IMPROVEMENTS TO FILTERBANK AND DELTA LEARNING WITHIN A DEEP NEURAL NETWORK FRAMEWORK

Tara N. Sainath[1], Brian Kingsbury[1], Abdel-rahman Mohamed[2], George Saon[1], Bhuvana Ramabhadran[1]

[1]IBM T.J. Watson Research Center, Yorktown Heights, NY 10598, USA.
[2]Department of Computer Science, University of Toronto, Canada.
[1]{tsainath, bedk, gsaon, bhuvana}@us.ibm.com, [2]asamir@cs.toronto.edu

ABSTRACT

Many features used in speech recognition tasks are hand-crafted and are not always related to the objective at hand, that is minimizing word error rate. Recently, we showed that replacing a perceptually motivated mel-filter bank with a filter bank layer that is learned jointly with the rest of a deep neural network was promising. In this paper, we extend filter learning to a speaker-adapted, state-of-the-art system. First, we incorporate delta learning into the filter learning framework. Second, we incorporate various speaker adaptation techniques, including VTLN warping and speaker identity features. On a 50-hour English Broadcast News task, we show that we can achieve a 5% relative improvement in word error rate (WER) using the filter and delta learning, compared to having a fixed set of filters and deltas. Furthermore, after speaker adaptation, we find that filter and delta learning allows for a 3% relative improvement in WER compared to a state-of-the-art CNN.

1. INTRODUCTION

Designing appropriate feature representations for speech recognition has been an active area of research for many years. For example, in large vocabulary systems, improvements in WER are observed by using speaker-adapted and discriminatively trained features [1, 2, 3]. However, oftentimes feature design is done separately from classifier design, and thus the designed features might not be best for the classification task. Deep neural networks are attractive because they have been shown to do feature extraction jointly with classification [4] such that features are tuned to the classification task.

Convolutional neural networks (CNNs) are a specific type of DNN that have shown state-of-the-art performance across a variety of small and large vocabulary tasks [5], [6]. The most popular features to use with CNNs are hand-crafted log-mel filter bank features. The mel-filter bank is inspired by auditory and physiological evidence of how humans perceive speech signals [7]. We argue that a filter bank that is designed from perceptual evidence is not always guaranteed to be the best filter bank in a statistical modeling framework where the end goal is word error rate (WER). Log-mel features are created by passing a power spectrum through a mel-filter bank, followed by a non-linear log operation, which can be modeled as one layer of a neural network, which we showed to be promising in [8].

In this paper, we extend the work in [8] in a variety of ways. First, it has been shown that log-mel features and their time dynamic information (represented by deltas (d) and double deltas (dd)) are better than using just static log-mel features [6]. The delta operation can be seen as a linear operation on log-mel features, and can also be learned by a neural network. The first goal of this paper is to incorporate delta learning into our filter learning framework.

Second, the work in [8] was presented on a speaker independent (SI) system. Oftentimes we see that gains demonstrated on an SI system disappear once speaker adaptation is incorporated [3]. We add speaker adaptation into the filter and delta learning framework using two methodologies. Vocal tract length normalization (VTLN) is a popular technique that warps the speech from different speakers and different vocal tract lengths into a canonical speaker with an average vocal tract length. While typically VTLN-warping is applied to the filterbank, it can also be applied directly on the power spectra [9], and we follow this implementation in our filter and delta learning framework. In addition, we explore adapting the CNN to the target speaker through the use of identity vectors (i-vectors) [10], which had been previously explored for speaker adaptation of DNNs.

Data-driven learning of filters has been explored in a variety of contexts. For example, [11] investigated deriving RASTA-like filters from phonetically labeled speech data using Linear Discrimininant Analysis (LDA). Furthermore, [12] looked at constructing temporal filters using principal component analysis (PCA) and the minimum classification error (MCE) criterion. In addition, [13] learned a discriminative filter bank model jointly with a classifier using a discriminative training criterion, though based on a relatively simple distance-based classifier. Our work differs from previous work in that filter and delta learning is performed within a neural network framework. As neural networks are state-of-the-art acoustic models [14], it is important that filter and delta learning is done on a strong acoustic model rather than a simpler classifier.

