RESHAPING DEEP NEURAL NETWORK FOR FAST DECODING BY NODE-PRUNING

Tianxing He *Yuchen Fan* *Yanmin Qian* *Tian Tan* *Kai Yu*

Institute of Intelligent Human-Machine Interaction
MOE-Microsoft Key Lab. for Intelligent Computing and Intelligent Systems
Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China
{cloudygoose,fyc0624,yanminqian,tantian,kai.yu}@sjtu.edu.cn

ABSTRACT

Although deep neural networks (DNN) has achieved significant accuracy improvements in speech recognition, it is computationally expensive to deploy large-scale DNN in decoding due to huge number of parameters. Weights truncation and decomposition methods have been proposed to speed up decoding by exploiting the sparseness of DNN. This paper summarizes different approaches of restructuring DNN and proposes a new *node pruning* approach to reshape DNN for fast decoding. In this approach, hidden nodes of a fully trained DNN are pruned with certain importance function and the reshaped DNN is retuned using back-propagation. The approach requires no modification on code and can directly save computational costs during decoding. Furthermore, it is complementary to weight decomposition methods. Experiments on a switchboard task shows that, by using the proposed node-pruning approach, DNN complexity can be reduced to 37.9%. The complexity can be further reduced to 12.3% without accuracy loss when node-pruning is combined with weight decomposition.

Index Terms— Deep Neural Networks, Node Pruning, Singular Value Decomposition, Speech Recognition

1. INTRODUCTION

The recently proposed Context-Dependent Deep-Neural-Network HMM (CD-DNN-HMM) has achieved significant improvement over the state-of-art CD-GMM-HMM for both phoneme recognition[1, 2] and large vocabulary continuous speech recognition(LVCSR) [3] tasks. It uses a deep neural network (DNN) to calculate posteriors of senones states and convert them into state-level conditional likelihood to be used in HMM. To ensure good recognition performance, deep neural network (DNN) used in speech recognition, especially in LVCSR, usually has a *wide* and *deep* structure. For example, a typical CD-DNN-HMM with 3000 senone states has 7 hidden layers with 2000 nodes per layer, leading to more than 30M parameters. The huge number of parameters incur significant computational costs (mainly due to large matrix multiplication) and memory requirements in decoding, which limits its application in real-world LVCSR[4]. Hence, there has been great interest in studying how to reduce the DNN model complexity while preserving its powerful modeling capacity[5, 6, 7], to speed up the decoding process.

Recent works all focused on exploiting the sparseness of weights in DNN, referred to as *weight operation* in this paper. One approach is direct *weight truncation*. It has been shown that by first throwing away the weights close to zero (up to 85% weights can be pruned) and then retuning the pruned network, DNN would not suffer any accuracy decline[5]. However, it is non-trivial to use this approach to actually speed up decoding and special code implementation is required. Another type of weight operation is to restructure DNN by exploiting redundancy of weights matrices calculation between layers. [7] aims to speed up both training and decoding by replacing the weight matrix below the output layer with the product of two small matrices. Further, [6] applies singular value decomposition (SVD) on all weight matrices of a fully trained DNN, and retune the restructured model. These approaches can dramatically reduce DNN complexity while keeping the original recognition performance.

In contrast to the previous *weight operation* approaches, this paper focuses on *node* level restructuring of DNN. The proposed algorithm reshapes a fully trained DNN by node-pruning, and then retunes it to prevent accuracy loss. Earlier works of node-pruning[8, 9, 10] were all applied to small-scale shallow neural networks with the motivation of alleviating the over-fitting problem for better performance. Most of early proposed importance functions examine the outputs of a node. It has not been investigated that how node-pruning can help reduce complexity of a large scale DNN. In this work, a new framework of node-pruning along with novel node importance functions are proposed for reshaping DNN. To the best of our knowledge, this is the first attempt to perform *node operation* in order to reduce complexity of DNN and apply it to LVCSR tasks. Considering that *node operation* is independent of weight operation, node-pruning can be com-

This work was supported by the Program for Professor of Special Appointment (Eastern Scholar) at Shanghai Institutions of Higher Learning, the China NSFC project No. 61222208 and JiangSu NSF project No. 201302060012.

bined with methods focusing on weights mentioned before to get additive improvements on model size reduction.

