Restructuring of Deep Neural Network Acoustic Models with Singular Value Decomposition

Jian Xue, Jinyu Li, and Yifan Gong

Microsoft Corporation, One Microsoft Way, Redmond, WA 98052

{jianxue; jinyli; ygong}@microsoft.com

Abstract

Recently proposed deep neural network (DNN) obtains significant accuracy improvements in many large vocabulary continuous speech recognition (LVCSR) tasks. However, DNN requires much more parameters than traditional systems, which brings huge cost during online evaluation, and also limits the application of DNN in a lot of scenarios. In this paper we present our new effort on DNN aiming at reducing the model size while keeping the accuracy improvements. We apply singular value decomposition (SVD) on the weight matrices in DNN, and then restructure the model based on the inherent sparseness of the original matrices. After restructuring we can reduce the DNN model size significantly with negligible accuracy loss. We also fine-tune the restructured model using the regular back-propagation method to get the accuracy back when reducing the DNN model size heavily. The proposed method has been evaluated on two LVCSR tasks, with context-dependent DNN hidden Markov model (CD-DNN-HMM). Experimental results show that the proposed approach dramatically reduces the DNN model size by more than 80% without losing any accuracy.

Index Terms: deep neural network, singular value decomposition, model restructuring

1. Introduction

Recent significant progress in deep learning has attracted a lot of interest in automatic speech recognition (ASR) [1][2][3][4][5][6][7]. The discovery of strong modeling capability of deep neural network (DNN) and the availability of high-speed hardware has made it feasible to train huge networks with tens of millions of parameters. Neural networks used in acoustic models in ASR have usually been trained to perform frame classification with cross-entropy criterion or perform sequential training in recent studies [8][9]. In the framework of context-dependent DNN Hidden-Markov-Model (CD-DNN-HMM) [1][2], the conventional Gaussian Mixture Model (GMM) is replaced by a DNN to evaluate the senone log likelihood. Besides CD-DNN-HMMs, DNN can also be used to provide the bottle-neck feature vectors of the GMM in a GMM-HMM system [10][11] Both applications of DNN in ASR achieved significant accuracy improvement. CD-DNN-HMM has been shown to achieve relative 16% [2] and 33% [4] word error reduction over discriminatively trained CD-GMM-HMMs, on a voice search task and a switchboard task, respectively. The work in [10] shows that DNN trained bottle-neck feature reduces word error rate by 16% relatively on a large vocabulary business search task

However, the outstanding performance of CD-DNN-HMM accompanies with huge computation cost – the immense CPU and memory usage, since it uses much more parameters than traditional GMM-HMM framework. Although the DNN training can be speeded up tremendously with Graphics Processing Unit (GPU)[12], the use of GPU in deployment machines is implausible. Also, GPU is not always available on all types of hardware, especially for some small devices, which limits the application of DNN in a lot of scenarios.

Although the ASR recognition accuracy typically improves as the network depth and width increase, it is still shown that a large portion of weight parameters in DNN are very small [13], which have negligible effect on the output values of each layer. So we believe that the model can be compressed to a large extent. The work in [13] exploits the sparseness in DNN, and presents a nice way to reduce the model size. The shortcoming of [13] is that the non-zero parameters always distribute randomly on each layer, so we need to use indices for the non-zero values, which bring us extra memory usage. And the optimum implementation for the work heavily depends on hardware architecture, which makes it hard to port the system between different frameworks.

In this work we propose a new method to reduce DNN model size. We apply singular value decomposition (SVD) to decompose the weight matrices in DNN model, and then restructure the model based on the sparseness of the original format. The restructured model has similar layout as original model, with a couple of extra layers, so all the advanced speed-up methods, including streaming SIMD extension (SSE) instructions and GPU implementation [12][14], can be applied on top of it. After SVD restructuring, the total model size is reduced extensively with possible accuracy loss (depend on the extent we compress the model). Then we can fine-tune the model with restructured model format to improve the accuracy back.