Our experiments are performed on a 50-hr English Broadcast News (BN) task [15]. The baseline system, a state-of-the-art deep CNN [6] trained on log-mel filter bank + d + dd features, has a WER of 19.5%. Using filter and delta learning, we achieve a WER of 18.6%, a 5% relative reduction. By incorporating VTLN and i-vectors, the baseline WER improves to 17.8%. Incorporating both of these techniques into the delta and filter learning network achieves a WER of 17.3%, a 3% relative reduction. This demonstrates that filter and delta learning still retain value after speaker adaptation.

The rest of this paper is organized as follows. Section 2 describes the basic architecture of doing filter and delta learning jointly with CNN training. The experimental setup and CNN architecture is discussed in Section 3. Section 4 presents experiments with filter and delta learning, while Section 5 discusses incorporating speaker adaption into the learning framework. Finally, Section 6 concludes the paper and discusses future work.

2. FILTER AND DELTA LEARNING

Convolutional neural networks (CNN) are commonly trained with log-mel filterbank features, as well as the delta and double-delta of

978-1-4799-2893-4/14/$31.00 ©2014 IEEE

these features [6]. While the process of generating these features is often separate from the CNN training process, both the filter and delta learning stages can be seen as different layers within a neural network, and can be learned jointly with the rest of the CNN.

The proposed model for filter and delta learning with a CNN framework is shown in Figure 1. Note that the input into the filter learning stage is the normalized log of the power spectrum, which is then taken to be positive using the exponent. Applying the normalization before the filter learning was found to be beneficial in [8, 16]. Below, we explain filter and delta learning in more detail.

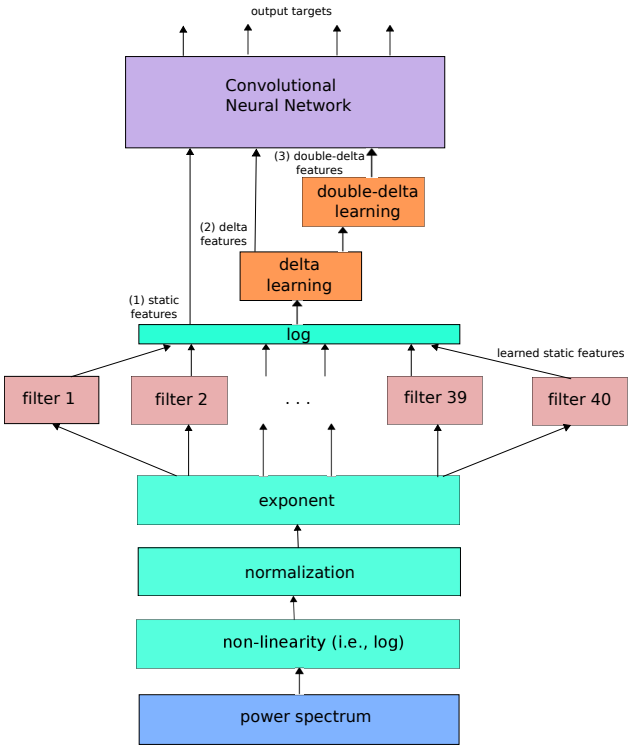

Fig. 1. Filterbank and Delta Learning Modules With a CNN

2.1. Feature Generation

To describe filter bank and delta learning more mathematically, first denote $\mathbf{f}$ as the input power spectral feature. Furthermore, denote $\exp(\mathbf{W}_i)$ as the weights for filter bank i, which span over a small local frequency region of the power spectrum. Here $\exp(\mathbf{W}_i)$ denotes an element-wise operation. The individual elements j of weight vector for filterbank i are denoted as $\exp(W_{i,j}) \in \exp(\mathbf{W}_i)$. The exponent operation ensures that the filterbank weights are positive. In addition, $\mathbf{f}_i \in \mathbf{f}$ are the power spectral components which correspond to filter bank i, and $f_{i,j} \in \mathbf{f}_i$, are the individual frequency components j that span over filter bank region i.

