LEARNING FILTER BANKS WITHIN A DEEP NEURAL NETWORK FRAMEWORK

Tara N. Sainath[1]*, Brian Kingsbury*[1]*, Abdel-rahman Mohamed*[2]*, 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, bhuvana}@us.ibm.com, [2]asamir@cs.toronto.edu

ABSTRACT

Mel-filter banks are commonly used in speech recognition, as they are motivated from theory related to speech production and perception. While features derived from mel-filter banks are quite popular, we argue that this filter bank is not really an appropriate choice as it is not learned for the objective at hand, i.e. speech recognition. In this paper, we explore replacing the filter bank with a filter bank layer that is learned jointly with the rest of a deep neural network. Thus, the filter bank is learned to minimize cross-entropy, which is more closely tied to the speech recognition objective. 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 bank learning approach, compared to having a fixed set of filters.

1. INTRODUCTION

Designing appropriate feature representations for speech recognition has been an active area of research for many years. For example, in large vocabulary continuous speech recognition systems, huge gains in performance are observed by using speaker-adapted and discriminatively trained-features [1], learned via objective functions such as feature-space maximum likelihood linear regression (fMLLR), and feature-space boosted maximum mutual information (fBMMI). In addition, designing appropriate classifiers given these features has been another active area of research, where popular modeling approaches include Deep Neural Networks (DNNs) or Gaussian Mixture Models which have been discriminatively trained.

Oftentimes feature design is done separately from classifier design. This has a drawback that 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 [2]. In fact [3] showed that the lower layers of DNNs produce speaker-adapted features, while the upper layers of DNNs perform class-based discrimination. For years, speech researchers have been using separate modules for speaker adaption (i.e. fMLLR) and discriminative training (i.e. fBMMI) for GMM training. One reason we believe DNNs are more powerful than GMMs is that this feature extraction is done jointly with the classification, such that features are tuned to the classification task at hand, rather than separately before classification.

One problem with DNNs is that they are not explicitly designed to reduce translational frequency variance within speech signals, which can exist due to different speaking styles. While DNNs could remove variance with a large enough number of parameters or having a lot of data, this can often be infeasible. Alternatively, fMLLR transformations look to address the issue of translational variance by mapping speech from different speakers into a canonical space. Therefore, fMLLR features are used in conjunction with DNNs to give optimal performance [4]. Convolutional Neural Networks (CNNs) [5] are better feature extractors than DNNs, as they reduce translational variance with far fewer parameters compared to DNNs, jointly while doing class-based discrimination. Therefore, with very simple features, i.e. VTLN-warped log-mel filter bank features, [6] showed than CNNs offered a 4-12% relative improvement in WER over DNNs across a variety of different LVCSR tasks. This result indicates that giving the CNN very simple features, and having it learn appropriate feature extraction and discrimination via an objective function related to speech recognition, is much more powerful than providing a DNN hand-crafted features.

Yet, one of the drawbacks with current CNN work in speech is that the most commonly used features are 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. We have seen examples of this with Hidden Markov Models (HMM) in acoustic modeling. HMMs have remained the dominant acoustic modeling technique to date, despite their frame independence assumption which is absent in human speech processing.

The creation of log-mel features is done by passing a power spectrum through a mel-filter bank, followed by a non-linear log operation. This process can be modeled by a layer of a neural network, which has a linear weight multiplication (i.e. filter bank layer), followed by a non-linearity (i.e., log). In this work, with even simpler features (i.e., power spectral features), we explore learning the mel-filter bank layer jointly with a deep CNN. This ensures that the filter bank is learned for the task at hand.

