Dynamic and Adaptive Threshold for DNN Compression from Scratch

Chunhui Jiang(✉), Guiying Li, and Chao Qian

School of Computer Science and Technology,
USTC-Birmingham Joint Research Institute in Intelligent Computation and Its Applications (UBRI), University of Science and Technology of China, Hefei 230027, Anhui, People's Republic of China
{beethove,lgy147}@mail.ustc.edu.cn, chaoqian@ustc.edu.cn

Abstract. Despite their great success, deep neural networks (DNN) are hard to deploy on devices with limited hardware like mobile phones because of massive parameters. Many methods have been proposed for DNN compression, i.e., to reduce the parameters of DNN models. However, almost all of them are based on reference models, which were firstly trained. In this paper, we propose an approach to perform DNN training and compression simultaneously. More concretely, a dynamic and adaptive threshold (DAT) framework is utilized to prune a DNN gradually by changing the pruning threshold during training. Experiments show that DAT can not only reach comparable or better compression rate almost without loss of accuracy than state-of-the-art DNN compression methods, but also beat DNN sparse training methods by a large margin.

Keywords: Deep neural networks · Pruning · DNN compression

1 Introduction

In last few years, deep neural networks (DNNs) have made impressive performance on various artificial intelligence tasks like image classification [1,2], speech recognition [3] and natural language processing [4]. Thus DNNs are promising to apply to all kinds of intelligent devices. However, DNNs usually have massive parameters, which result in severe memory overhead and energy consumption [5]. For example, the number of parameters in two popular DNN models AlexNet [1] and VGG16 [6] are 61 million and 138 million, and the storage cost are 240 MB and 550 MB respectively, such a large memory usage prevents DNNs to deploy on mobile devices with limited hardware like smart phones. Therefore, reducing the parameters of DNNs without significant performance degradation is important for DNN coming into utility.

It was shown that there existed high redundancy in the overall parameters of a DNN [7], which means only a small part of parameters are needed to guarantee the performance in testing. Since then a lot of methods are proposed to compress DNNs, almost all of them operate on the reference models like tensor

Y. Shi et al. (Eds.): SEAL 2017, LNCS 10593, pp. 858–869, 2017.
https://doi.org/10.1007/978-3-319-68759-9_70

decomposition, parameter sharing and network pruning, noting that reference models were firstly trained on the corresponding datasets. However, for a new dataset, it is very likely that there are no reference models which are trained on it, and the training of reference models themselves are time-consuming. For example, in [5], the training of AlexNet reference model took 75 h on NVIDIA Titan X, and the subsequent compression process took extra 173 h. So if the two previously separated steps, training of reference model and compression, can be combined as one, then significant time will be saved.

Network pruning is quite effective in compressing DNN models. The pruning thresholds play a key role in network pruning. The state-of-the-art pruning method, dynamic network surgery [8], computes pruning thresholds on reference models and keeps them fixed throughout the compression phase. With reference model, this fixed threshold scheme is effective because the weights distribution has little change during model compression. But it is not true for training, for example, either Xavier initialization [9] for LeNet-5 or Gaussian initialization for CIFAR10_CAFFE [10], is far away from compression model as shown in Fig. 1, which means their pruning thresholds are invalid in the training phase.

In this paper, we propose a dynamic and adaptive threshold framework (DAT) by which DNN training and compressing can be performed simultaneously. Dynamic threshold means we prune parameters gradually. Only a small part of weights is pruned at the beginning, but as the training goes on, more and more parameters are pruned. Compared to fixed threshold scheme which prunes huge number of weighs at the beginning, dynamic threshold is more reasonable

