Anonymous Model Pruning for Compressing Deep Neural Networks

Lechun Zhang[1,2], Guangyao Chen[2], Yemin Shi[2], Quan Zhang[2],
Mingkui Tan[4],Yaowei Wang[2,3], Yonghong Tian[2,3],Tiejun Huang[2,3]
[1]School of ECE, Peking University Shenzhen Graduate School, Shenzhen, P.R. China
[2]Peking University, Institute of Digital Media, Beijing 100871, China
[3]Peng Cheng Laboratory, Shenzhen 518055, China
[4]South China University of Technology 518055,China
{lechunzhang,gy.chen,shiyemin,zquan,yhtian,tjhuang}@pku.edu.cn,
wangyw@pcl.ac.cn,mingkuitan@scut.edu.cn

***Abstract*—Many deep neural network compression algorithms need to fine-tune on source dataset, which makes them unpractical when the source datasets are unavailable. Although data-free methods can overcome this problem, they often suffer from a huge loss of accuracy. In this paper, we propose a novel approach named Anonymous-Model Pruning (AMP), which seeks to compress the network without the source data and the accuracy can be guaranteed without too much loss. AMP compresses deep neural networks via searching pruning rate automatically and fine-tuning the compressed model under the teacher-student diagram. The key innovations are that the pruning rate is automatically determined, and the fine-tuning process is under the guidance of uncompressed network instead of labels. Even without the source dataset, compared with existing pruning methods, our proposed method can still achieve comparable accuracy with similar pruning rate. For example, for ResNet50, our AMP method only incur 0.76% loss in top-1 accuracy with 32.72% pruning rate.**

***Keywords*-network compression; knowledge distillation; pruning;**

I. Introduction

The success of Deep neural networks(DNN) mainly results from deeper network design, larger amount of weights and considerable scale of data, which make DNN hard to deploy on resource constrained platforms. Therefore, it is necessary to compress deep neural networks. However, most of the model compression methods try to compress deep neural networks and then retrain the model on original labeled datasets (such as ILSVRC-12 [1]) to regain a considerable accuracy. These kinds of methods may suffer from data dependence and over-fitting. Furthermore, we needn't download the whole dataset for fine-tuning after compression for two reasons: (**1**) a model cannot enrich its knowledge on the pre-trained datasets, which makes the original dataset redundant and less contributed; (**2**) pre-training datasets contain a huge number of items, on which fine-tuning is both space consuming and time consuming.

In this paper, we propose an Anonymous Model Pruning and knowledge distillation pipeline to compress and fine-tune convolution neural networks without the original datasets. Firstly, we set an acceptable loss threshold for each layer to prevent error accumulative. Secondly, we use binary search to determine the pruning rate of each layer keeping the loss under threshold. Meanwhile, pruning rate and pruning loss are not linear correlated, and our algorithm can better explore the redundancy of each layer. Experiments on Alexnet [5] and ResNet50 [6] for ImageNet classification demonstrate the effectiveness of our proposed method.

II. Related Work

Knowledge Distillation is first proposed by [7], where knowledge is transferred from high-capacity teacher to a more compact network for efficient deployment. Not only the ground truth component but also the dark knowledge term, containing information on the wrong outputs, have contribution on student network, which has been proved in [4]. This suggests that knowledge distillation, originally introduced to reduce parameters, can be applied to accuracy recovery. A pioneering work extended knowledge distillation to use both the outputs and intermediate representations, which is produced by teacher network, to train the narrower students [8]. In a different approach, the authors trained shallow nets to mimic deep networks with L2 norm of the difference between student's and teacher's logits [9].

Network Pruning is widely used in deep neural network compression since [10]. They approved that weight pruning is a valid way to reduce over-fitting and model complexity. Then some researchers pruned the small-weight connections and retrained the network to learn final weights for the remaining connections [11].In order to prevent weights from being wrongly pruned, soft-pruning has been proposed to update the pruned weight and apply pruning algorithm again to the new weights [12], [13]. Recently, filter level pruning has been proposed to compress and accelerate deep neural networks. A new formulation has been proposed for pruning convolution kernels which used Taylor expansion based importance criteria to prune filters [14]. We propose to assign pruning rate automatically by binary search to make network pruning simpler.

