Improved Bottleneck Features Using Pretrained Deep Neural Networks

Dong Yu and Michael L. Seltzer

Speech Research Group, Microsoft Research, USA

{dongyu, mseltzer}@microsoft.com

Abstract

Bottleneck features have been shown to be effective in improving the accuracy of automatic speech recognition (ASR) systems. Conventionally, bottleneck features are extracted from a multi-layer perceptron (MLP) trained to predict context-independent monophone states. The MLP typically has three hidden layers and is trained using the backpropagation algorithm. In this paper, we propose two improvements to the training of bottleneck features motivated by recent advances in the use of deep neural networks (DNNs) for speech recognition. First, we show how the use of unsupervised pretraining of a DNN enhances the network's discriminative power and improves the bottleneck features it generates. Second, we show that a neural network trained to predict context-dependent senone targets produces better bottleneck features than one trained to predict monophone states. Bottleneck features trained using the proposed methods produced a 16% relative reduction in sentence error rate over conventional bottleneck features on a large vocabulary business search task.

Index Terms: bottleneck features, pretraining, deep neural network, deep belief network,

1. Introduction

Bottleneck features generated by a multi-layer perceptron (MLP) can be considered a non-linear feature transformation and dimensionality reduction technique [1][2]. They can extract information useful for phoneme classification from multiple frames of the acoustic features and have been effective in improving the accuracy of speech recognition systems. Conventionally, bottleneck features are generated by a multi-layer perceptron (MLP) trained using the backpropagation (BP) algorithm. Typically, the MLP has three hidden layers and is trained to predict monophone state labels.

Historically, the practical use of MLPs was limited to architectures with only a few hidden layers. Deep neural networks (DNN) with many hidden layers were very difficult to train and were typically outperformed by shallower networks. However, a recent breakthrough in the training of DNNs has led to renewed interest in their use. This breakthrough occurred when unsupervised generative pretraining of the DNN parameters was proposed [3]. The basic idea of pretraining is to convert a DNN into a set of restricted Boltzmann machines (RBM). Each RBM is then trained greedily in a layer-by-layer manner. After pretraining, all layers are then jointly fine-tuned using the conventional BP algorithm. The role of the pretraining step is to initialize the DNN weights to a better starting point than random initialization and provide a form of regularization [4][5]. After pretraining, backpropagation typically converges to a better local optimum compared to systems with random initialization. In addition, pretraining enables deeper neural networks to be effectively used.

This method of training DNNs has proven to be effective on many tasks, including speech recognition. For example, the DNN-HMM which exploits the discriminative learning ability of pretrained DNNs and the sequential modeling ability of hidden Markov models (HMMs) outperformed the conventional Gaussian mixture model (GMM)-HMM for both phoneme recognition [6][7] and large vocabulary speech recognition [8] tasks. In the large vocabulary task, the DNN was trained to predict context-dependent senone posterior probabilities which then replaced the Gaussian computations in the HMM.

In this paper, we investigate whether pretrained DNNs can be used to generate bottleneck features that improve the accuracy of speech recognition systems compared to those trained using a conventional MLP. In addition, we study whether training the DNN using context-dependent target labels produces better bottleneck features given the success of this approach for DNN-HMMs. The effectiveness of these proposed improvements to bottleneck feature training was evaluated on a large vocabulary business search task. Our experiments show that bottleneck features generated by a pretrained DNN trained using senone labels produce a 16% relative reduction in sentence error rate compared to bottleneck features obtained using conventional MLP training and monophone state labels.

2. Bottleneck Features

Bottleneck features are generated from a multi-layer perceptron in which one of the internal layers has a small number of hidden units, relative to the size of the other layers. This small layer creates a constriction in the network that forces the information pertinent to classification into a low dimensional representation. Bottleneck features are most commonly used in an autoencoder which the neural network is trained to predict the input features themselves [3]. Because the activations at the bottleneck layer are a low-dimensional nonlinear function of the input features, an autoencoder can be viewed as a method of nonlinear dimensionality reduction. Bottleneck features for speech recognition are created from an MLP trained to predict phonemes or phoneme states. The inputs to the hidden units of the bottleneck layer are used as features for an HMM-based speech recognizer. These bottleneck features represent a nonlinear transformation and dimensionality reduction of the input features.

