FAST SPEAKER ADAPTATION OF HYBRID NN/HMM MODEL FOR SPEECH RECOGNITION BASED ON DISCRIMINATIVE LEARNING OF SPEAKER CODE

Ossama Abdel-Hamid and Hui Jiang

Department of Computer Science and Engineering
York University, Toronto, Ontario, M3J1P3 CANADA
Email: ossama@cse.yorku.ca hj@cse.yorku.ca

ABSTRACT

In this paper, we propose a new fast speaker adaptation method for the hybrid NN-HMM speech recognition model. The adaptation method depends on a joint learning of a large generic adaptation neural network for all speakers as well as multiple small speaker codes (one per speaker). The joint training method uses all training data along with speaker labels to update adaptation NN weights and speaker codes based on the standard back-propagation algorithm. In this way, the learned adaptation NN is capable of transforming each speaker features into a generic speaker-independent feature space when a small speaker code is given. Adaptation to a new speaker can be simply done by learning a new speaker code using the same back-propagation algorithm without changing any NN weights. In this method, a separate speaker code is learned for each speaker while the large adaptation NN is learned from the whole training set. The main advantage of this method is that the size of speaker codes is very small. 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 (i.e., just a few utterances). Experimental results on TIMIT have shown that it can achieve over 10% relative reduction in phone error rate by using only seven utterances for adaptation.

Index Terms— Fast Adaptation, Neural Network, Hybrid NN-HMM, Speaker Code

1. INTRODUCTION

Speaker adaptation techniques try to optimize system performance towards one target speaker (or a group of speakers). This is done by either modifying a trained model parameters to match the target speaker or modifying the target speaker features to match the trained system. Adaptation depends on collecting a limited amount of adaptation data from each speaker. And it is usually desirable to achieve better performance with only a small amount of adaptation data. For HMM, maximum a posteriori (MAP) [1, 2] has been popular and achieves good performance when a large amount of adaptation data is available. In MAP, all HMM parameters are re-estimated to optimize the model for the target speaker. While maximum likelihood linear regression (MLLR) [3, 4] and constrained MLLR (CMLLR) [5] works much better when only a small amount of adaptation data is used [6]. In these methods, all trained HMM parameters are transformed by a linear transform function that is learned from adaptation data. In CMLLR this transform can be viewed also as transforming the data itself to match the HMM model. In VTLN [7] a parametric frequency warping function is used to normalize speech features among different speakers. This method is successful and robust since only one free parameter needs to be optimized per speaker. However, its performance is somehow limited by a manually designed frequency warping function.

Recently, the hybrid NN-HMM model has gained more research interests in speech recognition because it has achieved very good performance in both small tasks like TIMIT [8] and large vocabulary tasks [9, 10, 11, 12] especially when a deep NN (DNN) is used. Some general adaptation techniques like VTLN can be used to normalize speech feature in these models as in [13, 12]. Moreover, a number of speaker adaptation techniques have been proposed to adapt hybrid NN-HMM models towards new target speakers. The simplest method is to modify all NN weights using the available adaptation data based on the standard BP training procedure [14]. But this method is very prone to over-fitting especially when some class labels are missing in the adaptation data. Another more successful method is to add a linear input network (LIN) as in [14] after the input. During adaptation, only those weights of this linear layer are learned based on the adaptation data. This solves the over-fitting problem to some extent. In LHN [15], the linear transformation layer is added before the output layer to process higher level features. In [16], a parametric NN activation function is adapted instead of the NN weights. All these methods depends on modifying part or all of the NN parameters using the limited speaker adaptation data. Due to the large number of parameters to be re-estimated in adaptation, these methods typically requires a relatively large amount of adaptation data to be effective. Even in LIN or LHN, the number of parameters is not easily controllable because it depends on the number of input or output nodes. A number of solutions have been proposed to reduce over-fitting. For example, conservative training in [15] helps to handle missing classes in the adaptation data. Even though, most of the existing adaptation methods fail to do any type of fast adaptation for the NN-based models since the over-fitting problem is always inevitable when only a small amount of adaptation data is available.