Data-driven learning of filter banks has been explored in a variety of contexts. For example, [8] derived filter banks directly from phonetically labeled speech data using Linear Discriminant Analysis (LDA), though the authors argued the derived filter banks were not optimal as LDA expects the data to have a Gaussian distribution, which is not true for power spectra. In addition, [9] investigated using the Kullback-Leibler (KL) distance as the measure in the filter-bank design, though the filter was still designed independently of the acoustic model. Alternatively, [10] learned a discriminative filter bank model jointly with a classifier using a discriminative training criterion. However, their work looked at a relatively simple distance-based classifier. In this work, we explore filter bank learning given a powerful, state-of-the art deep CNN acoustic model [11] where filter bank learning is incorporated into the learning of CNN parameters. The benefit of using a neural network is that filter bank learning can be seen as an extra layer of the neural network, where filter bank

978-1-4799-2756-2/13/$31.00 ©2013 IEEE

parameters are updated along with the parameters in subsequent layers. To our knowledge, this is the first attempt at doing filter bank learning with deep neural networks.

Our filter bank learning experiments are performed on a 50-hr English Broadcast News (BN) task [12]. The baseline system, a state-of-the art deep CNN [6] trained on log-mel filter bank features, has a WER of 22.3%. We find that applying filter bank learning directly into the CNN, we get a modest improvement of 22.0%. By normalizing features before passing them to the filter bank, which has been shown to be very important for neural network training [13], a WER of 21.3% is attained. Finally, incorporating pooling into the filter bank layer, provides a WER of 21.1%, which gives a 5% relative reduction in WER compared to a strong CNN baseline.

The rest of this paper is organized as follows. Section 2 describes the basic architecture of doing filter bank learning jointly with CNN training. The experimental setup and CNN architecture is discussion in Section 3, while Section 4 presents different experiments with filter bank learning, including results and analysis. Finally, Section 5 concludes the paper and discusses future work.

2. FILTER BANK LEARNING

The process of generating log mel-filter bank features from the power spectrum, and then training a convolutional neural network is depicted in Figure 1. Specifically, mel features are generated by passing the power spectrum through a set of mel-filters, depicted as "filter 1-40" in Figure 1. 40 mel-filters is a common number of filters to use in speech recognition tasks [14]. Then applying a log-compression gives log-mel filter bank features. The features are then normalized before passing them as input to the CNN, as this is critical to neural network training [13].

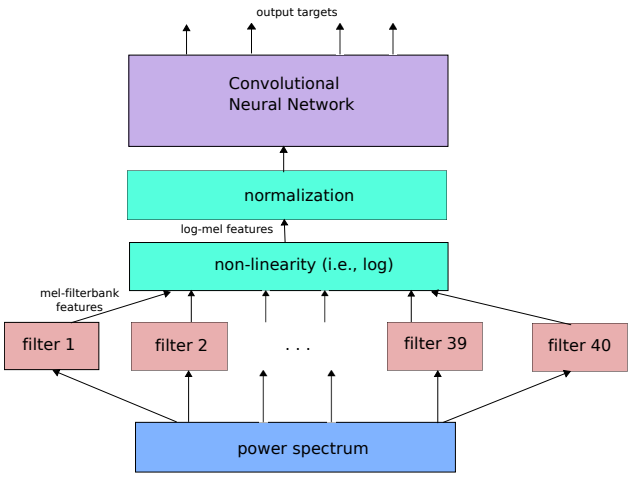

Fig. 1. Log Mel-filter bank Feature Generation as Input into a CNN

Typically, the feature-generation process is separate from the CNN training process. However, looking at the feature generation process in Figure 1, we can see that log-mel features are produced by multiplying power spectral features by a set of weights (i.e. filters), followed by a non-linearity (i.e., log). This is very similar to the behavior of one layer of a neural network. However, instead of having one set of weights which are used for all time and frequency components, a limited weight sharing idea is utilized [15], where weights are replicated at least once within a small localized frequency region. Because both the weight multiplication and non-linearity are differentiable functions, this means we should be able to "learn" the filters jointly with the rest of the neural network. The benefit of this approach is that filters are learned via a cross-entropy objective function and for the speech recognition objective at hand, rather than designed ahead of time.