The rest of the paper is organized as the following. Section 2 describes the structure of DNN used in ASR. Section 3 proposes how to apply SVD on weight matrices in DNN to restructure the model. Experimental results are presented in Section 4 to show the effectiveness of the proposed method. We conclude our work in Section 5.

2. DNN in Automatic Speech Recognition

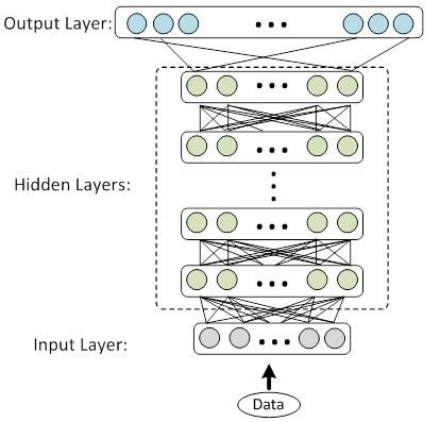


Fig. 1 DNN used in ASR systems

DNN is a feed-forward, artificial neural network which has more than one layer of hidden units between its inputs and outputs. Fig. 1 shows the structure of a DNN used in ASR systems, where the bottom layer is input layer, the mid-layers are hidden layers, and the top layer is output layer. Usually the network is fully connected between adjacent layers.

2.1 CD-DNN-HMM

The CD-DNN-HMM combines the discriminative modeling power of DNN with the sequential modeling power of HMM. In CD-DNN-HMM, we replace the GMM in a conventional GMM-HMM system with a DNN, in which the output layer consists of all the tied CD phone states (we also called senones), and each unit in the input layer corresponds to a data point in input feature vectors. We compute HMM's state emission probability density function $p_{x|y}(x|y=s)$ by converting the state posterior probability $p_{y|x}(y=s|x)$ obtained from the DNN to

$$p_{x|y}(x|y=s) = \frac{p_{y|x}(y=s|x)}{p_y(y=s)} \cdot p(x), \quad (1)$$

where s is a tied CD phone state, x is the input feature vector, $p_y(y=s)$ is the prior probability of state s, and $p(x)$ is independent of state s and can be crossed out during online evaluation.

We usually choose sigmoid function as the activation function of all hidden units, and softmax function for output layer units.

2.2 Training and Decoding

In our current implementation, CD-DNN-HMMs are initialized from traditional CD-GMM-HMMs. More specially, the CD-DNN-HMM inherits the model structure, including the phone set, HMM topology, and tying of context-dependent states, directly from the CD-GMM-HMM system. In addition, the senone labels used for training the DNNs are extracted from the forced alignment generated by CD-GMM-HMM. The detailed training procedure, including the bridge between CD-GMM-HMMs and momentum values used in the experiments, can be found in [2]. We also use GPU to speed up training.

Decoding is carried out by plugging the DNN into a conventional large vocabulary decoder.

3. SVD based Model restructuring

Currently used DNN in ASR system typically has 5-8 hidden layers, and each layer consists of a few thousands of units. With the same amount of training data, the DNN model usually has 2 to 10 times more parameters than traditional CD-GMM-HMMs. Also in CD-DNN-HMMs, the lower layers of DNNs are shared across all units in output layer and need to be calculated even only a small amount of states are active during search. Therefore, it is extremely important to reduce the DNN model size so that fast computation and small memory usage can be obtained for runtime evaluation, which is critical to real-world deployment.