In this paper, we propose a new fast speaker adaptation method for NN based models in speech recognition. This method relies on a joint training procedure to learn a generic adaptation NN from the whole training set as well as many small speaker codes, one of which is estimated for each speaker only using data from that particular speaker. 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. Experimental results on TIMIT have shown that it is possible to achieve over 10% relative reduction in phone error rate by using only seven adaptation utterances.

2. MODEL DESCRIPTION

The baseline model is a hybrid NN-HMM model similar to the one described in [17]. The NN computes posteriori probabilities of all HMM states given each input feature vector. The NN inputs are concatenated super-vector consisting of all speech feature vectors within a window of a number of consecutive frames. The baseline NN-HMM model is trained without using any speaker labels information. The NN training targets are HMM state labels. The standard back propagation procedure is used to optimize the NN weights where the cross entropy is used as an objective function.

As shown in the right side of Fig. 1, the proposed speaker adaptation method relies on learning another generic adaptation NN as well as some speaker specific codes. The adaptation NN is inserted above the input layer of original NN-HMM model. All layers of the adaptation NN are standard fully connected layers with a weight matrix, denoted as $\mathcal{W}_a^{(l)}$ with l representing l-th layer of the adaptation NN. The top layer of the adaptation NN represents the transformed features and its size matches the input size.

In addition, each layer of the adaptation NN receives all activation output signals of the lower layer along with a speaker-specific input vector, $\mathcal{S}$, named as *speaker code*. When we estimate the adaptation NN using the back-propagation (BP) algorithm, the derivatives of the objective function are calculated with respect to all weights $\mathcal{W}_a^{(l)}$ (for all l) as well as the associated speaker code $\mathcal{S}$. As a result, both of the weights and speaker codes will be learned. For example, when we apply a speech vector from i-th speaker to update the adaptation NN in BP, we use the computed derivatives to update all weights, $\mathcal{W}_a^{(l)}$ (for all l), and the speaker code $\mathcal{S}_i$ specific to the i-th speaker. In this way, we will be able to benefit from speaker labels to learn a generic adaptation NN as well as a whole bunch of speaker codes at the end of the BP training process. Each speaker has his/her own speaker code and each speaker code, $\mathcal{S}_i$, is a very compact feature vector representing speaker-dependent information. The speaker code is fed to the adaptation NN to control how each speaker's data is transformed to a general speaker-independent feature space by the generic adaptation NN. Moreover, this model configuration provides a very effective way to conduct speaker adaptation for the hybrid NN/HMM model. To adapt an existing hybrid NN/HMM model to a new speaker, only a new speaker code, $\mathcal{S}$, needs to be estimated without changing any weights in both original NN and adaptation NN in Fig. 1.

The advantage of our proposed method is that only a small speaker code needs to be estimated for each new speaker. This largely reduces the required amount of adaptation data per speaker particularly when a small speaker code is chosen for each speaker. As a result, it is possible conduct very rapid speaker adaptation for the hybrid NN-HMM model based on only a few utterances per speaker. On the other hand, if a large amount of adaptation data is available per speaker, the size of speaker code can be increased to allow a better representation of each speaker. Moreover, the generic adaptation NN is learned using all training data. This allows to build a large-scale adaptation NN that is powerful enough to model a complex transformation function between different feature spaces. This method is clearly superior to other speaker adaptation methods that learn a complete independent transform for each speaker, where each transformation needs to be linear.

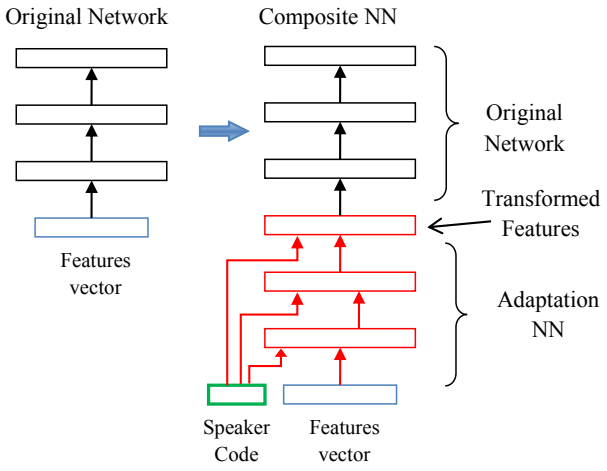

Fig. 1. Speaker adaptation of the hybrid NN-HMM model based on speaker code.