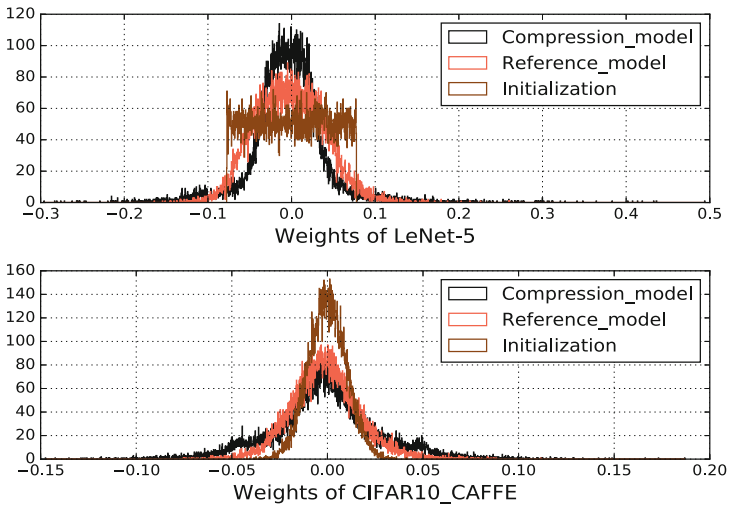


Fig. 1. Weights histogram of LeNet-5 (*top*) and CIFAR10_CAFFE (*bottom*). X axis is the range of weight values, and Y axis means the number of this value. Note that we only plot weights distribution of convolutional layers 2 (conv2) in above two models for demonstration, the similar pattern would be found in other layers.

because it is hard to judge the importance of initialized weights, and a good way is waiting for a few iterations as dynamic threshold does. Adaptive threshold means the thresholds change according to the weights distribution, noting that the weights distribution changes drastically over iteration in the training phase. In DAT framework dynamic threshold allows coarse-grained pruning and adaptive threshold allows fine-grained pruning. The experiments show that our proposed DAT framework is effective.

The rest of the paper is as follows. Section 2 presents related works. Preliminary and our proposed method are introduced in Sect. 3. In Sect. 4, we conduct comprehensive experiments and report the results. Conclusion and future work are made in Sect. 5.

2 Related Work

There have been many works aiming at DNN compression. [13] employed a fixed point implementation of DNN to replace float point scheme; [14] directly used binary weighs to alleviate the complexity of networks. The other works can be roughly divided into three categories: weight sharing, tensor decomposition and network pruning.

Weight sharing means grouping weights, within each group only one value or vector is needed like vector quantization [15] and HashedNets [16], which shares weights using hash buckets. Tensor decomposition is another way of DNN compression. [7,17] resort to low rank approximation of each layer matrix, [18] included the nonlinear part in the decomposition, [19] used a global error reconstruction to replace previous layer-wise decomposition. However, weight sharing and tensor decomposition will lose severe accuracy if high compression rate is needed.

By contrast, network pruning is a more promising DNN compression method, mainly because pruning and network retraining can be combined perfectly. Without loss of accuracy, network pruning could reach a very high compression rate. Moreover, the sparse matrix after pruning can be accelerated by extra hardware [20]. Network pruning set the unimportant weights to zero values, Optimal Brain Damage [21] and Optimal Brain Surgeon [22] computed hessian matrix to evaluate weights importance, but they need high computation overhead especially for DNNs. Magnitude-based methods prune weights whose absolute values are less than thresholds. This method, although simple, has the computational complexity of $\mathcal{O}(n)$, where n is the number of weights, thus is widely adopted by [5,8,23]. [23] employed a structured sparsity learning method to compress the convolutional layers, but as we know that most parameters of DNN are distributed in fully connected layers. In [5] pruning and retraining are repeated iteratively to reach a higher compression rate. However, the weight is discarded forever if be pruned, which constrain the compression and cause inefficient learning. This problem was solved by [8], which proposed dynamic network surgery and allowed the recovery of incorrect pruning. By this way, dynamic network surgery achieves state-of-the-art compression rate.

It should be emphasized that all methods of these three categories need reference models. Recently [12] studied l_1 and l_0 regularization of DNN systematically, and concluded that l_1 and l_0 regularization could lead to considerably sparse DNN. Note that our proposed DAT also works at the training stage, so we will compare DAT with regularization techniques in Sect. 4.

3 Dynamic and Adaptive Threshold (DAT)

In this section, we will first introduce dynamic network surgery (DNS) [8] in Preliminaries, then highlight the dynamic and adaptive threshold framework. Note that the DAT framework can also be combined with other pruning methods in addition to dynamic network surgery.

