Exploring Convolutional Neural Network Structures and Optimization Techniques for Speech Recognition

Ossama Abdel-Hamid[1], *Li Deng*[2], *Dong Yu*[2]

[1]Department of Computer Science and Engineering, York University, Toronto, Ontario, Canada
[2]Microsoft Research, Redmond, WA USA

ossama@cse.yorku.ca, {deng, dongyu}@microsoft.com

Abstract

Recently, convolutional neural networks (CNNs) have been shown to outperform the standard fully connected deep neural networks within the hybrid deep neural network / hidden Markov model (DNN/HMM) framework on the phone recognition task. In this paper, we extend the earlier basic form of the CNN and explore it in multiple ways. We first investigate several CNN architectures, including full and limited weight sharing, convolution along frequency and time axes, and stacking of several convolution layers. We then develop a novel weighted softmax pooling layer so that the size in the pooling layer can be automatically learned. Further, we evaluate the effect of CNN pretraining, which is achieved by using a convolutional version of the RBM. We show that all CNN architectures we have investigated outperform the earlier basic form of the DNN on both the phone recognition and large vocabulary speech recognition tasks. The architecture with limited weight sharing provides additional gains over the full weight sharing architecture. The softmax pooling layer performs as well as the best CNN with the manually tuned fixed-pooling size, and has a potential for further improvement. Finally, we show that CNN pretraining produces significantly better results on a large vocabulary speech recognition task.

Index Terms: Convolutional Neural Network, Hybrid Neural Network / Hidden Markov Models, Pretraining, Convolutional Restricted Boltzmann Machine

1. Introduction

Recently, deep neural network hidden Markov model (DNN/HMM) hybrid systems achieved remarkable performance in many large vocabulary speech recognition tasks [1, 2, 3, 4, 5, 6, 7]. This is attributed to the improved modeling power of the DNN that enables it to map complex patterns into class labels or posterior probabilities. This modeling power stems from the deep-layered structure and the distributed representation. Moreover, unsupervised pretraining of the DNN helps in achieving better performance for some tasks. This unsupervised pretraining is typically done by stacking single layered generative models called Restricted Boltzmann Machine (RBM).

More recently, Abdel-Hamid et al. [8] showed that a special neural network structure called the convolutional neural network (CNN) can further improve the hybrid model performance on the TIMIT phone recognition task. The CNN exploits domain knowledge about feature invariances within its structure and has been successfully applied to various image analysis and recognition tasks [9, 10]. For speech processing, CNN was theoretically proposed (with no experimental verification) in [11] but with convolution along the time axis to obtain features robust to small temporal shifts. In [12], a generative model named convolutional RBM (CRBM) was used to learn speech features without supervision but with one dimensional convolution along time. In [8], it has been shown that much gain can be obtained by applying convolution and max-pooling along frequency. Applying convolution along frequency results in features invariant to small frequency shifts of speech features, which often happen between different speakers and for the same speaker in different moods. Moreover, in [13], a heterogeneous pooling structure proved to be beneficial.

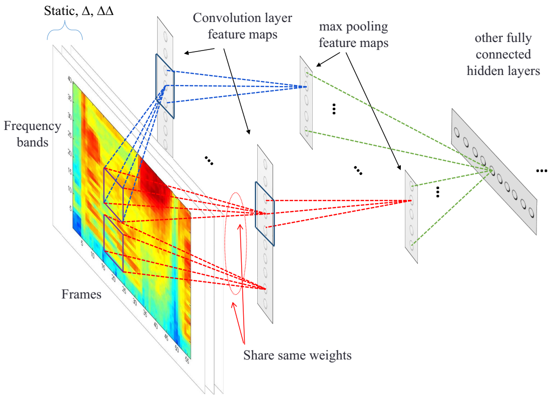

Figure 1: *A convolutional neural network with full weight sharing applied along the frequency axis.*

In this paper, we further explore CNNs in multiple dimensions. We first investigate different architectures of the CNN, including full and limited sharing of convolution filters, convolution along frequency and time axes, and the stacking of multiple convolution layers. We then propose a novel weighted softmax pooling layer to enable automatic learning of the pooling size. Lastly we evaluate the effect of CNN pretraining, which is achieved by stacking trained CRBMs and RBMs [14].