978-1-7281-4272-2/20/$31.00 ©2020 IEEE
DOI 10.1109/MIPR49039.2020.00040

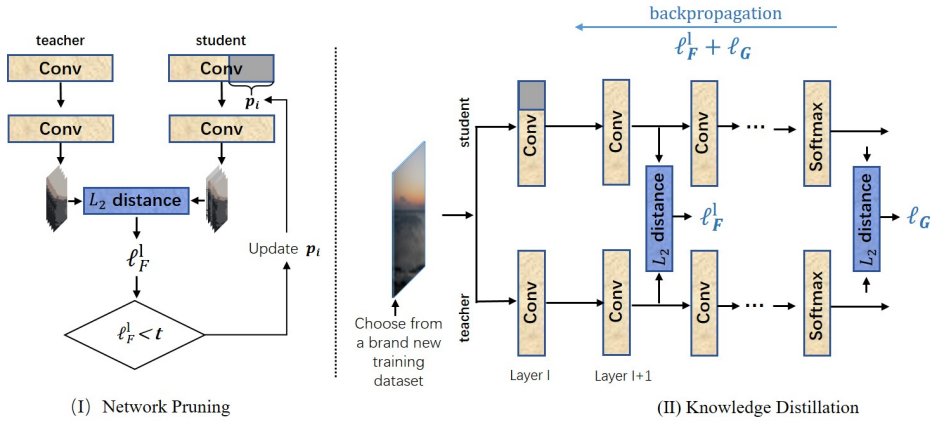

Figure 1. An overview of AMP algorithm. The gray parts represent network pruning. I. Use binary search to assign pruning rate for each layer. II. Use knowledge distillation to fine-tune student network. When pruning the l-th layer, both ℓ_F^l and ℓ_G will be used to guide backporpagation.

III. Notation and Preliminaries

Given a deep model Net_T, which is pre-trained on labeled dataset D_S, we hope to obtain a compressed model Net_S. We denote the parameters of each layer as a 4-D weight tensor and the feature maps of each layer as a 3-D tensor. Let $T^{(l)} \in \mathbb{R}^{N_l \times C_l \times h_l \times z_l}$ and $S^{(l)} \in \mathbb{R}^{N_l \times C_l \times h_l \times z_l}$ be the l-th layer weight tensor of the teacher and student models with kernel size $h_l \times z_l$, where N_l and C_l denote the number of output and input channels. We denote $||\mathbf{v}||^2$ as the L-2 norm of vector $\mathbf{v}$ and $S^{(l)}_{i,:,:,:}$ be the i-th channel of $S^{(l)}$. The output of the l-th layer is denoted by $S^{(l)}_O \in \mathbb{R}^{N_l \times h_o \times z_o}$, where h_o and z_o is the height and width of the feature maps. For the teacher network Net_T, the output of the l-th layer, denoted by $T^{(l)}_{O\ j,:,:}$, can be defined in a similar way.

IV. Anonymous Model Pruning

A. Overview of Proposed Methods

An overview of our approach is presented in Figure 1. Take the l-th layer of Net_S for example. We first prune the l-th layer of Net_S by the pruning rate p_l, initialized as p_0. p_0 is shared by all layers. Then we calculate the $L2$ loss of feature maps between the Net_T and the Net_S, which is named ℓ_F^l. Note that ℓ_F^l calculated from p_0 is defined as ℓ_F^0. Suppose t_{max} is the max pruning loss threshold to constrain the pruning proportions. We set a hyperparameter r to balance the pruning rate and model accuracy loss. So the max loss threshold in the l-th layer is calculated by $t^l_{max} = r * \ell_F^0$. Note that the whole training process shares the r.

Then the pruning rate p_l of l-th layer is automatically adjusted to a value as large as possible when the ℓ_F^l is less than t^l_{max}. To find a best p for each layer, we use binary search between 0 and 1. The process of training and binary search progresses iteratively for each layer.

Here, the feature maps loss ℓ_F^l restricts the pruning loss of each layer thereby reducing the reconstruction error. And the global pruning loss between Net_S and Net_T is defined as ℓ_G, which helps Net_S to mimic the final output of Net_T. Then we train Net_S layer by layer to minimize the distance between Net_T. This process uses knowledge distillation approach on another unlabeled dataset D_2 instead of the source dataset D_1. Note that Net_S and Net_T share the same structure and initial parameters at the beginning.

B. Pruning

We verify the feasibility of AMP on weight pruning and channel pruning. Weight pruning can prune unimportant weights precisely while channel pruning can bring convenience to calculation acceleration and practical deployment.