To describe filter bank 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. The mel-filter bank output m_i for filter bank i is given by Equation 1.

$$m_i = \exp(\mathbf{W}_i^T)\mathbf{f}_i = \sum_j \exp(W_{i,j}) f_{i,j} \tag{1}$$

Taking the log of m_i gives the log-mel filter bank coefficient for filter bank i, namely

$$l_i = \log(m_i) \tag{2}$$

Finally, a global mean-variance normalization is applied to the log-mel features. This is given by Equation 3, where $\{\mu_i, \sigma_i\}$ define the mean and variance parameters for feature dimension i. These parameters are estimated on the training data ahead of time.

$$n_i = \frac{l_i - \mu_i}{\sigma_i} \tag{3}$$

The goal of back propagation 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, the weight update equation for component j in filter bank i, denoted as weight $W_{i,j}$ is shown in Equation 4. Note that we really do want the weight update for W_{ij} and not $\exp(W_{i,j})$, as the exponent can be thought of as another operation to ensure positive weights.

$$W_{i,j} = W_{i,j} - \eta \frac{\partial \mathcal{L}}{\partial W_{i,j}} \tag{4}$$

The derivative of the objective function given weights can be easily calculated by back propagating error gradients from previous layers. Specifically, if n_i in Equation 3 is the output of the filter bank layer, then using the multivariate chain rule, the derivative of the objective function with respect to weight W_{ij} can be written as Equation 5. Here we assume that the term $\frac{\partial \mathcal{L}}{\partial n_i}$ is computed using the standard back propagation equations for neural networks [16].

$$\frac{\partial \mathcal{L}}{\partial W_{i,j}} = \frac{\partial \mathcal{L}}{\partial n_i} \frac{\partial n_i}{\partial W_{i,j}} \tag{5}$$

Given the definitions for m_i and l_i in Equations 1 and 2 respectively, we can further expand Equation 5 as follows.

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial W_{i,j}} &= \frac{\partial \mathcal{L}}{\partial n_i} \frac{\partial n_i}{\partial l_i} \frac{\partial l_i}{\partial m_i} \frac{\partial m_i}{\partial W_{i,j}} \\ &= \frac{\partial \mathcal{L}}{\partial n_i} \frac{1}{\sigma_i} \frac{1}{m_i} \exp(W_{i,j}) f_{i,j} \end{aligned} \tag{6}$$

Equation 6 demonstrates how gradients from the neural network stage can be back propagated into the filter bank learning stage.

Rather than having hand-crafted filter banks which are not necessarily tuned to the objective at hand, incorporating filter bank learning as an extra stage in the neural network allows the weights to be updated according to objective function $\mathcal{L}$. In the next few sections, we will discuss results with filter bank learning.

3. EXPERIMENTS

Experiments are conducted on a 50-hour English Broadcast News (BN) task [12]. 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 speaker-independent CNN system is trained with 40 dimensional log mel-filter bank coefficients, which are global mean-and-variance normalized. The architecture 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.

Following a recipe similar to [4], during fine-tuning, after one pass through the data, loss is measured on a held-out set and the learning rate is reduced by a factor of 2 if the held-out loss has not improved sufficiently over the previous iteration. Training stops after we have reduced the step size 5 times. All CNNs are trained with cross-entropy, and results are reported in a hybrid setup.

4. RESULTS

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

4.1. Filter Learning

First, we explore a direct application of filter learning based on Figure 1. Specifically, the "learned" filter banks are applied directly on the magnitude of the power spectral coefficients, and then a log is taken. A global mean/variance normalization is applied after the log, and these normalized "learned" log-mel features as passed as input into the CNN. Since it is critical to initialize weights in the lower layers compared to higher layers [11], the filter bank layer is initialized to be the mel-filter bank, rather than starting from random initialization.

