IEICE TRANS. INF. & SYST., VOL.E103–D, NO.5 MAY 2020

LETTER

Loss-Driven Channel Pruning of Convolutional Neural Networks

Xin LONG[†a)], **Xiangrong ZENG**[†], ***Nonmembers***, **Chen CHEN**[†], ***Student Member***, **Huaxin XIAO**[†], ***and*** **Maojun ZHANG**[†], ***Nonmembers***

SUMMARY The increase in computation cost and storage of convolutional neural networks (CNNs) severely hinders their applications on limited-resources devices in recent years. As a result, there is impending necessity to accelerate the networks by certain methods. In this paper, we propose a loss-driven method to prune redundant channels of CNNs. It identifies unimportant channels by using Taylor expansion technique regarding to scaling and shifting factors, and prunes those channels by fixed percentile threshold. By doing so, we obtain a compact network with less parameters and FLOPs consumption. In experimental section, we evaluate the proposed method in CIFAR datasets with several popular networks, including VGG-19, DenseNet-40 and ResNet-164, and experimental results demonstrate the proposed method is able to prune over 70% channels and parameters with no performance loss. Moreover, iterative pruning could be used to obtain more compact network.

key words: *channel pruning, convolutional neural networks, Taylor expansion, fine-tuning, iterative pruning*

1. Introduction

In recent years, deep neural networks play important roles in variety of computer vision tasks [1]–[3]. However, large networks appear in practical applications due to the specification of different task, which impose more parameters and FLOPs consumption burden on training devices. There is impending necessity to compress the deep networks without affecting the performance. Considerable efforts have been proposed to study this scheme, including parameter pruning and sharing [4], [5], low rank approximation [6], [7], quantization [8]–[10], architecture designment [11]–[13] and sparse pruning [14]–[16]. As discussed in [17], channel pruning would provide a great tradeoff between flexibility and ease of implementation for CNNS while other approaches not.

In this paper, we propose a loss-driven channel pruning method which prunes unimportant channels identified by using Taylor expansion technique. The flow chart of the proposed algorithm is shown in Fig. 1. First of all, our action object is a trained model aiming to different datasets and networks. Next, we evaluate the importance of each channel by designed criteria for channel pruning. Then removing unimportant channels through defined threshold. Finally, a compact network is obtained by fine-tuning which also make performance degradation compensated. Besides, the above process can be repeated for several time, called iterative channel pruning which leads to more compact network.

Manuscript received November 19, 2019.
Manuscript publicized February 17, 2020.
[†]The authors are with the National University of Defense Technology, China.
a) E-mail: longxin14@nudt.edu.cn
DOI: 10.1587/transinf.2019EDL8200

2. Proposed Method

In this section, we propose our method to achieve channel pruning of CNNs. The pipeline is shown in Fig. 1. Batch normalization [18] is a wide-used technology which has been adopted by most CNNs to achieve fast convergence and better performance. The general algorithm of batch normalization is shown in algorithm 1, where the last step is the affine transformation operation, γ and β are trainable parameters (scale and shift) which provides the possibility of linearly transforming normalized activation back to initial input scales to avoid decreasing model performance.

Obviously, scaling and shifting parameters are channel-wise in each layer. Therefore, we could achieve

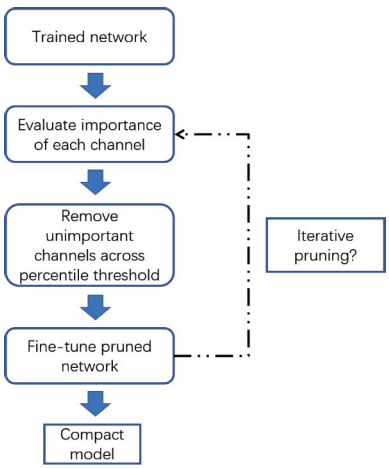


Fig. 1 Pipeline of loss-ware channel pruning.

