APPLYING CONVOLUTIONAL NEURAL NETWORKS CONCEPTS TO HYBRID NN-HMM MODEL FOR SPEECH RECOGNITION

Ossama Abdel-Hamid[†] *Abdel-rahman Mohamed*[‡] *Hui Jiang*[†] *Gerald Penn*[‡]

[†] Department of Computer Science and Engineering, York University, Toronto, Canada
[‡] Department of Computer Science, University of Toronto, Toronto, Canada
{ossama@cse.yorku.ca asamir@cs.utoronto.ca hj@cse.yorku.ca gpenn@cs.utoronto.ca }

ABSTRACT

Convolutional Neural Networks (CNN) have showed success in achieving translation invariance for many image processing tasks. The success is largely attributed to the use of local filtering and max-pooling in the CNN architecture. In this paper, we propose to apply CNN to speech recognition within the framework of hybrid NN-HMM model. We propose to use local filtering and max-pooling in frequency domain to normalize speaker variance to achieve higher multi-speaker speech recognition performance. In our method, a pair of local filtering layer and max-pooling layer is added at the lowest end of neural network (NN) to normalize spectral variations of speech signals. In our experiments, the proposed CNN architecture is evaluated in a speaker independent speech recognition task using the standard TIMIT data sets. Experimental results show that the proposed CNN method can achieve over 10% relative error reduction in the core TIMIT test sets when comparing with a regular NN using the same number of hidden layers and weights. Our results also show that the best result of the proposed CNN model is better than previously published results on the same TIMIT test sets that use a pre-trained deep NN model.

Index Terms— acoustic modeling, neural networks, speech recognition, local filtering, max-pooling

1. INTRODUCTION

The long term goal for automatic speech recognition (ASR) is to effectively deal with a wide range of speaking, channel, and environmental conditions that human can handle quite well. These types of acoustic variations have been found to be quite challenging to the state-of-the-art ASR systems that use Hidden Markov Models (HMMs) to model the sequential structure of speech signals, where each HMM state uses a Gaussian Mixture model (GMM) to model short-time spectral representation of speech signal. Better acoustic models should be able to model a variety of acoustic variations in speech signals more effectively to achieve robustness against various speaking and environmental conditions. More recently, deep neural networks have been proposed to replace Gaussian Mixture model (GMM) as the basic acoustic models for HMM-based speech recognition systems [1]. And it has been demonstrated that NN-based acoustic models can achieve very competitive recognition performance in some very difficult LVCSR tasks [2, 3]. A key advantage of neural networks is its "distributed representations" of input features (i.e., many neurons are active simultaneously to represent input features) that makes them more efficient than GMMs. This property allows NNs to model a diversity of speaking styles and background conditions with much less training data because NN can share similar portions of the input space to train some hidden units but keep other units sensitive to a subset of the input features that are significant to recognition.

In this paper we propose to use some concepts developed in Convolutional Neural Networks (CNN) literature [4] to explicitly normalize speech spectral features to achieve speaker invariance and enforce locality of features. CNN has been proven to be successful in many vision tasks [5] and it has been proposed for time series data as in [6, 7], where convolution-based filtering is performed along the time axis. The novelty of this paper is to apply the CNN concepts in the frequency domain to exploit CNN invariance to small shifts along the frequency axis through the use of local filtering and max-pooling. In this way, some acoustic variations can be effectively normalized and the resultant feature representation may be immune to speaker variations, colored background and channel noises. Our experimental results on TIMIT have shown that the proposed CNN method can achieve over 10% relative error reduction over regular NNs using the same number of hidden layers and comparable number of trainable weights under the same hybrid NN-HMM framework.

2. CONVOLUTIONAL NEURAL NETWORK

CNN consists of one or more pairs of convolution and max pooling layers [4]. A convolution layer applies a set of filters that process small local parts of the input where these filters are replicated along the whole input space. A max-pooling layer generates a lower resolution version of the convolution layer activations by taking the maximum filter activation from different positions within a specified window. This adds translation invariance and tolerance to minor differences of positions of objects parts. Higher layers use more broad filters that work on lower resolution inputs to process more complex parts of the input. Top fully connected layers finally combine inputs from all positions to do the classification of the overall inputs. This hierarchical organization generates good results in image processing tasks [5].