Following the filter-bank learning idea presented in [8], as shown in Equation 1, we take the log of the power spectrum $f_{i,j} \in \mathbf{f}$. Then, as shown by Equation 2, the features are normalized using mean and variance parameters $\{\mu_{i,j}, \sigma_{i,j}\}$ to get $n_{i,j}$. After the normalization, an exponent is applied to $n_{i,j}$ in Equation 3, to ensure that the input features into the filter bank, $e_{i,j}$, are positive. The normalized features $e_{i,j} \in e_i$ are then passed through the filter bank i, and the log is taken, to produce output m_i, given by Equation 4.

$$l_{i,j} = \log(f_{i.j}) \tag{1}$$

$$n_{i,j} = \frac{l_{i,j} - \mu_{i,j}}{\sigma_{i,j}} \tag{2}$$

$$e_{i,j} = \exp(n_{i,j}) \tag{3}$$

$$m_i = \log\left(\exp(\mathbf{W}_i^T)\mathbf{e}_i\right) = \log\left(\sum_j \exp(W_{i,j})e_{i,j}\right) \tag{4}$$

Given the filterbank features m_i, we then compute a time derivative delta of this feature. A common equation for computing delta features [17] is shown in Equation 5 , where α_t are the delta coefficients assumed to be integers. A popular choice for α values is $\alpha_1 = 1$ and $\alpha_2 = 2$. Notice that this equation assumes that the coefficients α_t and $-\alpha_t$ are applied to symmetric points $m_{i+\alpha_t}$ and $m_{i+\alpha_t}$. It also assumes that no scaling is applied to the current frame m_i when computing m_i^d, and that this delta coefficient is zero.

$$m_i^d, \texttt{htk} = \frac{\sum_{t=1}^{N} \alpha_t (m_{i+\alpha_t} - m_{i-\alpha_t})}{2\sum_{t=1}^{N} \alpha_t^2} \tag{5}$$

In this paper, we give more freedom to delta coefficients, not requiring them to be symmetric around zero and also allowing for a non-zero delta coefficient at time $t = 0$. Our proposed method for computing deltas is given by Equation 6.

$$m_i^d = \frac{\sum_{t=-N}^{N} \alpha_t m_{i+t}}{\sum_{t=-N}^{N} \alpha_t^2} \tag{6}$$

Similarly, the double-delta of the filterbank feature, is computed by taking the time derivative of the delta filterbank feature m_i^d. Our proposed equation for computing double-delta is given by Equation 7, where β_t are the double-delta coefficients.

$$m_i^{dd} = \frac{\sum_{t=-M}^{M} \beta_t m_{i+t}^d}{\sum_{t=-M}^{M} \beta_t^2} \tag{7}$$

After computing filterbank and delta features, namely m_i, m_i^d and m_i^{dd}, these features are then passed as input to the CNN, as indicated by streams (1), (2), and (3) in Figure 1 .

2.2. Delta Learning

The goal of backpropagation is to learn a set of weights that optimize some objective function $\mathcal{L}$. Typically, these weights are learned through stochastic gradient descent, by taking the derivative of the objective function with respect to the weights, and then updating the weights. For example, using stochastic gradient descent optimization, the weight update for the double-delta coefficient β_t is given by Equation 8, where γ is the learning rate.

$$\beta_t = \beta_t - \gamma \frac{\partial \mathcal{L}}{\partial \beta_t} \tag{8}$$

The derivative of the objective function w.r.t the weights can be easily calculated by back propagating error gradients from previous layers. Specifically, if m_i^{dd} in Equation 7 is the output after computing the double-delta features, then using the multivariate chain rule, the derivative of the objective function with respect to coefficient β_t can be written as Equation 9. Here we assume that the term $\frac{\partial \mathcal{L}}{\partial m_i^{dd}}$ is computed using the standard back propagation equations [18].

