DEEP BELIEF NETWORKS USING DISCRIMINATIVE FEATURES FOR PHONE RECOGNITION

Abdel-rahman Mohamed[1], *Tara N. Sainath*[2], *George Dahl*[1]
Bhuvana Ramabhadran[2], *Geoffrey E. Hinton*[1] *and Michael A. Picheny*[2]

[1]Department of Computer Science, University of Toronto
[2]IBM T. J. Watson Research Center, Yorktown Heights, NY 10598
asamir@cs.toronto.edu[1], tsainath@us.ibm.com[2], gdahl@cs.toronto.edu[1],
bhuvana@us.ibm.com[2], hinton@cs.toronto.edu[1], picheny@us.ibm.com[2]

ABSTRACT

Deep Belief Networks (DBNs) are multi-layer generative models. They can be trained to model windows of coefficients extracted from speech and they discover multiple layers of features that capture the higher-order statistical structure of the data. These features can be used to initialize the hidden units of a feed-forward neural network that is then trained to predict the HMM state for the central frame of the window. Initializing with features that are good at generating speech makes the neural network perform much better than initializing with random weights. DBNs have already been used successfully for phone recognition with input coefficients that are MFCCs or filterbank outputs [1, 2]. In this paper, we demonstrate that they work even better when their inputs are speaker adaptive, discriminative features. On the standard TIMIT corpus, they give phone error rates of 19.6% using monophone HMMs and a bigram language model and 19.4% using monophone HMMs and a trigram language model.

Index Terms— Discriminative feature transformation, Deep belief networks, Phone recognition.

1. INTRODUCTION

For the past three decades, most automatic speech recognition systems have used Hidden Markov Models (HMMs) to model the sequential structure of speech signals, with each HMM state using a mixture of Gaussians (GMMs) to model a spectral representation of the sound wave. Early systems used the maximum likelihood (ML) criterion to train GMMs to fit the training data. Significant word error rate reductions were achieved by fine-tuning the ML trained GMMs using discriminative training criteria [3, 4]. Speaker adaptation techniques which map different speakers' characteristics to a canonical speaker, applied to the GMMs, provided complementary improvements [5]. Both speaker adaptation and discriminative training can also be applied in the feature domain to create a better set of features that are speaker adaptive and discriminative by nature [5, 4]. Most of these techniques have been extensively used in LVCSR tasks while in [6] they were applied to the phone recognition task in the TIMIT database.

Some researchers have proposed feed-forward neural networks (NN) [7, 8] as an alternative to GMMs because:

- Their estimation of the posterior probabilities of HMM states does not require detailed assumptions about the data distribution.
- They allow an easy way of combining diverse features, including both discrete and continuous features.
- They use far more of the data to constrain each parameter because the output on each training case is sensitive to a large fraction of the weights.

Recently, a significantly better way of training feed-forward neural networks has been discovered [9]. Instead of initializing the weights at random values they are initialized by first fitting a multilayer generative model, called a deep belief net (DBN), to the input data. The layers of features found by the DBN have not yet used any of the information in the labels. A subsequent stage of discriminative fine-tuning that only changes the features slightly produces a neural net that trains faster and overfits less than the same net with randomly initialized weights.

DBNs have already been successfully used for phone recognition [1, 2] using MFCCs or log spectrograms as their inputs. This paper shows that DBNs can achieve even better performance by using speaker adaptive and discriminative features as inputs to the DBN.

2. FEATURE DESCRIPTION

In this section, we describe various features used in the paper.

2.1. MFCC Features

Following the steup in [1], A speech utterance is analyzed using a 25-ms Hamming window with a 10-ms frame rate. Each frame is represented by 12th-order Mel frequency cepstral coefficients (MFCCs) and energy, along with their first and second temporal derivatives. The data were normalized so that, averaged over the training cases, each coefficient or first derivative or second derivative had zero mean and unit variance.

2.2. LDA Features

Following the setup in [6], to create a set of Linear Discriminant Analysis (LDA) features, a speech utterance is first chunked into 20ms frames, with a frame-shift of 5 ms, a different frame rate than that used in the previous section and [1]. Each frame is represented by a 13 dimensional MFCC feature vector. Features are then mean and variance normalized on a per utterance basis. Then, at each frame, a context of 8 frames to the left and right of the current frame are joined together and an LDA transform is applied to project the feature vector down to 40 dimensions.

