LOW-RANK MATRIX FACTORIZATION FOR DEEP NEURAL NETWORK TRAINING WITH HIGH-DIMENSIONAL OUTPUT TARGETS

Tara N. Sainath, Brian Kingsbury, Vikas Sindhwani, Ebru Arisoy, Bhuvana Ramabhadran

IBM T. J. Watson Research Center, Yorktown Heights, NY 10598
{tsainath, bedk, vsindhw, earisoy, bhuvana}@us.ibm.com

ABSTRACT

While Deep Neural Networks (DNNs) have achieved tremendous success for large vocabulary continuous speech recognition (LVCSR) tasks, training of these networks is slow. One reason is that DNNs are trained with a large number of training parameters (i.e., 10-50 million). Because networks are trained with a large number of output targets to achieve good performance, the majority of these parameters are in the final weight layer. In this paper, we propose a low-rank matrix factorization of the final weight layer. We apply this low-rank technique to DNNs for both acoustic modeling and language modeling. We show on three different LVCSR tasks ranging between 50-400 hrs, that a low-rank factorization reduces the number of parameters of the network by 30-50%. This results in roughly an equivalent reduction in training time, without a significant loss in final recognition accuracy, compared to a full-rank representation.

Index Terms— Deep Neural Networks, Speech Recognition

1. INTRODUCTION

Deep Neural Networks (DNNs) have become popular in the speech community over the last few years, showing significant gains over state-of-the-art GMM/HMM systems on a wide variety of small and large vocabulary tasks [1, 2, 3, 4]. However, one drawback of DNNs is that training is very slow, in part because DNNs can have a much larger number of parameters compared to GMMs [3, 4].

There have been a few attempts in the speech recognition community to reduce the number of parameters in the DNN without significantly decreasing final recognition accuracy. One common approach, known as "sparsificiation" [3], is to zero out weights which are close to zero. However, this reduces parameters after the network architecture has been defined, and therefore does not have any impact on training time. Second, convolutional neural networks (CNNs) [5] have also been explored to reduce parameters of the network, by sharing weights across both time and frequency dimensions of the speech signal. However, experiments show that in speech recognition, the best performance with CNNs can be achieved when matching the number of parameters to a DNN [6], and therefore parameter reduction with CNNs does not always hold in speech tasks.

In this paper, we explore reducing parameters of the DNN before training, such that overall training time is reduced but recognition accuracy is not significantly increased. Typically in speech, DNNs are trained with a large number of output targets (i.e., 2,000 - 10,000), equal to the number of context-dependent states of a GMM/HMM system, to achieve good recognition performance. Having a larger number of output targets contributes significantly to the large number of parameters in the system, as over 50% of parameters in the network can be contained in the final layer. Furthermore, few output targets are actually active for a given input, and we hypothesize that the output targets that are active are probably correlated (i.e. correspond to a set of confusable context-dependent HMM states). The last weight layer in the DNN is used to project the final hidden representation to these output targets. Because few output targets are active, we suspect that the last weight layer (i.e. matrix) has low rank. If the matrix is low-rank, we can use factorization to represent this matrix by two smaller matrices, thereby significantly reducing the number of parameters in the network before training. Another benefit of low-rank factorization for non-convex objective functions, such as those used in DNN training, is that it constrains the space of search directions that can be explored to maximize the objective function. This helps to make the optimization more efficient and reduce the number of training iterations, particularly for 2nd-order optimization techniques.

The use of low-rank matrix factorization for improving optimization problems has been explored in a variety of contexts. For example, in multivariate regression involving a large-number of target variables, the low-rank assumption on model parameters has been effectively used to constrain and regularize the model space [7], leading to superior generalization performance. DNN training may be viewed as effectively performing nonlinear multivariate regression in the final layer, given the data representation induced by the previous layers. Furthermore, low-rank matrix factorization algorithms also find wide applicability in matrix completion literature (see, e.g., [8] and references therein). Our work extends these previous works by exploring low-rank factorization specifically for DNN training, which has the benefit of reducing the overall number of network parameters and improving training speed.

