Rapid and Effective Speaker Adaptation of Convolutional Neural Network Based Models for Speech Recognition

Ossama Abdel-Hamid and Hui Jiang

Department of Computer Science and Engineering
York University, Toronto, Ontario, Canada
ossama@cse.yorku.ca, hj@cse.yorku.ca

Abstract

Recently, we have proposed a novel fast adaptation method for the hybrid DNN-HMM models in speech recognition [1]. This method relies on learning an adaptation NN that is capable of transforming input speech features for a certain speaker into a more speaker independent space given a suitable speaker code. Speaker codes are learned for each speaker during adaptation. The whole multi-speaker training dataset is used to learn the adaptation NN weights. Our previous work has shown that this method is quite effective in adapting DNNs even when only a very small amount of adaptation data is available. However, the proposed method does not work well in the case of convolutional neural network (CNN). In this paper, we investigate the fast adaptation of CNN models. We first modify the speaker code based adaptation method to better suit to the CNN structure. Moreover, we investigate a new adaptation scheme using speaker specific adaptive nodes output weights. These weights scale different nodes outputs to optimize the model for new speakers. Experimental results on the TIMIT dataset demonstrates that both methods are quite effective in terms of adapting CNN based acoustic models and we can achieve even better performance by combining these two methods together.

Index Terms: Fast Adaptation, Convolutional Neural Network, Hybrid NN-HMM, Speaker Code, Adaptive Output Weight

1. Introduction

In automatic speech recognition (ASR), speaker adaptation techniques attempt to optimize speaker-independent acoustic models toward a target speaker based on a small amount of adaptation data from the target speaker [2, 3, 4, 5]. There are a number of successful speaker adaptation techniques proposed for HMM-based models, such as MAP [2, 6], MLLR [3, 7], and CMLLR [4]. However, these old adaptation techniques can not be easily applied to hybrid DNN-HMM models. Some methods have been proposed to adapt large neural networks towards a target speaker, such as linear input network (LIN) as in [8], linear hidden transformation in [9], parametric NN activation functions in [10], and so on. Despite of these, speaker adaptation still remains as a very challenging problem for the DNN-HMM models especially when only a limited amount of adaptation data is available. Though, there has been a number of attempts for fast speaker adaptation either using the knowledge of possible speaker variations as in [11] or by restricting the features transform by learning transformation space as in [12].

In our previous work in [1], we have proposed a fast speaker adaptation for the DNN-HMM model that works well even when only a few adaptation utterances are available. This method relies on a joint training procedure to learn a generic adaptation neural network (NN) from the whole training set as well as many small speaker codes for all different speakers. Each speaker code is estimated from single speaker data. The speaker code is fed to the generic adaptation NN to form an effective nonlinear transformation in feature space to normalize speaker variations. This transformation is controlled by the speaker code. During adaptation, a new speaker code for a new speaker is learned such that the performance of the new speaker is optimized on the adaptation data. This method is appealing because the large adaptation network can be reliably learned from the entire training data set while only a small speaker code is needed for each speaker. Moreover, the speaker code size can be freely adjusted according to the amount of available adaptation data. As a result, it is possible to conduct a very fast adaptation of the hybrid NN/HMM model for each speaker based on only a small amount of adaptation data. As shown in [1], the proposed speaker code based adaptation method works well with DNN based acoustic models while it does not yield similar performance gain in case of convolutional neural network (CNN). As shown in [13, 14], CNN can achieve significantly better performance than DNN in speaker-independent ASR tasks. The reasons for that failure may be attributed to the ability of CNN to normalize speaker differences or the use of max-pooling and convolution operations in CNN making it more difficult to learn good adaptation NN weights beneath the convolution layers.