1) Weight Pruning: Considering that smaller weights have less impact on the model, we sort the weights of one layer from small to large. With the pruning rate p, we recurrently set the first p weights to zero in the forward pass and then re-train this layer until binary search is finished. This operation will be applied to each convolution layer and fully connected layer respectively in order. For weight pruning, ℓ_F^l can be computed by

$$\begin{aligned} \ell_F^l &= ||S_O^{(l)} - T_O^{(l)}||^2 \\ &= \frac{1}{C_{l+1}} \sum_{k=0}^{C_l} ||S^{(l)}_{O\ k,:,:} - T^{(l)}_{O\ k,:,:}||^2 \end{aligned} \tag{1}$$

2) Channel Pruning: In contrast to weight pruning, channel pruning needs to evaluate the comprehensive performance of one channel to determine whether to prune it or not. We apply group-sparse regularization l_2-norms to zero-out filters with small l_2-norm, which is also been used in [15], [16]. In order to unify the two strategies, we try to simulate the output of the pruned channels by null matrices, thus making the output dimension of Net_S and Net_T consistent. However, our experiments show that the loss between null matrices and the corresponding layers of Net_T can not reduce through backpropagation, because the

related channels have been removed. Considering the output of the pruned layer is inconsistent after pruning, we choose the output of the layer next to the pruned layer to calculate pruning loss. For channel pruning, ℓ_F^l can be computed by

$$\begin{aligned}\ell_F^l &= ||S_O^{(l)} - T_O^{(l)}||^2 \\ &= \frac{1}{C_{l+1}} \sum_{k=0}^{C_l} ||{S_O^{(l+1)}}_{k,:,:} - {T_O^{(l+1)}}_{k,:,:}||^2 \end{aligned} \tag{2}$$

C. Knowledge Distillation

After network pruning, we use knowledge distillation [7] to further compensate the performance degradation. For the following considerations, we train all the layers together: (1) the deep representations are more expressive [17], so the error can accumulate while pruning the front layers; (2) focusing on reducing the loss of pruned layer ℓ_F^l may lead to local optimal solutions. Conversely, parameters in the mid-layers may suffer from non-convergence if merely under the supervision of the output of teacher's last layer. Therefore, we constrain mid-layers as well as the last fully connected layer, which can reduce the loss caused by pruning , and prevent accumulated error at the same time.

V. Experiments

A. Implementation details

We evaluate AMP on three popular deep neural networks: AlexNet [5], VGG16 [18], Resnet50 [6] on ILSVRC-12 [1]. For training dataset, we randomly select a subset of Open Images V4 [19] containing 7000 categories, in which each category contains only 10 images. The dataset mainly contains random images of the real world. Consequently, it has the similar data distribution to ILSVRC-12. In the testing phase, we record the accuracy of compressed model on their source dataset. We use pre-trained models from Pytorch as the initial teacher and student networks. The learning rate is set to 0.0001 in the beginning and Adam [20] optimizer is adopted. Empirically, we set p_0 to 0.2 and λ to 1.

B. AMP with Weight Pruning

We prune each layer separately and evaluate the differences between the pruned network and the original network. We find that the pruning loss and the pruning rate are not strictly linear correlated. This reveals that the sensitivity of each layer is different, which has also been proved by [21]. This inspires us that the nonlinear correlation between pruning loss and pruning rate can be used to balance pruning rate and network accuracy.

1) AlexNet on ILSVRC-12: We report the relationship between hyperparameter r, compression rate and top-1 accuracy loss in Table I. The batch size is set to 64 and training epoch is set to 32. For equal spaced changes in r, the growth of compression rate is nonlinear. When the student network has already been pruned by 60.25%, the remaining parameters in the student network are relatively important and have more effect on performance. Therefore, by increasing the same size of r, the pruning rate increases by a small proportion. Compared with [22] which achieved 35% pruning rate with 2.2% accuracy damage, we prune 51.03% parameters with only 0.17% accuracy damage.

Table I
AlexNet on ILSVRC-12.

r	Pruning Rate %	Top 1%
0.0	-	56.522
1.0	20.63	56.384
1.05	51.03	56.352
1.10	60.25	56.208
1.15	64.95	55.720

2) ResNet50 on ILSVRC-12: ResNet50 consists of several sub-modules, called residual block. We take a residual block as a unit to apply the pruning algorithm. We report the comparison of our method with other pruning algorithms in Table II. Hyperparameter r in this experiment is set to 1.2. The result of soft filter pruning [3] is not fine-tuned after pruning, we choose this result because the fine-tuned accuracy at the same pruning rate sharply dropped by 14.04% and the authors did not explain why. Considering that the authors only listed the one-view accuracy, we don't compare with the abnormal data for the sake of fairness.

Since the pruning rate of our method is obtained automatically by binary search within the given loss threshold, we can't precisely control the pruning rate. Therefore we choose the closest pruning result to other methods, which makes them comparable.