Results with this proposed method of filter learning are shown in Table 1. The learning provides a modest 0.3% improvement in WER over the baseline system with a WER of 22.3%. In the next section, we will discuss what assumptions are not appropriate for direct filter bank learning, and why improvements are small.

Method	WER
Baseline	22.3
Filter Learning	**22.0**

Table 1. Direct Application of Filter bank Learning

4.2. Feature Normalization

Feature normalization is critical in neural network training to achieve good convergence in training. As discussed in [13], when features are not centered around zero, network updates will be biased towards a particular direction and this will slow down learning. The paper even discusses an extreme case when all inputs into a layer are positive. This causes all weights to increase or decrease together for a given input, and thus the weight vector can only change direction by zigzagging which is extremely slow. This extreme case discussed in [13] is exactly the problem with the direct application of filter bank learning for neural networks, as all the inputs into the filter bank layer are from the magnitude of the power spectrum and are thus positive. In this section, we discuss how to normalize power spectral features for filter learning.

4.2.1. Algorithm

One could directly normalize the power spectral features, and then pass them to a filter bank. However, because the features would be negative, taking the log is not possible. The non-linear log operation is also critical in compressing and spreading out features so they can be better used for classifiers.

Given the constraints that we want to normalize the power spectral features, but want the output of the filter bank stage to be positive, we explore an idea that is similar to that done in RASTA processing [17]. First, as shown in Equation 7, we take the log of the power spectrum $f_{i,j} \in \mathbf{f}$, where again i denotes the filter bank and j is the individual frequency component which spans over filter bank i. Then, as shown by Equation 8, the features are normalized to get $l_{i,j}$, which is done completely on the power spectral dimension now. After the normalization is done, an exponent is applied to $l_{i,j}$ in Equation 9, to ensure that the input features into the filter bank, $e_{i,j}$, are positive. Because of the exponent taken after normalization, the log is taken before normalization in Equation 8, to ensure that the new "normalized" features are roughly in the same range as the "unnormalized" features passed into the filter bank in Section 4.1. The normalized features $e_{i,j} \in e_i$ are then passed through the filter bank i to produce output m_i, given by Equation 10, which is then passed as input into the CNN. Equations 7-10 ensure that a normalized and positive feature is passed to the filter bank stage.

$$l_{i,j} = \log(f_{i.j}) \tag{7}$$

$$n_{i,j} = \frac{l_{i,j} - \mu_{i,j}}{\sigma_{i,j}} \tag{8}$$

$$e_{i,j} = \exp(n_{i,j}) \tag{9}$$

$$m_i = \exp(\mathbf{W}_i^T)\mathbf{e}_i = \sum_j \exp(W_{i,j})e_{i,j} \tag{10}$$

With the addition of power spectrum-normalization, the back propagation equations change slightly as well. Given objective function $\mathcal{L}$ and the error gradient from the previous layer, $\frac{\partial \mathcal{L}}{\partial m_i}$, the weight update is now given by Equation 11.

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial W_{i,j}} &= \frac{\partial \mathcal{L}}{\partial m_i}\frac{\partial m_i}{\partial W_{i,j}} \\ &= \frac{\partial \mathcal{L}}{\partial m_i}\exp(W_{i,j})e_{i,j}\end{aligned} \tag{11}$$

4.2.2. Results and Analysis

Results with filter bank learning using normalized input features are shown in Table 2. By normalizing the input features, we can achieve a 0.7% absolute improvement over un-normalized features, and a 1%

Method	WER
Baseline	22.3
Filter Learning	22.0
Normalization With Filter Learning	**21.3**

Table 2. Normalized Features for filter bank Learning

absolute improvement in WER over the baseline system. This points to the importance of feature normalization in neural networks.

A visual comparison of the 40 mel-filter banks and learned filter banks is shown in Figure 2. The power spectral frequency components the filters span over are also shown. For visual purposes, the filters have been normalized so they can be compared on the same plot. Notice that for low-frequency regions, both the learned and mel-filters look similar. In the mid-frequency regions, the learned filters seem to have multiple peaks instead of one, indicating they are picking up multiple important critical frequencies rather than just one like the mel. In the high-frequency regions, the filters appear to be high-pass filters compared to the band-pass mel filters.

