing overfitting in datasets with limited training samples as observed by [18] in computer vision.

Table 3. CNN Word Error Rates: We fix the first convolution layer with 8×3 filter size, and vary the max pooling layer size across spectrum and time.

		Spectrum				
		1	**2**	**3**	**4**	**5**
Time	1		70.8	70.4	70.3	70.5
	2	71.0	70.6	70.0	70.0	70.2
	3	70.5	70.5	70.0	69.9	70.4
	4	70.8	70.5	70.0	69.9	70.3
	5	70.7	70.7	70.3	70.2	70.4

We vary the size of the max pooling across both spectrum and time. Table 3 gives the experimental results with a first layer convolutional filter size fixed at 8×3, and Table 4 gives the results for convolutional filter size fixed at 6×6. For reference, the best convolutional results from Section 3.2 gives the best WER of 71.0. We find using a first layer of convolution size 6×6, followed by max pooling in both time and spectrum gave the best results with a WER of 69.6. The best result pooling size was 3×4 and 4×4 and gave an equal FA of 41.6.

3.4. Second Layer Convolution Filter Size

We now investigate adding a second convolutional layer on top of the max pooling layer in Table 5. We replace the fully connected layer following the max pooling layer with a convolutional layer. We see a very small improvement increasing the number of feature maps to 128 / 128 with our WER improving to 69.2 with FA of 40.6 (filter size 3×6), and the square filter size of 5×5 to perform very closely at 69.3 WER. The experiments suggest there is much less local structure for

Table 4. CNN Word Error Rates: We fix the first convolution layer with 6×6 filter size, and vary the max pooling layer size across spectrum and time.

		Spectrum				
		1	**2**	**3**	**4**	**5**
Time	1		70.8	70.1	70.2	70.6
	2	70.8	70.5	70.1	70.0	69.8
	3	70.4	70.2	69.9	69.6	70.1
	4	70.5	70.0	70.0	69.6	70.1
	5	70.3	70.2	69.9	69.9	70.2

Table 5. CNN Word Error Rates: We fix the first convolutional layer with 6×6 filter size followed by 4×4 max pooling and vary the size of the second convolution layer filter size across spectrum and time.

		Spectrum						
		1	**2**	**3**	**4**	**5**	**6**	**7**
Time	1		69.9	69.7	69.8	69.8	69.9	69.9
	2	69.9	69.8	70.0	69.6	69.5	69.5	69.8
	3	69.8	69.9	70.1	69.3	69.6	69.2	69.2
	4	70.1	69.8	70.0	70.0	69.9	69.3	69.8
	5	70.4	70.2	69.6	69.9	69.3	69.7	69.4
	6	70.1	69.9	69.5	69.4	70.0	69.6	69.6
	7	70.5	69.8	69.4	69.8	69.8	69.3	69.9

the second convolution layer to take advantage of compared to the first convolution layer.

3.5. Feature Maps

In this set of experiments, we vary the number of feature maps in the convolutional layers. Table 6 gives the results, we find adding additional feature maps did not yield any meaningful gains in performance, with only a 0.1 absolute improvement in WER using 128 / 128 feature maps. We surmise our limited training dataset to be too small for the bigger model to learn new useful filters. When using 256 / 256 feature maps, the FA improves the most, however there is a small degradation in the WER, it is unclear why this is happening.

3.6. Dropout and Additional Hidden Layers

Dropout [19] adds robustness to our model by preventing feature co-adaptation. Dropout is helpful in the low resource condition since it is very easy to overfit the small training dataset. We want to see the effects of dropout on our CNN. We apply a 50% dropout rate on all the fully connected layers for our 6 layer architecture (with 128 / 128 feature maps). We report an improvement of FA and WER of 43.6 and 67.3 respectively.

In Section 3.1 we saw substantial gains by using many hidden layers, here we append additional fully connected layers (without dropout) to our CNN. Table 7 gives the results. We find using a deep 19 layer model gives a WER 67.7.

Table 6. CNN Word Error Rates: We fix the first convolutional layer with 6×6 filter size, followed by 4×4 max pooling, and 5×5 convolution, we then vary the number of feature maps in the convolution layers.

First Layer	**Second Layer**	**FA**	**WER**
64	64	40.7	69.3
64	128	40.0	69.3
128	64	41.1	69.6
128	128	41.0	69.2
128	256	51.1	69.5
256	128	41.1	69.6
256	256	41.3	69.5

Table 7. CNN Frame Accuracy (FA) and Word Error Rates (WER): we vary the number of fully connected hidden layers.