In this paper, we study how to conduct effective adaptation for CNN based acoustic models, particularly when only a very small amount of adaptation data is available. More specifically, we first modify the speaker code based adaptation method to better suit to the structure of CNN. We reconfigure the adaptation NN and insert the adaptation NN above the convolution and pooling layers. As a result, we adapt the features computed from the convolution and pooling layers rather than adapting speech features directly as in the original method. This may overcome the difficulty of learning the adaptation NN effectively beneath the convolution and pooling layers. Secondly, we propose a new adaptation scheme using adaptive nodes output weights, which are capable of scaling the NN activations from layer to layer according to the estimated weights. The nodes output weights are learned for each speaker using only the speaker adaptation utterances. The adaptation of the node output weights is more appealing than adapting a standard NN layer because this significantly reduces the number of weights to be learned per speaker from the square of the node number to the same number of nodes in each layer. The experimental results on the TIMIT database have shown that both methods can significantly improve adaptation performance of CNN based on a small number of adaptation utterances per speaker. More-

over, it leads to even better performance when the two methods are combined together, which yields a very competitive performance on TIMIT, namely 18.86% in phone error rate (PER). To our knowledge, this is one of the best results published in TIMIT so far. This work also indicates that it is possible to apply speaker adaptation to CNN to further improve performance even though CNN is regarded to be quite effective in normalizing speaker variations.

2. Speaker code based adaptation

Speaker code based adaptation has been proposed in [1] for DNN based models. In this section, we briefly review the method in [1] and then extend it to perform effective adaptation for CNN.

2.1. Adaptation of DNN

In [1], the initial speaker independent model is based on DNN [15]. This speaker adaptation method consists of two phases. The first phase is the learning of an adaptation NN and the second phase is the learning of speaker codes for new speakers. The adaptation NN is a NN that takes the input speech features along with a speaker code and transforms the features and feeds them back as a new input of the initial speaker independent NN as shown in Figure 1(a). The features transformation aims at removing speaker specific differences so that the speech recognition accuracy is improved. This transformation function depends on the general adaptation NN weights and the speaker code. The adaptation NN is learned using the whole training dataset. The adaptation NN weights are the same for all speakers while each speaker in the training dataset has a different speaker code. Both the adaptation NN weights and training dataset speakers codes are optimized using the training dataset. By the end of this phase we get an adaptation NN capable of adapting new speakers features given a suitable speaker code is provided.

The second phase of adaptation is learning speakers codes for new speakers. During this phase, only the speaker code is optimized for a new speaker using adaptation utterances of this speaker. The whole NN (including the initial speaker independent NN and the adaptation NN) weights are not modified.

The training of both the adaptation NN and the speaker codes uses the same training procedure as the initial NN. The cross entropy between the target state labels and NN outputs is minimized. The weights and speaker codes derivatives are computed regarding the objective function and are used to update them using the stochastic gradient descent algorithm. The initial NN weights are not modified except possibly the lower (first) layer weights which are fine tuned to optimize the whole NN in the existence of the adaptation NN.

2.2. Adaptation of CNN

The above-mentioned method does not work well in adapting CNN models as reported in [1]. This may mainly be attributed to the difficulty of training fully connected NN layers beneath the convolution layer, which involves the complexity of pooling and convolution operations that assume that the input is grouped in a number of frequency bands, which is not the case in the fully connected layers of the adaptation NN. In order to overcome this problem, we re-configure the adaptation NN and move the adaptation NN above the convolution and pooling layers as shown in Figure 1(b). As a result, the adaptation NN will transform the features computed by the convolution and pooling layers instead of the original input speech features.

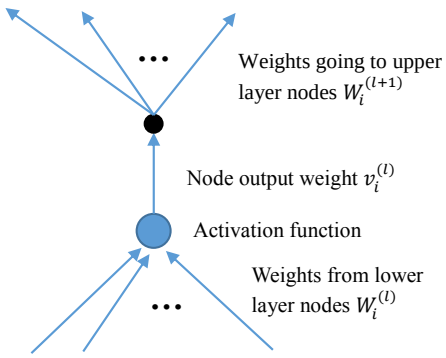

Figure 2: *Node output weight.*

There are a number of possibilities in configuring and fine-tuning of the whole adaptation method. Firstly, the speaker code can be fed to the first CNN layer in addition to the adaptation NN. Secondly, we may choose to fine-tune the first CNN layer (the one beneath the adaptation NN) only or the second layer (the one that receives adaptation NN output) only or both of them during the adaptation stage. We will experimentally evaluate these options in section 4.