3.1 Preliminaries

Dynamic network surgery maintains a mask $\boldsymbol{M}_l$ for weight matrix $\boldsymbol{W}_l$ in layer l, where $1 \leq l \leq L$. $\boldsymbol{M}_l$ takes binary values in which 0 indicates being pruned and 1 indicates being kept, so $\boldsymbol{W}_l$ would keep dense throughout the compression. At the beginning of one iteration, $\boldsymbol{M}_l$ is updated as:

$$\boldsymbol{M}_l^{(i,j)} = \begin{cases} 0, |\boldsymbol{W}_l^{(i,j)}| < t_l \\ 1, |\boldsymbol{W}_l^{(i,j)}| \geq t_l \end{cases} \tag{1}$$

and the change of $\boldsymbol{M}_l^{(i,j)}$ from 0 to 1 means splicing, which is vital for DNS. t_l is a layer-wise threshold that is computed before model compression and keep fixed during the whole compression. After the update of $\boldsymbol{M}_l^{(i,j)}$, weight matrix $\boldsymbol{W}_l$ is wrapped into $\boldsymbol{H}_l = \boldsymbol{W}_l \bigodot \boldsymbol{M}_l$, noting that $\boldsymbol{H}_l$ is not dense because $\bigodot$ indicates element-wise product. It is $\boldsymbol{H}_l$, not $\boldsymbol{W}_l$ which plays the role of "weights" in forward and backward propagation until the update of $\boldsymbol{W}_l$:

$$\boldsymbol{W}_l^{(i,j)} \leftarrow \boldsymbol{W}_l^{(i,j)} - \alpha \frac{\partial Loss}{\partial (\boldsymbol{W}_l^{(i,j)} \boldsymbol{M}_l^{(i,j)})} \tag{2}$$

Note that Eq. 2 is not a standard gradient descent algorithm because the gradient is partial derivative of loss function to $\boldsymbol{H}_l$ but not $\boldsymbol{W}_l$, on the other hand, Eq. 2 also updates the previously pruned entries in $\boldsymbol{W}_l$, which makes the model compression dynamic.

3.2 Dynamic and Adaptive Threshold

Recall that our goal is to perform DNN training and compression simultaneously, now we elaborate how to achieve this goal by introducing a dynamic and adaptive threshold framework. The fixed threshold scheme computes t_l as:

$$t_l = \mu_l + c_l \sigma_l \tag{3}$$

where $\mu_l = \mathbb{E}(|\boldsymbol{W}_l|)$, $\sigma_l = \sqrt{Var(|\boldsymbol{W}_l|)}$ and c_l is a layer-wise hyper-parameter. Obviously this scheme is suitable to Gaussian-like distribution, which is true for reference models, but not true for initialization models. For example, Xavier initialization [9], which is used in standard LeNet-5 [11] training procedure of Caffe [10], initializes $\boldsymbol{W}_l$ by a uniform distribution $\boldsymbol{W}_l \sim U[-a, a]$, in this case, we have $\mu = \frac{a}{2}$, $\sigma = \frac{\sqrt{3}}{6}a$, so the choice of c need to be specially careful, $c \geq 2$ will cause $t = \mu + c\sigma > a$ and then all weights in this layer will be pruned!

The other problem of fixed threshold scheme is over pruning at the beginning. After initialization, the magnitude of weights distributes randomly, it is difficult to evaluate the importance of weights. If we prune a large part of weights this moment as fixed threshold scheme does, many important weights will be pruned and the performance of DNN models may suffer serious damage. We propose that these problems can be solved by making c_l **dynamic**. More concretely, c_l should be a function of iteration i. It is expected that c_l would range from $-c_{l,max}$ to $c_{l,max}$ as the iteration increases. We find the transformed $tanh$ function meeting our expectation perfectly, that is,