Algorithm 1: Procedure of batch normalization

Input : Value of x over a mini-batch: $\mathcal{B} = \{x_1, \cdots, x_m\}$;
Learned parameter: γ, β
Output: $y_i = BN_{\gamma,\beta}(x_i)$

$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i$
$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2$
$\hat{x}_i \leftarrow \frac{x_i - \mu_B}{\sqrt{\sigma^2_{\mathcal{B}} + \varepsilon}}$
$y_i \leftarrow \gamma\hat{x}_i + \beta \equiv BN_{\gamma,\beta}(x_i)$

channel pruning based on batch normalization layer by considering γ and β parameters. Different from previous research, it has great advantage of pending no much extra computation resource to the network.

2.1 Criteria for Channel Pruning

Considering a batch of training examples $\mathcal{B} = \{x_1, \cdots, x_m\}$, and corresponding target output $y = \{y_1, \cdots, y_m\}$. After training process, we obtain a trained model with L-layer network parameters $W = \{(w_1, b_1), (w_2, b_2), \cdots (w_L, b_L)\}$, and assuming cost function $L(\cdot)$. As introduced above, γ and β are trained parameters during training process, tiny changes may occur changes in the loss function. To approximate loss changes $\Delta L(\cdot)$, we adopt the first-degree Taylor polynomial. In general, for a binary function $f(x, y)$, the Taylor expansion at point (x_0, y_0) is

$$f(x_0 + h, y_0 + k) = \sum_{n=0}^{N} \frac{1}{n!} \left(h\frac{\partial}{\partial x} + k\frac{\partial}{\partial y} \right)^2 f(x_0, y_0) + \Theta(x_0, y_0)$$

On the other hand, the first-degree Taylor expansion for loss function $L(\cdot)$ at channel (γ_0, β_0) is

$$L(W, (\gamma_0 + h, \beta_0 + k)) = L(W, (\gamma_0, \beta_0)) + h \cdot \frac{\partial L(W, (\gamma_0, \beta_0))}{\partial \gamma} + k \cdot \frac{\partial L(W, (\gamma_0, \beta_0))}{\partial \beta} + \Theta(\gamma_0, \beta_0)$$

where $\Theta(\gamma_0, \beta_0)$ is the high order remainder. Therefore the loss change $\Delta L(\cdot)$ regarding scaling and shifting parameters could be formed as:

$$\Delta L(W, (\gamma_0, \beta_0)) = h \cdot \frac{\partial L(W, (\gamma_0, \beta_0))}{\partial \gamma} + k \cdot \frac{\partial L(W, (\gamma_0, \beta_0))}{\partial \beta} + \Theta(\gamma_0, \beta_0)$$

which is the criterion for evaluating importance of layer channel. Intuitively, this criterion prunes layer channels that have small value of loss change $\Delta L(\cdot)$ and ignores the high order remainder. This method requires accumulation of gradient of γ and β parameters, which could be easily computed in back-propagation process.

2.2 Channel Pruning and Fine-Tuning

After training in general conditions, we obtain full-precision model. Then we can prune layer channels regarding to proposed criterion by removing corresponding connections. Similar to [17], we prune unimportant channels with a global percentile threshold across the whole network. We rank the importance of each channel by the value of $\Delta L(\cdot)$, and prune channels with corresponding lower value of $\Delta L(\cdot)$ under above stated percentile threshold.

For example, we will prune 50% unimportant channels with lower value of $\Delta L(\cdot)$ when the percentile threshold is 50%. Obviously, we obtain a more compact model with less parameters and computational resources in this way. Besides, pruning channel would lead to some accuracy loss when the percentile threshold is pretty high. In experimental sections, this can be largely compensated by fine-tuning process which needs less training epochs and time.