The remainder of the this paper is organized as follows. Section 2 describes the framework of node-pruning on DNN. Then experimental setup and results on TIMIT and switchboard English tasks are given in section 3, including results of combining node-pruning with the SVD method[6]. Section 4 concludes the paper and discusses future research directions.

2. RESHAPING DNN BY NODE-PRUNING

The basic assumption of the proposed node-pruning framework is that the DNN prior to pruning has redundant nodes. Since there is no sound theoretical guidance of choosing the the shape(number of hidden nodes of each layer) or scale to achieve optimal DNN structure, people normally employ certain level of redundancy to guarantee good decoding performance. Sometimes, the redundancy can be large, for example, DNN of 2048 nodes per layer may be adopted for the TIMIT experiments. This practical choice makes the assumption in this paper reasonable. Therefore, a pre-requisite of *node-pruning* is a sufficiently wide and deep neural network, referred to as *original DNN*. In this paper, the original DNN is the one after parameter fine-tuning using the back-propagation (BP) algorithm [1].

At each time instance, DNN accepts an input observation vector $\mathbf{o}$ and converts it to posterior probability $P(s|\mathbf{o})$, where s is usually a senone state. Assuming that the DNN has L *hidden* layers (layer l has N^l hidden nodes) with layer 0 as the input layer and layer $L+1$ as the output layer, for each input feature vector $\mathbf{o}$, the output of node i of hidden layer l, $y_i^l(\mathbf{o})$, can be computed as follows:

$$y_i^l(\mathbf{o}) = f\left(\sum_j w_{ji}^l y_j^{l-1}(\mathbf{o}) + b_i^l\right), \quad 1 \le l \le L \qquad (1)$$

where w_{ji}^l is the transition weight on the arc between node j of layer $l-1$ and node i of layer l, b^l is the bias vector of layer l, and f is the activation function, normally a sigmoid function, defined as:

$$f(x) = \frac{1}{1+e^{-x}} \qquad (2)$$

Node-pruning is only applied to hidden nodes in this work. Prior to pruning, it is necessary to evaluate the importance of each hidden node. In this paper, three *node importance functions* are proposed as below.

- *Entropy*

  The *entropy* importance function directly examines the activation distribution of each node. A node is *activated* if the output value is greater than a threshold (0.5 is used in this paper). Let $|\mathbf{O}|$ be the total number of input frame set $\mathbf{O}$, which is chosen to be the whole training dataset in this work. Let $a_i^l(\mathbf{O})$ and $d_i^l(\mathbf{O})$ be the total number of frames which activate or de-activate node i of layer l respectively ($a_i^l(\mathbf{O}) + d_i^l(\mathbf{O}) = |\mathbf{O}|$), the *entropy* importance function of node (l,i) is then defined as

  $$\mathcal{J}_e(l,i) = \frac{d_i^l(\mathbf{O})}{|\mathbf{O}|}\log_2\left(\frac{d_i^l(\mathbf{O})}{|\mathbf{O}|}\right) + \frac{a_i^l(\mathbf{O})}{|\mathbf{O}|}\log_2\left(\frac{a_i^l(\mathbf{O})}{|\mathbf{O}|}\right)$$

  The intuition is that if one node's outputs are almost identical on all training data, these outputs do not generate variations to later layers and consequently the node may not be useful. The *entropy* importance function is similar to the ones proposed in early works[10, 11] since it focuses on the outputs of a node.

- *Output-weights Norm (onorm)*

  onorm measures the importance of a node (l,i) by the average L1-norm of the weights of its outgoing links, which is formulated as

  $$\mathcal{J}_o(l,i) = \frac{1}{N^{l+1}}\sum_{j=1}^{N^{l+1}} |w_{ij}^{l+1}|$$

  where N^{l+1} is the number of nodes of the layer $l+1$. The intuition is that a node's importance should be reflected by its related weights after the BP process.

- *Input-weights norm (inorm)*

  Similiar to *onorm*, *inorm* reflects the importance of a node by the average L1-norm of the weights of its incoming links, which is formulated as

  $$\mathcal{J}_i(l,i) = \frac{1}{N^{l-1}}\sum_{j=1}^{N^{l-1}} |w_{ji}^l|$$

After training, a score is calculated for each hidden node using one of the above importance functions. Then all the nodes are sorted by their scores and nodes with less importance values are removed. Along with the removal of a node, all relevant incoming and outgoing links are also removed. It's worth noting that although every node removal will affect the scores of some other nodes, in this paper, all nodes are only examined once with the scores calculated using the original DNN.