When generating neural-network-based features, the input is typically generated using variants of TRAPS features, in which long temporal windows of critical band energies are processed [9][10]. Other features, including multi-resolution variants of RASTA have also been used [11]. In most systems, the best performance is obtained by combining the bottleneck features derived from these inputs with traditional features, e.g., MFCC or PLP. It is believed that by using alternative input features to the neural network, the bottleneck features capture information that is complementary to conventional features derived from the short-time spectra of the input.

3. Pretrained Deep Neural Networks

Pre-trained DNNs can be seen as improved MLPs. In fact, when performing classification, pretrained DNNs and MLPs are identical. The improvement arises from the training strategy employed by the pretrained DNNs. The trick is to learn the parameters of each layer greedily by treating each pair of layers of the DNN as an RBM before doing a joint optimization of all the layers. This learning strategy enables deeper neural networks to be trained by moving the weights to good initial values.

3.1. Restricted Boltzmann Machines

An RBM can be represented as a bipartite graph in which the stochastic units in the visible layer only connect to the stochastic units in the hidden layer. The units in the visible and hidden layers are represented by distributions from the exponential family. Bernoulli or Gaussian distributions are typically used in the visible layer while Bernoulli distributions are typically used in the hidden layer. Gaussian-Bernoulli RBMs are typically used to convert real-valued stochastic variables to binary stochastic variables which can then be further processed using the Bernoulli-Bernoulli RBMs.

Given the model parameters θ, the joint distribution $\mathrm{p}(\mathbf{v},\mathbf{h};\theta)$ over the visible units $\mathbf{v}$ and hidden units $\mathbf{h}$ in the RBMs can be defined as

$$\mathrm{p}(\mathbf{v},\mathbf{h};\theta)=\frac{exp\left(-\mathrm{E}(\mathbf{v},\mathbf{h};\theta)\right)}{Z}, \quad (1)$$

where $\mathrm{E}(\mathbf{v},\mathbf{h};\theta)$ is an energy function and $Z=\sum_v\sum_h exp\left(-\mathrm{E}(\mathbf{v},\mathbf{h};\theta)\right)$ is the partition function. The marginal probability that the model assigns to a visible vector $\mathbf{v}$ is

$$\mathrm{p}(\mathbf{v};\theta)=\frac{\sum_{\boldsymbol{h}} exp\left(-\mathrm{E}(\mathbf{v},\mathbf{h};\theta)\right)}{Z}. \quad (2)$$

For Bernoulli-Bernoulli and Gaussian-Bernoulli RBMs, the energy functions are

$$\mathrm{E}(\mathbf{v},\mathbf{h};\theta)=-\sum_{i=1}^{V}\sum_{j=1}^{H}w_{ij}\,v_ih_j-\sum_{i=1}^{V}b_iv_i-\sum_{j=1}^{H}a_jh_j, \quad (3)$$

and

$$\begin{aligned}\mathrm{E}(\mathbf{v},\mathbf{h};\theta)=&-\sum_{i=1}^{V}\sum_{j=1}^{H}w_{ij}\,v_ih_j\\&+\frac{1}{2}\sum_{i=1}^{V}(v_i-b_i)^2-\sum_{j=1}^{H}a_jh_j,\end{aligned} \quad (4)$$

respectively, where w_{ij} represents the symmetric interaction term between visible unit v_i and hidden unit h_j, b_i and a_j are the bias terms, and V and H are the numbers of visible and hidden units.