$$\frac{\partial \mathcal{L}}{\partial \beta_t} = \frac{\partial \mathcal{L}}{\partial m_i^{dd}} \frac{\partial m_i^{dd}}{\partial \beta_t} \tag{9}$$

The update equation for the delta coefficient α is a bit more complicated, as the delta feature m_i^d is computed by back propagating error gradients from both the CNN and double-delta layer, as shown streams (2) and (3) in Figure 1.

$$\frac{\partial \mathcal{L}}{\partial \alpha_t} = \frac{\partial \mathcal{L}}{\partial m_i^d}\frac{\partial m_i^d}{\partial \alpha_t} + \frac{\partial \mathcal{L}}{\partial m_i^{dd}}\frac{\partial m_i^{dd}}{\partial m_i^d}\frac{\partial m_i^d}{\partial \alpha_t} \tag{10}$$

2.3. Filter Learning

After the double-delta and delta coefficients are updated, the last step is to update the filter learning weights. The derivative of the objective function given the filter weights for component j in filter bank i, denoted as weight $W_{i,j}$ is shown in Equation 11. Notice the derivative includes a back propagation term from both the CNN and delta-layer, as shown by streams (1) and (2) in Figure 1.

$$\frac{\partial \mathcal{L}}{\partial W_{i,j}} = \frac{\partial \mathcal{L}}{\partial m_i}\frac{\partial m_i}{\partial W_{i,j}} + \frac{\partial \mathcal{L}}{\partial m_i^d}\frac{\partial m_i^d}{\partial m_i}\frac{\partial m_i}{\partial W_{i,j}} \tag{11}$$

Equations 9, 10 and 11 demonstrate how both the filter bank and delta computations can be learned jointly with the rest of a CNN.

3. EXPERIMENTS

Experiments are conducted on a 50-hour English Broadcast News (BN) task [15]. The acoustic models are trained on 50 hours of data from the 1996 and 1997 English Broadcast News Speech Corpora. Results are reported on the EARS `dev04f` set.

The baseline CNN system is trained with 40 dimensional log mel-filter features, along with the delta and double-deltas, which are per-speaker mean-and-variance normalized, rather than the speaker-independent globally normalized filter learning system proposed in [8]. The architecture and training recipe of the CNN is similar to [6], which was found to be optimal for BN. Specifically, the CNN has 2 full weight sharing convolutional layers with 256 hidden units, and 3 fully connected layers with 1,024 hidden units per layer. The final softmax-layer consists of 512 output targets. All CNNs are trained with cross-entropy, and results are reported in a hybrid setup.

4. RESULTS WITH DELTA LEARNING

In this section, we present experiments and results with various modifications to the filter and delta learning idea presented in Section 2.

4.1. Delta Learning

We explore learning delta and double-delta coefficients in Equations 6 and 7. First, we investigate the optimal delta size for both static and learned deltas. A delta size of 3, for example, corresponds to fixed delta coefficients of [-1, 0, 1], while a delta size of 5 corresponds to fixed delta coefficients of [-2, -1, 0, 1, 2]. Note that these fixed deltas are used as an initialization in delta learning.

Table 1 shows the WER for different delta sizes. The baseline log-mel + delta (d) + double-delta (dd) system is a 19.5%, with a delta size of 3. If filter learning is applied, and deltas are computed on the output of the filter layer, but not learned, the WER is at 19.1%, with a delta size of 3. Learning the filter and delta coefficients with the strategy outlined in Section 2, drops the WER to 18.8%. Note that with delta learning, the optimal size is 5, and a bit more freedom can be given to the deltas. Overall, filter and delta leaning offers a 4% relative improvement in WER over the baseline log-mel+d+dd.