CNN introduces three extra concepts over the simple fully connected feed-forward NN: local filters, max-pooling, and weight sharing. In next section, we will discuss these concepts in details and consider to apply them to speech signals.

3. APPLYING CNN CONCEPTS TO SPEECH

In this section, we are concerned with applying CNN principles along the frequency axis of speech signals. The variability along the time axis is handled by the HMM model and the dependency between adjacent speech frames is dealt with a long time context window feeding as an input to the NN as in standard hybrid NN-HMM models. Each input of NN is a stack of consecutive feature frames O_t centering at time t, while the target output is the probability $P(s|O_t)$ of the centered frame belonging to each HMM state s at time t.

3.1. Locality

Speech signals enjoy some locality characteristics along the frequency axis, which means that different phonemes many have energy concentrations in different local bands along the frequency axis. These local energy concentrations become the critical clues to distinguish different phonemes. For example, voiced phonemes have a number of formants appearing at different frequencies. As a result, filters that work on local frequency region will provide an efficient way to represent these local structures and their combinations along the whole frequency axis may be eventually used to recognize each phone. This strategy is better than a traditional acoustic model that represents the entire frequency spectrum as a whole, such as GMM. Another benefit of local filters is the potential to achieve better robustness against ambient noises especially when noises are only concentrated in parts of spectrum, where local filters in relatively cleaner parts of spectrum can still detect speech features well to compensate ambiguity of noisy parts.

CNN is capable of modeling these local frequency structures by allowing each node of the so-called convolutional layer to receive input only from features representing a limited bandwidth (the receptive field of the node) of the whole speech spectrum. In order to locally process spectrum, we need to represent speech inputs in a frequency scale that can be divided into a number of local bands. Therefore, the standard MFCC features are not suitable here because of DCT-based decorrelation transform. However, linear spectrum, Mel-scale spectrum, or filter-bank features are all perfect for local filtering in our CNN configuration.

Let's assume speech input to CNN is $\mathbf{v}$ that is divided into B frequency bands as: $\mathbf{v} = [\mathbf{v}_1\ \mathbf{v}_2\ ...\ \mathbf{v}_B]$, where $\mathbf{v}_b$ is the feature vector representing band b. As shown in figure 1, this feature vector $\mathbf{v}_b$ includes speech spectral features, delta and acceleration parameters from local band b of all feature frames within the current context window. Activations of the convolution layer are divided into K bands where each band contains J filters activations. Let's assume each band activation is denoted as $\mathbf{h}_k = [h_{k,1}\ h_{k,2}\ ...\ h_{k,J}]$. The convolution layer activations can be computed as a convolution-like operation of each filter on the lower layer bands followed by a non-linear activation function:

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

where $\theta(x)$ is the activation function, s is the filter size in the number of input bands, $\mathbf{w}_{b,j}$ is the weight vector representing the bth component of the jth filter. Note that we have ignored flipping the filter index in above equation to make notation simple. This convolutional layer can be viewed as a standard Neural Network layer, where all J nodes of the hidden layer are grouped into K bands and each node receives inputs only from s bands of the lower layer. Moreover, weights and biases going into the jth node of each band are shared among different hidden layer bands as shown in figure 1. Note that this weight sharing scheme is popular in image processing but it will be modified in section 3.3 to better fit speech signals.

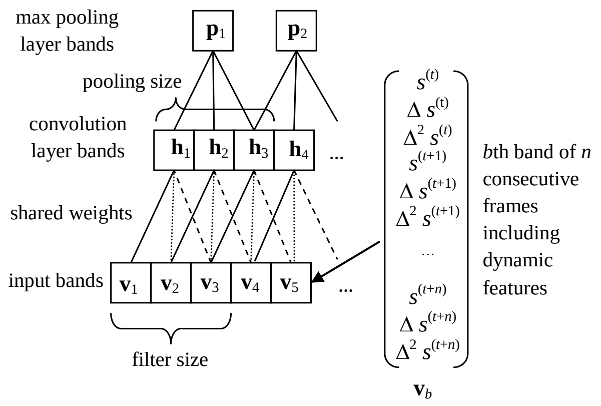
max pooling layer bands
pooling size
convolution layer bands
shared weights
input bands
filter size
p1 p2
h1 h2 h3 h4 ...
v1 v2 v3 v4 v5 ...
s(t) Δ s(t) Δ2 s(t) s(t+1) Δ s(t+1) Δ2 s(t+1) ... s(t+n) Δ s(t+n) Δ2 s(t+n)
vb
bth band of n consecutive frames including dynamic features

Fig. 1. Diagram to shown a pair of CNN convolution layer and max-pooling layers, where weights represented by the same line style are shared among all convolution layer bands.