In contrast to weight truncation or decomposition, directly pruning hidden nodes will result in significant performance degradation, as will be shown later in section 3.1. Hence, the pruned DNN needs to be re-finetuned. This means node-pruning requires more training time. However, different from other methods, the reshaped DNN can be directly used without any code or resources modification during decoding.

[1] Some early experiments on TIMIT showed that using the RBM-pretrained DNN as the original DNN resulted in noticeable performance degradation. Hence it is not further investigated in the paper.

3. EXPERIMENTS

Various aspects of the node-pruning approach was first investigated in detail using the TIMIT corpus. With the best importance function chosen from experiments on TIMIT, the approach was applied to a LVCSR task: switchboard English. Combination with SVD based weight decomposition was then performed and achieved the largest reduction of DNN complexity without significant performance degradation.

Note that *"complexity"* of DNN here refers to the total number of parameters of the weight transition matrices, which was also adopted in [12]. The *complexity* is proportional to the computational cost of calculating state posteriors from DNN and directly reflects decoding speed of a DNN-based speech recognition system.

3.1. Experiments on TIMIT

The proposed node-pruning approach was first investigated on a phone recognition task using the TIMIT corpus. The standard training set consisting of 462 speakers was used for RBM pre-training and DNN fine-tuning, and the 24-speaker core test set was used for evaluation. 40-dimensional Mel-scale filter bank coefficients along with their first and second derivatives were used as features. 11 consecutive frames were concatenated to form the input vectors of DNN. 183 target classes labels (3 states/phone, 61 phones) and mini-batches of size 128 were used during training.

The RBM-pretrained DNN was fine-tuned with cross-entropy objective function, along with a L2-norm weight-dacay term of coefficient 10^{-6}, a learning rate annealing and early stopping strategies as in [1] was used. After decoding, the 61 phone classes were mapped to a standard set of 39 classes for scoring. A 5-layer DNN with 1024 nodes per layer was trained as baseline. The phone error rate (PER) is 20.79% and the complexity is 5.76M parameters.

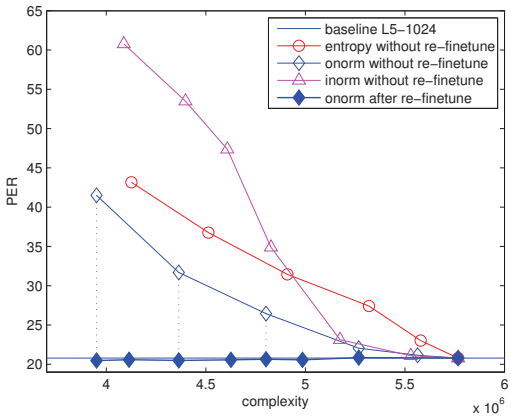

Fig. 1: Performance of node-pruned DNN without re-finetuening on TIMIT.

To investigate how much damage would node-pruning do to the baseline DNN, results of node-pruning with and without re-finetuning were compared. 100, 250, 500, 750 hidden nodes were pruned using the three importance functions defined in section 2 respectively. Figure 1 shows that without retraining, PER of the pruned DNN (1000 nodes pruned) increases from 20.79% to around 40%. The severe performance degradation has not been observed in weight decomposition methods[6]. This implies that node pruning changes DNN structure more significantly and retraining is necessary to obtain comparable performance to the original DNN. This is justified by the performance of the *onorm* node pruning with retraining in figure 1, which yielded almost the same PER as the baseline DNN.

