Investigation of Deep Neural Networks (DNN) for Large Vocabulary Continuous Speech Recognition: Why DNN Surpasses GMMs in Acoustic Modeling

Jia Pan[1], Cong Liu[1], Zhiguo Wang[1], Yu Hu[1], Hui Jiang[2]
[1]*iFlytek Research, Hefei, Anhui, P. R. China*
[2]*Department of Computer Science and Engineering, York University, Toronto, Canada*
{jiapan,conliu2,zgwang,yuhu}@iflytek.com, hj@cse.yorku.ca

Abstract

Recently, it has been reported that context-dependent deep neural network (DNN) has achieved some unprecedented gains in many challenging ASR tasks, including the well-known Switchboard task. In this paper, we first investigate DNN for several large vocabulary speech recognition tasks. Our results have confirmed that DNN can consistently achieve about 25-30% relative error reduction over the best discriminatively trained GMMs even in some ASR tasks with up to 700 hours of training data. Next, we have conducted a series of experiments to study where the unprecedented gain of DNN comes from. Our experiments show the gain of DNN is almost entirely attributed to DNN's feature vectors that are concatenated from several consecutive speech frames within a relatively long context window. At last, we have proposed a few ideas to re-configure the DNN input features, such as using logarithm spectrum features or VTLN normalized features in DNN. Our results have shown that each of these methods yields over 3% relative error reduction over the traditional MFCC or PLP features in DNN.

Index Terms: speech recognition, deep neural networks, pre-training, acoustic modeling

1. Introduction

Artificial neural network (ANN) [1] has been used to model the state emission probabilities of hidden Markov model (HMM) speech recognizers since early 90's. Although this ANN-HMM outperformed Gaussian mixture HMM for some tasks, the improvements were typical very small. Very recently, deep neural network (DNN) has been proposed to replace the traditional GMMs in HMMs as basic acoustic models for automatic speech recognition (ASR) [6,2]. The main differences between this so-called context-dependent DNN-HMMs hybrid model and the classical ANN-HMMs in the 90's include that output nodes of neural network are expanded from a small number of phonemes into a large number of tied-states of tri-phone HMMs and network depth is significantly increased to over six hidden layers. Thanks to Hinton's pre-training procedure [7], the parameters of the deep neural network with a lot of hidden layers and a huge output layer can still be learned in a very reliable way. It has been reported that this type of context-dependent DNN-HMMs has achieved an unprecedented gain in many challenging ASR tasks, including business-search task [6] and the well-known Switchboard task [2,5].

In this paper, we first investigate DNN for several large vocabulary speech recognition tasks, including 320 hours English Switchboard task and three other Mandarin Chinese recognition tasks (ranging from 70 hours to 700 hours of training data). Our results in all of these four large tasks have confirmed that DNN can consistently achieve about 25-30% relative error reduction over the best discriminatively trained GMMs. Even in a 700-hour Mandarin speech recognition task, DNN still yields over 28% error reduction over the best discriminatively trained GMMs. Next, we have conducted a series of experiments to study where the unprecedented gain of DNN comes from. Our experiments show the gain of DNN is almost entirely attributed to DNN's feature vectors that are concatenated from several consecutive speech frames within a relatively long context window. Inspired by the above conclusion we have proposed a few methods to re-configure the DNN inputs, such as using logarithm spectrum features and VTLN normalized features in DNN. Our results have shown that each of these methods yields over 3% relative error reduction over the traditional MFCC or PLP features in DNN.

2. The Context-Dependent Deep Neural Network HMM

Hidden Markov model (HMM) has been the dominant technique for ASR for at least two decades. One of the critical parameters of HMM is the state observation probability distribution. In conventional HMM for ASR, Gaussian mixture models (GMMs) are used to model the state observation probabilities. The Gaussian mixture HMMs are typically trained based on maximum likelihood criterion or other discriminative training strategies [3]. Recently, the so-called context dependent deep neural network (DNN) has been proposed to replace GMMs to compute state observation probabilities for all tied states in the HMM set [6, 4]. It has been reported that this so-called context-dependent DNN/HMM hybrid model has achieved an unprecedented gain in many challenging ASR tasks [2, 5]. In this section, we briefly review how context dependent DNN is combined with HMM for ASR.