Method	Delta Size	WER
Log-mel + fixed d + dd	3	19.5
Log-mel + fixed d + dd	5	19.6
Filter and delta learning	3	19.0
Filter and delta learning	5	**18.8**
Filter and delta learning	7	19.0

Table 1. WER, Filter and Delta Learning

A closer look at the learned d+dd coefficients is given in Table 2, compared to the initial d and dd starting point of $[-0.2, -0.1, 0, 0.1, 0.2]$. First, we see that learned deltas which are symmetric about zero, do not sum to zero, which is very different than the hand-crafted delta filters. Second, the 0^{th} delta coefficient is found to be non-zero, again different than hand-crafted delta filters. Overall, this leads to a much different shape for the delta and double-delta learned filters compared to hand-crafted filters. To understand if the non-zero delta sum and 0^{th} coefficient are beneficial, in the next section we explore various regularization techniques to force the learned deltas to behave more like hand-crafted ones.

Index	$t=-2$	$t=-1$	$t=0$	$t=1$	$t=2$
α_t	0.155	-0.003	-0.089	0.002	0.153
β_t	-0.454	-0.121	0.011	0.130	0.481

Table 2. Learned Delta and Double-Delta Coefficients

4.2. Delta Learning Regularization

4.2.1. Total Delta Sum = 0

The first regularization we explore is to ensure that the sum of all delta coefficients is zero. The new loss function $\mathcal{L}$ is defined as the sum of the unpenalized $\mathcal{L}_b$ plus a penalty term which tries to drive the sum of the deltas to zero, as shown in Equation 12. Here λ is a constant which weights the penalty term. The same regularization is also applied to the double-delta filters.

$$\mathcal{L} = \mathcal{L}_b + \lambda \left(\sum_{t=-N}^{N} \alpha_t \right)^2 \tag{12}$$

4.2.2. Symmetric Delta Only Sum = 0

The second regularization we explore is to ensure that symmetric delta terms, in other words $\alpha_- t$ and α_t, sum to zero. The new loss in Equation 13 incudes a penalty term which tries to drive the sum of symmetric deltas to zero, as shown in Equation 13.

$$\mathcal{L} = \mathcal{L}_b + \lambda \sum_{t=1}^{N} (\alpha_{t-N} + \alpha_{t+N})^2 \tag{13}$$

4.2.3. Enforcing 0^{th} Delta = 0

Finally, the third regularization explored is to force the 0^{th} delta coefficient to go to zero after updating the delta weights.

4.2.4. Results

Results with different delta regularizations are shown in Table 3. We can see that all of the regularization techniques degrade the WER.

This helps justify our results that an appropriate delta and double-delta need not have coefficients symmetric about zero sum to zero, and also that the 0^{th} delta coefficient need not be zero.

Filter and delta learning, No Regularization	**18.8**
Filter and delta learning, Total Sum = 0	19.0
Filter and delta learning, Symmetric Sum = 0	19.0
Filter and delta learning, 0^{th} delta = 0	19.2

Table 3. WER, Delta Regularization

4.3. Delta Learning Per Dimension

Another drawback of static deltas is that the same coefficient α_t is applied to all dimensions of the feature vector $m_i \in \Re^N$. There is no reason to believe that each feature dimension should have exactly the same delta. For example, low-frequency regions have different time-dynamic information than high-frequency regions, and it seems plausible that speech is more suited to having a different delta per dimension. Table 4 shows that delta learning per dimension offers a small improvement over learning one delta for all dimensions. Overall, we see that delta and filter learning offers a 5% relative improvement over using fixed filters and deltas.

Method	WER
Log-mel + d + dd	19.5
Filter learning + d + dd	19.1
Filter and delta learning	18.8
Filter and delta learning, per dimension	**18.6**

Table 4. WER, Filter and Delta Learning

5. SPEAKER ADAPTATION

In this section, we discuss experiments incorporating speaker adaptation into the filter and delta learning framework.