2.3 Iterative Channel Pruning

In order to obtain more compact network, we extend proposed method to iterative channel pruning, which is shown by the dotted portion in Fig. 1. There are two ways to conduct iterative channel pruning. 1), by setting a fixed percentile threshold, the pruning is carried out according to the threshold every iteration until pruning is stopped due to excessive precision loss. 2), single pruning target can be obtained by multiple channel pruning iterations. For instance, if we want to prune 60% channels, we can get it through two iterations of pruning, such as pruning 20% channels first and then pruning 50%. By doing so, we could prevent the percentile threshold of single pruning is too large, and precision compensation cannot be obtained through fine-tuning.

3. Experiments

3.1 Implementation Details

In this paper, we empirically conduct experiments on CIFAR-10 and CIFAR-100. The same standard data augmentation strategy in [14] is adopted by this paper. For network architectures, we evaluate our method on three frequently-used network: VGG-19, DenseNet-40 and ResNet-164. Because the limitation of our poor experimental conditions, we cannot conduct proposed method on ImageNet and other complex networks with more layers. During training process, the learning rate starts at 0.1 and we use learning rate decay equal to 0.1 at epochs number 80 and 120 for the whole 160 epochs. Moreover, we use a weight decay of 5×10^{-4} and momentum of 0.9 for stochastic gradient descent (SGD) optimizer.

3.2 Experimental Results and Discussions

Comparison on CIFAR-10 and CIFAR-100 with [17]: The results on CIFAR-10 and CIFAR-100 are shown in Table 1, we define the global percentile threshold of channel pruning as θ.

As shown in Table 1, it could be seen that the proposed method outperform a little more than [17]. Moreover, it could lead to better results than baseline when pruning many layer channels.

Effect of different pruning percentile threshold: The goal of channel pruning is to pruning more parameters without affecting the accuracy. We train DenseNet-40 and ResNet-164 on CIFAR-100 with a varying threshold, and

Table 1 Test error comparison.

(a) Test error comparison on CIFAR-10

Model	Test error (%)	Parameter pruned (%)	FLOPs pruned (%)
VGG-19-Baseline	6.34	-	-
VGG-19-Ours (θ = 70%)	6.09	87.5	46.6
VGG-19-[17] (θ = 70%)	6.20	88.5	51.0
DenseNet-40-Baseline	6.11	-	-
DenseNet-40-Ours (θ = 40%)	5.16	36.9	34.3
DenseNet-40-[17] (θ = 40%)	5.19	35.7	28.4
DenseNet-40-Ours (θ = 70%)	5.51	65.0	59.9
DenseNet-40-[17] (θ = 70%)	5.65	65.2	55.0
ResNet-164-Baseline	5.42	-	-
ResNet-164-Ours (θ = 40%)	4.88	28.1	26.1
ResNet-164-[17] (θ = 40%)	5.08	14.9	23.7
ResNet-164-Ours (θ = 60%)	5.10	54.7	50.8
ResNet-164-[17] (θ = 60%)	5.27	35.2	44.9