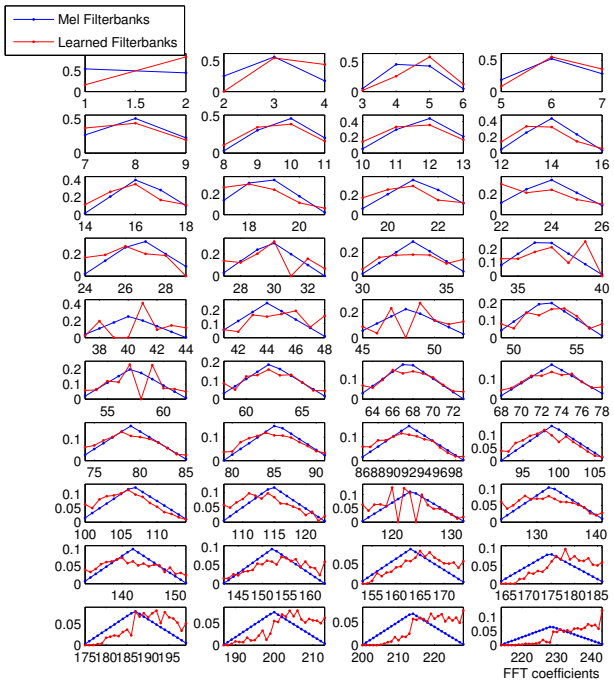

Fig. 2. Mel-Fiterbank vs. Learned Filter banks

Figure 3 also plots the log-mel features and learned filter bank features for a specific utterance. The learned features appear similar to the log-mel features, indicating the learned features are meaningful. In the high frequency regions, the features are not as smooth because of the high-pass filters in Figure 2.

Notice that using a limited weight sharing filter bank layer with one output seems to preserve locality in frequency and allows us to feed the outputs from this layer into a convolutional layer with full weight sharing. Previous work with limited weight sharing used multiple outputs per layer, which did not allow for multiple convolutional layers with full weight sharing as the locality in frequency was not preserved [15], [18].

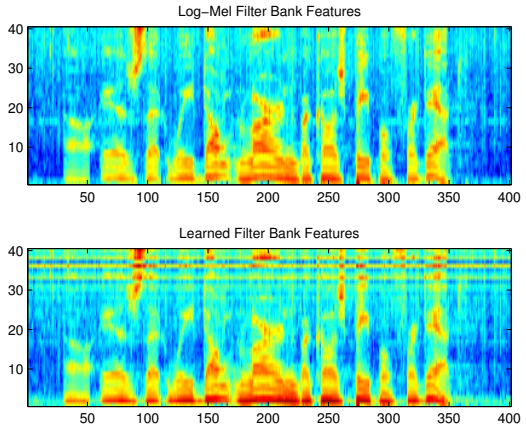

Fig. 3. Log-Mel Features and Learned Features

4.3. Pooling

Because the filter bank layer weights span over small frequency regions and there are multiple weights, this layer can be seen as a convolutional layer with limited weight sharing [15]. Pooling is an important concept in convolutional layers which helps to reduce variance in the input features. A pooling strategy which varies the pooling size for each filter bank layer makes sense, as each power spectrum band is linearly spaced in frequency and contains an unequal amount of information, as reflected by mel-fiters having constant Q spacing apart.

[18] introduces a varied pooling strategy for each weight that spans across a localized frequency region (i.e., filter bank), with the acoustic knowledge that it makes sense to pool more in higher frequency regions and less in lower frequency regions. This strategy has been coined "heterogeneous pooling". One of the problems with heterogeneous pooling is that the filters are generally shifted by a fixed amount (i.e., one) along the frequency axis during pooling.