Here we present a SVD based model restructuring method for DNN models. Fig. 2 depicts how to decompose a weight matrix into two matrices with smaller dimensions.

$$\begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix} = \begin{bmatrix} u_{11} & \cdots & u_{1n} \\ \vdots & \ddots & \vdots \\ u_{m1} & \cdots & u_{mn} \end{bmatrix} \cdot \begin{bmatrix} \epsilon_{11} & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & \cdots & \epsilon_{kk} & \cdots & 0 \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & \epsilon_{nn} \end{bmatrix} \cdot \begin{bmatrix} v_{11} & \cdots & v_{1n} \\ \vdots & \ddots & \vdots \\ v_{n1} & \cdots & v_{nn} \end{bmatrix}$$

$$\approx \begin{bmatrix} u_{11} & \cdots & u_{1n} \\ \vdots & \ddots & \vdots \\ u_{m1} & \cdots & u_{mn} \end{bmatrix} \cdot \begin{bmatrix} \epsilon_{11} & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & \cdots & \epsilon_{kk} & \cdots & 0 \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 \end{bmatrix} \cdot \begin{bmatrix} v_{11} & \cdots & v_{1n} \\ \vdots & \ddots & \vdots \\ v_{n1} & \cdots & v_{nn} \end{bmatrix}$$

$$= \begin{bmatrix} u_{11} & \cdots & u_{1k} \\ \vdots & \ddots & \vdots \\ u_{m1} & \cdots & u_{mk} \end{bmatrix} \cdot \begin{bmatrix} \epsilon_{11} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \epsilon_{kk} \end{bmatrix} \cdot \begin{bmatrix} v_{11} & \cdots & v_{1k} \\ \vdots & \ddots & \vdots \\ v_{k1} & \cdots & v_{kk} \end{bmatrix}$$

$$= \begin{bmatrix} u_{11} & \cdots & u_{1k} \\ \vdots & \ddots & \vdots \\ u_{m1} & \cdots & u_{mk} \end{bmatrix} \cdot \begin{bmatrix} n_{11} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & n_{kk} \end{bmatrix}$$

Fig. 2 SVD decomposition on weight matrices in DNN models

For a $m \times n$ weight matrix A, if we apply SVD on it, we get

$$A_{m\times n} = U_{m\times n} \sum_{n\times n} V^T_{n\times n}, \quad (2)$$

where $\sum$ is a diagonal matrix with A's singular values on the diagonal in the decreasing order. The m columns of U and the n columns of V are called the left-singular vectors and right-singular vectors of A, respectively. Since A is a sparse matrix, a large part of A's singular values should be very small. Fig. 3 illustrates the distribution of singular values for a 2048×2048 weight matrix in a 5-hidden-layer DNN, where x-axis is the number of singular values, and y-axis is the accumulated percentage of total singular values.

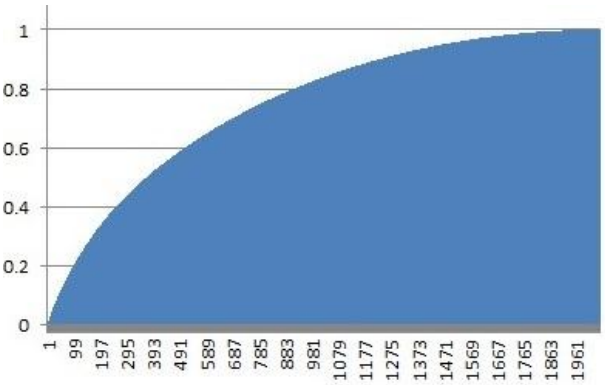


Fig. 3 Distribution of singular values for a weight matrix in a 5-hidden layer DNN

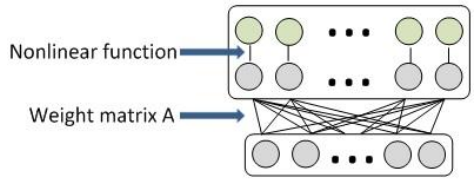
From Fig. 3 we can see that around 15% of singular values contribute 50% of total values, and around 40% of singular values contribute 80% of total values. So if we set those small values to 0, it won't considerably change the values of elements in matrix A. Assume we only keep k biggest singular values of A, we can rewrite formula (2) as

$$A_{m\times n} = U_{m\times k} \sum_{k\times k} V^T_{k\times k} = U_{m\times k} N_{k\times k}, \quad (3)$$