Although the joint and marginal probabilities are expensive to estimate due to the partition function, the conditional probabilities can be efficiently calculated as

$$p\left(h_j=1|\mathbf{v};\theta\right)=\sigma\left(\sum_{i=1}^{V}w_{ij}\,v_i+a_j\right), \quad (5)$$

$$p(v_i=1|\mathbf{h};\theta)=\sigma\left(\sum_{j=1}^{H}w_{ij}\,h_j+b_i\right). \quad (6)$$

for the Bernoulli-Bernoulli RBM and

$$p\left(h_j=1|\mathbf{v};\theta\right)=\sigma\left(\sum_{i=1}^{V}w_{ij}\,v_i+a_j\right), \quad (7)$$

$$p(v_i|\mathbf{h};\theta)=N\left(v_i;\sum_{j=1}^{H}w_{ij}\,h_j+b_i,1\right). \quad (8)$$

for the Gaussian-Bernoulli RBM, where $\sigma(x)=1/\left(1+exp(x)\right)$ is the sigmoid function.

The parameters in RBMs can be optimized to maximize log likelihood $\log p(\mathbf{v};\theta)$ and can be updated as

$$\Delta w_{ij}=\langle v_ih_j\rangle_{data}-\langle v_ih_j\rangle_{model}, \quad (9)$$

where $\langle v_ih_j\rangle_{data}$ is the expectation that v_i and h_j occur together in the training set and $\langle v_ih_j\rangle_{model}$ is that same expectation under the distribution defined by the model. Because $\langle v_ih_j\rangle_{model}$ is extremely expensive to compute exactly, the contrastive divergence (CD) approximation to the gradient is used, where $\langle v_ih_j\rangle_{model}$ is replaced by running the Gibbs sampler initialized at the data for one full step [12].

3.2. Training Deep Neural Networks

The last layer in the pretrained DNNs transforms a number of Bernoulli distributed units into a multinomial distribution using the softmax operation

$$\mathrm{p}(l=k|\mathbf{h};\theta)=\frac{exp\left(\sum_{i=1}^{H}\lambda_{ik}h_i+a_k\right)}{Z(\boldsymbol{h})}, \quad (10)$$

where $l=k$ denotes the input been classified into the kth class, and λ_{ik} is the weight between hidden unit h_i at the last layer and class label k. The training labels for the bottleneck features are typically monophone or monophone states, though we will show that using finer labels such as senones can improve performance.

To learn the pretrained DNNs, we first train a Gaussian-Bernoulli RBM generatively in which the visible layer is the continuous input vector constructed from 11 frames of MFCC features in this study. Bernoulli-Bernoulli RBMs are used for the remaining layers. When pretraining the next layer, $E\left(h_j|\mathbf{v};\theta\right)=p\left(h_j=1|\mathbf{v};\theta\right)$ from the previous layer is used as the visible input vector based on the mean-field theory. This process continues until the last layer at which time BP is used to fine-tune all the parameters jointly by maximizing the frame-level cross-entropy between the true and the predicted probability distributions over class labels.

4. Experiments

The goal of this work is to investigate the effectiveness of three proposed sources of improvement for bottleneck features: pretraining the neural network, exploiting deeper networks, and using context-dependent target labels. We are also interested in knowing how bottleneck features perform compared to the context-dependent DNN-HMMs developed recently [8] For these purposes, we conducted a series of experiments using the Windows Live Search for Mobile (WLS4M) corpus collected from real users of a smartphone application for business search [13][14].

4.1. Corpus Description

The WLS4M corpus consists of spoken queries for local businesses in the United States collected from real users of a deployed smartphone voice search application. The audio is sampled at 8 kHz and encoded by GSM. The data contains almost all sources of variability seen in the real-world such as noise, music, side-speech, accents, hesitation, repetition, sloppy pronunciation, and interruption.

The training set for all the models consisted of 24 hours of speech from 32,057 utterances. The decoding parameters were tuned using a 6.5-hour development set (8777 utterances) and

Table 1. Sentence Error Rate (SER) of Baseline Systems

Acoustic Model	*Dev % SER*	*Test % SER*
GMM-HMM ML	37.1	39.6
GMM-HMM MPE	34.5	36.2
DNN-HMM	28.2	30.4

Table 2. Performance comparison of bottleneck features trained with different supervision labels.