Alternatively, we propose a vocal-tract-length-normalization (VTLN)-inspired pooling strategy. Frequency pooling is performed to reduce formant translations due to different speaking styles, vocal tract lengths, gender, accents, etc. VTLN is another popular technique to reduce frequency transformations of the input signal. VTLN tries to find the optimal frequency warp factor for each speaker, and map the speech back to a canonical space. Each warp factor generates a shifted and scaled version of the mel filter banks in each frequency region (i.e., 1-40). The warp factor for a given speaker is generally selected via maximum likelihood [19]. For LVCSR tasks, we typically use about 21 different warp factors [14].

In this paper, we explore a VTLN-inspired pooling strategy in an unsupervised manner, and use just one filter per frequency region. For each region, we compute the unique locations of the center frequency of the VTLN filters in this region, but ignore the differences in shape of the VTLN filters. During pooling, this corresponds to having filters that are shifted in frequency at center locations defined by the VTLN filters, rather than spaced by a fixed amount as in heterogeneous pooling. However, there is no "optimal" warp factor and corresponding filter bank that is selected for each speaker as is done in normal VTLN, just one filter is used per region.

Results with heterogeneous and VTLN-inspired pooling for the filter bank layer are shown in Table 3. Note that pooling is also performed in the convolutional layers of the CNN, though this is just fixed-size pooling as in [6]. All pooling layers use a max-pooling

strategy. For heterogenous pooling, we tuned the pooling size P in each region, using a linear relationship that pooling in the lower frequency regions should be small, and pooling in the upper layers should be high. We found using a pooling distribution between $P = 1 - 8$ was optimal, as shown in Table 3. However, heterogeneous pooling does not seem to improve over the baseline.

For VTLN-style pooling, we also tuned the pooling size P. This was done by taking a percentage of the total unique center frequencies for each region. In lower frequencies, we find few unique center frequencies (i.e., between 1-3), while in the higher frequency regions, there are 21 unique center frequencies. Table 3 shows the WER for different % of unique warp factors selected per region, along with the variance of the actual pooling size. Notice that by using 15% of the total unique center frequencies, which corresponds to having a pooling size between 1-3, we can achieve a WER of 21.1%, a 0.2% reduction in WER compared to no pooling.

Method	Pooling Size (P)	WER
Baseline	none	21.3
Heterogeneous pooling	1-8	21.3
VTLN-inspired pooling	1-3 (15%)	**21.1**
VTLN-inspired pooling	1-5 (25%)	21.5
VTLN-inspired pooling	1-7 (35%)	22.3
VTLN-inspired pooling	1-16 (75%)	23.0

Table 3. WER with Different Pooling Strategies

A further analysis into the shape of these filters is shown in Figure 4. The biggest thing to notice is that in the higher-frequency bands where the pooling size is increased, the filters learned from pooling appear to having multiple peaks and are less smooth compared to the no pooling filters. One hypothesis is that pooling is mimicking having the peaks of the multiple VTLN filters, though now in one filter. The real multiple-VTLN filters however span a much greater region in frequency, compared to the learned filter. This inspires us to increase the filter bank size to see if any further improvements are possible.

4.4. Increasing Filter Bank Size

In this section, we explore giving more freedom to each filter. With increased filter sizes, this indicates that the mel-filters cannot be used as an initial filter. Thus, we explore using a Gaussian filter as the initial filter. Before increasing filter size, we first check if there is a change in WER by using a Gaussian filter as the initial filter. The Gaussian filters peaks at the same point the mel-filters do, but tapers off in a Gaussian manner rather than a triangular manner like the mel-filters. Table 4 shows there is no change in WER by using a Gaussian initialization. This justifies using this type of initialization as we increase filter size.