2.3. Speaker Adapted Features

Two steps are performed to create a set of speaker adapted (SA) features. First vocal tract length normalization (VTLN) is applied to the LDA features, followed by a feature/model space adaptation. Both steps are discussed below in more detail.

2.3.1. Vocal Tract Length Normalization

The length of a speaker's vocal tract is often a large factor in speaker variability. VTLN is a popular technique used to reduce this variability. In this procedure, a scaling factor is learned for each speaker that warps the speech from this speaker into a canonical speaker with an average vocal tract length. The warp is applied to the given set of acoustic features before they are LDA transformed. After the warp, features are again spliced together at each frame and an LDA transform is applied to produce a set of 40 dimensional "warped" features.

2.3.2. Feature/Model Space Adaptation

After VTLN, the "warped" features are adapted for each speaker using feature space Maximum Likelihood Linear Regression (fMLLR) [5]. In this method, a linear regression based transformation is applied to the VTLN features for each speaker to create a set of features in a canonical features space.

2.4. Discriminative Feature

Discriminatively trained features and models have been shown to significantly improve error rates, as these discriminative models have more power to better differentiate between confusable sounds, such as "ma" and "na". In this work, we use a large margin discriminative training approach using the Boosted Maximum Mutual Information (BMMI) criterion [4]. In this approach, a set of ML-trained models are used to bootstrap the training of a set of feature space Boosted Maximum Mutual Information (fBMMI) features. In this paper, we explore creating fBMMI features after the SA features.

3. DEEP BELIEF NETWORKS

Feed-forward neural networks can be trained to output a probability distribution over HMM states for the central frame in a window of feature values. The "correct" state is obtained by using a forced alignment with some pre-existing model and the weights of the neural network are adjusted to increase the log probability assigned to the correct HMM state. Typically, the weights are initialized to small random values. If there are many hidden layers in the neural network and many hidden units in each layer, it is easy for the neural network to overfit. Overfitting can be substantially reduced if a generative model is used to find sensible features without making any use of the labels. The hidden units of the neural net can then be initialized to implement these features. The very limited information in the labels is then only used to slightly adjust the pre-existing features to give better discrimination, not to discover features.

To discover multiple layers of non-linear features without using the labels, we need to fit a multi-layer generative model to the input data. For directed graphical models (called belief nets) with dense connectivity between layers, this is usually very difficult because it is generally very hard to infer the posterior distribution over each hidden layer given the input data. This distribution is required for maximum likelihood learning. [9] showed that the difficult inference problem can be finessed by learning the features one layer at a time using an *undirected* graphical model called a Restricted Boltzmann Machine (RBM). The RBM has a bipartite connectivity structure that makes it very easy to infer the states of its hidden variables. Once the weights of the RBM have been learned the vectors of hidden feature activations can be used as data for training another RBM that learns a higher layer of features. Building a deep generative model one layer at a time is much more efficient than trying to learn all of the layers at once. After a stack of RBMs has been trained, they can be composed to form a deep generative model which has undirected connections between its top two layers and top-down, directed connections between adjacent lower layers. This model is called a deep belief net (DBN).

One important property of DBNs is that their hidden states can be inferred very efficiently and fairly correctly by a single bottom-up pass in which the top-down generative weights are used in the reverse direction. Another important property is that each time an extra layer of learned features is added to a DBN, the new DBN has a variational lower bound on the log probability of the training data that is better than the variational bound for the previous DBN, provided the extra layer is learned in the right way [9].

The weights of a DBN can be used to initialize the hidden layers of a feedforward neural network which is given an additional, "softmax" output layer. For a wide variety of tasks, discriminative "fine-tuning" of a DBN-initialized neural network gives much better performance than the same neural network initialized with small random weights [10]. Many of the generatively learned features may be irrelevant for the discrimination task, but those that are relevant are usually much more useful than the input features because they capture the complex higher-order statistical structure that is present in the input data.

3.1. Learning a Restricted Boltzmann Machine

An RBM is a bipartite graph in which visible units that represent observations are connected to hidden units that learn to represent features using undirected weighted connections. They are restricted in the sense that there are no visible-visible or hidden-hidden connections. In the simplest type of RBM, both the hidden and visible units are binary and stochastic. To deal with real-valued input data, we use a Gaussian-Bernoulli RBM in which the hidden units are binary but the input units are linear with Gaussian noise. RBMs have an efficient training procedure which makes them suitable as building blocks for learning DBNs.