Labels used for bottleneck feature training	*Dev % SER*	*Test % SER*
None (pretraining only)	39.4	42.1
Monophone states	35.2	37.0
Converted monophone states	34.0	35.7
Senones	33.4	34.8

the final evaluation was performed using a 9.5-hour test set (12,758 utterances). To preserve the time-varying nature of a deployed application, all queries in the training set were collected before those in the development set, which were in turn collected before those in the test set. The lexicon and trigram language model (LM) used for decoding were the same as used in our previous work [8]. The lexicon contains 65K words and was extracted from the CMU public lexicon. The perplexity of the LM is 117.

We evaluated the performance on this task using sentence error rate (SER) for three reasons. First, business search queries are short with an average sentence length of 2.1 words and users want all words correctly recognized to find the right company. Second, there is significant inconsistency in spelling that makes using sentence accuracy more convenient. Third, all previous results on this task were reported using sentence accuracy.

4.2. Baseline Systems

The baseline system in this task was a standard triphone GMM-HMM, trained on 39-dimensional MFCC features (static, delta, and delta-delta coefficients). Log energy was used in place of C0 and the features were pre-processed with cepstral mean normalization. These baseline systems are optimized for the training set size and have 53K logical and 2K physical triphones with 761 tied states (senones), each of which is a GMM with 24 mixture components. Acoustic models were built using both maximum likelihood (ML) and minimum phone error (MPE) criteria. In addition, we also report the accuracy of a DNN-HMM system that used a 5-hidden-layer pretrained DNN to predict the same 761 senones used in the GMM-HMM systems. The performance of the baseline systems are shown in Table 1.

4.3. Experiments on Bottleneck Features

In this study we derive the bottleneck features directly from MFCC features. This is different from most other bottleneck features that are generated from alternative features, e.g., TRAPS-DCT features. We made this decision in order to remove any gains obtained from using additional information and isolate the effectiveness of the bottleneck features extracted with different techniques. In all the results reported here, we followed the DNN training recipe in [8]. More specifically, we used 11-frames of 39-dimensioanl MFCC features as the input to the DNNs, 39 hidden units in the bottleneck layer, and 2048 hidden units for non-bottleneck hidden layers. We used a learning rate of 0.004 for all layers during pretraining, and used a learning rate of 0.08 for the first 6 epochs and a learning rate of 0.002 for the remaining 6 epochs during fine-tuning. The DNN training was carried out using stochastic mini-batch gradient descend with a minibatch size of 256 samples.

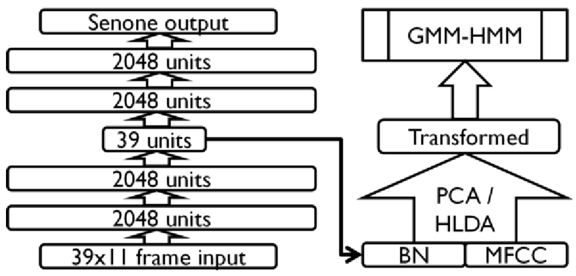

Figure 1: Typical 5-hidden layer bottleneck feature extraction architecture used in this study.

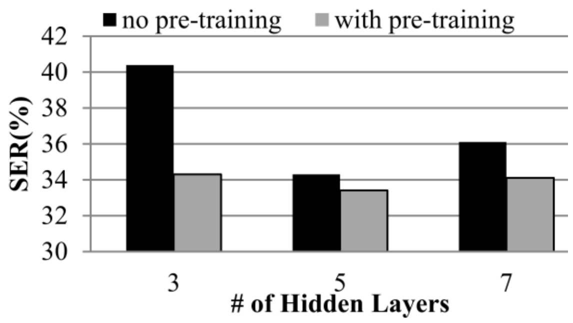

Figure 2: Development set SER using BN features with and without pretraining.