2.1. Training

During training, we want to learn three sets of parameters: the original NN weights, the adaptation weights, and the training speakers codes. First of all, the original NN weights is learned without inserting the adaptation weights in the same way as a standard hybrid NN-HMM model without using any speaker information. This results in a speaker independent (SI) NN-HMM model.

Secondly, the adaptation layers are inserted and all adaptation weights, $\mathcal{W}_a^{(l)}$ (for all l), and speakers codes $\mathcal{S}_i$ for all speakers in the training set, are learned jointly in such a way that the frame-wise classification performance is optimized. In this paper, these parameters are optimized using the standard back-propagation algorithm with the cross entropy objective function. Both adaptation weights and speaker codes are initialized randomly at the beginning. No weight in the original NN is modified during this phase.

Of course, other training scenarios are possible here. For example, all or part of the original NN weights can be further fine tuned when learning the adaptation NN to further optimize the whole network because speaker labels are considered in this phase. Another possibility is to learn all the three sets of parameters at the same time. However, this may result in two inseparable NNs and they eventually become one large deep NN with only a number of lower layers receiving a speaker code. Another possibility is to use the learned adaptation NN to transform all training data and a new NN is learned from scratch. This NN receives speaker-normalized features instead of the original features. This can be considered as a form of speaker adaptive training for the NN-HMM model.

2.2. Adaptation

After learning all adaptation NN weights using all training data as above, adaptation to a new speaker is done by learning a new speaker code for each new speaker who is not observed in the training set.

In this paper, we use the supervised adaptation method where a small set of labelled adaptation utterances are available for each new speaker. State level alignments are obtained from the trained HMM model using the standard forced alignment method. We still use the same back-propagation training procedure when we train the adaptation NN in the above except that only the speaker code $\mathcal{S}$ is updated based on its derivatives while all NN weights are kept unchanged in this adaptation stage. After the speaker code is learned from a small amount of adaptation data, the speaker code will be fed to the NNs along with new test data as shown in Fig. 1 for testing.

3. EXPERIMENTS

3.1. Experimental setup

We have performed some phone recognition experiments on the TIMIT corpus[1] to evaluate effectiveness of the proposed fast adaptation method. 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 filter-banks 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 [18] 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 [17]. 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 adaptation and the remaining $8 - n_a$ utterances for test. 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 all eight runs.

3.2. Performance on different adaptation set sizes

In the first set of experiments, we measure the performance of the proposed fast adaptation method using different amounts of adaptation data. In this experiment, a baseline NN with two hidden layers is first trained. We use an adaptation NN as shown in figure 1. The adaptation NN has two hidden layers in addition to the output layer which has a linear activation function. All hidden layers have 1000 nodes each. We used speaker code size of 50. The bottom layer of the baseline NN is fine-tuned during learning of the adaptation NN.

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

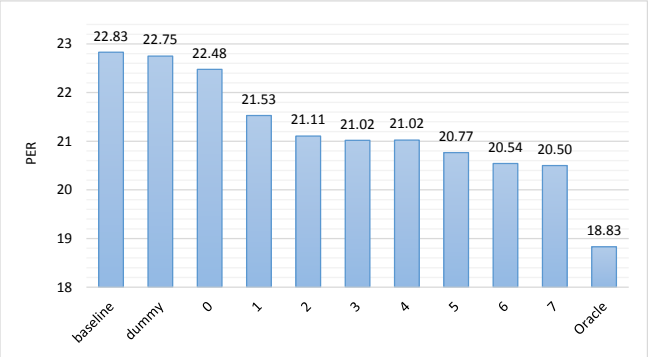

Fig. 2. Adaptation performance (phone error rate in %) as a function of the number of adaptation utterances.