5.1. Incorporating VTLN

First, we explore incorporating VTLN into our model. Typically VTLN-warping is applied by constructing filter banks with different frequency warps, and choosing the optimal warped filter bank for each speaker via maximum-likelihood [19]. Since we have just one filterbank (as it is learned), this type of approach for VTLN will not work in our framework. Alternatively, VTLN can also be applied on the power spectra itself [9], and therefore just one filterbank can be used, which fits much better into our model framework.

Table 5 shows the results with vtln-warping for both the log-mel baseline, and proposed filter+delta learning. Note that for the log-mel system, warping is performed on the filterbank rather than the power-spectra due to implementation efficiency, though we have found no difference in performance by warping either. The table shows that even after vtln-warping, the filter+delta learning system continues to show gains over the baseline.

Method	WER
VTLN + log-mel + d + dd	18.7
VTLN + Filter + delta learning	**18.0**

Table 5. WER, Filter and Delta Learning with VTLN

5.2. Incorporating I-Vectors

Finally, we explore incorporating i-vectors into our model. I-vectors were first explored in [10] but for DNNs. Since CNNs require features which obey a frequency (and time) locality property, i-vectors cannot be concatenated with the full dimension of learned filter features, which have this locality property [8]. We compare two different methodologies to incorporate i-vectors into CNNs.

I-vectors can be incorporated into the convolutional layer by concatenating the feature with every localized frequency patch . For example, if the CNN sees a 9x9 time-frequency patch of localized features, we concatenate the 100-dimensional i-vectors into this feature so that the new filter size becomes 9x109. Every time the CNN shifts in frequency, the same i-vector is concatenated to the current set of localized features. This idea has been explored before when incorporating the non-localized energy feature into a CNN [5]. Alternatively, since we know i-vectors can be incorporated into fully connected DNN layers, we can use a joint CNN/DNN approach. Specifically, we can feed the i-vectors into one fully connected DNN layer, and then join this output into the first fully connected layer of the CNN. This joint CNN/DNN approach has been explored before for CNNs when combining different feature streams [20, 21].

Table 6 shows the WER for the two different methodologies. Just for simplicity to avoid the extra dimensions with d+dd, we compare the two different ideas of incorporating i-vectors with just filter learning. We see there is an improvement in WER when i-vectors are incorporated, but there is not a huge difference in final performance when incorporating i-vectors at the CNN or DNN level. Incorporating at the DNN layer is a bit faster, as we do not need to add i-vectors into the localized features for each CNN shift. For this reason, we use this approach for i-vectors in subsequent experiments.

Method	WER
Filter learning only	19.5
Filter Learning only + I-vector at CNN level	18.8
Filter Learning only + I-vector at DNN level	18.7

Table 6. WER, Incorporating I-vectors into CNNs

Table 7 compares the results of VTLN and i-vectors for both the baseline log-mel+d+dd and filter+delta learning systems. After including i-vectors, filter and delta learning still maintains a 3% relative improvement over the baseline, showing the value of this technique over a strong speaker-adapted, state-of-the-art CNN baseline.

Method	WER
VTLN log-mel + d + dd + i-vectors	17.8
VTLN + Filter + delta learning + i-vectors	**17.3**

Table 7. WER, Filter and Delta Learning with VTLN and I-Vectors

6. CONCLUSIONS

In this paper, we improved the filter learning idea proposed in [8] by incorporating delta learning into this framework. We also presented results on a strong baseline, after incorporating speaker-adaptation techniques such as VTLN and i-vectors. On a 50-hour BN task, the proposed filter and delta learning strategy has a WER of 18.6%, a 5% relative improvement over a baseline log-mel+d+dd CNN. After incorporating speaker adaptation, the filter and delta learning approach has a WER of 17.3%, still showing a 3% relative improvement over the speaker-adapted CNN baseline.

7. REFERENCES

[1] 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.

[2] D. Povey, D. Kanevsky, B. Kingsbury, B. Ramabhadran, G. Saon, and K. Visweswariah, "Boosted MMI for Model and Feature-Space Discriminative Training," in *Proc. ICASSP*, 2008, pp. 4057–4060.