The weights on the connections and the biases of the individual units define a probability distribution over the joint states of the visible and hidden units via an energy function. For binary RBMs, the energy of a joint configuration is:

$$E(\mathbf{v}, \mathbf{h}; \theta) = -\sum_{i=1}^{\mathcal{V}} \sum_{j=1}^{\mathcal{H}} w_{ij} v_i h_j - \sum_{i=1}^{\mathcal{V}} b_i v_i - \sum_{j=1}^{\mathcal{H}} a_j h_j \qquad (1)$$

where $\theta = (\mathbf{w}, \mathbf{b}, \mathbf{a})$ and w_{ij} represents the symmetric interaction term between visible unit i and hidden unit j while b_i and a_j are their bias terms. $\mathcal{V}$ and $\mathcal{H}$ are the numbers of visible and hidden units. The probability that the model assigns to a visible vector $\mathbf{v}$ is:

$$p(\mathbf{v}; \theta) = \frac{\sum_{\mathbf{h}} e^{-E(\mathbf{v}, \mathbf{h})}}{\sum_{\mathbf{u}} \sum_{\mathbf{h}} e^{-E(\mathbf{u}, \mathbf{h})}} \qquad (2)$$

Since there are no hidden-hidden or visible-visible connections, the conditional distributions $p(\mathbf{v}|\mathbf{h})$ and $p(\mathbf{h}|\mathbf{v})$ are factorial and are

given by:

$$p(h_j = 1|\mathbf{v};\theta) = \sigma(\sum_{i=1}^{\mathcal{V}} w_{ij}v_i + a_j)$$
$$p(v_i = 1|\mathbf{h};\theta) = \sigma(\sum_{j=1}^{\mathcal{H}} w_{ij}h_j + b_i), \quad (3)$$

where $\sigma(x) = (1 + e^{-x})^{-1}$.

To perform steepest ascent in the log likelihood, the update rule for the weights is

$$\Delta w_{ij} \propto \langle v_i h_j \rangle_{data} - \langle v_i h_j \rangle_{model} \quad (4)$$

The expectation $\langle v_i h_j \rangle_{data}$ is the frequency with which visible unit i and hidden unit j are on together in the training set and $\langle v_i h_j \rangle_{model}$ is that same expectation under the distribution defined by the model. The term $\langle . \rangle_{model}$ takes exponential time to compute exactly, but the efficient Contrastive Divergence (CD) approximation to the gradient can be used instead [11]. The new update rule becomes:

$$\Delta w_{ij} \propto \langle v_i h_j \rangle_{data} - \langle v_i h_j \rangle_{recon} \quad (5)$$

where $\langle . \rangle_{recon}$ represents the expectation with respect to the distribution of reconstructions produced by initializing at a data vector then updating the hidden units followed by the visible units followed by the hidden units again.

For Gaussian-Bernoulli RBM's the learning procedure is very similar except that the visible activities are measured in units equal to the standard deviation of the noise. More details of Gaussian-Bernoulli RBM's can be found in [12].

4. EVALUATION SETUP

4.1. Corpus

Phone recognition experiments were performed on the TIMIT corpus.[1] We used the 462 speaker training set and removed all SA records (i.e., identical sentences for all speakers in the database) since they could bias the results. A separate development set of 50 speakers was used for tuning all of the meta parameters, such as the number of layers and the size of each layer. Results are reported using the 24-speaker core test set, which excludes the dev set.

4.2. DBN Training

All DBNs were pre-trained with a fixed recipe using stochastic gradient descent with a mini-batch size of 128 training cases. For Gaussian-binary RBMs, we ran 150 epochs with a fixed learning rate of 0.005 while for binary-binary RBMs we used 50 epochs with a learning rate of 0.08.

For fine-tuning, we used stochastic gradient descent with the same mini-batch size as in pre-training. The learning rate started at 0.1. At the end of each epoch, if the phone error rate (PER) on the development set increased, the weights were returned to their values at the beginning of the epoch and the learning rate was halved. This continued until the learning rate fell below 0.001. During both pre-training and fine-tuning, a small weight-cost of 0.0002 was used and the learning was accelerated by using a momentum of 0.9 (except for the first epoch of fine-tuning which did not use momentum).

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

[12] gives a detailed explanation of weight-cost and momentum and sensible ways to set them.