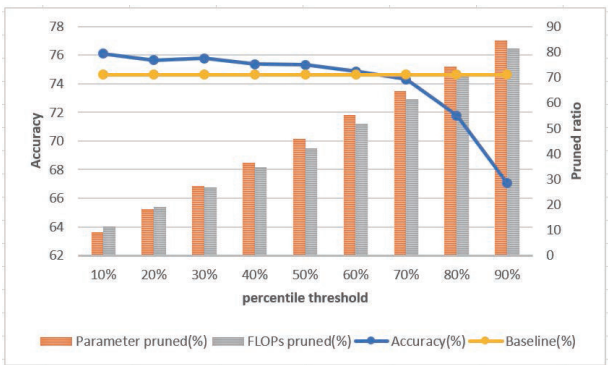
(b) Test error comparison on CIFAR-100

Model	Test error (%)	Parameter pruned (%)	FLOPs pruned (%)
VGG-19-Baseline	26.74	-	-
VGG-19-Ours (θ = 50%)	26.02	72.3	34.2
VGG-19-[17] (θ = 50%)	26.52	75.1	37.1
DenseNet-40-Baseline	25.36	-	-
DenseNet-40-Ours (θ = 40%)	24.63	36.4	34.6
DenseNet-40-[17] (θ = 40%)	25.28	37.5	30.3
DenseNet-40-Ours (θ = 60%)	25.14	55.1	51.7
DenseNet-40-[17] (θ = 60%)	25.72	54.6	47.1
ResNet-164-Baseline	23.37	-	-
ResNet-164-Ours (θ = 40%)	22.41	27.0	30.4
ResNet-164-[17] (θ = 40%)	22.87	15.5	33.3
ResNet-164-Ours (θ = 60%)	23.24	50.5	53.8
ResNet-164-[17] (θ = 60%)	23.91	29.7	50.6

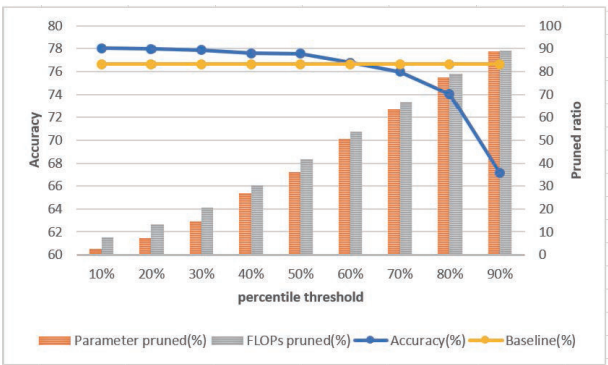
the pruned ratio of parameters and FLOPs are recorded for each pruning pro-cess. The results are summarized in Fig. 2. Form Fig. 2, it can be concluded that the performance of pruned models degrade much when the pruning ratio surpasses a large threshold. If we prune few channels, the performance of mode is even better than baseline due to the overfitting phenomenon in training is eliminated. On the other hand, both networks have large redundancy over parameters and computation consumption, and we could obtain a compact network by our method with a proper threshold.

Iterative channel pruning: We employ the two ways of iterative channel pruning on CIFAR10/CIFAR-100 by using VGG-19. The test accuracy of each iteration are shown in Tables 2 and 3. For iterative pruning, we set pruning threshold as 50%, and conduct the pipeline in Fig. 1. In order to compare iterative pruning and single pruning, we set up experiments in (b) to maintain same pruned target.

As the pruning process goes, we will obtain more and more slimming network while inevitably hurt the model performance. On the other hand, single pruning may hurt the performance drastically than iterative pruning while achieving the same pruning goal. On CIFAR-10/100, the accuracy by single pruning is reduced by nearly 14/54 percent points than iterative pruning when pruning 87.5% channels, which is very fatal for image classification. Thus, we suggest to adopt iterative pruning with low percentile thresholds in applications.



(a) DenseNet-40



(b) ResNet-164

Fig. 2 Model parameter and FLOPs savings by different pruning percentile threshold on CIFAR-100. The horizontal axis represents the pruning percentile, and the vertical axis is the precision and parameter/FLOPs pruned.

Table 2 Test error comparison.

(a) Iterative pruning on CIFAR-10 (θ = 50%)

Iteration	Test accuracy (%)	Parameter pruned (%)	FLOPs pruned (%)
1	93.89	75.93	35.33
2	93.86	91.51	59.22
3	92.81	97.54	85.73
4	90.46	99.37	95.36
5	83.08	99.85	98.72

(b) Single pruning on CIFAR-10 for comparison with (a)

Pruning ratio (%)	Test accuracy (%)	Parameter pruned (%)	FLOPs pruned (%)
50.00	93.89	75.93	35.33
75.00	93.74	90.37	54.58
87.50	78.72	97.15	83.06
93.75	10.00	99.19	95.39

Table 3 Test error comparison.