$$c_l(i) = c_{l,max} \cdot tanh(\frac{i - i_0}{\lambda}) \tag{4}$$

in which i_0 makes $c_l(i)$ equal to 0, and λ is a scaling factor. It should be noted that the threshold is low at the beginning, indicating only few weights are pruned. i_0 is an important demarcation point. When $i < i_0$, $c_l(i)$ takes negative values and the threshold t_l takes no more than μ_l, in this case, not too much weights are pruned. However, if $i \geq i_0$, t_l increases rapidly and approaches its extreme value, and most weights would be pruned. So in order to waiting for weights to exhibit their own importance, i_0 can not be too small. In this work, we empirically set i_0 to $i_{max}/4$. Note that there may exist other functions that are suitable for $c_l(i)$. However, in this work, we mainly want to demonstrate the effectiveness of a dynamic strategy, but not which strategy is best.

Considering the dramatic change of weight distribution during training, the fixed μ_l and σ_l computed from the initialization cannot cover the whole compression, which is different from compression with reference model, so μ_l and σ_l need to be **adaptive** instead of staying fixed. That is, μ_l and σ_l should also be the function of iteration i. Perhaps the easiest way to threshold adaption is employing

$$\mu_l(i) = \mathbb{E}(|\boldsymbol{W}_l(i)|) \tag{5}$$

$$\sigma_l(i) = \sqrt{Var(|\boldsymbol{W}_l(i)|)} \tag{6}$$

however, this strategy will cause drastic change of DNN structure, resulting in training inefficiency. So we adopt a smoother adaption method:

$$\mu_l(i) \leftarrow \mu_l(i) + \epsilon \cdot \Delta\mu_l(i) \tag{7}$$

$$\sigma_l(i) \leftarrow \sigma_l(i) + \epsilon \cdot \Delta\sigma_l(i) \tag{8}$$

in which $\Delta\mu_l(i)$ and $\Delta\sigma_l(i)$ are computed as:

Algorithm 1. Dynamic and Adaptive Threshold (DAT)

Input: $\{\boldsymbol{X}, \boldsymbol{y}\}$: training datum, i_{max}: maximum iterations
Output: $\boldsymbol{W}, \boldsymbol{M} = \{\boldsymbol{W}_l, \boldsymbol{M}_l : 1 \leq l \leq L\}$: weight and mask matrices of the compressed model

```
Initialize: W_l, μ_l = 𝔼(|W_l|), σ_l = sqrt(Var(|W_l|)), ∀1 ≤ l ≤ L
repeat
  for l = 1 to L do
    Compute Δμ_l(i), Δσ_l(i) by Eq. 9, Eq. 10
    Update μ_l(i), σ_l(i) by Eq. 7, Eq. 8
    Compute c_l(i) by Eq. 4
    Compute threshold t_l(i) = μ_l(i) + c_l(i) · σ_l(i)
  end for
  Update M by Eq. 1 and forward propagation
  Back propagation and update W by Eq. 2
  i = i + 1
until convergence or i ≥ i_max
```

$$\Delta\mu_l(i) = (\mathbb{E}(|\boldsymbol{W}_l(i)|) - \mathbb{E}(|\boldsymbol{W}_l(i-1)|)) \cdot e^{i/i_{max}} \tag{9}$$

$$\Delta\sigma_l(i) = (\sqrt{Var(|\boldsymbol{W}_l(i)|)} - \sqrt{Var(|\boldsymbol{W}_l(i-1)|)}) \cdot e^{i/i_{max}} \tag{10}$$

Now $\mu_l(i)$ and $\sigma_l(i)$ can still be updated by the change of weight distribution, but the step is controlled by ϵ, thus this strategy is smoother. The exponential term is designed to impose more importance to the update near maximum of iteration, because at that time $c_l(i)$ becomes almost saturated, and update of μ_l and σ_l should be more sensitive to the change of weights distribution. The whole algorithm procedure is summarized in Algorithm 1.

4 Experiments

In this section, comprehensive experiments are conducted on MNIST [11] and CIFAR-10 [26] to evaluate the performance of our proposed DAT framework. We claim that DAT is firstly a DNN compression method, so we compare DAT with state-of-the-art DNN compression methods, which include iterative network pruning (INP) [5] and dynamic network surgery (DNS) [8], noting that INP and DNS need reference models but DAT does not need. Then we compare DAT with DNN sparse training methods like l_1 and l_0 regularization [12].