Figure 2 shows the adaptation performance using different numbers of adaptation utterances (varying n_a from 0 to 7). It shows that even with one adaptation utterance, we can achieve a performance gain of about 1.3% (about 5.7% relative) reduction in error rate. After using seven utterances for adaptation, phone error rate (PER) drops from 22.83% to 20.5%. The "dummy" adaptation means we add adaptation NN without feeding any speaker codes. The results show that adding these dummy layers achieves only minor improvement. This confirms that the gain is obtained actually from adaptation not from just adding more layers. Using zero adaptation utterances means that the speaker code is not tuned during adaptation and left as zeros. It indicates that this kind of adaptive training even without adaptation during testing may be a little bit helpful. The Oracle score is the PER when we use the same eight utterances per speaker for both adaptation and testing.

3.3. Fine-tuning of baseline NN

In this section we compare different schemes of fine tuning the original network. Since the original NN was trained on the features before adaptation. Using it with the adapted features may limit the benefit of speaker code. We would like to test fine-tuning the original NN. We used exactly the same structure as the previous section. Table 1 shows the different fine-tuning schemes. It shows that fine tuning only the first layer results in the best performance. Fine-tuning the whole network either from random weights or starting from the baseline NN weights is worse than fine tuning only the first layer. This may be attributed to overfitting on the adaptation sentences and the possible encoding of some phonemes labels information from within the speaker code. In other words, when the whole NN is retrained when speaker code information is available, this may bias classification towards seen labels in adaptation sentences. However it does not yield as good performance too when using the baseline NN without fine tuning.

3.4. Linear vs sigmoid layer in adaptation NN

The aim of the adaptation NN is to transform the input features from the speaker space into a speaker-independent space that is more suitable for recognition. In this section, we compare using linear and sigmoid activation functions for the adaptation network top layer.

Table 1. Comparison of different fine-tuning schemes.

Fine tuning scheme	PER
No fine tuning	21.24%
Fine tune first layer	20.50%
Fine tune the whole NN	21.34%
randomly initialized NN	21.48%

Table 2. PER (in %) of different activation functions used in the top layer of the adaptation NN. Results are shown for 1 and 7 adaptation utterances.

Activation function	1 utt.	7 utt.
Linear	21.75%	21.24%
Sigmoid	22.21%	21.61%
Linear + fine tune first layer	21.53%	20.50%
Sigmoid + fine tune first layer	21.65%	20.91%

Moreover, we fine tune the baseline NN first layer to compensate the difference between the input linear domain and the sigmoid function domain. The results in Table 2 shows that a linear top layer yields a slightly better performance. This is because linear layer can generate features in the same domain as the original features. Hence these features will better suite the baseline NN weights.

The table also shows that fine tuning the first layer of the baseline NN is much better. Because it helps in reducing the discrepancy between the transformed features and the baseline NN weights.

3.5. Performance with more complex NN architectures

In this section, we try to improve the overall system performance by using some good baseline NNs. We use a pre-trained deep NN (DNN) [17] that contains 5 hidden layers and a convolutional NN (CNN) [19]. As for CNN, we use the same model structure and parameters as the best performing one in [19], i.e, filter size of 8, pooling size of 6, 21 frequency bands, and 84 features maps per frequency band.

Table 3 shows the results of applying the adaptation technique to these models. The first column shows the baseline NN architecture. The second column shows the adaptation NN architecture.

Speaker adaptation improves DNN performance. Although, the relative improvement is not as big as with a smaller NN without pre-training as in section 3.2. Speaker adaptation of a CNN baseline is not good. CNN complex structure is suitable for extracting better representation at the lower layers of the baseline CNN. Although, it may make learning the adaptation NN weights more difficult.

On the other hand, using a deeper adaptation NN didn't help improving the result. While using CNN as adaptation NN structure produces a better performance. It is only slightly better than using a CNN without speaker information. This may be attributed to the ability of CNN of doing inherent speaker normalization.

4. RELATION TO PRIOR WORK

The proposed method can be considered as a feature space transformation technique. Previous feature transformation based speaker adaptation techniques include CMLLR [5] (also known as feature-space MLLR or fMLLR), VTLN [7], and LIN [14]. In both CMLLR and LIN a linear transformation is learned from the adaptation data. In VTLN, a fixed-form transformation function is designed and adaptation data are used to optimize its parameters. In our proposed method, we use a learnable non-linear transformation function optimized on the whole training data set. Its parameters are the speaker code values, which are optimized on the adaptation data.