The rest of the paper is organized as follows. In section 2 we introduce the CNN and its various architectures. In section 3 we describe the weighted softmax pooling. The CNN pretraining is discussed in section 4. We report the experimental results in section 5, and conclude the paper in section 6.

2. The convolutional neural nework

2.1. Basic structure

The CNN is a neural network with a special structure. Figure 1 illustrates an example CNN with full weight sharing. In this CNN the first layer, which consists of a number of feature maps, is called a convolution layer. Each neuron in the convolution layer receives input from a local receptive field representing features of a limited frequency range. Neurons that belong to the same feature map share the same weights (also called filters or kernels) but receive different inputs shifted in frequency. As a result, the convolution layer does a convolution of the kernels with the lower layer activations.

Suppose the NN input is $\mathbf{V} \in \mathbf{R}^{A \times B}$, where A is the number of features representing an input frequency band and B is the number of the input frequency bands. In the case of filter bank features, B represents the size of the filter bank feature vector. Let's assume that $\mathbf{v} = [\mathbf{v}_1 \ \mathbf{v}_2 \ \ldots \ \mathbf{v}_B]$, where $\mathbf{v}_b$ is the feature vector representing band b. The activations of the convolution layer can be computed as:

$$h_{j,k} = \theta\left(\sum_{b=1}^{s} \mathbf{w}_{b,j}^{T} \mathbf{v}_{b+k-1} + a_j\right), \qquad (1)$$

where $h_{j,k}$ is the convolution layer's output of the jth feature map on the kth convolution layer band, s is the filter size, $\mathbf{w}_{b,j}$ is a weight vector representing the bth band of the jth filter, a_j is the bias of the jth feature map, and $\theta(x)$ is the activation function, which is set to the sigmoid function in this work.

A pooling layer is added on top of the convolution layer to compute a lower resolution representation of the convolution layer activations through sub-sampling. The pooling function, which computes some statistics of the activations, is typically applied to the neurons along a window of frequency bands and generated from the same feature map in the convolution layer. In [8] a max pooling function, which simply computes the maximum value of the feature over the corresponding frequency bands, was used. The max-pooling activations can be computed as:

$$p_{j,m} = \max_{k=1}^{r}(h_{j,(m-1)\times n+k}), \qquad (2)$$

where $p_{j,m}$ is the pooling layer's output of the jth feature map and mth pooling layer band, n is the sub-sampling factor, and r is the pooling size, which is the number of bands to be pooled together.

Both weight sharing and max pooling play a vital role in achieving invariance to small frequency shifts. This is a desired property because, for example, the formants of the same phoneme may appear on slightly different frequencies for different speakers or even for the same speaker in different states. Moreover, weight sharing helps in reducing over-fitting due to the reduced number of trainable parameters.

The convolution-pooling pairs can be stacked up to obtain higher level features, on top of which the standard fully connected layers can be added to combine the features of different bands.

2.2. Full and limited weight sharing

We call the weight sharing scheme in Figure 1 the *full weight sharing*. This is the standard scheme used for image processing. It is suitable for image processing because the same image pattern can appear at any position in an image. In speech, however, different patterns appear in different frequency bands. For this reason, a *limited weight sharing* scheme as shown in Figure 2 is more suitable. The difference here is that a different (not shared) kernel is used for the different frequency window in the convolution layer. Each neuron in the pooling layer summarizes convolution layer's activations generated from one particular feature map defined by the kernels. It is as if the convolution and pooling layers are divided into many sections, each of which processes only a limited range of the input bands and generates only one output band in the pooling layer. In mathematical terms, the convolution and pooling layer activations can be computed as:

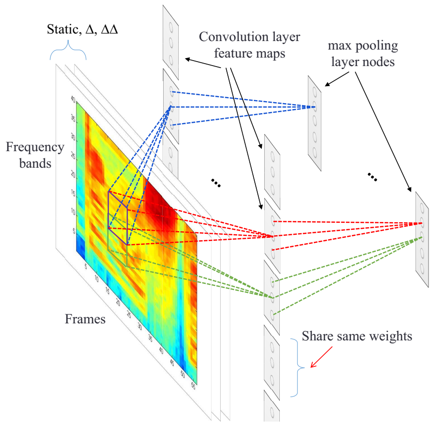

Figure 2: *A convolution neural network with* limited *weight sharing applied along the frequency axis.*

$$h_{j,k}^{(m)} = \theta\left(\sum_{b=1}^{s} \mathbf{w}_{b,j}^{(m)} \mathbf{v}_{(m-1)\times n+b+k-1}^{T} + a_j^{(m)}\right), \qquad (3)$$

and

$$p_{j,m} = \max_{k=1}^{r}(h_{j,k}^{(m)}), \qquad (4)$$

where $h_{j,k}^{(m)}$ is the value of the kth band of the jth feature map at the mth convolution layer section, and $p_{j,m}$ is the value of the jth feature of the mth pooling layer band.

This scheme also helps reduce the number of neurons in the pooling layer. Because each band uses kernels that consider only the patterns appearing in the corresponding frequency range, the number of kernels per band is reduced and the resulted representation has better ability to distinguish patterns in different frequency bands. On the other hand, this limited weight sharing scheme has the disadvantage of preventing the addition of more convolution layers on top. This is because the features in different pooling layer bands in this scheme are unrelated and cannot be convolved. An alternative solution, which may lead to better performance, is to apply the limited weight sharing convolution layer on top of a full weight sharing one.

2.3. Convolution along the time axis

The CNN architecture can also be applied along the time axis to reduce temporal variability. In the hybrid NN/HMM framework, the HMM is supposed to handle the speech temporal variability. However, since the scores of each frame are computed from a window of consecutive frames, invariance to small shifts within this context window can be desirable. Note that the pooling and sub-sampling doesn't affect the time resolution seen by the HMM. It only affects the time resolution received by higher layers of the CNN.

Combining convolution along both frequency and time axes generates a 2D CNN similar to the ones used for image analysis, and allows for further performance improvement. Different from that in the image analysis, though, the convolution and pooling parameters along the frequency and time dimensions are independent in the speech analysis case. This increases the number of model parameters to optimize and requires careful handling of these parameters.

3. Weighted softmax pooling

At the same time when max-pooling introduces robustness to small frequency shifts, it reduces discrimination ability to some features in some bands, esp. when a fixed pooling size is used as in the CNN presented in the previous section. To alleviate this problem, we propose adding learnable weights to the pooling layer. More specifically, instead of using a max-pooling function, we use a softmax pooling function which is differentiable. The softmax pooling replaces Eq. 4 by:

$$p_{j,m} = \sum_{k=1}^{r} \frac{exp(u_{j,m,k})exp(\epsilon h_{j,k}^{(m)})}{\sum_{i=1}^{r} exp(u_{j,m,i})exp(\epsilon h_{j,i}^{(m)})} h_{j,k}^{(m)}, \qquad (5)$$

where $u_{j,m,i}$ represents the logarithm of the pooling weights, and ϵ is a softmax function parameter that controls its smoothness.

The pooling weights can be learned using the backpropagation algorithm by computing the derivative of Eq. 5. By modifying the pooling weights, a similar effect to modifying pooling size can be obtained.

4. Pretraining

RBM-based pretraining improves DNN performance when the training set is small. For convolutional structures, a convolutional RBM (CRBM) has been proposed in [14]. Similar to the RBM, the training of the CRBM maximizes the training data likelihood using the approximate *contrastive divergence* algorithm. In the CRBM, the convolutional layer activations are stochastic. The CRBM model defines the probability of a certain configuration of input and hidden units and the conditional probability of hidden unit activations given the visible units and vice versa. The CRBM defines a multinomial distribution over the pooled hidden nodes and at most one node in a pooled set can be active. The conditional probability of activation of a hidden node $h_{j,k}^{(m)}$ given the CRBM input is defined by the following softmax function:

$$P(h_{j,k}^{(m)} = 1|\mathbf{V}) = \frac{exp(I(h_{j,k}^{(m)})}{\sum_{k=1}^{r} exp(I(h_{j,k}^{(m)})}, \qquad (6)$$

where $I(h_{j,k}^{(m)})$ is the summation of weighted signal reaching node $h_{j,k}^{(m)}$ from the input layer and it is defined as

$$I(h_{j,k}^{(m)}) = \sum_{b=1}^{s} \mathbf{w}_{b,j}^{(m)} \mathbf{v}_{(m-1)\times n+b+k-1}^{T} + a_j^{(m)}. \qquad (7)$$

The conditional probability distribution of $v_{i,b}$, the visible units of the bth band of the ith feature map, given the hidden unit states can be computed by the following Gaussian distribution:

$$P(v_{i,b}|\mathbf{H}) = \mathcal{N}(v_{i,b}; \sum_{j,(k,m)\in\mathbf{C}(i,b)} h_{j,k}^{(m)} w_{i,f(b,k,m),j}^{m}, \sigma^2) \qquad (8)$$

where the mean of this Gaussian is the summation of the weighted signal arriving from the hidden units that are connected to the visible units. $\mathbf{C}(i,b)$ represents this connection as the set of indices of convolution bands and sections that receive input from the visible unit $v_{i,b}$. $w_{i,f(b,k,m),j}^{m}$ is the weight on the link from the bth band of the ith input feature map to the kth band and the jth feature map of the mth convolution section. σ^2 is the variance of the Gaussian distribution and it is a fixed model parameter.

The weights of a trained CRBM are good initial values for the convolution layer. After the first convolution layer weights are learned, they are used to compute the convolution and pooling layer outputs using equations 3 and 4. The outputs of the pooling layer are used as inputs to pretrain the next layer as done in deep belief network training [15].

5. Experimental results

5.1. Experimental setup

Experiments were done on two datasets: TIMIT and Microsoft-internal voice search (VS). The feature extraction is similar. Speech was analyzed using a 25-ms Hamming window with a 10-ms fixed frame rate. The speech feature vector was generated by a Fourier-transform-based filter-bank, which includes 40 coefficients distributed on a Mel scale, along with their first and second temporal derivatives. Energy was added to TIMIT feature vector only when limited weight sharing is used, while VS has no energy component. All speech data were normalized so that each coefficient has zero mean and unit variance. The original mean and variance were computed once over the whole training dataset and were used for normalizing test data as well. In all experiments a NN with three hidden layers is used. Each layer is either a fully connected layer with 1000 nodes or a convolution and pooling pair.

5.1.1. The TIMIT phone recognition task

In TIMIT, we used the 462-speaker training set and removed all SA sentences (they are two sentences read by all speakers). A separate development set of 50 speakers was used for tuning the models parameters and controlling the NN training progress (the learning rate and the number of iterations). Results are reported using the 24-speaker core test set, which has no overlap with the development set.

We used 183 target class labels (i.e., 3 states for each one of the 61 phones). After decoding, the 61 phone classes were mapped to a set of 39 classes as in [16] for scoring. In our experiments, a bi-gram language model over phones, estimated from the training set, was used in decoding. To prepare the NN targets, a mono-phone HMM was trained on the training dataset, and it was used to generate state-level forced alignments. For neural network training, a learning rate annealing and an early stopping strategies were utilized as in [15].

5.1.2. The voice search task

VS is a large vocabulary voice search dataset containing 18 hours of speech data. Initially, a tri-phone HMM was built with state tying. The state labels were used as the NN targets. NN training was done using a fixed recipe. The first 15 epochs were run with a learning rate of 0.08. Another 10 epochs were run with a learning rate of 0.002.