Filter Initalization	Filter Size	WER
Mel-Filter	Baseline Size	21.3
Gaussian	Baseline Size	21.3

Table 4. WER With Different Filter Initalizations

We explored increasing filter size for VTLN-style pooling. Results are shown in Table 5. Notice that for VTLN pooling, increasing the filter size does not help, and keeping the filter-size the same as the mel (i.e. Filter Size Multiple 1.0) seems to be the best. One hypothesis is that perhaps when filter size increases, there is more overlap in

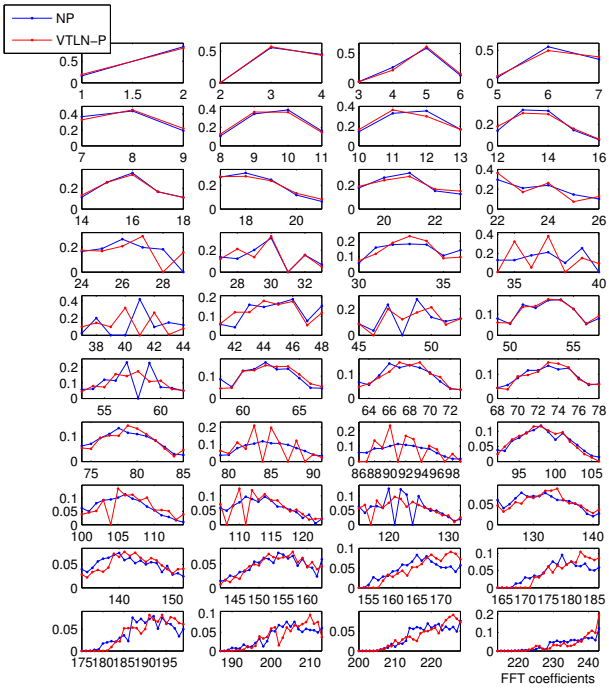

Fig. 4. Learned Filters with No Pooling and VTLN-style pooling

frequency between the different filters. Therefore, filters might co-adapt together to explain the same frequency regions, while placing less importance on other frequency components. While dropout [20] is a methodology to prevent co-adaptation, it is effective only where there are a large number of hidden units, which is not the case here.

Pooling Type	Filter Size Multiple	WER
VTLN-inspired	1.00	21.1
VTLN-inspired	1.25	21.8
VTLN-inspired	1.50	21.8

Table 5. WER With Different Filter Size

4.5. Regularization

Figures 2 and 4 indicate that the learned filters are not smooth and have multiple peaks. It is not clear if this is a good behavior, meaning we are picking up multiple important signals in each band, or this behavior is bad because we give too much freedom to the filter bank weights. In this section, we attempt to answer this question by exploring if enforcing some smoothing between neighboring weights might help. Neighboring weight smoothing for neural networks was first introduced in [21]. Given weights $W_{i,j} \in \mathbf{W}_i$ for filter bank i, unsmoothness is measured as follows:

$$\sum_{j=2}^{|\mathbf{W}_i|} [W_{i,j} - W_{i,j-1}]^2 \quad (12)$$

Given this, the new loss function $\mathcal{L}$ is defined as the sum of the old loss $\mathcal{L}_b$ plus a second term which measures unsmoothness across all filter banks, as shown in Equation 13. Here λ is a constant which

weights the loss due to unsmoothness.

$$\mathcal{L} = \mathcal{L}_b + \lambda \sum_{i=1}^{40} \sum_{j=2}^{|\mathbf{W}_i|} [W_{i,j} - W_{i,j-1}]^2 \quad (13)$$

Finally, the new weight update given this reguarlization is shown in Equation 14. Boundary conditions for the weights when $j = 0$ and $j = |\mathbf{W}_i|$ are discussed in more detail in [21].

$$\frac{\partial \mathcal{L}}{\partial W_{i,j}} = \frac{\partial \mathcal{L}_b}{\partial W_{i,j}} + 2\lambda[2W_{i,j} - W_{i,j-1} - W_{i,j+1}] \quad (14)$$

Table 6 shows the WER with and without neighboring weight smoothing. Note that λ is tuned on a held-out set. For a non-zero λ, we find that smoothing causes an increase in WER. This indicates that multiple peaks in each filter is good.