3.2. Max Pooling

As discussed above, speech spectrum includes many local structures and these local structures are distributed over a range of frequency axis, where each local structure typically appears to center around one particular frequency that can vary within a limited range. For example, central frequencies of formants for the same phoneme may vary within a limited range and they normally differ between different speakers and sometimes even between different utterances from the same speaker. Some models try to remove the variability between speakers by transforming the speech features of each speaker into a canonical speaker space [8, 9].

In CNN, this variability problem can be easily solved through the use of max-pooling, where a max-pooling layer is added on top of each convolution layer. The max-pooling layer activations are divided into M bands. Each band of the max-pooling layer receives input from r convolution layer neighbouring bands to generate J values representing the maximum activations received from the J convolution filters within these r bands as shown in figure 1. The max-pooling layer usually generates a lower resolution version of the convolution layer by doing this maximization operation every n bands, where n is the sub-sampling factor. As a result a smaller number of bands are obtained that provide a lower *frequency* resolution features that contain more useful information that can be further processed by higher layers in the NN hierarchy.

The activations of the mth band of the max-pooling layer are denoted as $\mathbf{p}_m = [p_{m,1}\ p_{m,2}\ ...\ p_{m,J}]^T$. Each activation is computed as:

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

where r is called pooling size. n could be smaller than r to have some overlap between adjacent pooling bands. Figure 1 shows an example of pooling layer which has a sub-sampling factor of 2 and pooling size 3.

3.3. Weight Sharing

Weight sharing is a critical principle in CNN since it helps to reduce the total number of trainable parameters. Furthermore, weight sharing may lead to more efficient training and more effective model especially when some similar local structures may appear in may places in the input space. In a standard CNN, the local filter weights

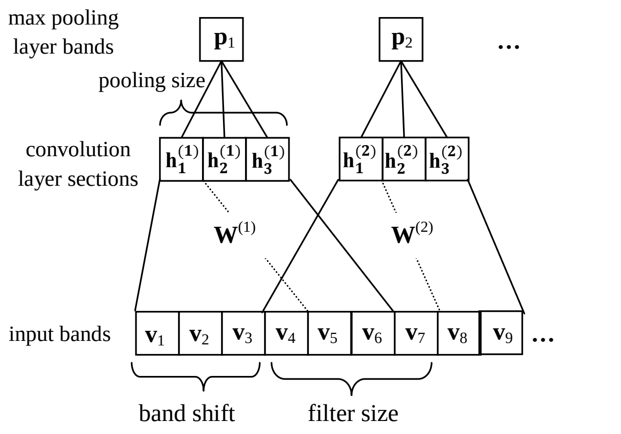
max pooling layer bands
$\mathbf{p}_1$
$\mathbf{p}_2$
...
pooling size
convolution layer sections
$\mathbf{h}_1^{(1)}$ $\mathbf{h}_2^{(1)}$ $\mathbf{h}_3^{(1)}$
$\mathbf{h}_1^{(2)}$ $\mathbf{h}_2^{(2)}$ $\mathbf{h}_3^{(2)}$
$\mathbf{W}^{(1)}$
$\mathbf{W}^{(2)}$
input bands
$\mathbf{v}_1$ $\mathbf{v}_2$ $\mathbf{v}_3$ $\mathbf{v}_4$ $\mathbf{v}_5$ $\mathbf{v}_6$ $\mathbf{v}_7$ $\mathbf{v}_8$ $\mathbf{v}_9$ ...
band shift
filter size

Fig. 2. CNN with limited weight sharing. The figure shows weights sharing within convolution layer sections. For example, in the figure $\mathbf{W}^{(1)}$ represents the weights matrix shared between bands $\mathbf{h}_1^{(1)}$, $\mathbf{h}_2^{(1)}$, and $\mathbf{h}_3^{(1)}$, where $\mathbf{h}_1^{(1)}$ receives input from bands 1-4 in input layer, $\mathbf{h}_2^{(1)}$ receives input from bands 2-5, and so on.

are tied and shared for all positions within the whole input space, as in figure 1. In this case, computation of all filters activations can be simply viewed a convolution of the filter weights and the input signals. This type of weight sharing works well in image processing. For example, a set of filters that work as edge detectors can be uniformly applied to the whole image irrelevant of any particular position.