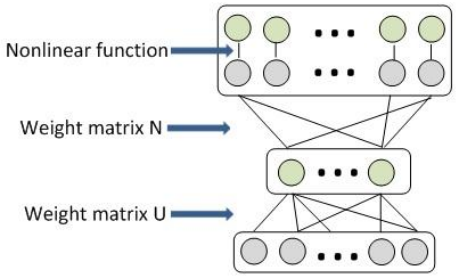
where $N_{k\times k} = \sum_{k\times k} V^T_{k\times k}$.

In this way we decompose matrix A into two smaller matrices U and N. Fig. 4 describes how we apply them back to the original DNN model. For one single layer in a DNN model, we replace it with two layers, while the first one has no nonlinear function, and the second one does. The number of parameters changes from mn to $(m+n)k$. We reduce the model size significantly if k is much smaller than n. In implementation, the value of k can be set to a pre-decided value. We can also choose the value of k so that a major part

of *A*'s singular values are kept.



(a) One layer in original DNN model



b) Two corresponding layers in new DNN model

Fig. 4 Model conversion in restructured DNN

If we restructure the DNN model aggressively with accuracy loss, we can also fine-tune the model with the same back-propagation method which is used to train original DNN model.

4. Experiments

Experiments were done on two different LVCSR tasks, to completely evaluate the proposed approach.

4.1 Experiments on the LVCSR task with single data resource

We first evaluated the proposed approach on a Microsoft internal task. The training data, called Train-1, consists of 750 hours of audios. The test set, called Test-1, has 31829 words in 9562 utterances.

The input feature to CD-DNN-HMM system is a 13-dimension mean-normalized MFCC feature with up to third-order derivatives. We augment the feature vectors with previous and next 5 frames (5-1-5). The speaker-independent 3-state cross-word triphones share 5976 senones, determined by the baseline CD-GMM-HMM system.

The original DNN used in CD-DNN-HMM has 5 hidden layers, each with 2048 units. The output layer has 5976 units corresponding to the 5976 senones. The DNN is initialized with DBN-pretraining procedure, and then refined with back-propagation using senone labels derived from the MLE model alignment [1].

We first apply SVD restructuring on the weight matrix below the output layer, since it's the largest one in the model. Table 1 summarizes the experimental results. The first column describes the setup of the model, and the number in bracket means that how many singular values we keep after SVD decomposition. The third column is the number of parameters in each model. For example, in the original DNN model the number of parameters is $572 \times 2048 + (2048 \times 2048) \times 4 + 2048 \times 5976 \approx 29\text{M}$.

Table 1 Results of SVD restructuring on output layer on task 1

Acoustic Model		WER	Number of parameters
Baseline, GMM model		29.1%	11M
Original DNN model		25.6%	29M
SVD (1024)		25.6%	25M
SVD (512)		25.7%	21M
SVD (256)	Before fine-tune	28.6%	19M
	After fine-tune	25.6%	

From Table 1 we can see that model size of our original DNN model is nearly 3 times of GMM model, which was trained with both feature space and model space discriminative training technology: feature minimum phone error (fMPE) and boosted maximum mutual information (BMMI). We reduce WER at 12% relatively by replacing GMM model with DNN model. The following rows in Table 1 verify the effect of the proposed approach. When we keep only 1/4 of largest singular values (the SVD-512 case) on the matrix below the output layer, WER is almost the same as the original model, while we reduce the overall model size around 30%. If we compress the model further, keeping only 1/8 of largest singular values, WER increases a lot, but the following fine-tuning can bring the accuracy back.

We also apply the method on other weight matrices, except the one above the input layer, since the number of parameters in this matrix is much smaller than others, and restructuring it does not affect the model size considerably. Table 2 summarizes the results.