2.1. Structure of Deep Neural Network

The structure of DNN is a multi-layer perceptron (MLP). A (L+1)-layer MLP is used to model the posterior probability $P_{s|o}(s \mid o)$ of an HMM tied-state s given an observation vector o. The first L layers, $l = 0 \ldots L-1$, are hidden layers that model posterior probabilities of hidden nodes h^l given input vectors v^l from previous layer while the top layer L is used to compute the posterior probability for all tied states using softmax:

$$P^l_{h_j|v}(h^l_j \mid v^l) = 1/(1+e^{-z^l_j(v^l)}) = \sigma(z^l_j(v^l)), 0 \le l < L \tag{1}$$

$$P^L_{s|v}(s \mid v^L) = \frac{e^{z^L_s(v^L)}}{\sum_{s'} e^{z^L_{s'}(v^L)}} = soft\max_s(z^L(v^L)) \tag{2}$$

$$z^l(v^l) = (W^l)^T v^l + a^l \tag{3}$$

where W^l and a^l denote weight matrix and bias vectors for hidden layer l, and h_j^l and $z_j^l(v^l)$ denote the j-th component of hidden node, h^l, and its activation $z^l(v^l)$ respectively.

2.2. Training and Initialization

2.2.1. DNN Training with error back-propagation

MLPs are often trained with the error back-propagation procedure (BP) with stochastic gradient descent:

$$(W^l, a^l) \leftarrow (W^l, a^l) + \eta \frac{\partial J}{\partial (W^l, a^l)}, 0 \le l \le L \tag{4}$$

where J is an objective function and η is the learning rate. In ASR, the objective function J is set to maximize the total log posterior probability over all T training samples O = {o(t)} given the ground-truth state labels s(t), i.e.

$$J(O) = \sum_{t=1}^{T} \log P_{s|o}(s(t) \mid o(t)) \tag{5}$$

In this case, the gradients can be computed as follows:

$$\begin{aligned} &\frac{\partial J}{\partial W^l} = \sum_t v^l(t)(w^l(t)e^l(t))^T \qquad \frac{\partial J}{\partial a^l} = \sum_t w^l(t)e^l(t) \\ &e^L(t) = (\log soft\max)'(z^L(v^L(t))) \\ &e^{l-1}(t) = W^l \cdot w^l(t) \cdot e^l(t), 0 \le l \le L \\ &w^l(t) = \begin{cases} diag(\sigma'(z^l(v^l(t)))) & 0 \le l \le L \\ 1 & l = L+1 \end{cases} \end{aligned} \tag{6}$$

where error signals $e^l(t) = \partial J / \partial v^{l+1}(t)$ are back-propagated from the layers l+1 and above, and component-wise derivatives of sigmoid are $\sigma_j'(z) = \sigma_j(z) \cdot (1 - \sigma_j(z))$ and derivative of log softmax is computed as $(\log soft\max)_j'(z) = \delta_{s(t),j} - soft\max_j(z)$.

2.2.2. Initialization with DBN-based Pre-training

The problem of training MLP with BP algorithm is that the objective function is non-convex. When more hidden layers are added, it becomes very difficult to find good local optimum. As a result, it is very important to initialize all MLP parameters by some efficient pre-training algorithm.

In [7], it has been proposed to use a pre-training algorithm based on restricted Boltzmann machine (RBM) to initialize DNN. RBM is a two-layer generative model based on an energy function assigned to every configuration of visible and hidden state vectors. An efficient unsupervised algorithm based on one-step contrastive divergence as in [7] is available to learn all connection weights between two layers in RBM. This pre-training algorithm has been found to be also effective in training DNN, where the learned weights of RBM can be directly used to initialize a two-layer feed-forward neural network with sigmoid hidden units. Once we have trained an RBM from training data, we can use its hidden activation probabilities as training data to train another layer of RBM, which in turn is used to initialize next layer in DNN. In such a way, we will be able to initialize all weights in all layers of DNN using the above RBM connection weights. This is called pre-training. After that, we add a randomly initialized soft max output layer and use the standard BP algorithm to fine-tune all parameters in DNN. See [7][6] for more details on RBM-based pre-training.

2.3. Decoding of CD-DNN-HMMs

When we use context dependent DNN-HMMs for decoding, we compute the state emission probabilities using DNN as

$$p(O \mid S) = \frac{p(S \mid O) * p(O)}{p(S)} \tag{9}$$

where O is the observation vectors that are regular acoustic feature vectors augmented with neighbor frames within a context window and S is the tied state of tri-phone HMM based on a phonetic decision tree as in a normal tri-phone GMM-HMM model. The posterior probability $p(S \mid O)$ comes from the output layer of DNN for the corresponding tied-state S, and $p(O)$ can be ignored since it is irrelevant to S. The prior probability $p(S)$ can be approximately computed by counting the frames belonging to each state S based on state forced-alignment using GMM-HMMs or other models.