Our initial experiments are conducted on a 50-hour English Broadcast News (BN) task [9], where a DNN is trained with 2,220 output targets. We show by imposing a rank of 128 on the final matrix, we can reduce the number of parameters of the DNN by 28% with no loss in accuracy. Furthermore, we show that when low-rank matrices are used with 2nd order Hessian-free sequence-training [10], we can reduce the overall number of training iterations by roughly 40%, leading to further training speed improvements. Second, we explore the behavior of low-rank factorization on two larger tasks with larger number of output targets: namely a 300-hour Switchboard (SWB) task with 9,300 output targets and a 400-hour English BN task with 6,000 output targets. On BN we can reduce the number of parameters of the network by 49% and for SWB by 32%, with nearly no loss in accuracy. Finally, we extend the use of low-rank factorization beyond acoustic modeling, exploring the versatility of the low-rank technique on DNNs used for language modeling (DNN-LM). We show that with low-rank factorization, we can reduce the number of parameters of a DNN-LM trained with 10,000 output targets by 45% without a significant loss in accuracy,

The rest of this paper is organized as follows. Low-rank matrix factorization is discussed in Section 2. Section 3 analyzes the low-

rank factorization on a small 50-hr BN task. Results using low-rank factorization for larger acoustic modeling tasks are discussed in Section 4 and for language modeling in Section 5. Finally, Section 6 concludes the paper and discusses future work.

2. LOW-RANK MATRIX FACTORIZATION

The left-hand side of Figure 1 shows a typical neural network architecture for speech recognition problems, namely 5 hidden layers with 1,024 hidden units per layer, and a softmax layer with 2,220 output targets. In this paper, we look to represent the last weight matrix in Layer 6, by a low-rank matrix. Specifically, let us denote the layer 6 weight by A, which is of dimension $m \times n$. If A has rank r, then there exists [11] a factorization $A = B \times C$ where B is a full-rank matrix of size $m \times r$ and C is a full-rank matrix of size $r \times n$. Thus, we want to replace matrix A by matrices B and C. Notice there is no non-linearity (i.e. sigmoid) between matrices B and C. The right-hand side of Figure 1 illustrates replacing the weight matrix in Layer 6, by two matrices, one of size $1,024 \times r$ and one of size $r \times 2,220$.

We can reduce the number of parameters of the system so long as the number of parameters in B (i.e., mr) and C (i.e., rn) is less than A (i.e., mn). If we would like to reduce the number of parameters in A by a fraction p, we require the following to hold.

$$mr + rn < pmn \tag{1}$$

solving for r in Equation 1 gives the following requirement needed to reduce overall parameters by fraction p.

$$r < \frac{pmn}{m+n} \tag{2}$$

In Sections 3, 4 and 5, we will discuss our choice of r for specific tasks, and the reduction in the number of network parameters.

3. LOW-RANK ANALYSIS

3.1. Experimental Setup

Our initial low-rank experiments are conducted on a 50 hour English Broadcast News (BN) transcription task [9] and results are reported on 100 speakers in the EARS `dev04f` set. The initial acoustic features are 13-dimensional MFCC features. An LVCSR recipe described in [12] is used to create a set of feature-space speaker-adapted (FSA) features, using vocal-tract length normalization (VTLN) and feature-space maxiumum likelihood linear regression (fMLLR).

All pre-trained DNNs use FSA features as input, with a context of 9 frames around the current frame. In [4], it was observed that a 5-layer DNN with 1,024 hidden units per layer and a sixth softmax layer with 2,220 output targets was an appropriate architecture for BN tasks. All DNNs are pre-trained generatively using the procedure outlined in [4]. During fine-tuning, the DNN is first trained using the cross-entropy (CE) objective function. With this criterion, after one pass through the data, loss is measured on a held-out set[1] and the learning rate is annealed (i.e. reduced) by a factor of 2 if the held-out loss has grown from the previous iteration [13]. Training stops after we have annealed the weights 5 times. After CE training, Hessian-free sequence-training [10] is performed to better adjust the weights for a sequence-level speech task.

[1]Note that this held out set is different than `dev04f`.

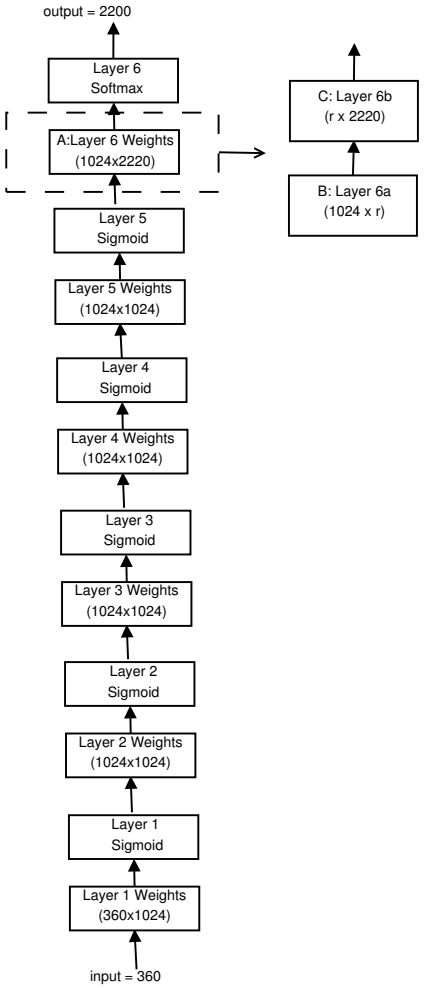

Fig. 1. Diagram of Deep Neural Network Architecture Commonly Used in Speech Recognition