During training and decoding, 183 target class labels (i.e., 3 states for each one of the 61 phones) were used, which we will refer to as monophone class labels. After decoding, the 61 phone classes were mapped to a set of 39 classes as in [13] for scoring. We also explored using 2400 target class labels (i.e., triphone class labels), corresponding to the 2400 context-dependent (CD) states used in the IBM system [6]. Initial experiments with triphone targets result in comparable PERs with the monophone target based training and this needs to be investigated further.

4.3. Decoder

We experimented with two different decoders in this work. The decoder used at Toronto is HVite, which is part of the HTK package [14]. This decoder only supports up to bigram language models. For all experiments, we fixed the parameters for the Viterbi decoder. Specifically, we used a zero word insertion probability and a unit language model scale factor. The decoder used at IBM is a static FSM [15] with no limit on the number of ngrams, though in this work we explore using the IBM decoder with just trigrams. We will denote which decoder is used for various experiments.

5. EVALUATIONS

In this section, we report results using DBNs with various features.

5.1. DBN Performance with Different Feature Sets

5.1.1. Results

Figure 1 shows the effect using different input features and model depth when the number of input frames was fixed at 11. Note that in this experiment, 1024 hidden units per layer are used and results are reported using the HTK decoder. The main trend visible in the figure is that adding more hidden layers gives better performance, though the gain diminishes as the number of layers increases. There is no significant difference in performance between the MFCC features and LDA as the network is capable of capturing local discriminative information present in the context window. A jump of about 2% in accuracy is observed when speaker information is utilized to transform all speakers to a canonical speaker. Performing fBMMI complements the DBN local discrimination within the input context window by maximizing the margin between competing phone classes using an objective function that spans the whole utterance. Table 1 shows the best achieved PER for each type of input features on the core test set. Notice that as the feature type improves, the PER decreases, a trend which was also observed in [6] when posteriors were created from a GMM.

Feature Type	PER
MFCC	22.1
LDA	22.2
LDA+SA	20.3
LDA+SA+fBMMI	**19.6**

Table 1. PER of TIMIT Core Test Set

5.1.2. Error Analysis

Figure 2 shows the breakdown of error rates for the four different feature types within 6 broad phonetic classes (BPCs), namely vow-

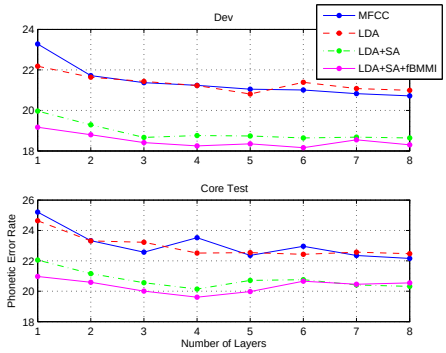

Fig. 1. Effect of Increasing DBN Hidden Layers with Diff. Features

els/semivowels, nasals, strong fricatives, weak fricatives, stops and closures/silence. Here the error rate was calculated by counting the number of insertions, deletions and substitutions that occur for all phonemes within a particular BPC. As the feature set is improved, gains are observed across all BPCs. The biggest decrease in PER can be seen in the vowel/semi-vowel class.

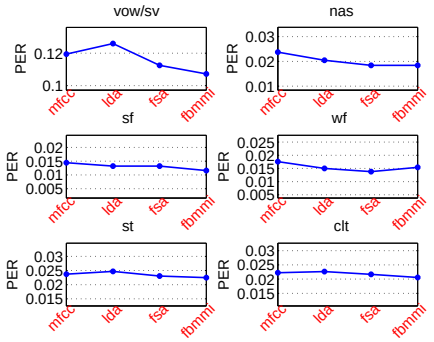

Fig. 2. Errors within BPCs Across Different Feature Types

5.2. Comparison with GMMs

Finally, we compared the performance when posteriors are generated using DBNs and GMMs with fBMMI features as input[2]. Note once posteriors are generated, a Viterbi search is performed with an HMM to find the best phone sequence. In this experiment, the IBM decoder is used. Figure 3 shows the relative change in PER from the GMMs to DBNs system across 6 different BPCs. Notice that the two systems appear to be complementary within each of the different BPCs. This suggests that a system which combines both GMM and DBN posteriors might offer further improvements. Table 2 shows the results for the DBN, GMM and posterior combination systems. Combining individual systems offers a slight improvement in PER.