4.1 Experimental Setting

Datasets and Reference Models. Both MNIST and CIFAR-10 have 50000 training images and 10000 testing images of 10 classes. MNIST is a handwritten digits database and CIFAR-10 is a natural images database. For MNIST, the classical LeNet-5 [11] is trained on it, and LeNet-5 has 4 learnable layers. For CIFAR-10, we choose CIFAR10_CAFFE which is defined in Caffe [10] and has 5 learnable layers. Both reference models are trained using standard protocols in

Caffe community. Finally, the LeNet-5 reference model achieves an accuracy rate of **99.09%** using 10000 iterations, and the CIFAR10_CAFFE reference model achieves **75.54%** accuracy rate using 5000 iterations.

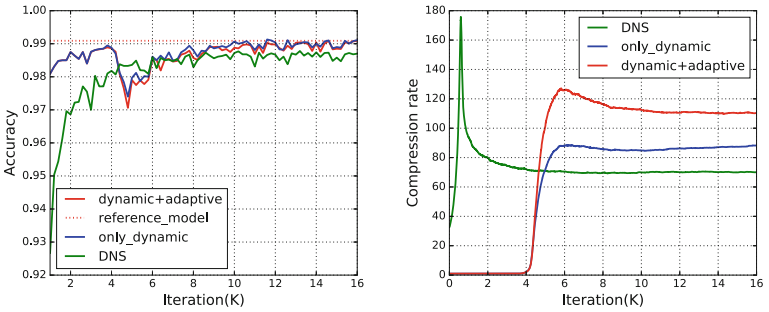
Implementation Details. We conduct experiments using Caffe platform and the open source code from [8] on NVIDIA GTX TITAN X graphics card. Moreover, we follow the default setting of corresponding *.prototxt* files in Caffe unless otherwise specified. The random number seed is fixed when we initialize a model for fair comparison. We employ classification accuracy and compression rate as the evaluation metrics, in which compression rate is defined as total number of weights divided by number of weights after compression.

4.2 Demonstration of DAT's Effectiveness

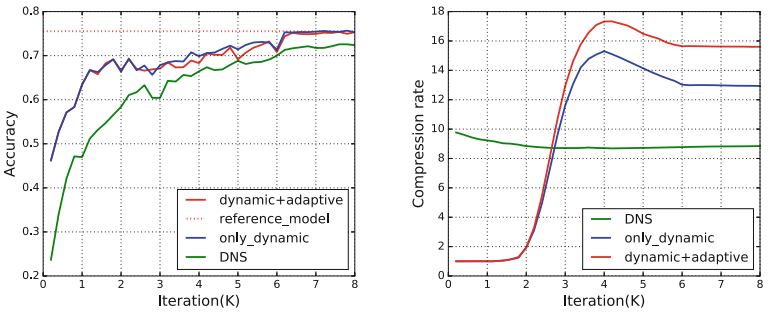
We firstly demonstrate the effectiveness of our proposed DAT framework, including dynamic threshold scheme alone and the whole DAT scheme. Dynamic network surgery (DNS) [8] is taken as the comparative method. Although DNS is designed to work on reference models, in this subsection all methods are used for training from scratch for fair comparison. Because large scale pruning would inevitablely result in loss in performance, more iterations are needed to ensure the recovery of accuracy. For example, the number of iterations is increased from 10000 to 16000 for training LeNet-5, and from 5000 to 8000 for training CIFAR10_CAFFE respectively. Note that drastic pruning will happen after i_0, so the learning rate from i_0 is increased for better compensating for the loss of accuracy, especially for LeNet-5 that originally adopts a monotone decreasing learning policy in Caffe. For each method, we explore the best $\boldsymbol{c}_{max}$ and then plot accuracy and compression rate curves over iterations in Fig. 2.