5.2. Results on CNN structure

In this set of experiments we compare different convolution structures on TIMIT. Table 1 shows the results. With a full weight sharing CNN we got relative reduction in phone error rate (PER) of more than 5% compared to the DNN without convolution. With limited weight sharing the relative reduction

Table 1: *Comparisons of TIMIT phone recognition accuracy among different CNN architectures. LWS: limited weight sharing; FWS: full weight sharing; K: # of feature maps; PS: pooling size; FS: filter size; B: # of bands.*

Convolution architecture	PER
No convolution	22.9 %
Freq FWS (K:200, PS:6, FS:8, B:20)	21.6%
Freq LWS (K:84, PS:6, FS:8, B:20)	20.5%
Time FWS (K:400, PS:2, FS:8, B:7)	22.5%
2D Multi-layers (K:40, PS:2,2, FS:3,3, B:20,7), (K:200, PS:3,1, FS:5,7, B:18,1)	21.5%

exceeded 10%. Convolution through time improved the score as well, though, the improvement was much smaller. This indicates that convolution through frequency is more important. There are a number of possibilities for the limited improvement with convolution through time. Firstly, the pooling and sampling may affect label localization through time because it decreases time resolution for higher CNN layers. Thus it would be more difficult for higher layers to attribute the label to the center frame. Secondly, the number of frames within the time window is only 15. This may decrease the benefit of convolution along time as compared with the 40 frequency bands. The last row shows the result of an attempt to use a 2D convolution with two convolution layers. Full weight sharing was used so that more than one convolution layers can be stacked. A small improvement was obtained over one layer of full weight sharing, but we think that with more optimization of the parameters we can get a better performance.

5.3. Results on weighted softmax pooling

In this experiment, we tested the performance of weighted softmax pooling. An architecture similar to the limited weight sharing CNN used in the previous sub-section was used here. The pooling weights were initialized with one. Then, the weights were updated using the back-propagation algorithm. In one experiments weights were not tied and we got 20.8% PER. While in another experiment, weights were tied within three groups where each group has 28 feature maps sharing the same pooling weights. In this case we got 20.4% PER, which is slightly better than that achieved with max-pooling.

Figure 3 shows the learned weights. We notice that some bands receive wider inputs than others. This indicates that different bands and feature maps are more suited to different pooling sizes.

5.4. Results on Pretraining

To test the effects of pretraining a CNN, we conducted a set of experiments on TIMIT and VS datasets. The pretraining implementation was based on the technique of [15] with detail described in Section 4, where CRBMs are used with convolution layers and RBMs are used with fully connected layers.

Table 2 shows the results on the tasks of TIMIT (in PER) and VS (in word error rate (WER)). We observe that the pretraining improves both the DNN and CNN except for the CNN on TIMIT where pretraining didn't help. In general, the relative improvement of using pretraining for the CNN is less than that on the DNN. This may be attributed to the better structure of the CNN that reduces the benefit of pretraining.

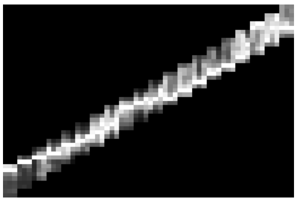

Figure 3: *Learned softmax pooling weights. Each column represents a set of pooling weights. Each pooling band has three sets of weights. The vertical axis represents the input bands.*

Table 2: *Effects of CRBM pretraining (PT)*

Score Function	TIMIT PER	VS WER
DNN, no PT	22.8%	37.1%
DNN, PT	21.9%	35.4%
CNN no PT	20.5 %	34.2%
CNN PT	20.7 %	33.4%

6. Summary and conclusion

In this work we explored multiple aspects of CNNs. Our experimental results confirmed the improved performance of the CNN, previously obtained on the TIMIT phone recognition task, over a DNN, and showed that similar performance improvement can be obtained on large vocabulary tasks as well. The success of the CNN can be attributed to the learned features that are invariant to small frequency shifts of speech patterns (e.g. formants) which increases the robustness to speaker variations. Results also demonstrated that both full and limited weight sharing architectures perform better than the DNN without convolution. However, the limited weight sharing scheme provides more gains than the full weight sharing scheme. This is because distinct feature patterns are expected to appear in each frequency band. However, full weight sharing preserves consistent features along all frequency bands, which can be further convoluted and pooled in higher convolution layers.