Method	WER
No smoothing	21.1
Smoothing	21.3

Table 6. WER With Neighboring Weight Smoothing

4.6. Exploration of Negative Weights and Non-Linearities

Having a log as a non-linearity requires that both input features and corresponding weights are positive. In this section, we explore changing the non-linearity, which concurrently removes the constraint that the weights must be positive. Specifically, we explore using a sigmoid non-linearity, which is a popular non-linearity used in neural networks. We also investigate using a cube-root, which is inspired by its use in Perceptual Linear Predictive (PLP) features. Since these non-linearities are centered around zero, we allow the weights to have negative values, which removes the exponent in Equation 10.

Table 7 shows the results for different non-linearities. It appears that using the log non-linearity with positive weights is the best. This experimentally justifies that a filter bank which has a logarithmic non-linearity that corresponds to human perception of loudness, is a sensible choice.

Non-Linearity	Weight Constraints	WER
Log	Positive	21.3
Sigmoid	None	23.5
Cube Root	None	24.3

Table 7. WER With Different Non-Linearities

5. CONCLUSIONS

In this paper, we explored adding a filter bank layer as an extra layer into a CNN. The filter bank is learned jointly with the rest of the network parameters to optimize the cross-entropy objective function. Thus, instead of having a perceptually motivated filter bank which is not necessarily correlated to the speech recognition objective, the filter is learned for the task at hand. However, we do find that using a non-linear perceptually motivated log function is appropriate. We introduce a novel idea of normalizing filter-bank features while still ensuring they are positive so that the logarithm non-linearity can be applied. Second, we explore a VTLN-inspired pooling strategy. On a 50-hour BN task, the proposed filter-learning strategy has a WER of 21.1%, a 5% relative improvement over a baseline CNN with hand-crated mel-filter bank features with a WER of 22.3%.

6. REFERENCES

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

[2] Y. LeCun, "Learning Invariant Feature Hierarchies," in *European Conference on Computer Vision (ECCV)*. 2012, vol. 7583 of *Lecture Notes in Computer Science*, pp. 496–505, Springer.

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

[4] 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 *Proc. ASRU*, 2011.

[5] Y. LeCun and Y. Bengio, "Convolutional Networks for Images, Speech, and Time-series," in *The Handbook of Brain Theory and Neural Networks*. MIT Press, 1995.

[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] L. Burget and H. Heřmanskỳ, "Data Driven Design of Filter Bank for Speech Recognition," in *Text, Speech and Dialogue*. Springer, 2001, pp. 299–304.

[9] Y. Suh and H. Kim, "Data-Driven Filter-Bank-based Feature Extraction for Speech Recognition," in *Proc. SPECOM*, 2004.

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

[11] 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.

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

[13] Y. LeCun, L. Bottou, G. Orr, and K. Muller, "Efficient Backprop," in *Neural Networks: Tricks of the Trrade*, G. Orr and Muller K., Eds. 1998, Springer.

[14] H. Soltau, G. Saon, and B. Kingsbury, "The IBM Attila Speech Recognition Toolkit," in *Proc. SLT*, 2010.

[15] 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.

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

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

[18] L. Deng, O. Abdel-Hamid, and D. Yu, "A Deep Convolutional Neural Network using Heterogeneous Pooling for Trading Acoustic Invariance with Phonetic Confusion," in *Proc. ICASSP*, 2013.

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

[20] G.E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Improving Neural Networks by Preventing Co-Adaptation of Feature Detectors," *The Computing Research Repository (CoRR)*, vol. 1207.0580, 2012.

[21] J. Jean and Jin Wang, "Weight Smoothing to Improve Network Generalization," *Neural Networks, IEEE Transactions on*, vol. 5, no. 5, pp. 752–763, 1994.