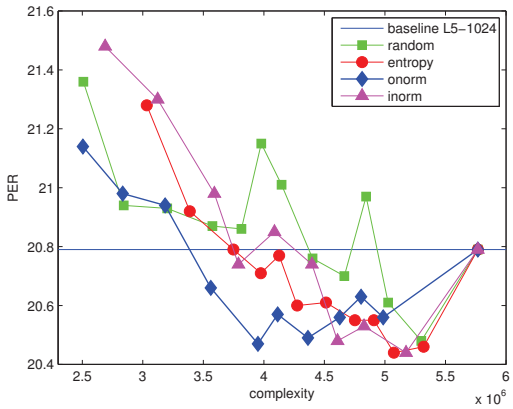

Fig. 2: Performance of different node importance functions after re-finetuning on TIMIT

To investigate the impact of using different importance functions, the three importance functions defined in section 2 and a random score function were evaluated respectively. The results are illustrated in figure 2. It is shown that node-pruning with the proposed importance functions can get the same or even better performance compared to the baseline DNN after retraining, and *onorm* still has the best performance. In contrast, random importance function generally performed worse than the proposed ones and showed instability in performance change. This demonstrates that the proposed ones give better and more stable indications about the importance of hidden-nodes. From the above results, node pruning can effectively obtain 37.50% complexity reduction (from 5.6M to 3.5M) without hurting the decoding performance. Since *onorm* showed the best and most stable performance, it was used for the following experiments.

3.2. Experiments on Swithboard English

In this section node-pruning is evaluated on a LVCSR task, Switchboard English. For fast experiments, a subset consisting of data of 810 speakers(approximately 50 hours) was first selected from the whole 309 hours data set for training. 13-dimensional PLP features with per-speaker CMN and CVN, along with first and second derivatives were extracted. Cross-word triphone models with 3001 tied-states was used. State alignment for DNN training was generated using a GMM model. A trigram language model which was trained on

the transcription of the 2000h Fisher corpus and interpolated with a background trigram model was used for decoding. For all the experiments, the switchboard (swb) part of the Hub5'00 data set and the fisher (fsh) part of the RT03S data set were used as the test set in this paper, which is the same as [13]. The baseline DNN has 7 hidden layers of 2048 nodes per layer. Node pruning with *onorm* importance function was applied and the results of the subset and full training set are shown in table 1.

System	#Pruned Nodes	DNN Complexity	WER (%)	
			swb	fsh
Baseline	—	32.2M	25.7	28.8
Node-Pruning	6500	13.1M	25.6	28.9
	7000	12.2M	25.5	28.8
	8000	10.4M	26.0	29.1

Table 1: Word error rate(WER) versus pruned complexity on switchboard 50 hours data

It can be seen that with 7000 hidden nodes pruned, about 62.1% DNN complexity reduction is obtained on the 50-hour task without any performance loss. This again demonstrates the effectiveness of the node-pruning approach.

3.3. Combination with Weight Matrices Decomposition

As stated in section 1, node-pruning is complementary to weight matrices decomposition methods and may be combined to get better complexity saving. This is investigated on the 50 hours switchboard English task in this section. Here, SVD[6] was applied on the weight matrices of a node-pruned DNN. For comparison, standard SVD restructuring with different ranks was also performed. Note that SVD was applied on all matrices except the one above the input layer.

SVD Rank	#Pruned Nodes	DNN Complexity	WER (%)	
			swb	fsh
192	—	6.5M	25.3	28.3
128	—	4.6M	25.9	29.0
192	6500	3.95M	25.7	28.5

Table 2: WER versus pruned SVD complexity on switchboard 50 hours data

Table 2 shows that DNN complexity can be effectively reduced to 20.1% (rank 192) by pure SVD restructuring. SVD with smaller rank will further reduce complexity at a cost of slight WER increase. By combining SVD rank-192 with node-pruning[2], a more compact DNN (87.7% complexity reduction) can be obtained. It is worth noting that the resultant compact DNN is smaller than SVD rank-128 and suffer no accuracy loss from baseline DNN(table 1), which means

[2]SVD is only applied on matrices that has fairly large size (a r-rank SVD can reduce complexity of a $m \times n$ matrix only if $m \times r + r \times n < m \times n$)

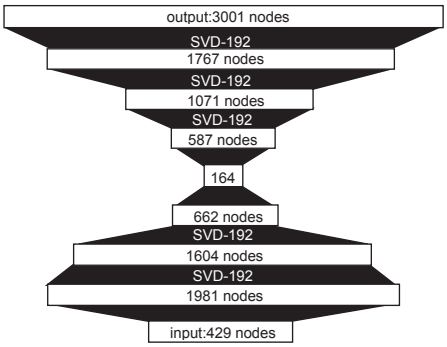

Fig. 3: illustration of node-pruning combined with SVD restructuring

node-pruning combined with SVD can achieve better tradeoff between complexity and performance.