As is shown in Fig. 2, both dynamic threshold scheme alone and the whole DAT scheme can reach a rather high compression rate almost without loss of accuracy, which proves that our proposed method is highly effective in DNN compression. For both models, the dynamic only and DAT followed the same pattern in accuracy curves, they undergo a small drop after i_0, noting that for LeNet-5 $i_0 = 4000$ and for CIFAR10_CAFFE $i_0 = 2000$. Eventually the dynamic only and DAT reached the accuracy of 99.11% and 99.09% for LeNet-5, 75.30% and 75.33% for CIFAR10_CAFFE. These results are comparable to reference models (red dash line in Fig. 2), and they are much better than those of DNS, which are only 98.70% and 72.39%. This is also true for compression rate curve, i.e., the compression rates of the dynamic only and DAT are much higher than those of DNS. Different from accuracy curves, the dynamic only and DAT begin to rise in compression rate after i_0, and then exhibit different behaviours. DAT can reach a higher compression rate for both models, which is mainly due to the adaptive threshold scheme's adaptability to weight distribution.

In this subsection, it is shown that DAT can outperform DNS in both accuracy and compression rate, which demonstrate the effectiveness of our proposed dynamic and adaptive threshold framework clearly.



(a) LeNet-5



(b) CIFAR10_CAFFE

Fig. 2. Accuracy and compression rate comparison of different methods on LeNet-5 (a) and CIFAR10_CAFFE (b). Note that for each subfigure, accuracy curves (*left*) and compression rate curves (*right*) are plotted. (Color figure online)

4.3 Comparison with DNN Compression Methods

In this subsection, DAT is compared with state-of-the-art DNN compression methods. Under the condition of compression without accuracy loss, network pruning is a more efficient method than others like weight sharing and tensor decomposition, so we choose state-of-the-art DNN pruning methods, i.e., iterative network pruning (INP) [5] and dynamic network surgery (DNS) [8], as the compared methods. As for the evaluation metrics, overall iterations is used, in addition to accuracy and compression rate. For INP and DNS, overall iterations equals to the number of iterations in training phase plus that in compression phase, and for DAT, overall iterations is just the maximum number of iterations. The iterative number in INP is fixed to 3, which means the procedure of pruning and retraining are repeated 3 times.

Generally we keep the accuracy being comparable with that of reference model and then compare the compression rate of different methods, due to the tradeoff between accuracy and compression rate. For CIFAR10_CAFFE, this target is a little bit difficult because the range of accuracy is relatively large,

which is different from LeNet-5. As a result, the accuracy of INP, DNS and DAT do not strictly equal to that of reference model, i.e. 75.54%.

The results are presented in Table 1. It is obvious that DAT and DNS outperform INP in compression rate by a large margin. Besides, INP need much more iterations to compress a DNN model, which is consistent with the results in [5] and [8]. Our proposed DAT is comparable with DNS in both accuracy and compression rate. For LeNet-5, DAT reaches a compression rate of 110×, which is slightly better than 108× of DNS. For CIFAR10_CAFFE, DAT is slightly worse than DNS in compression rate with 15.6× versus 16.0×, and is better than DNS in accuracy with 75.33% versus 75.19%. However, the biggest difference between DAT and DNS is that DAT does not need reference model, which results in dramatic decrease in overall iterations.

Table 1. Comparison of DAT and state-of-the-art DNN compression methods including iterative network pruning (INP) and dynamic network surgery (DNS). Note that the accuracy of reference model is 99.09% for LeNet-5 and 75.54% for CIFAR10_CAFFE.

	Method	Accuracy	Overall iterations	Compression rate	Need reference model?
LeNet-5	INP	99.09%	70 K	20×	YES
	DNS	99.09%	26 K	108×	YES
	DAT	99.09%	**16 K**	**110×**	**NO**
CIFAR10_CAFFE	INP	**75.78%**	23 K	8.8×	YES
	DNS	75.19%	13 K	**16.0×**	YES
	DAT	75.33%	**8 K**	15.6×	**NO**