3. Experiments

In this section, we will first use several large vocabulary ASR tasks to evaluate the performance of context-dependent DNN-HMM and more importantly we will conduct some experimental studies to investigate why and how the DNN-HMM has surpassed the conventional GMM-HMM by such a surprisingly large margin. In this paper, we use four large ASR data sets, including the well-known 320 hours Switchboard task, 70 hours Mandarin PSC task, 500 hours Mandarin short message dictation (SMD) task and 700 hours Mandarin conversational telephony speech (CTS) recognition task. For the Switchboard task, the training data contains Switchboard-I training set about 300 hours and Call Home English training set containing 80 telephone calls in America about 20 hours. We have used two standard test sets: the NIST 1998 and 2001 Hub5 evaluate sets. For Mandarin PSC task, the data are collected from computer tests to evaluate language proficiency for Mandarin native speakers in Anhui, China. The PSC train set contains 76,843 utterances (about 70 hours) from 1,500 speakers and the test set contains 3720 utterances (about 3 hours) from 50 speakers. The SMD data are collected from an online service for mobile devices. It includes 500 hours training data from more than 5,000 speakers and 30 hours test set from 500 speakers. The CTS task is similar to Switchboard task and the data are collected from many telephone calls in China. It contains 700 hours training data and 20 hours test data.

3.1. GMM-HMM Baseline Systems

As our baseline systems, we have trained state-tied cross-word tri-phone GMM-HMMs based on both maximum likelihood (ML) and minimum phone error (MPE) criteria. The features for PSC task and SMD task are 39-dim vectors including 13-dim static MFCC's and first and second derivatives while Switchboard task and CTS task use 39-dim feature vectors including 13-dim PLP and the first and second derivatives. A 4-dim pitch feature is extracted and concatenated with the above 39-dim features for the three Mandarin ASR tasks. In PSC and SMD, the features are normalized with cepstral mean normalization algorithm per utterance while cepstral mean and variance normalization are used for Switchboard and CTS per conversation side. The GMM-HMMs are trained with 30 Gaussians per state for PSC and 40 Gaussian mixtures per state for the rest three tasks. The numbers of tri-phone tied states are

3969 in PSC, 8991 for Switchboard, 5996 for CTS and 6004 for SMD respectively. The language model for Switchboard is tri-gram model trained with all training transcripts. The language models for Mandarin tasks are trained from some available large Chinese text corpora. The CTS task uses a tri-gram model consisting of 26 million n-grams, and SMD use a four-gram model with 400 million n-grams. The perplexity of CTS is 110 while the perplexity of SMD is about 90. In Table 1, we give the baseline recognition performance for the baseline GMM-HMM models in all of the four tasks in the first two rows, where the performance in Switchboard is comparable with the best single-pass performance reported under the same training condition in [8]. We report word error rate (WER) for Switchboard and character error rate (CER) for the rest three Chinese tasks.

Table 1 *Performance (WER or CER in %) of different models and relative error reduction of DNN over MPE-GMM*

	Switchboard		PSC	CTS	SMD
	Hub98	Hub01			
ML GMM	46.6	35.4	18.2	53.6	24.4
MPE GMM	43.4	32.8	16.7	48.7	22.3
DNN	**31.2**	**23.7**	**13.4**	**34.8**	**16.8**
rel gain	*28.1%*	*27.7%*	*19.8%*	*28.5%*	*24.7%*

3.2. DNN-HMMs

In this section, we use the above-mentioned methods to train DNN for HMMs. We follow the standard DNN setup in [6][2], where each feature vector is augmented with its neighboring 10 frames within a context window (5+1+5) to form a 11-frame vector as DNN input feature. In our experiments, we use RBM-based pre-training to initialize DNN layer by layer. All parameters in RBM are experimentally tuned in the PSC task and are kept the same for the other three larger tasks. In RBM pre-training, we use a learning rate of 0.0025 for all layers and a momentum term of 0.9 in updating weights. We use 12 full sweeps through all training data for Gaussian-Bernoulli RBM and 10 full sweeps for Bernoulli-Bernoulli RBM. In our experiments, we average updates over mini-batches of 1024 training cases before applying them. The whole training corpus is normalized to zero mean and unit variance prior to pre-training since RBMs are not scale-invariant.