Table 3. Comparison of different models structures.

Baseline NN	Adaptation NN	PER
NN(5×1000) + PreTraining	None	21.61%
CNN	None	20.1%
CNN	NN(2×1000)	21.65%
NN(5×1000) + PreTraining	NN(2×1000)	20.70%
NN(5×1000) + PreTraining	NN(4×1000)	20.73%
NN(5×1000) + PreTraining	CNN	19.77%

In [20], hidden factors are used in the top layer to represent speaker and environment in a factorial NN model. The probability distribution of frame labels and these factors given the features vector can be estimated from the model. The label posterior probability is estimated by marginalizing over all possible factors values but this is done for each frame separately. While in our proposed model the speaker code is optimized over all speaker adaptation data. Moreover, the speaker code is fed to more than one layer in the adaptation network instead of only the top layer.

Previous hybrid NN-HMM adaptation methods like LIN [14] and LHN [15] depended on modifying the NN weights. In our proposed method, adaptation is achieved by modifying only the small speaker codes while keeping all weights fixed. This leads to a very fast adaptation algorithm for NN-based models.

5. CONCLUSION

In this paper, we have proposed a new fast adaptation method for the hybrid NN-HMM model. The proposed method relies on learning an adaptation NN from the training data and a speaker-dependent code from the adaptation data. It transforms speech features into a speaker-independent space based on the speaker code. The speaker code is found by optimizing the overall composite network performance using the back-propagation algorithm. Experimental results show a 10% relative reduction in phone error rate using only 7 utterances for adaptation. Even with one utterance we get more than 5% relative reduction in phone error rate. Moreover, the experiments show that this improvement is attributed mainly to the use of adaptive speaker codes not only to the increased model complexity.

The proposed method helps us to look at the adaptation problem from a different perspective. Instead of modifying the existing models or learning speaker-dependent transformation, we only learn a small speaker code as a compact description of each speaker. This makes it particularly suitable for very fast adaptation using only a very limited amount of adaptation data. Moreover, the size of speaker codes can be freely adjusted based on the amount of available adaptation data. Experimental results on the TIMIT task have shown that a very large-scale NN model, such as DNN can also be effectively adapted by only a few utterances. Moreover, the adaptation NN can generate better performance by using more complex NN architectures like CNN. Although, further research is needed to investigate the ability to adapt a CNN using speaker information.

6. REFERENCES

[1] 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.

[2] 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.

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

[5] V.V. Digalakis, D. Rtischev, and L.G. 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.

[6] 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.

[7] L. Lee and R.C. Rose, "Speaker normalization using efficient frequency warping procedures," in *Acoustics, Speech, and Signal Processing, 1996. ICASSP-96. Conference Proceedings., 1996 IEEE International Conference on*, may 1996, vol. 1, pp. 353 –356 vol. 1.

[8] M. Abdel-Rahman, G. Dahl, and G. Hinton, ""deep belief networks for phone recognition," in *NIPS Workshop on Deep Learning for Speech Recognition and Related Applications*, Whistler, BC, Canada, 9 2009.

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

[10] G.E. Dahl, Dong Yu, Li Deng, and A. Acero, "Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 20, no. 1, pp. 30–42, Jan. 2012.

[11] 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.

[12] 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.

[13] F. Seide, G. Li, X. Chen, and D. Yu, "Feature engineering in context-dependent deep neural networks for conversational speech transcription," in *Automatic Speech Recognition and Understanding (ASRU), 2011 IEEE Workshop on*, dec. 2011, pp. 24 –29.

[14] 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, p. 21712174.

[15] R. Gemello, F. Mana, S. Scanzio, P. Laface, and R. De Mori, "Linear hidden transformations for adaptation of hybrid ann/hmm models," *Speech Communication*, vol. 49, no. 1011, pp. 827 – 835, 2007.

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

[17] A. Mohamed, G.E. 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.

[18] 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.

[19] 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), 2012 IEEE International Conference on*, march 2012, pp. 4277 –4280.

[20] Dong Yu, Xin Chen, and Li Deng, "Factorized deep neural networks for adaptive speech recognition," in *Int. Workshop on Statistical Machine Learning for Speech Processing*, 2012.