4.4 Comparison with DNN Sparse Training Methods

Finally we compare DAT with two well-known DNN sparse training methods, i.e., l_1 and l_0 regularization. Note that most weights w satisfy $|w| < 1$, thus their l_1 regularization will be significantly larger than their l_2 regularization, so the weight decay factor should be much smaller. In experiments, the weight decay is set to 1/10 of the original value for LeNet-5 and 1/5 for CIFAR10_CAFFE [10]. The other problem of l_1 regularization is that l_1 encourages many weights near zero, but does not output exactly zero value weights! Therefore, for fair comparison, we add a pruning process after training while keeping the accuracy not decreasing. The l_0 regularization [12] directly set all weights to zero except the t largest-magnitude ones every n iterations. The main target of this subsection is the comparison between DAT and l_1, l_0 regularization, and for simplicity we use the same training iterations as the reference models, that is, 10000 for LeNet-5 and 5000 for CIFAR10_CAFFE.

The accuracy curve over iterations is plotted in Fig. 3. For LeNet-5, only DAT can meet the accuracy of reference model (the red dash line in Fig. 3), and

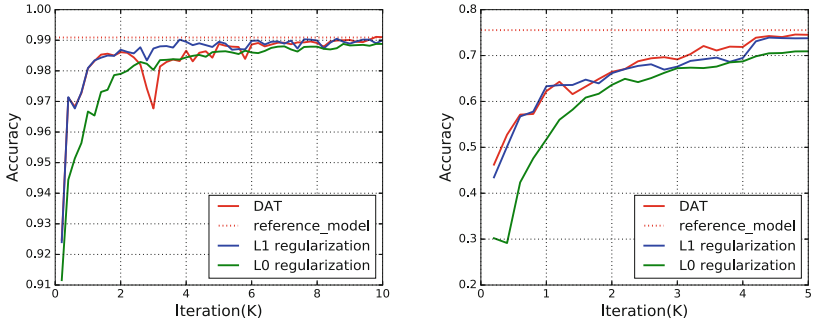


Fig. 3. Accuracy curves of different DNN sparse training methods on LeNet-5 (*left*) and CIFAR10_CAFFE (*right*). Note that the training iterations are 10000 for LeNet-5 and 5000 for CIFAR10_CAFFE. (Color figure online)

Table 2. Comparison of DAT with l_1 and l_0 regularization.

	LeNet-5		CIFAR10_CAFFE	
Method	Accuracy	Compression rate	Accuracy	Compression rate
l_1	99.0%	10×	73.55%	2.3×
l_0	98.88%	10×	70.92%	2.1×
DAT	**99.10%**	**63**×	**74.52%**	**10.8**×

for CIFAR10_CAFFE, no methods can reach the accuracy of reference model, mainly because of reduction of iterations. Besides, the accuracy of l_0 drop a lot for both models, compared with accuracy of DAT and reference model, which is mainly due to the fixed hard constraint t adopted by l_0 regularization.As for the compression rate, DAT reaches 63× and 10.8× for LeNet-5 and CIFAR10_CAFFE respectively as shown in Table 2. These results are surprising because the compression rate of DAT is several times better than the other two methods. Considering the accuracy of DAT is also higher, so we can draw the conclusion that DAT is quite an effective DNN sparse training method.

5 Conclusion

In this paper, we explore to compress DNN models without using reference models. A dynamic and adaptive threshold (DAT) framework is proposed to prune a DNN gradually by changing the pruning threshold during training, thus DNN training and compression can be performed simultaneously. Experiment results demonstrate that DAT can compress LeNet-5 and CIFAR10_CAFFE by a factor of 110× and 15.6× respectively, without the usage of reference models and almost without loss of accuracy. These compression rates are comparable or better than state-of-the-art DNN compression method. Also, DAT can beat DNN sparse training methods like l_1 and l_0 regularization by a large margin.

Although effective, our proposed method imports some hyper parameters and the searching of these hyper parameters is time-consuming. Therefore, in the future we plan to explore DNN compression from scratch with fewer parameters.

Acknowledgments. We want to thank the reviewers for their valuable comments. This work was supported by the NSFC (U1605251, U1613216), the Young Elite Scientists Sponsorship Program by CAST (2016QNRC001), the CCF-Tencent Open Research Fund and the Royal Society Grant on "Data Driven Metaheuristic Search".