However, in speech signals the local structures appearing at different frequency bands may behave in a quite different way. Therefore, it may be better to limit weight sharing only to those local filters that are close to each other and will be pooled together in the max-pooling layer. This weight sharing strategy is depicted in figure 2 where one set of filter weights is used for each pooling band. As a result, we divide the convolution layer into a number of convolution sections, where all convolution bands in each section are pooled together into one pooling layer band and are computed by convolving section filters with a small number of the input layer bands. In this case, the pooling layer activations are computed as:

$$p_{m,j} = \max_{k=1}^{r}(h_{k,j}^{(m)}), \tag{3}$$

with

$$h_{k,j}^{(m)} = \theta\left(\sum_{b=1}^{s-1} \mathbf{w}_{b,j}^{(m)} \mathbf{v}_{m\times n+b+k}^{T} + a_j^{(m)}\right) \tag{4}$$

Where $h_{k,j}^{(m)}$ is the activation of the jth filter of the mth section of the convolution layer applied at the kth band position. In this context, it's more suitable to call n *band shift* in the max-pooling layer.

One disadvantage of this weight sharing is that other pooling layers can not be added on top of it because the filters outputs in different pooling bands are not related. Therefore, this type of weight sharing is normally used only in the topmost pooling layer.

3.4. Model Structure

The CNN consists of one or more pairs of convolution and max-pooling layers, where the lowest layers process a small number of input frequency bands independently to generate higher level representation with lower frequency resolution. The number of bands decreases in higher layers. The input to each convolution layer can be padded to ensure that the first and last input bands are processed by a suitable number of filters in the convolution layer. In this work, each input is padded by adding half of filter size of dummy bands before and after the first and last bands so that the number of bands stays the same in both the input and convolution layers. Usually the top layers in CNN are fully connected just like that of a normal forward-feeding NN. These fully connected top layers are expected to combine different local structures extracted in the lower layers for the final recognition purpose.

When it is used in a hybrid NN-HMM model for speech recognition, posterior probabilities of HMM states are computed using a top softmax layer. The CNN processes each input speech utterance by generating all HMM state probabilities for each frame. Then a Viterbi decoder is used to get the sequence of labels corresponding to the input utterance as done in [1].

In training stage, CNN is estimated using the standard back-propagation algorithm to minimize cross entropy of targets and output layer activations. For a max-pooling layer, the error signal is back-propagated only to the convolution layer node that generates the maximum activation within the pooled nodes. The training targets are obtained from forced alignments generated from a trained HMM model.

4. EXPERIMENTS AND RESULTS

4.1. Experimental Setup

Phone recognition experiments are performed on the TIMIT corpus[1] to evaluate effectiveness of the proposed CNN models. We use the 462-speaker training set and remove all SA records (i.e., identical sentences for all speakers in the database) since they could bias the results. A separate development set of 50 speakers is used for tuning all of the meta parameters. Results are reported using the 24-speaker core test set, which has no overlap with the development set.

In feature extraction, speech is analyzed using a 25-ms Hamming window with a 10-ms fixed frame rate. The speech feature vector is generated by a Fourier-transform-based filter-banks, which includes 40 coefficients distributed on a mel scale and energy, along with their first and second temporal derivatives.

All speech data are normalized by averaging over all training cases so that each coefficient or first derivative or second derivative all has zero mean and unit variance. We use 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 [10] for scoring. In our experiments, a bi-gram language model over phones, estimated from the training set, is used in decoding.

For network training, a learning rate annealing and early stopping strategies are utilized as in [1]. The NN input layer includes a context window of 15 frames. The input of CNN is divided into 40 bands. Each band includes one of the 40 filter-bank coefficients along the 15 frames context window including their first and second derivatives. Moreover, all bands of the first convolution layer receive the energy as an extra input because it is not suitable to treat it as a frequency band. Moreover the inputs of convolution layers are padded as described in section 3.4.