Our results indicate that doing convolution along time, while outperforming the DNN, performs significantly worse than doing it along frequency. We conjecture that the performance gap is caused by the implicit label shift and the smaller range of the time dimension. The experimental results suggest that combining convolution along both frequency and time axes in a way that takes into account speech trajectory confusion may bring more performance gain.

In this paper, we also proposed weighted softmax pooling to enable automatic determination of the pooling size. Experimental results showed that different feature-map groups organized in different bands can effectively learn different pooling weight structures, some of which are wider than others. This observation indicates that separate pooling sizes and structures are desired for different features and bands. The recognition performance reported in this work is only slightly better than that achieved using the limited weight sharing CNN. However, we believe this is an interesting direction and better learning algorithms may further improve the performance.

Finally, we proposed to use CRBM to pretrain the convolution layers in the CNN. We evaluated the CNNs with and without pretraining and found out that pretraining improved performance on a large vocabulary speech recognition task.

7. References

[1] G. Dahl, D. Yu, L. Deng, and A. Acero, "Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition," *IEEE Trans. Audio, Speech, and Language Proc.*, vol. 20, no. 1, pp. 30 –42, jan. 2012.

[2] F. Seide, G. Li, and D. Yu, "Conversational speech transcription using context-dependent deep neural networks," in *Proc. Interspeech*, 2011.

[3] T. N. Sainath, B. Kingsbury, B. Ramabhadran, P. Fousek, P. Novak, and A. Mohamed, "Making deep belief networks effective for large vocabulary continuous speech recognition," in *ASRU*, 2011.

[4] G. Hinton, L. Deng, D. Yu, G. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82 –97, nov. 2012.

[5] L. Deng, J. Li, J.-T. Huang, K. Yao, D. Yu, F. Seide, M. Seltzer, G. Zweig, X. He, J. Williams, Y. Gong, , and A. Acero, "Recent advances in deep learning for speech research at Microsoft," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, May 2013.

[6] S. M. Siniscalchi, D. Yu, L. Deng, and C. hui Lee, "Speech recognition using long-span temporal patterns in a deep network model," *IEEE Signal Processing Letters*, March 2013.

[7] D. Yu, L. Deng, and F. Seide, "The deep tensor neural network with applications to large vocabulary speech recognition," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 21, no. 2, pp. 388–396, 2013.

[8] O. Abdel-Hamid, A. Mohamed, H. Jiang, and G. Penn, "Applying convolutional neural networks concepts to hybrid NN-HMM model for speech recognition," in *Proc. ICASSP*, march 2012, pp. 4277 –4280.

[9] Y. LeCun, F. Huang, and L. Bottou, "Learning methods for generic object recognition with invariance to pose and lighting," in *Proceedings of CVPR'04*. IEEE Press, 2004.

[10] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," in *Proceedings of the IEEE*, 1998, pp. 2278–2324.

[11] Y. LeCun and Y. Bengio, "Convolutional networks for images, speech, and time-series," in *The Handbook of Brain Theory and Neural Networks*, M. A. Arbib, Ed. MIT Press, 1995.

[12] H. Lee, P. Pham, Y. Largman, and A. Ng, "Unsupervised feature learning for audio classification using convolutional deep belief networks," in *Advances in Neural Information Processing Systems 22*, 2009, pp. 1096–1104.

[13] L. Deng, O. Abdel-Hamid, and D. Yu, "A deep convolutional neural network using heterogeneous pooling for trading acoustic invariance with phonetic confusion," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, May 2013.

[14] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng, "Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations," in *Proceedings of the 26th Annual International Conference on Machine Learning*, ser. ICML '09. New York, NY, USA: ACM, 2009, pp. 609–616.

[15] A. Mohamed, G. Dahl, and G. Hinton, "Acoustic modeling using deep belief networks," *IEEE Trans. Audio, Speech, and Language Processing,*, vol. 20, no. 1, pp. 14 –22, jan. 2012.

[16] K. F. Lee and H. W. Hon, "Speaker-independent phone recognition using hidden markov models," *IEEE Transactions on Audio, Speech and Language Proc.*, vol. 37, no. 11, pp. 1641 – 1648, November 1989.