References

1. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems, pp. 1097–1105 (2012)
2. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)
3. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature **521**(7553), 436–444 (2015)
4. Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., Kuksa, P.: Natural language processing (almost) from scratch. J. Mach. Learn. Res. **12**, 2493–2537 (2011)
5. Han, S., Pool, J., Tran, J., Dally, W.: Learning both weights and connections for efficient neural network. In: Advances in Neural Information Processing Systems, pp. 1135–1143 (2015)
6. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
7. Denil, M., Shakibi, B., Dinh, L., deFreitas, N., et al.: Predicting parameters in deep learning. In: Advances in Neural Information Processing Systems, pp. 2148–2156 (2013)
8. Guo, Y., Yao, A., Chen, Y.: Dynamic network surgery for efficient DNNs. In: Advances in Neural Information Processing Systems, pp. 1379–1387 (2016)
9. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: AISTATS, vol. 9, pp. 249–256 (2010)
10. Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., Darrell, T.: Caffe: convolutional architecture for fast feature embedding. In: Proceedings of the 22nd ACM International Conference on Multimedia, pp. 675–678. ACM (2014)
11. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proc. IEEE **86**(11), 2278–2324 (1998)
12. Collins, M.D., Kohli, P.: Memory bounded deep convolutional networks. arXiv preprint arXiv:1412.1442 (2014)
13. Lin, D.D., Talathi, S.S., Annapureddy, V.S.: Fixed point quantization of deep convolutional networks. arXiv (2015)
14. Courbariaux, M., Bengio, Y., David, J.P.: Binaryconnect: training deep neural networks with binary weights during propagations. In: Advances in Neural Information Processing Systems, pp. 3123–3131 (2015)
15. Gong, Y., Liu, L., Yang, M., Bourdev, L.: Compressing deep convolutional networks using vector quantization. arXiv preprint arXiv:1412.6115 (2014)

16. Chen, W., Wilson, J.T., Tyree, S., Weinberger, K.Q., Chen, Y.: Compressing neural networks with the hashing trick. In: ICML, pp. 2285–2294 (2015)
17. Denton, E.L., Zaremba, W., Bruna, J., LeCun, Y., Fergus, R.: Exploiting linear structure within convolutional networks for efficient evaluation. In: Advances in Neural Information Processing Systems, pp. 1269–1277 (2014)
18. Zhang, X., Zou, J., Ming, X., He, K., Sun, J.: Efficient and accurate approximations of nonlinear convolutional networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1984–1992 (2015)
19. Lin, S., Ji, R., Guo, X., Li, X., et al.: Towards convolutional neural networks compression via global error reconstruction. In: International Joint Conferences on Artificial Intelligence (2016)
20. Han, S., Liu, X., Mao, H., Pu, J., Pedram, A., Horowitz, M.A., Dally, W.J.: EIE: efficient inference engine on compressed deep neural network. In: Proceedings of the 43rd International Symposium on Computer Architecture, pp. 243–254. IEEE Press (2016)
21. LeCun, Y., Denker, J.S., Solla, S.A., Howard, R.E., Jackel, L.D.: Optimal brain damage. In: NIPs, vol. 2, pp. 598–605 (1989)
22. Hassibi, B., Stork, D.G., et al.: Second order derivatives for network pruning: optimal brain surgeon. In: Advances in Neural Information Processing Systems, p. 164 (1993)
23. Wen, W., Wu, C., Wang, Y., Chen, Y., Li, H.: Learning structured sparsity in deep neural networks. In: Advances in Neural Information Processing Systems, pp. 2074–2082 (2016)
24. Zaabab, A.H., Zhang, Q.J., Nakhla, M.S.: Device and circuit-level modeling using neural networks with faster training based on network sparsity. IEEE Trans. Microw. Theor. Tech. **45**(10), 1696–1704 (1997)
25. Ishikawa, M.: Structural learning with forgetting. Neural Networks **9**(3), 509–521 (1996)
26. Krizhevsky, A., Hinton, G.: Learning multiple layers of features from tiny images (2009)