Layers	**FA**	**WER**	**Layers**	**FA**	**WER**
4	41.0	69.7	12	41.6	68.6
5	41.0	69.1	13	41.4	68.4
6	41.0	69.2	14	41.7	68.5
7	41.1	69.5	15	41.9	68.0
8	41.2	69.3	16	42.0	68.1
9	41.3	69.1	17	42.0	68.1
10	41.5	69.1	18	42.2	67.9
11	41.4	68.6	19	42.1	67.7

4. CONCLUSION

We have conducted a detailed empirical study of CNN acoustic models in low resource language ASR. We emphasize the importance to use two dimensional convolution filters modelling signal patterns that span across spectrum and time in the acoustic signal. Our final CNN model gives a 3.5 absolute WER improvement over our baseline DNN. We find CNNs to improve over fully connected DNNs providing more robustness and better generalization in the low resource condition.

5. ACKNOWLEDGEMENTS

This work was supported by the Intelligence Advanced Research Projects Activity (IARPA) via Department of Defense U.S. Army Research Laboratory (DoD / ARL) contract number W911NF-12-C-0015. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, DoD/ARL, or the U.S. Government.

6. REFERENCES

[1] Geoffrey Hinton, Li Deng, Dong Yu, George Dahl, Abdel rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara Sainath, and Brian Kingsbury, "Deep Neural Networks for Acoustic Modeling in Speech Recognition," *IEEE Signal Processing Magazine*, November 2012.

[2] George Dahl, Dong Yu, Li Deng, and Alex Acero, "Context-Dependent Pre-Trained Deep Neural Networks for Large-Vocabulary Speech Recognition," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, pp. 30–42, January 2012.

[3] George Dahl, Tara Sainath, and Geoffrey Hinton, "Improving Deep Neural Networks for LVCSR Using Rectified Linear Units and Dropout," in *IEEE International Conference on Acoustic Speech and Signal Processing*, 2013.

[4] Yann LeCun, Leon Bottou, Genevieve Beth Orr, and Klaus Robert Mller, "Efficient BackProp," *Neural Networks: Tricks of the Trade*, pp. 5–50, 1998.

[5] Kevin Jarrett, Koray Kavukcuoglu, MarcAurelio Ranzato, and Yann LeCun, "What is the Best Multi-Stage Architecture for Object Recognition," in *International Conference on Computer Vision*, 2009.

[6] Yann LeCun, Leon Bottou, Yoshua Bengio, and Patrick Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, pp. 2278–2324, November 1998.

[7] Ossama Abdel-Hamid, Abdel rahman Mohamed, Hui Jiang, and Gerald Penn, "Applying convolutional neural networks concepts to hybrid NN-HMM model for speech recognition," in *IEEE International Conference on Acoustic Speech and Signal Processing*, 2012.

[8] Ossama Abdel-Hamid, Li Deng, and Dong Yu, "Exploring Convolutional Neural Network Structures and Optimization Techniques for Speech Recognition," in *INTERSPEECH*, 2013.

[9] Laszlo Toth, "Convolutional Deep Maxout Networks for Phone Recognition," in *INTERSPEECH*, 2014.

[10] Tara Sainath, Abdel rahman Mohamed, Brian Kingsbury, and Bhuvana Ramabhadran, "Deep Convolutional Neural Networks for LVCSR," in *IEEE International Conference on Acoustic Speech and Signal Processing*, 2013.

[11] Tara Sainath, Brian Kingsbury, Abdel rahman Mohamed, George Dahl, George Saon, Hagen Soltau, Tomas Beran, Aleksandr Aravkin, and Bhuvana Ramabhadran, "Improvements to Deep Convolutional Neural Networks for LVCSR," in *Automatic Speech Recognition and Understanding Workshop*, 2013.

[12] Yajie Miao and Florian Metze, "Improving Language-Universal Feature Extraction with Deep Maxout and Convolutional Neural Networks," in *INTERSPEECH*, 2014.

[13] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton, "On the Importance of Initialization and Momentum in Deep Learning," in *International Conference on Machine Learning*, 2013.

[14] John Duchi, Elad Hazan, and Yoram Singer, "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization," *Journal of Machine Learning Research*, vol. 12, pp. 2121–2159, July 2011.

[15] Andrew Senior, Georg Heigold, MarcAurelio Ranzato, and Ke Yang, "An Empirical study of learning rates in deep neural networks for speech recognition," in *IEEE International Conference on Acoustic Speech and Signal Processing*, 2013.

[16] Matthew Zeiler, Marc'Aurelio Ranzato, Rajat Monga, Mark Mao, Ke Yang, Quoc Viet Le, Patrick Nguyen, Andrew Senior, Vincent Vanhoucke, Jeffrey Dean, and Geoffrey Hinton, "On Rectified Linear Units for Speech Processing," in *IEEE International Conference on Acoustic Speech and Signal Processing*, 2013.

[17] Andrew Maas, Awni Hannun, and Andrew Ng, "Rectifier Nonlinearities Improve Neural Network Acoustic Models," in *International Conference on Machine Learning: Workshop on Deep Learning for Audio, Speech, and Language Processing*, 2013.

[18] Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Neural Information Processing Systems*, 2012.

[19] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, June 2014.