After pre-training, the DNN is fine-tuned using BP with state labels obtained through forced alignment by ML trained GMM-HMM models. All BP parameters are also experimentally tuned in the PSC task. During BP fine-tuning, we use 10 training epochs of the whole training set. A learning rate of 0.002 is used for the first three epochs and the learning rate was set to be half of the previous epoch for the remaining epochs. We average updates over mini-batches of 1024 training cases on all tasks and no momentum term is added. We have trained and evaluated DNNs with 3 to 6 hidden layers and each hidden layer has 1024 to 2048 hidden nodes.

The best DNN performance is listed in the third row of Table 1 for all four tasks and the relative error reductions over the discriminatively trained MPE-GMM are also given in the fourth row. We can see that DNN yields a consistent performance gain (ranging from 20% to 30% error reduction) across all evaluated ASR tasks, including CTS with over 700 hours of training data. This is really a very impressive gain in ASR that has not been seen for decades.

3.3. Why DNN-HMM Surpasses GMM-HMM

Given the above DNN results, it is interesting to ask the question where this impressive performance gain comes from in DNN-HMM method. In this section, we will conduct a series of experiments to answer this question. In order to have a fair comparison of modeling capability between DNN and GMMs, we consider using the input features for both models. Since GMMs cannot handle highly correlated features so that it does not make sense to use 11 concatenated frames in GMM. We have chosen to use only the current frame (without augmenting any neighboring frames) for both DNN and GMMs. We first conduct the experiments in the smaller PSC task. The results are shown in upper half of Table 2. It is quite surprising that DNN does NOT yield any better performance than discriminatively trained GMMs if they both use the current frame only. This is also true even when we increase the hidden layers from 1 to 6. On the other hand, as shown in the bottom half of Table 2, if we extend the context window to augment more neighboring frames as DNN input features, performance improves dramatically from 18.0% to 13.4% (using 11 frames). These results clearly demonstrate the impressive gain of the context-dependent DNN-HMM model can be almost entirely attributed to DNN's input feature vectors that are concatenated from several consecutive speech frames within a relatively long context window. DNN-HMM does not necessarily yield better modeling capability than the normal GMMs for standard speech features but DNN is indeed very powerful in terms of leveraging highly correlated features.

Table 2 *DNN Performance (CER in %) in the PSC task using various context window for input features (NN-1 for MLP with one hidden layer and DNN-6 for DNN with 6 hidden layers)*

Context window	NN-1	DNN-6	MPE GMM
1	*18.0*	*17.0*	*16.7*
Context window	3	5	7
DNN-6	*14.2*	*13.7*	*13.5*
Context window	9	11	13
DNN-6	*13.5*	***13.4***	*13.6*

Moreover, we also repeat the above experiments in the Switchboard task. The results in Table 3 lead to the same conclusion. If we use only the current frame, DNN does not surpass MPE-trained GMMs even when we increase the number of hidden layers up to 5. However, as shown in the second part of Table 3, performance of DNN is significantly improved when we use a longer (5+1+5) context window, WER is quickly brought down to 31.2% from 42.6% in Hub98 and 33.1% to 23.7% in Hub01, respectively.

Table 3 *DNN Performance (WER in %) in Switchboard using various context windows for input features. NN-1 for MLP with one hidden layer, DNN-3 (DNN-5) for DNN with 3 (5) hidden layers with 2000 hidden nodes per layer.*

Context window		NN-1	DNN-3	DNN-5	MPE GMM
1	Hub98	44.8	43.9	42.6	43.4
	Hub01	35.4	34.8	33.1	32.8
11	Hub98	39.7	33.4	**31.2**	n/a
	Hub01	31.4	25.6	**23.7**	n/a

The above experimental results demonstrate that almost the entire gain of DNN is attributed to DNN input feature vectors that are concatenated from several consecutive speech frames within a relatively long context window. Because these augmented features provide more context information to distinguish among different phones. Moreover, these neighboring frames are highly correlated because of large overlaps in speech analysis windows. It is difficult to use GMMs to model these augmented features since it leads to ill-formed covariance matrices in Gaussians. On the other hand, DNN is quite powerful to harness these highly correlative features since neural networks only use linear perceptron as their basic units for classification. It seems fairly difficult for GMMs to compete with DNN unless we have a good way to use these highly correlated features in GMMs.

3.4. Some improvements to DNN-HMMs

Since we have known DNN can handle highly correlated feature, it is easy to think of directly using log filter bank energies instead of using MFCCs in DNN. The role of DCT in MFCC extraction is to de-correlated features. This is important for GMM modeling but it becomes unnecessary in DNN. Moreover, DCT reduces the feature dimension and this may lead to information loss. In this experiment, we directly use 24 filter bank log energies (prior to DCT in MFCC extraction) as static features to replace the standard MFCC's. Next, we calculate dynamic features in the same way to generate 76-dim (24*3+4) feature vector for each frame. These features are augmented with the neighboring frames in (5+1+5) context window as input features to DNN. The recognition results of using these features for 6 hidden layers DNN in the PSC task are shown as Table 4.