Table 2 Results of SVD restructuring on the whole model on task 1

Acoustic model		WER	Number of parameters
All hidden layers (512)	Before fine-tune	26.0%	21M
	After fine-tune	25.6%	
All hidden layers (256)	Before fine-tune	27.0%	17M
	After fine-tune	25.8%	
All hidden and output layers (256)	Before fine-tune	29.7%	7M
	After fine-tune	25.4%	
All hidden and output layer (192)	Before fine-tune	36.7%	5.6M
	After fine-tune	25.5%	

From Table 2 we observe that when we apply SVD on all hidden layers and keep only 1/4 of largest singular values, WER increases less than 1% relatively. When we apply SVD on all hidden layers and keep 1/8 of largest singular values, we lose 40% of the accuracy improvement, but reduce model size by 41%. If we compress the model more aggressively, word accuracy will drop further after SVD decomposition, but fine-tuning can get the lost accuracy back. Our best setup (last row) shows that we can reduce the model size more than 80% without losing any accuracy, which only has half of parameters compared to GMM model.

To further verify the advantage of the proposed method, we also built a DNN model with the same model structure as SVD-256 case on all hidden and output layers from the beginning. The model got 26.3% WER, which is nearly 4% worse than the one using proposed method. Also the training converged slowly. The number of iterations needed to get the final model is more than the one we built the baseline model

plus the one we used to do fine-tuning after SVD restructuring.

4.2 Experiments on the LVCSR task with multiple data resources

Our second task is to train a CD-DNN-HMM model for two different scenarios by mixing the training data from these two scenarios. Model structure is the same as the one in last session. Besides the training set Train-1, we also have another commercial set called Train-2, which has 300 hours of audio. We use these two training sets to train a single DNN model for two scenarios. We evaluate the model on test set Test-1 and another one Test-2 on the second scenario, which has 16028 words in 2286 utterances.

We first apply SVD restructuring using the similar setup as the one used on last task. Table 3 shows us the results.

Table 3 Results of SVD restructuring on the whole model on task 2

Acoustic model		WER	
		Test-1	Test-2
Original DNN model		25.6%	21.0%
All hidden and output layers (512)	Before fine-tune	26.2%	22.8%
	After fine-tune	25.7%	21.0%
All hidden and output layers (256)	Before fine-tune	30.3%	26.3%
	After fine-tune	26.2%	21.3%
All hidden and output layer (192)	Before fine-tune	33.0%	29.1%
	After fine-tune	26.2%	21.5%

From Table 3 we observe that if we keep 512 singular values for each matrix, we can improve the accuracy back after fine-tuning. In this setup the model size is reduced by 55%. But if we compress the model further, keeping 256 or 192 singular values during SVD restructuring as in Table 2, we lose accuracy around 2.5% relatively even after fine-tuning. We believe the reason is that we have multiple data resource on this task, which is more variant than the single data resource. The DNN model learned from the mixed training data set thus needs more parameters to handle the variation, which reduces its sparseness.

Through the analysis of the singular values of each weight matrix, we found that their distributions are different. So keeping the same number of singular values for all the layers may not be the best way to compress the model. Table 4 gives us the detail information about it, where the second to fourth columns describe how many singular values contribute to the certain percentage of total singular values.

Table 4 Distribution of singular values in different weights matrices

Weight matrix	No. of singular values			
	20%	30%	40%	50%
Hidden layer 2	95	164	253	366
Hidden layer 3	98	171	263	378
Hidden layer 4	78	140	220	320
Hidden layer 5	87	155	241	343
Output layer	125	232	363	519

From Table 4 we can see that the singular values of weight matrix for output layer are more spread out than others, which means that the matrix is denser. In the following experiments we use the 40% setup to restructure the model and then do fine-tuning. The final model got 25.9% for WER on Test-1, and 21.0% on Test-2, which is similar with the setup that we keep 512 singular values for all the matrices. But we reduce the model size by another 40%, which is 73% reduction compared to the original model.