3. Adaptive nodes output weights

In order to further improve the adapted CNN model, we propose a new adaptation scheme using adaptive nodes output weights. These adaptive weights may further help in adapting CNN models towards the target speaker. These weights scale the output generated by a NN node after applying the activation function as shown in Figure 2. These weights can be viewed as an extra linear layer with a diagonal weight matrix. These weights can be applied to any layer of the NN or even the input layer. In the case of standard fully connected layers, these weights simply increase or decrease the activities of different nodes from layer to layer. In the case of convolution layers, these adaptive weights may serve as a mechanism to force the pooling operations towards certain more preferable frequency shifts. For example, for a male speaker there may be tendency to decrease activations of nodes representing shifts towards higher frequencies of the same features map kernel. This may adaptively decrease the effective pooling size, and hence reduce confusion between similar phonemes [14].

Let's assume that the ith node in a certain NN layer has the output o_i. The scaled output (after multiplying by the node output weight, v_i) is:

$$\hat{o}_i = o_i \exp(v_i) \tag{1}$$

Note that we represent the weight as an exponential of the parameter v_i to guarantee its positivity during training. The derivative $\frac{\partial E}{\partial \hat{o}_i}$ will be the same as the standard derivative computed using the back-propagation algorithm. But when the output weight is used the derivative of o_i and v_i can be computed using:

$$\frac{\partial E}{\partial o_i} = \frac{\partial E}{\partial \hat{o}_i} \exp(v_i) \tag{2}$$

and

$$\frac{\partial E}{\partial v_i} = \frac{\partial E}{\partial \hat{o}_i} o_i \exp(v_i) \tag{3}$$

For a convolution layer the output of the ith feature map of the jth band at the sth shift location after applying the output

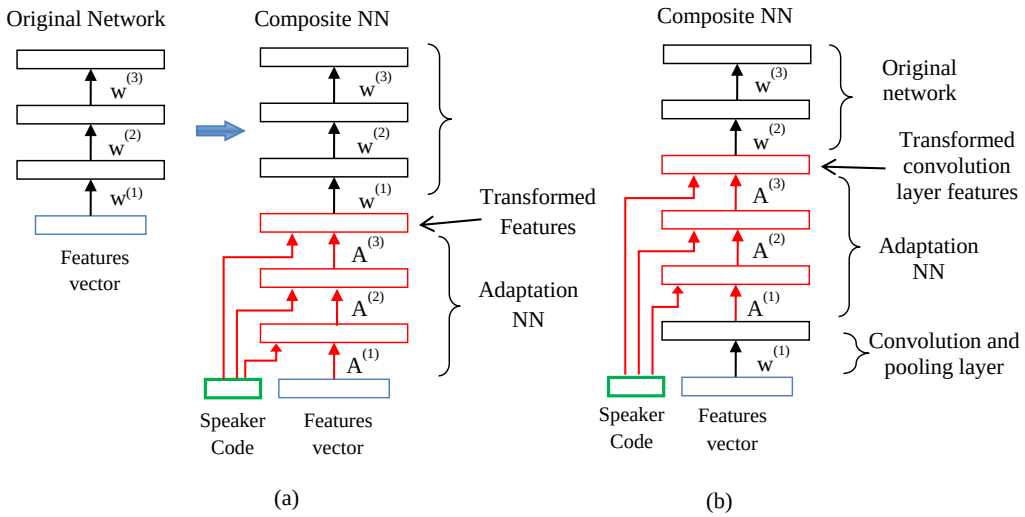

Figure 1: *Speaker adaptation based on speaker codes: (a) previously proposed model for DNN; (b) the proposed model for CNN*

weight is:

$$\hat{o}_{i,j,s} = o_{i,j,s} * \exp(v_{i,j,s}) \quad (4)$$

And after applying max pooling it becomes:

$$p_{i,j} = \max_s \hat{o}_{i,j,s} = \max_s o_{i,j,s} * \exp(v_{i,j,s}) \quad (5)$$

Equation 5 shows that if the weight $\exp(v_{i,j,s})$ is zero for some s, this sth shift will not be used in the maximization operation. This will force the maximization operation towards certain shifts depending on the speaker characteristics.