Table 4 *DNN performance comparison (CER in %) between log filter bank energy and MFCCs in the PSC task*

feature type	PSC
MFCC_0_D_A + pitch	13.38
log Filter bank + pitch	13.27
log Filter bank_D_A + pitch	**12.93**

The results show that if we only use static logarithm filter bank features, performance of DNN improves slightly but if we add delta and accelerate features, performance of DNN improves from 13.38% to 12.93%, which is about 3.4% relative error reduction over the standard MFCC features.

It is well known that vocal-tract length normalization (VTLN) is a very effective normalization method widely used in GMM-HMM systems. VTLN warps the frequency axis to account for the fact that the precise locations of vocal-tract resonances vary roughly monotonically with the physical size of the speaker. Generally speaking, VTLN features are better for phoneme classification since VTLN can normalize some feature variations due to speaker difference. In this experiment, we use VTLN features instead of the PLP features in the Switchboard task to evaluate DNN-HMMs. It is noted that all warping factors (in both training and testing stages) are estimated from ML-trained GMM-HMM models. The recognition performance of DNN using the VTLN features in the Switchboard task is shown in Table 5. From the above results we can see that VTLN features give almost 4% relative improvement over PLP for DNN-3 (with three hidden layers) are used the gain of VTLN is almost 4% relatively. For DNN-6 (with 6 hidden layers), the gain of VTLN features is still about 3% relatively. Note that this gain is still quite significant but it is less than that in GMM-HMMs (about 7% in this case). It is interesting that our conclusion is not the same as in [5], where it is claimed that VTLN features cannot bring any significant gain than PLP features if the warping factors are estimated by GMM-HMMs or when the hidden layer number is large.

Table 5 *DNN performance (WER in %) using VTLN normalized PLP features in Switchboard*

	Hub98		Hub01	
	DNN-3	DNN-6	DNN-3	DNN-6
PLP	33.4	30.9	25.6	23.6
VTLN	**32.0**	**30.0**	**24.4**	**22.8**

4. Conclusions

In this paper, we have investigated DNN in several large vocabulary speech recognition tasks. Our results have confirmed that DNN can consistently achieve about 25-30% relative error reduction over the best discriminatively trained GMMs. Moreover, we have also shown that the gain of DNN is almost entirely attributed to DNN feature vectors that are concatenated from several consecutive speech frames within a relatively long context window. At last, we have proposed a few ideas to re-configure the DNN input features, such as using logarithm spectrum features or VTLN normalized features in DNN.

As for future work, we will develop effective speaker-adaptation techniques DNN in feature and model spaces, and use convolutional neural network (CNN) to improve feature extraction in DNN [9], and improve efficiency of training to scale up further.

5. References

[1] N. Morgan and H. Bourlard, "Continuous speech recognition using multilayer perceptrons with hidden Markov models," *Proc. of ICASSP*, 1990.

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

[3] H. Jiang, "Discriminative training for automatic speech recognition: A survey", *Computer and Speech, Language*, pp.589-608, Vol. 24, Issue 4, October 2010.

[4] D. Yu, L. Deng, and G. Dahl, "Roles of Pre-training and Fine-Tuning in Context-Dependent DNN-HMMs for Real-World Speech Recognition," *Proc. of NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, December 2010.

[5] F. Seide, G. Li, X. Chen and D. Yu, "Feature Engineering in Context-Dependent Deep Neural Networks for Conversational Speech Transcription," *Proc. ASRU 2011*, pp. 24-29, 2011.

[6] G. Dahl, D. Yu, L. Deng, and A. Acero, "Context-Dependent Pre-Trained Deep Neural Networks for Large Vocabulary Speech Recognition," *IEEE Trans. Speech and Audio Proc., Special Issue on Deep Learning for Speech and Language Processing*, 2011.

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

[8] P. C. Woodland, T. Hain, G. Evermann and D. Povey, "CU-HTK March 2001 Hub5 system." *Proc. of DARPA Hub5E Conversational Speech Recognition Workshop*, 2001.

[9] O. Abdel-Hamid, A. Mohamed, H. Jiang and G. Penn, "Applying Convolutional Nerual Networks to Hybrid NN-HMM Model for Speech Recognition," *Proc. of ICASSP 2012*, Kyoto, Japan, 2012.