System	PER
fBMMI-GMM	19.5
fBMMI-DBN	19.4
Posterior Combination	**19.3**

Table 2. PER of TIMIT Core Test Set at fBMMI Level

6. CONCLUSIONS

In this paper, we explored using speaker adaptive, discriminative features as inputs to a DBN. On TIMIT, we found that these features

[2]The GMM system is the best number on TIMIT with speaker adapted, discriminative features [16].

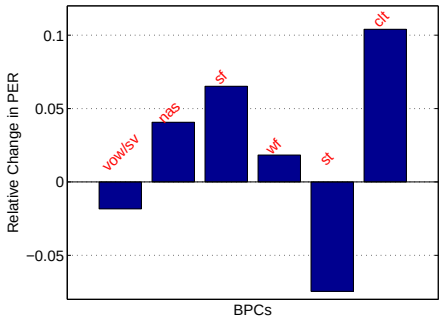

Fig. 3. Relative Change in PER of GMM and DBN systems

provided a PER of 19.6% using monophone HMMs and a bigram LM and 19.4% using a trigram LM, comparable to the best reported GMM number on TIMIT with speaker adapted, discriminative features. In the future, we would like to further optimize DBN parameters for different input feature streams to improve performance.

7. REFERENCES

[1] A. Mohamed, G. Dahl, and G. Hinton, "Acoustic Modeling using Deep Belief Networks," *IEEE Trans. on Audio, Speech and Language Processing.*, 2011.

[2] G.E. Dahl, M. Ranzato, A. Mohamed, and G.E. Hinton., "Phone Recognition with the Mean-Covariance Restricted Boltzmann Machine.," in *NIPS*, 2010.

[3] B.H. Juang, W. Hou, and C.H. Lee, "Minimum classification error rate methods for speech recognition," *IEEE Trans. on Speech and Audio Processing*, vol. 5, no. 3, pp. 257 –265, 1997.

[4] D. Povey, D. Kanevsky, B. Kingsbury, B. Ramabhadran, G. Saon, and K. Visweswariah, "Boosted MMI for model and feature-space discriminative training," 2008, pp. 4057 –4060.

[5] M.J.F. Gales, "Maximum Likelihood Linear Transformations for HMM-Based Speech Recognition," *Computer Speech and Language*, vol. 12, pp. 75–98, 1998.

[6] T. N. Sainath, B. Ramabhadran, and M. Picheny, "An Explortation of Large Vocabulary Tools for Small Vocabulary Phonetic Recognition," in *Proc. ASRU*, 2009.

[7] H. Bourlard and N. Morgan, *Connectionist Speech Recognition: A Hybrid Approach*, Kluwer Academic Publishers, 1993.

[8] H. Larochelle, D. Erhan, A. Courville, J. Bergstra, and Y. Bengio, "An empirical evaluation of deep architectures on problems with many factors of variation," in *ICML*, 2007, pp. 473–480.

[9] G. E. Hinton, S. Osindero, and Y. Teh, "A fast learning algorithm for deep belief nets," *Neural Computation*, vol. 18, pp. 1527–1554, 2006.

[10] D. Erhan, Y. Bengio, A. Courville, P. Manzagol, and P. Vincent, "Why does unsupervised pre-training help deep learning?," *Journal of Machine Learning Research*, vol. 11, pp. 625–660, 2010.

[11] G. E. Hinton, "Training products of experts by minimizing contrastive divergence," *Neural Computation*, vol. 14, pp. 1771–1800, 2002.

[12] G. E. Hinton, "A Practical Guide to Training Restricted Boltzmann Machines," in *Technical report 2010-003, Machine Learning Group, University of Toronto*, 2010.

[13] K. F. Lee and H. W. Hon, "Speaker-independent phone recognition using hidden Markov models," *IEEE Transactions on Audio, Speech & Language Processing*, vol. 37, no. 11, pp. 16411648, 1989.

[14] S. Young et. al., *The HTK Book*, Cambridge University, 2002.

[15] H. Soltau, G. Saon, and B. Kingsbury, "The IBM Attila speech recognition toolkit," in *Proc. IEEE Workshop on Spoken Language Technology*, 2010, to appear.

[16] T. N. Sainath, B. Ramabhadran, M. Picheny, D. Nahamoo, and D. Kanevsky, "Exemplar-Based Sparse Representation Features: From TIMIT to LVCSR," *Submitted to TSAP*, 2010.