3.2. Low-Rank for Cross-Entropy Training

We first explore the behavior of the low-rank network, and the appropriate choice of r, for a cross-entropy trained DNN. The baseline DNN system contains 6.8M parameters and has a word-error-rate (WER) of 17.7% on the `dev04f` set.

In the low-rank experiments, we replace the final layer full-rank matrix of size $1,024 \times 2,220$, with two matrices, one of size $1,024 \times r$ and one of size $r \times 2,220$. Table 1 shows the WER for different choices of the rank r and percentage reduction in parameters compared to the baseline DNN system. Furthermore, Figure 2 plots the relative spectral norm error between the final layer full-rank matrix and its best rank-r approximation, as a function of percentage of parameters reduced by using a rank-r matrix. The figure shows that between 0 to 30%, the increase in relative spectral norm error is not large, approximately 0.05. In addition, the table shows that with a 28% reduction in parameters (i.e., $r = 128$), the softmax outputs are not significantly effected and we can achieve the same WER of 17.7% as the baseline system.

Table 1. WER for Different Choices of Rank r on Softmax Layer

r = Rank	WER	# of Params (% Reduction)
Full Rank	17.7	6.8M
512	17.3	6.2M (10%)
256	17.6	5.4M (20%)
128	17.7	5.0M (28%)
64	18.0	4.8M (30%)
32	18.3	4.7M (31%)

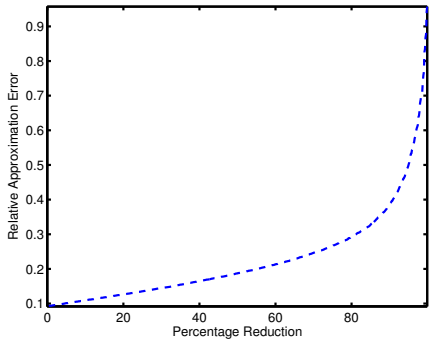

Fig. 2. Spectral Norm between Full and Rank r-Matrix

To further justify why a rank of 128 was feasible for this task, we took a full-rank matrix and explored how many total of the 2,220 output targets were above a threshold of 1e-03, which we will term as "active". Figure 3 plots the histogram of the number of active output targets for 50 utterances in the `dev04f` set. We can see that typically 128 or less output targets are active, illustrating why a rank of 128 was appropriate for the Broadcast News task.

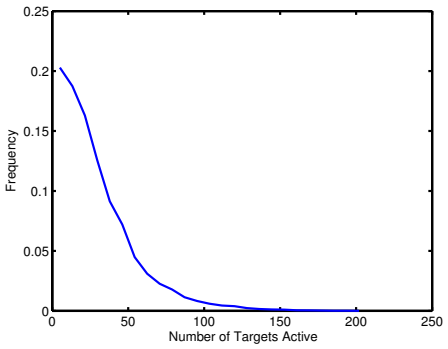

Fig. 3. Number of Active Output Targets

Finally, because the low-rank network has an extra weight multiplication compared to the full-rank system, we also check to make sure that the reduction in parameters with the low-rank method improves overall DNN training speed compared to the full-rank system. All timing experiments in this paper were run on an 8 core Intel Xeon X5570@2.93GHz CPU. Matrix/vector operations for DNN training are multi-threaded using Intel MKL-BLAS. We find that the full-rank system takes approximately 41 hours to train, while the low-rank system takes 33 hours to train, roughly a 20% speedup in overall training with low-rank. We do not find that the overall number of training iterations is reduced when using a low-rank versus full-rank approximation. One possible reason for this is that mini-batch SGD already helps to constrain search space as search directions are computed frequently from small batches, and therefore low-rank factorization does not help to further improve the optimization.