The bottleneck features extracted from the DNNs are then used as alone or concatenated with the original MFCC features to train the GMM-HMMs. In both the bottleneck (BN) only and BN+MFCC configurations, the features were decorrelated using principle component analysis (PCA) and converted to 39 dimensions. Figure 1 illustrates the typical 5-hidden-layer bottleneck feature extraction architecture used in this study. To train the GMM-HMMs using the BN or BN+MFCC features, single pass retraining was performed using the baseline GMM-HMMs trained from MFCC features. Four iterations of EM were then performed using the BN or BN+MFCC features.

Figure 2 shows the SER on the development set using bottleneck features only with and without using pretraining. The bottleneck features were trained using senone labels generated by forced alignment using the ML-trained GMM-HMMs.

We make two observations from this figure. First, the bottleneck features learned with pretraining consistently outperform those learned without pretraining. Second, the bottleneck features in the 5-hidden-layer DNN perform better than those in the 3- and 7-hidden-layer DNNs. This indicates that while a sufficiently deep network is helpful for bottleneck features, deeper networks do not always improve performance. Note that this is in contrast to the DNN-HMMs where deeper networks seem to consistently perform better than shallower networks. We believe this is because when networks become very deep, the middle bottleneck layer gets less information from the supervision. Because bottleneck features extracted from 5-hidden-layer DNN have the best performance, we keep this configuration for the remainder of our experiments.

It was hypothesized that using monophone state labels was sufficient for training bottleneck features [10]. In the CD-DNN-HMM work, however, significant gains were obtained using senone labels rather than monophone state labels. We thus are interested in knowing whether senone labels also produce better bottleneck features. Table 2 compares the performance of the BN systems using labels generated from

Table 3. Comparison of BN performance with traditional MLP training and pretrained DNN's with senone labels.

Bottleneck Feature Training				
# Hidden Layers	***Pre-Train***	***Target Labels***	***Dev % SER***	***Test % SER***
3	No	Monophone State	39.9	41.6
3	Yes	Senone	34.3	36.0
5	Yes	Senone	33.4	34.8

different acoustic models. The first row in the table shows that if supervised fine-tuning is omitted and the bottleneck features are only pretrained using the generative criterion, a 39.4% SER is obtained on the development set.

Using monophone state alignment from the monophone GMM-HMM reduces the SER to 35.2%. This highlights the importance of supervised training for bottleneck features. Using monophone state labels converted from the alignment performed by the triphone acoustic model further reduces the SER by 1.2% on the development set. Finally, if the senone labels are used directly we got additional improvement of 0.6% and 0.9% on the development and test sets, respectively. There was no additional gain obtained using labels generated from the MPE-HMM or the DBN-HMM so those results are not reported here.

The total effect of the training strategies proposed in this paper is shown in Table 3. The table shows the performance obtained using a conventional MLP with three hidden layers and monophone state labels as well as that obtained using the same neural network with pretraining and senone labels. Adding pretraining and senone labels provides approximately 14% relative improvement in SER on the development and test sets. The relative improvement grows to 16% if a deeper network with five hidden layers is used.

Finally, we evaluated the performance of bottleneck features when combined with the original MFCC features. As shown in Table 4, the combination of bottleneck and MFCC features is 0.8% better than bottleneck features alone on the development set. Surprisingly, no improvement is seen on the test set. Comparing Tables 1 and 3, we can see that an ML HMM system that uses BN+MFCC features outperforms a discriminatively trained MPE system that uses MFCC features alone. If we train the BN+MFCC system using MPE , relative reductions in SER of 12% and 13% are observed on the development and test sets compared to the baseline MFCC MPE system. These results are not quite as good as those obtained using the DNN-HMM since the training of BN features is disconnected from the manner in which such features are processed during decoding. However, this performance gap can be reduced further by applying multi-pass decoding strategies and speaker adaptation techniques.