[3] T. N. Sainath, B. Ramabhadran, M. Picheny, D. Nahamoo, and D. Kanevsky, "Exemplar-Based Sparse Representation Features: From TIMIT to LVCSR," 2011.

[4] A. Mohamed, G. Hinton, and G. Penn, "Understanding how Deep Belief Networks Perform Acoustic Modelling," in *ICASSP*, 2012.

[5] O. Abdel-Hamid, A. Mohamed, H. Jiang, and G. Penn, "Applying Convolutional Neural Network Concepts to Hybrid NN-HMM Model for Speech Recognition," in *Proc. ICASSP*, 2012.

[6] T. N. Sainath, A. Mohamed, B. Kingsbury, and B. Ramabhadran, "Deep Convolutional Neural Networks for LVCSR," in *Proc. ICASSP*, 2013.

[7] S. Davis and P. Mermelstein, "Comparison of Parametric Representations for Monosyllabic Word Recognition in Continuously Spoken Sentences ," *IEEE Transacations on Acoustics, Speech and Signal Processing*, vol. 28, no. 4, pp. 357 – 366, 1980.

[8] T. N. Sainath, B. Kingsbury, A. Mohamed, and B. Ramabhadran, "Learning Filter Banks Within a Deep Neural Network Framework," in *to appear in Proc. ASRU*, 2013.

[9] P. Zhan and A. Waibel, "Vocal Tract Length Normalization for Large Vocabulary Continuous Speech Recognition," Tech. Rep., CMU Computer Science Technical Reports, 1997.

[10] G. Saon, H. Soltau, M. Picheny, and D. Nahamoo, "Speaker Adaptation of Neural Network Acoustic Models Using I-Vectors," in *to appear in Proc. ASRU*, 2013.

[11] S. Vuuren and H. Hermansky, "Data-Driven Design of RASTA-like Filters," in *Proc. Eurospeech*, 1997.

[12] J. W. Hung and L. S. Lee, "Optimization of Temporal Filters for Constructing Robust Features in Speech Recognition," *IEEE Transactions on Audio, Speech and Language Processing*, 2006.

[13] A. Biem, E. Mcdermott, and S. Katagiri, "A Discriminative Filter Bank Model For Speech Recognition," in *Proc. ICASSP*, 1995.

[14] G. Hinton, L. Deng, D. Yu, G. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury, "Deep Neural Networks for Acoustic Modeling in Speech Recognition," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[15] B. Kingsbury, "Lattice-Based Optimization of Sequence Classification Criteria for Neural-Network Acoustic Modeling," in *Proc. ICASSP*, 2009.

[16] H. Hermansky and N. Morgan, "RASTA Processing of Speech," *IEEE Transactions on Speech and Audio Processing*, vol. 2, no. 4, pp. 578 – 589, 1994.

[17] S. J. Young, G. Evermann, M. J. F. Gales, T. Hain, D. Kershaw, X. Liu, G. Moore, J. Odell, D. Ollason, D. Povey, V. Valtchev, and P. C. Woodland, *The HTK Book (for HTK Version 3.4)*, University of Cambridge, 2006.

[18] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Neurocomputing: foundations of research*, pp. 696–699, 1988.

[19] L. Lee and R. C. Rose, "Speaker Normalization using Efficient Frequency Warping Procedures," in *Proc. ICASSP*, 1996.

[20] T. N. Sainath, B. Kingsbury, A. Mohamed, G. Dahl, G. Saon, H. Soltau, T. Beran, A. Y. Aravkin, and B. Ramabhadran, "Improvements to Deep Convolutional Neural Networks for LVCSR," in *Proc. ASRU*, 2013.

[21] H. Soltau, G. Saon, and T. N. Sainath, "Joint Training of Convolutional and Non-Convolutional Neural Networks," in *submitted to Proc. ICASSP*, 2014.