One interesting observation is that, in all the node-pruning experiments(especially on SWB task), without explicitly restraining the layer size, the reshaped DNN reveals an extremely small middle hidden layer as shown in figure 3. This natural bottleneck shape not only implies some rationality of using bottleneck features, but also may reveal new properties of DNN. The relevant investigation will be a future work.

4. CONCLUSION AND FUTURE WORK

In this work, a new framework of reshaping DNN by node-pruning is proposed, which is complementary to the previously proposed approaches based on weight operation. Several novel node importance functions are proposed and experimented on the TIMIT task. The L1 norm of outgoing links, referred to as *onorm*, is shown to be most effective. With *onorm* node-pruning, on a 50 hours switchboard English task, the DNN complexity can be reduced to 37.9% without losing any performance. By further combination with SVD based weight matrices decomposition, up to 87.7% DNN complexity reduction can be achieved without affecting the recognition accuracy. The node pruning also reveals natural bottle-neck shape of DNN, which will be investigated in the future.

It is worth noting that the complexity reduction depends on the redundancy of the DNN. Node-pruning experiments on the 309 hours SWB data set have also been conducted and achieved exactly 50% complexity reduction with a bottleneck middle layer of 547 nodes. More sophisticated investigation on the model redundancy issue will also be carried in the future.

5. ACKNOWLEDGEMENTS

We would like to thank Jian Xue for valuable discussions about the SVD method.

6. REFERENCES

[1] Abdel rahman Mohamed, George E. Dahl, and Geoffrey E. Hinton, "Acoustic modeling using deep belief networks," *IEEE Transactions on Audio, Speech & Language Processing*, vol. 20, no. 1, pp. 14–22, 2012.

[2] Abdel rahman Mohamed, Dong Yu, and Li Deng, "Investigation of full-sequence training of deep belief networks for speech recognition," in *Proc. InterSpeech*, 2010.

[3] George E. Dahl, Dong Yu, Li Deng, and Alex Acero, "Context-dependent pre-trained deep neural networks for large vocabulary speech recognition," *IEEE Transactions on Audio, Speech, & Language Processing*, vol. 20, pp. 30–42, 2012.

[4] Vincent Vanhoucke, Andrew Senior, and Mark Z. Mao, "Improving the speed of neural networks on cpus," in *Proc. Deep Learning and Unsupervised Feature Learning Workshop, NIPS*, 2011.

[5] Dong Yu, Frank Seide, Gang Li, and Li Deng, "Exploiting sparseness in deep neural networks for large vocabulary speech recognition," in *Proc. ICASSP*, 2012.

[6] Jian Xue, Jinyu Li, and Yifan Gong, "Restructuring of deep neural network acoustic models with singular value decomposition," in *Proc. InterSpeech*, 2013.

[7] Tara N. Sainath, Brian Kingsbury, Vikas Sindhwani, Ebru Arisoy, and Bhuvana Ramabhadran, "Low-rank matrix factorization for deep neural network training with high-dimensional output targets," in *Proc. ICASSP*, 2013.

[8] Russel Reed, "Pruning algorithms-a survey," *IEEE transactions on Neural Networks*, vol. 4, no. 5, pp. 740–747, 1993.

[9] Bruce E. Segee and Michael J.Carter, "Fault tolerance of pruned multilayer networks," *Neural Networks*, vol. 2, pp. 447–452, 1991.

[10] Shinji Yamamoto, T.Oshino, T.Mori, A.Hashizume, and J.Motoike, "Gradual reduction of hidden units in the back propagation algorithm, and its application to blood cell classification," in *Proc. International Joint Conference on Neural Networks*, 1993.

[11] J. Sietsma and R. J. F. Dow, "Creating artificial networks that generalize," *Neural Networks*, vol. 4, pp. 67–69, 1991.

[12] Oriol Vinyals and Nelson Morgan, "Deep vs. wide: Depth on a budget for robust speech recognition," in *Proc. InterSpeech*, 2013.

[13] Frank Seide, Gang Li, Xie Chen, and Dong Yu, "Feature engineering in context-dependent deep neural networks for conversational speech transcription," in *Proc. ASRU*, 2011.