The introduced nodes output weights will be optimized using the speaker adaptation sentences in the same way as the speaker codes in the previous section except that there is no need to use an adaptation NN. The weight parameters v_i will be initialized with zero then optimized using a number of stochastic gradient descent epochs. Obviously, both the speaker code based adaptation and nodes output weights adaptation can be combined, where both speaker code and output weights are jointly learned based on adaptation data in the adaptation stage.

4. Experiments

4.1. Experimental setup

We perform phone recognition experiments on the TIMIT corpus to evaluate effectiveness of the proposed adaptation methods. We use the standard 462-speaker training set and remove all SA records (i.e., identical sentences for all speakers in the database) since they may 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. Each speaker in the test set has eight utterances.

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 filterbanks which include 40 coefficients distributed on a Mel scale and energy, along with their first and second temporal derivatives. This leads to a 123-dimension feature vector per speech frame. All speech data are normalized by averaging over all training samples so that all feature vectors have zero mean and unit variance. We use 183 target class labels (i.e., 3 states for each one of the 61 phones) for NN training. After decoding, the 61 phone classes were mapped to a set of 39 classes as in [16] for scoring purpose. In our experiments, a bi-gram language model in phone level, estimated from the training set, is used in decoding.

For training the weights of the original NN and the adaptation NN, a learning rate annealing and early stopping strategies are utilized as in [15]. The NN input layer includes a context window of 15 consecutive frames. During adaptation we used a fixed learning rate of 0.1 and 0.025 for sigmoid layers and linear layers in order. The number of epochs is determined using the development set and it is optimized independently for each adaptation data set size. Since each test speaker has eight utterances in total. Testing is conducted for each speaker based on a cross validation method. In each run, for each speaker, eight utterances are divided into n_a utterances for supervised adaptation and the remaining $8 - n_a$ utterances for testing. This is repeated eight times for each speaker. Each time, different adaptation and test utterances are randomly selected in such a way that each utterance is assigned the same number of times for both adaptation and testing. The overall recognition performance is the average of the eight runs. In the following sections we use 7 utterances for adaptation unless otherwise mentioned.

In the following experiments, we use the best performing CNN structure as in [13], which has a convolution and pooling layer, two hidden layers with 1000 nodes each, and a soft-max output layer. On the other hand, the used DNN contains two hidden layers and each one has 1000 nodes. The adaptation NN contains two hidden layers and each one has 1000 nodes in addition to the output layer which has the same number of nodes as the input layer.

4.2. CNN adaptation using speaker codes

In this set of experiments, we first evaluate performance of the speaker code based adaptation method with the adaptation NN re-configuration as proposed in section 2. The baseline speaker independent CNN has one convolution and pooling layer and two hidden layers and its structure is the same as the best performing network in [13]. In these experiments, we have tested different design and training options. The results are listed in Table 1, where rows 3 and 4 compares the performance of feed-

ing speaker code to the convolution layer, and rows from 4 to 7 compare different fine-tuning options of the original CNN weights. As shown in Table 1, the proposed modifications for the speaker code based adaptation yield significant performance gain for CNN than the original method in [13], namely improving PER from 20.07% to 19.14% by using only 7 adaptation utterances which we cannot get without the proposed modification as in row 2.