3.3. Low-Rank for Hessian-Free Sequence Training

Because sequence-training is often applied after cross-entropy training, we also explore network behavior with low-rank factorization for sequence training. Given that $r = 128$ was best architecture for cross-entropy training, we keep this architecture for sequence training. Table 2 shows the performance after sequence-training for the low-rank and full-rank networks. Notice that the WER of both systems is the same, indicating that low-rank factorization does not hurt DNN performance during sequence training. Furthermore, notice that the number of iterations for the low-rank system is significantly reduced compared to the full-rank system. With a second-order Hessian-free technique, the introduction of low-rank factorization helps to further constrain the space of search directions and makes the optimization more efficient. This leads to an overall sequence training time of 8.3 hours, a 55% speedup in training time compared to the full-rank system with a training time of 18.7 hours.

Table 2. Low-Rank for Sequence Training

Method	WER	# Iters	Training Time (hrs)
Full Rank	14.7	24	18.7
Low Rank (r=128)	14.7	14	8.4

3.4. Rank On Other Layers

We also check to see if network parameters can be further reduced by applying the low-rank methodology to other layers. For this experiment, we applied the low-rank factorization to the final hidden layer, which has dimension $1,024 \times 1,024$. Table 3 shows the WER and number of parameters for different rank choices of the hidden layer matrix. The baseline system has a low-rank of 128 applied on the final softmax layer, but no other factorization. As to be expected, the reduction in parameters with low-rank is smaller compared to Table 1 because the softmax weight layer is larger than the hidden weight layer. Notice that the WER goes up when rank factorization is used. This indicates that while the softmax-layer has a low-rank factorization, the hidden layers do not have the same properties.

Table 3. WER for Different Choices of r on Final Hidden Layer

r = Rank	WER	Number of Params (% Reduction)
Baseline	17.7	5.0M
256	18.0	4.5M (11%)
128	18.1	4.2M (16%)

4. RESULTS ON A LARGER TASK

To make sure that the low-rank factorization is generalizable, we also explored the behavior on two other larger data sets, which have even larger numbers of output targets.

4.1. 400 Hr Broadcast News

4.1.1. Experimental Setup

First, we explore scalability of the proposed low-rank factorization on 400 hours of English Broadcast News (BN) [9]. Results are re-

Table 4. WER and perplexity for different choice of rank r

r = Rank	WER		Perplexity		Params (% Reduction)
	DNN LM	DNN LM + 4-gram LM	DNN LM	DNN LM + 4-gram LM	
Baseline	20.8	20.5	102.8	92.6	8.2M
256	20.9	20.5	101.8	92.0	5.8M (29%)
128	21.0	20.4	102.8	91.9	4.5M (45%)

ported on the DARPA EARS `dev04f` set. The initial acoustic features are 19-dimensional PLP features. Again, FSA features are created by utilizing VTLN and fMLLR speaker-adaptation techniques, and are used as input to the DNN. All DNNs use FSA features as input, with a context of 9 frames around the current frame. The architecture consists of 5 hidden layers with 1,024 hidden units per layer and a sixth softmax layer with 5,999 output targets. DNN results are only reported for cross-entropy training, to demonstrate the parameter reduction with low-rank.

4.1.2. Results

Table 5 shows the WER and number of parameters for both the baseline and low-rank DNNs. Notice, we can reduce parameters by 49% without loss in accuracy. Furthermore, we find that training time for the full-rank DNN takes roughly 14.8 days, while training time for the low-rank DNN takes 7.7 days, roughly a 2-times speedup in training time with low-rank.

Table 5. Low-Rank Behavior, 400 Hr BN

Method	WER	# of Params (% Reduction)
Full-Rank DNN	16.7	10.7M
Low-Rank DNN (r=128)	16.6	5.5M (49%)

4.2. 300 Hr Switchboard

4.2.1. Experimental Setup

Second, we demonstrate scalability of the proposed low-rank factorization on 300 hours of conversational American English telephony data from the Switchboard (SWB) corpus. Results are reported on the `Hub5'00` set. Again, FSA features are created by utilizing VTLN and fMLLR speaker-adaptation techniques, and are used as input to the DNN. All DNNs use FSA features as input, with a context of 11 frames around the current frame. Following the setup in [3], the architecture consists of 6 hidden layers with 2,048 hidden units per layer and a seventh softmax layer with 9,300 output targets. Again, DNN results are only reported for cross-entropy training only, to demonstrate the parameter reduction with low-rank.

4.2.2. Results

Table 6 shows the WER and number of parameters for both the baseline and low-rank DNNs. For SWB, we find the best performance with a rank of 512, compared to 128 for the BN tasks. Notice, we can reduce parameters by 32% with very little loss in accuracy.