(a) Iterative pruning on CIFAR-100 (θ = 50%)

Iteration	Test accuracy (%)	Parameter pruned (%)	FLOPs pruned (%)
1	73.98	72.32	34.18
2	71.46	91.25	70.33
3	62.60	97.66	90.21
4	46.86	99.41	97.29

(b) Single pruning on CIFAR-100 for comparison with (a)

Pruning ratio (%)	Test accuracy (%)	Parameter pruned (%)	FLOPs pruned (%)
50.00	73.98	72.32	34.18
75.00	63.28	90.70	66.25
87.50	8.65	97.61	89.80
93.75	6.22	99.34	97.16

4. Conclusion

In this paper, we propose loss-driven channel pruning method to learn compact CNNs. It identifies unimportant channels by using Taylor expansion technique regarding to scaling and shifting factors, then we prune channels by fixed percentile threshold. Accuracy loss caused by pruning can be compensated by following fine-tuning process. Experimental results show that the proposed method is able to prune over 70% channels and parameters with no accuracy loss, and even better results compared to baseline. More significantly, we found iterative pruning which would be better than single pruning process.

References

[1] A. Krizhevsky, I. Sutskever, and G.E. Hinton, "ImageNet classification with deep convolutional neural networks," Neural Information Processing Systems, pp.1097–1105, 2012.

[2] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," 2014 IEEE Conference on Computer Vision and Pattern Recognition, pp.580–587, 2014.

[3] H. Xiao, B. Kang, Y. Liu, M. Zhang, and J. Feng, "Online meta adaption for fast video object segmentation," IEEE Trans. Pattern Anal. Mach. Intell., Early Access, 2019.

[4] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," Neural Information Processing Systems, pp.1135–1143, 2015.

[5] S. Han, H. Mao, and W.J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding," International Conference on Computer Representations, 2016.

[6] X. Yu, T. Liu X. Wang, and D. Tao, "On compressing deep models by low rank and sparse decomposition," 2017 IEEE Conference on Computer Vision and Pattern Recognition, pp.67–76, 2017.

[7] E.L. Denton, W. Zaremba, J. Bruna, Y. LeCun, and R. Fergus, "Exploiting linear structure within convolutional networks for efficient evaluation," Neural Information Processing Systems, pp.1269–1277, 2014.

[8] M. Courbariaux, Y. Bengio, and J. David, "BinaryConnect: Training deep neural networks with binary weights during propagations," Neural Information Processing Systems, pp.3123–3131, 2015.

[9] F. Li and B. Liu, "Ternary weight networks," Computer Vision and Pattern Recognition, 2016.

[10] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "XNOR-Net: ImageNet classification using binary convolutional neural networks," European Conference on Computer Vision, Lecture Notes in Computer Science, vol.9908, pp.525–542, Springer, Cham, 2016.

[11] A. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, and H. Adam, "MobileNets: Efficient convolutional neural networks for mobile vision applications," Computer Vision and Pattern Recognition, 2017.

[12] X. Zhang, X. Zhou, M. Lin, and J. Sun, "ShuffleNet: An extremely efficient convolutional neural network for mobile devices," 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp.6848–6856, 2018.

[13] B. Baker, O. Gupta, N. Naik, and R. Raskar, " Designing neural network architectures using reinforcement learning," International Conference on Computer Representations, 2017.

[14] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H.P. Graf, "Pruning filters for efficient convnets," arXiv preprint arXiv:1608.08710, 2016.

[15] W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li, "Learning structured sparsity in deep neural networks," Neural Information Processing

Systems, pp.2074–2082, 2016.

[16] X. Long, Z. Ben, X. Zeng, Y. Liu, M. Zhang, and D. Zhou, "Learning sparse convolutional neural networks via quantization with low rank regularization," IEEE Access, vol.7, pp.51866–51876, 2019.

[17] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, and C. Zhang, "Learning efficient convolutional networks through network slimming," arXiv preprint arXiv:1708.06519, 2017.

[18] S. Ioffe and C. Szegedy, "Batch normalization: Accelarating deep network training by reduing internal covariate shift," arXiv preprint arXiv:1502.03617, 2015.