Table 1: *PER (Phone error rate in %) of CNN adaptation based on speaker code only. The firs column shows whether the adaptation NN is added above the convolution layer (Reordering) or not (No reordering), and whether the convolution layer receives the speaker code (SC) or not. The "FT" column shows the index of the fine tuned layers of the original CNN.*

#	Configuration	FT	PER
1	CNN Baseline [13]	None	20.07%
2	No reordering	1	24.57%
3	Reordering	1	**19.14**%
4	Reordering + CNN gets SC	1	19.25%
5	Reordering + CNN gets SC	2	19.44%
6	Reordering + CNN gets SC	1, 2	19.55%
7	Reordering + CNN gets SC	None	19.31%

4.3. CNN adaptation using adaptive output weights

In this section, we evaluate the adaptation performance using adaptive nodes output weights. As shown in Figure 2, adaptive output weights can be applied to any layer of CNN and the weights are learned during adaptation only based on the adaptation data. In Table 2, we show several different configurations to apply adaptive weights to various layers of CNN for adaptation. It is shown that it is not useful to adapt input scale (i.e. layer 0) but the proposed adaptive weights can significantly improve adaptation performance when they are applied to upper layers in CNN. For instance, when applying adaptive weights to layers 1 and 2, we can improve PER from 20.07% to 19.20% by using only 7 adaptation utterances.

Table 2: *Adaptation performance based on adaptive nodes output weights, which are applied to various layers of the CNN baseline. The layer 0 indicates multiplying the input features by adaptive scalar weights directly.*

#	Configuration	PER
0	baseline [13]	20.07%
1	layer 0	20.28%
2	layer 0+1	19.79%
3	layer 0+1+2	19.65%
4	layer 1	19.64%
5	layer 1+2	**19.20**%
6	layer 2	19.21%

4.4. Combining output weights and speaker code methods for DNN and CNN

In this section, we consider to combine the above two different adaptation methods, one is based on adaptive nodes output weights and the other on speaker codes.

First of all, we consider the DNN-based models as studied in [1]. It is straightforward that the proposed adaptive output weights can be equally applied to DNN as well. As shown in the first row of Table 3, adaptive weights based adaptation also works for DNN, improving PER from 22.83% to 21.56% in PER. It is noted that it gives worse results than that of speaker codes in case of DNN. When we combining both methods for DNN, it yields 20.69% in PER, which is still not as good as speak code adaptation in [1], i.e. 20.47% in PER.

Secondly, we consider to combine both adaptation methods for CNN based models. The second row in Table 3 shows adaptation performance of the two methods and their combination. After combining the two adaptation methods, the performance of CNN is largely improved to 18.86%, which represents over 6% relative error reduction of PER over the CNN baseline. To our knowledge, this is one of the lowest PERs published on TIMIT so far.

Table 3: *Adaptation performance by combining and speaker code adaptation (SC) and nodes output weight adaptation (OW) for both DNN and CNN.*

Model	Baseline	SC	OW	SC+OW
DNN	22.83 [1]	20.47 [1]	21.56	20.69
CNN	20.07	19.14	19.20	**18.86**

5. Conclusions

In this paper, we have investigated speaker adaptation methods of the CNN. Firstly, we have modified the speaker code based adaptation method to suit to CNN, where the adaptation NN is moved onto the top of the convolution and pooling layers. As a result, the adaptation NN transforms the features computed by the convolution and pooling layers instead of transforming the input itself. This method can improve the performance and reduce the PER by 4% (relative). Secondly, we have proposed a new adaptation scheme that uses adaptive output node weights. These weights provide a more efficient way to adapt NN without modifying the original NN weights. The number of nodes output weights is equal to the number of nodes in a layer. As a result, the number of parameters to be learned per speaker is much less than that of modifying the layer weights. Experimental results show that this method achieves PER relative reduction of more than 4% for CNN and of more than 5% for DNN. Moreover, the combination of the two methods achieves better results for CNN giving 18.86% in PER. For DNN, nodes output weights do not add to the performance when combined with the speaker code based adaptation.

This work shows that although CNN performs quite effective speaker normalization, further improvement is still achievable by using different speaker adaptation techniques. The success with the modified speaker code based adaptation with CNN indicates that the poor adaptation performance of the previously proposed speaker code based method can be attributed mainly to the difficulty of training an adaptation NN beneath the convolution layer. This difficulty is caused by the use of convolution and max pooling operations in the convolution layer. Further investigation is needed to determine whether a different type of pooling function that has smoother derivatives, such as soft-max pooling, can solve this problem or not. Finally, it is also interesting to investigate the proposed adaptation methods in some large scale ASR tasks such as in [17, 18].