4.2. Influence of Pooling

In the first set of our experiments, we first evaluate effect of different pooling size n in the CNN configuration. Assume each CNN is

[1]http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC93S1.

composed of a convolution layer with the limited weight sharing and filter size of 8 bands, a max-pooling layer with a sub-sampling factor of 2, and one top fully-connected hidden layer having 1000 nodes and one additional the softmax top layer to generate state posterior probabilities. We have tested CNNs with pooling sizes from 1 (no max-pooling) to 8. For comparison, CNNs are compared with a fully connected NN having two hidden layers with 1000 nodes each. All networks used logistic activation functions. The number of filters per band in the convolution layer is selected to have similar number of trainable weights as the baseline NN and they range from 97 to 80 filters.

The recognition results are shown in figure 3, which clearly indicate a siginificant and consistent reduction of the error rate as the pooling size is increased up to 6 bands. In this case, the recognition error is reduced from 22.57% to 20.1%, which accounts for about 11% relative error reduction. We also note that the use of local filters without max-pooling only gives a small error reduction about 0.5% absolute.

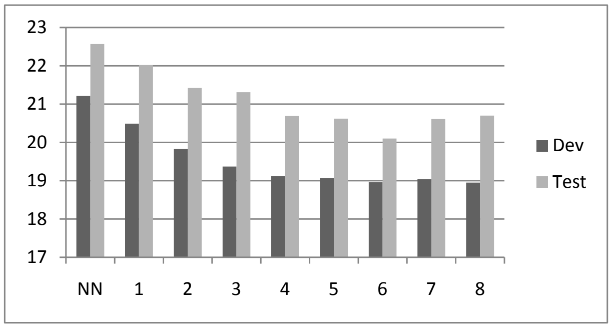

Fig. 3. Influence of pooling size on phone error rate. The horizontal axis represent the pooling size of the CNN.

4.3. Deep Model Performance

In this section, we evaluate a deep CNN that consists of one pair of convolution and max-pooling layers pair and TWO fully connected hidden layers. Each of the fully connected hidden layers still has 1000 hidden nodes. The convolution layer has 84 filters per band and a filter size of 8 bands, and pooling size is set to 6 with limited weight sharing scheme and a sub-sampling factor of 2. Here, we compare this CNN model with another NN consisting of three hidden layers (each has 1000 hidden nodes) which has a similar number of trainable weights. As shown in table 1, this CNN model achieves phone error rate of 20.07% which is significantly better than the 3-layer NN baseline. Moreover, this result is also better than previously published results on the same TIMIT core test sets using the same speech features.

5. CONCLUSION

In this paper, we have proposed to use the CNN principles in frequency domain to normalize acoustic variations for speech recognition. The results have shown that the use of max-pooling significantly improves recognition performance on the TIMIT data sets. The best results using CNN are shown to overtake some previously published results under similar experimental setup, which are obtained by using Deep belief networks as well as an RBM-based pretraining stage [1].

Table 1. Comparison of our best CNN with 3-hidden-layer NN as well as other published results in terms of phone error rates on TIMIT core test set.

Method	PER
NN with 3 hidden layers of 1000 nodes	22.95%
CNN with no pre-training (this work)	**20.07%**
NN with DBN pre-training [1]	20.70%
NN with DBN pre-training and mcRBM features extraction [11]	20.50%

Experiments on larger and noisy databases are needed to further study the robustness of the technique. Further improvements may be achieved by using unsupervised pre-training and repeating pooling operations in more than one layer.

6. REFERENCES

[1] A. Mohamed, G. Dahl, and G. Hinton, “Acoustic modeling using deep belief networks,” *Audio, Speech, and Language Processing, IEEE Transactions on*, 2011.

[2] 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.

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

[4] 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.

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

[6] 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. 1995, MIT Press.

[7] 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*, pp. 1096–1104. 2009.

[8] A. Andreou, T. Kamm, and J. Cohen, “Experiments in vocal tract normalization,” in *Proc. the CAIP Workshop: Frontiers in Speech Recognition II*, 1994.

[9] M.J.F. Gales, “Maximum likelihood linear transformations for hmm-based speech recognition,” *Computer Speech and Language*, vol. 12, pp. 75–98, 1998.

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

[11] G. E. Dahl, M. Ranzato, A. Mohamed, and G. E. Hinton, “Phone recognition with the mean-covariance restricted boltzmann machine,” in *Advances in Neural Information Processing Systems*, 2010, number 23.