5. Conclusion

In this paper we described the work of DNN model restructuring. We apply SVD on the weight matrices in DNN, and then restructure the model based on the sparseness of the original matrices. The same accuracy as the original DNN can be maintained when we just apply modest parameter reduction. If number of parameters of the DNN is reduced heavily, we can fine-tune the model with the new structure to get the lost accuracy back. We evaluate the approach on two LVCSR tasks. On the first task with single training data resource we can reduce the model size by more than 80% without any accuracy loss. On another LVCSR task with multiple data resources, the restructured DNN needs additional parameters to handle the data source variability. By keeping the singular values proportional to the total of them in each layer, we can finally reduce the model size by 73% with less than 1% relative accuracy loss. These two experiments also show that the training data variability affects the capacity of DNN – more parameters are needed for restructured DNN to model the variability inside heterogeneous training data in addition to the phonetic variability.

6. Acknowledgements

The authors would like to thank Dong Yu for valuable discussion and the support on DNN training tool, Emilian Stoimenov for helpful discussion, and also thank Jui-Ting Huang for some experimental results.

7. References

[1] D. Yu, L. Deng, and G. Dahl, "Roles of pre-training and fine-tuning in context-dependent DBN-HMMs for real-word speech recognition," in *proceedings of NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2010.

[2] F. Seide, G. Li, and D. Yu, "Conversational speech transcription using context-dependent deep neural networks," *Proc. Interspeech*, pp. 437-440, 2011.

[3] N. Jaitly, P. Nguyen, A. Senior, and V. Vanhoucke, "An application of pretrained deep neural networks to large vocabulary conversational speech recognition,"*Proc. Interspeech*, 2012.

[4] T. N. Sainath, B. Kingsbury, and B. Ramabhadran, "Improvements in using deep belief networks for large vocabulary continuous speech rcognition," *Tech. Rep. UTML TR 2010-003*, Speech and Language Algorithm Group, IBM, February, 2011.

[5] T. N. Sainath, B. Kingsbury, B. Ramabhadran, P. Fousek, P. Novak, A.-r. Mohamed, "Making deep belief networks effective for large vocabulary continuous speech recognition," *Proc. ASRU*, pp. 30-35, 2011.

[6] G. E. Dahl, D. Yu, L. Deng, and A. Acero, "Large vocabulary continuous speech recognition with context-dependent DBN-HMMs," *Proc. ICASSP*, pp. 4688-4691, 2011.

[7] A. Mohamed, G. E. Dahl, and G. Hinton, "Acoustic modeling using deep belief networks," *IEEE Trans. Audio Speech and Language Process.*, vol. 20, no. 1, pp. 14-22, Jan. 2012.

[8] B. Kingsbury, T. Sainath, and H. Soltau, "Scalable minimum Bayes risk training of deep neural network acoustic models using distributed Hessian-free optimization," *Proc. Interspeech*, 2012.

[9] H. Su, G. Li, D. Yu, and F. Seide, "Error back propagation for sequence training of context-dependent deep networks for conversational speech transcription," *Proc. ICASSP*, 2013.

[10] D. Yu, and M. L. Seltzer, "Improved bottleneck features using pretrained deep neural networks," *Proc. Interspeech*, pp. 237-240, 2011.

[11] T. N. Sainath, B. Kingsbury, and B. Ramabhadran, "Auto-encoder bottleneck features using deep belief networks," *Proc. ICASSP*, pp. 4153-4156, 2012.

[12] K.-S. Oh, and K. Jung, "GPU implementation of neural networks," *Pattern Recognition*, vol. 37, issue 6, pp. 1311-1314, June 2004.

[13] D. Yu, F. Seide, G. Li, and L. Deng, "Exploiting sparseness in deep neural networks for large vocabulary speech recognition," *Proc. ICASSP*, pp. 4409-4412, 2012.

[14] V. Vanhoucke, A. Senior, and M. Z. Mao, "Improving the speed of neural networks on CPUs," in *proceedings of NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.