6. References

[1] O. Abdel-Hamid and H. Jiang, "Fast speaker adaptation of hybrid nn/hmm model for speech recognition based on discriminative learning of speaker code," in *Acoustics, Speech and Signal Processing (ICASSP), IEEE International Conference*, 2013.

[2] J.-L. Gauvain and C.-H. Lee, "Maximum a posteriori estimation for multivariate gaussian mixture observations of markov chains," *Speech and Audio Processing, IEEE Transactions on*, vol. 2, no. 2, pp. 291–298, Apr 1994.

[3] C. J. Leggetter and P. C. Woodland, "Maximum likelihood linear regression for speaker adaptation of continuous density hidden markov models," *Computer Speech and Language*, vol. 9, no. 2, pp. 171 – 185, 1995.

[4] V. Digalakis, D. Rtischev, and L. Neumeyer, "Speaker adaptation using constrained estimation of gaussian mixtures," *Speech and Audio Processing, IEEE Transactions on*, vol. 3, no. 5, pp. 357 –366, Sep 1995.

[5] P. C. Woodland, "Speaker adaptation for continuous density HMMs: A review," in *ISCA ITR-Workshop on Adaptation Methods for Speech Recognition*, Aug 2001, pp. 11–19.

[6] S. M. Ahadi and P. C. Woodland, "Combined bayesian and predictive techniques for rapid speaker adaptation of continuous density hidden markov models," *Computer Speech and Language*, vol. 11, no. 3, pp. 187 – 206, 1997.

[7] M. Gales, "Maximum likelihood linear transformations for hmm-based speech recognition," *Computer Speech and Language*, vol. 12, no. 2, pp. 75 – 98, 1998.

[8] J. Neto, L. Almeida, M. Hochberg, C. Martins, L. Nunes, S. Renals, and T. Robinson, "Speaker-adaptation for hybrid hmm-ann continuous speech recognition system," in *Eurospeech*, Madrid, Spain, 9 1995, pp. 2171–2174.

[9] R. Gemello, F. Mana, S. Scanzio, P. Laface, and R. D. Mori, "Linear hidden transformations for adaptation of hybrid ann/hmm models," *Speech Communication*, vol. 49, pp. 827 – 835, 2007.

[10] S. M. Siniscalchi, J. Li, and C.-H. Lee, "Hermitian based hidden activation functions for adaptation of hybrid hmm/ann models," in *Interspeech*, 2012.

[11] N. Strm, "Automatic continuous speech recognition with rapid speaker adaptation for human/machine interaction," Ph.D. dissertation, KTH, Stockholm, 1997.

[12] S. Dupont and L. Cheboub, "Fast speaker adaptation of artificial neural networks for automatic speech recognition," in *Acoustics, Speech, and Signal Processing, 2000. ICASSP '00. Proceedings. 2000 IEEE International Conference on*, vol. 3, 2000, pp. 1795–1798 vol.3.

[13] O. Abdel-Hamid, A. Mohamed, H. Jiang, and G. Penn, "Applying convolutional neural networks concepts to hybrid nn-hmm model for speech recognition," in *Acoustics, Speech and Signal Processing (ICASSP), IEEE International Conference on*, March 2012, pp. 4277–4280.

[14] 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), IEEE International Conference*, 2013.

[15] A. Mohamed, G. Dahl, and G. Hinton, "Acoustic modeling using deep belief networks," *Audio, Speech, and Language Processing, IEEE Transactions on*, 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 Processing*, vol. 37, no. 11, pp. 1641 – 1648, November 1989.

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

[18] J. Pan, C. Liu, Z. Wang, Y. Hu, and H. Jiang, "Investigation of deep neural networks (DNN) for large vocabulary continuous speech recognition: Why DNN surpasses GMMs in acoustic modelling," in *The 8th International Symposium on Chinese Spoken Language Processing (ISCSLP-2012)*, 2012.