Table 6. Low-Rank Behavior, 300 Hr SWB

Method	WER	# of Params (% Reduction)
Baseline DNN	14.2	41M
Low-Rank DNN (r=512)	14.4	28M (32%)

5. LANGUAGE MODELING

We also explore low-rank factorization for DNNs in Language Modeling (DNN-LM) [2] using the set-up given in [14]. We apply low-rank factorization to the best scoring DNN architecture in [14], which consists of one projection layer where each word is represented with 120 dimensional features, three hidden layers with 500 hidden units per layer and a softmax layer with 10,000 output targets.

DNN language models are explored on a Wall Street Journal (WSJ) task [15]. The language model training data consists of 900K sentences (23.5M words). Development and evaluation sets consist of 977 utterances (18K words) and 2,439 utterances (47K words) respectively. Acoustic models are trained on 50 hours of Broadcast News. Baseline 4-gram language models trained on 23.5M words result in 22.3% WER on the evaluation set. DNN language models are evaluated using lattice rescoring. The performance of each model is evaluated using the model by itself and by interpolating the model with the baseline 4-gram language model. Interpolation weights are chosen to minimize the WER on the development set. The baseline DNN language model yields 20.8% WER by itself and 20.5% after interpolating with the baseline 4-gram language model.

In the low-rank experiments, we replace the final layer matrix of size $500 \times 10,000$, with two matrices, one of size $500 \times r$ and one of size $r \times 10,000$. Table 4 shows both the perplexity and WER on the evaluation set for different choices of the rank r and percentage reduction in parameters compared to the baseline DNN system.

The table shows that with a rank=128 in the interpolated model, we can achieve almost the same WER and perplexity as the baseline system, with a 45% reduction in number of parameters.

6. CONCLUSIONS

In this paper, we explored a low-rank matrix factorization of the final weight layer in a DNN. We explored this factorization for acoustic modeling on three different LVCSR tasks, including 50 and 400 hr English Broadcast News tasks and a 300 hr Switchboard telephony tasks. We found that this allowed us to reduce the number of parameters of the network between 30-50%, resulting in roughly a 30-50% speedup in training time, with little loss in final recognition accuracy compared to a full rank representation. Furthermore, we explored the low-rank factorization for language modeling, also demonstrating improvements in training speed with no loss in final WER.

7. ACKNOWLEDGEMENTS

The authors would like to thank Hagen Soltau, George Saon and Stanley Chen for their contributions towards the IBM toolkit and recognizer utilized in this paper.

[2] We used the term DNN-LM to be consisted with acoustic modeling. However, we did not use generative pre-training while training the language models.

8. REFERENCES

[1] G. Hinton, L. Deng, D. Yu, G. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury, "Deep Neural Networks for Acoustic Modeling in Speech Recognition," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.

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

[3] F. Seide, G. Li, and D. Yu, "Conversational Speech Transcription Using Context-Dependent Deep Neural Networks," in *Proc. Interspeech*, 2011.

[4] T. N. Sainath, B. Kingsbury, B. Ramabhadran, P. Fousek, P. Novak, and A. Mohamed, "Making Deep Belief Networks Effective for Large Vocabulary Continuous Speech Recognition," in *Proc. ASRU*, 2011.

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

[6] T. N. Sainath, A. Mohamed, B. Kingsbury, and B. Ramabhadran, "Exploring Convolutional Neural Networks for Large Vocabulary Speech Recognition," in *submitted to Proc. ICASSP*, 2013.

[7] M. Yuan, A. Ekici, Z. Lu, and R. Monteiro, "Dimension Reduction and Coefficient Estimation in Multivariate Linear Regression," *J. R. Statist. Soc. B.*, vol. 69, no. 3, pp. 329–346, 2007.

[8] B. Recht and C. Re, "Parallel Stochastic Gradient Algorithms for Large-Scale Matrix Completion," *Optimization Online*, 2011.

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

[10] B. Kingsbury, T. N. Sainath, and H. Soltau, "Scalable Minimum Bayes Risk Training of Deep Neural Network Acoustic Models Using Distributed Hessian-free Optimization," in *Proc. Interspeech*, 2012.

[11] G. Strang, *Introduction to Linear Algebra*, Wellesley-Cambridge Press, 4th edition, 2009.

[12] H. Soltau, G. Saon, and B. Kingsbury, "The IBM Attila speech recognition toolkit," in *Proc. IEEE Workshop on Spoken Language Technology*, 2010, pp. 97–102.

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

[14] E. Arisoy, T. N. Sainath, B. Kingsbury, and B. Ramabhadran, "Deep neural network language models," in *Proceedings of the NAACL-HLT 2012 Workshop*. June 2012, pp. 20–28, Association for Computational Linguistics.

[15] Stanley F. Chen, "Performance prediction for exponential language models," Tech. Rep. RC 24671, IBM Research Division, 2008.