5. Conclusions

In this paper, we have studied the use of deep neural networks for bottleneck feature extraction. In order to obtain good performance with deep networks, unsupervised pretraining is applied, in which the DNN into converted into a series of Restricted Boltzmann Machines and greedily trained layer by layer. Using pretraining to initialize the parameters prior to back propagation has two benefits: first, performance is improved for all neural network topologies. Second, deep architectures with many layers that perform poorly with random initialization now can achieve good performance.

We have also examined what impact the choice of target labels used to train the neural network has on performance. Conventionally, neural networks used to generate probabilistic or bottleneck features use monophone or monophone state labels. We have shown that improvements can be obtained by using context-dependent labels generated by the senones of the recognizer. The combination of DNN pretraining and the senone target labels resulted in a 16% improvement in performance compared to the conventional method for training bottleneck features.

Table 4. Summarization of sentence accuracy obtained using pretrained DNN based bottleneck features.

Features	***HMM Training***	***Dev % SER***	***Test % SER***
BN	ML	33.4	34.8
BN + MFCC	ML	32.6	34.8
BN + MFCC	MPE	30.4	32.2

6. Acknowledgements

The authors would like to thank Li Deng, Frank Seide, and Nikko Strom for valuable discussions.

7. References

[1] V. Fontaine, C. Ris, and J.-M. Boite, "Nonlinear discriminant analysis for improved speech recognition," in *Proc. Eurospeech*, 1997.

[2] F. Grezl, M. Karafiat, S. Kontar, and J. Cernocky, "Probabilistic and bottle-neck features for LVCSR of meetings," in *Proc. ICASSP* 2007, pp. 757–760.

[3] G. Hinton and R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504 – 507, 2006.

[4] D. Erhan, A. Courville, Y. Bengio, and P. Vincent, "Why does unsupervised pretraining help deep learning?" in *Proc. of AISTATS* 2010, vol. 9, May 2010, pp. 201–208.

[5] D. Yu, L. Deng, and G. Dahl, "Roles of pretraining and fine-tuning in context-dependent DBN-HMMs for real-world speech recognition," in *Proc. NIPS Workshop on Deep Learning and Unsupervised Feat. Learning*, 2010.

[6] A. Mohamed, G. E. Dahl, and G. E. Hinton, "Acoustic modeling using deep belief networks," *IEEE Trans. on Audio, Speech, and Lang. Proc.- Special Issue on Deep Learning for Speech and Lang. Proc., 2011*, (in press).

[7] A. Mohamed, D. Yu, and L. Deng, "Investigation of Full-Sequence Training of Deep Belief Networks for Speech Recognition", in *Proc. Interspeech* 2010, pp. 1692-1695.

[8] G.E. Dahl, D. Yu, L. Deng, and A. Acero, "Context-Dependent Pretrained Deep Neural Networks for Large Vocabulary Speech Recognition", *IEEE Trans. Audio, Speech, and Lang. Proc. - Special Issue on Deep Learning for Speech and Lang. Proc.*, 2011, (in press).

[9] H. Hermansky and S. Sharma, "TRAPS – classifiers of temporal patterns," Proc. ICSLP, Sydney Australia, 1998.

[10] F. Grezl and P. Fousek, "Optimizing bottle-neck features for LVCSR," in *Proc. ICASSP*. 2008, pp. 4729–4732.

[11] C. Plahl, R. Schluter, and H. Ney, "Hierarchical bottle neck features for LVCSR," in *Proc. Interspeech* 2010.

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

[13] D. Yu, Y. C. Ju, Y. Y. Wang, G. Zweig, and A. Acero, "Automated directory assistance system - from theory to practice," in *Proc. Interspeech*, 2007, pp. 2709–2711.

[14] A. Acero, N. Bernstein, R. Chambers, Y. Ju, X. Li, J. Odell, P. Nguyen, O. Scholtz, and G. Zweig, "Live search for mobile: Web services by voice on the cellphone," in *Proc. ICASSP* 2008, pp. 5256–5259.