In the experiments, we also test the student network immediately after weight pruning and get 68.450%, which has -7.68% Δ accuracy. The accuracy of the student network gradually increases in the training progress, which proves the success of knowledge distillation process in AMP algorithm.

Table II
ResNet50 on ILSVRC-12.

Method	Pruning Rate %	ΔTop1 %
Data-Free Pruning [22]	35.08	-2.2
Soft filter pruning [3]	30	-1.54
ThiNet(70) [2]	30	-0.84
Soft weight sharing [13]	6.6	-2.02
Anonymous Model Pruning	**32.72**	**-0.76**

C. AMP with Channel Pruning

we prune VGG-16 and evaluate the pruned model on ILSVRC-12. Experimental results are shown in Table III. At the same pruning rate, channel pruning does not achieve competitive accuracy as weight pruning does. Therefore, channel pruning preserves relatively less dimensional information than weight pruning, and it is relatively difficult to

Table III
PERFORMANCE OF AMP WITH CHANNEL PRUNING.

Network	Pruning Rate %	Top1%	ΔTop1%	Top5 %	Δ Top5 %	Pruned FLOPs %	r
VGG-16	-	71.59	-	90.38	-	-	-
VGG-16	34.63	68.38	-3.21	88.80	-1.58	35.43	1.1
VGG-16	40.97	67.20	-4.39	87.95	-2.43	38.73	1.15
VGG-16	46.30	66.78	-4.81	88.02	-2.36	42.20	1.2

recover the network accuracy without training on the original dataset.

VI. CONCLUSION

This paper proposes the Anonymous Model Pruning (AMP) algorithm that can compress deep neural network without the original dataset. We use binary search to find the practical pruning rate automatically. All the experimental results have revealed that AMP algorithm can achieve a competitive result with other weight pruning methods even without the source dataset. We think that our method will help further research in data-independent model compression.

ACKNOWLEDGMENT

This work is supported in part by the National Key R&D Program of China under Grant 2017YFB1002400, in part by the National Natural Science Foundation of China under Contract U1611461 and 61825101, in part by Shenzhen Municipal Science and Technology Program under Grant JCYJ20170818141146428.

REFERENCES

[1] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *CVPR*. Ieee, 2009, pp. 248–255.

[2] J.-H. Luo, J. Wu, and W. Lin, "Thinet: A filter level pruning method for deep neural network compression," *arXiv preprint arXiv:1707.06342*, 2017.

[3] Y. He, X. Dong, G. Kang, Y. Fu, and Y. Yang, "Progressive deep neural networks acceleration via soft filter pruning," *arXiv preprint arXiv:1808.07471*, 2018.

[4] T. Furlanello, Z. C. Lipton, M. Tschannen, L. Itti, and A. Anandkumar, "Born again neural networks," *arXiv preprint arXiv:1805.04770*, 2018.

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *NeurIPS*, 2012, pp. 1097–1105.

[6] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016, pp. 770–778.

[7] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *Computer Science*, vol. 14, no. 7, pp. 38–39, 2015.

[8] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio, "Fitnets: Hints for thin deep nets," *Computer Science*, 2014.

[9] J. Ba and R. Caruana, "Do deep nets really need to be deep?" in *NIPS*, 2014, pp. 2654–2662.

[10] Y. LeCun, J. S. Denker, and S. A. Solla, "Optimal brain damage," in *NIPS*, 1990, pp. 598–605.

[11] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *Fiber*, vol. 56, no. 4, pp. 3–7, 2015.

[12] F. Tung and G. Mori, "Clip-q: Deep network compression learning by in-parallel pruning-quantization," in *CVPR*, 2018, pp. 7873–7882.

[13] K. Ullrich, E. Meeds, and M. Welling, "Soft weight-sharing for neural network compression," *arXiv preprint arXiv:1702.04008*, 2017.

[14] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz, "Pruning convolutional neural networks for resource efficient inference," *arXiv preprint arXiv:1611.06440*, 2016.

[15] W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li, "Learning structured sparsity in deep learning." NIPS, 2016.

[16] Z. Hao, J. M. Alvarez, and F. Porikli, "Less is more: Towards compact cnns," in *ECCV*, 2016.

[17] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *TPAMI*, vol. 35, no. 8, pp. 1798–1828, 2013.

[18] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[19] A. Kuznetsova, H. Rom, N. Alldrin, J. Uijlings, I. Krasin, J. Pont-Tuset, S. Kamali, S. Popov, M. Malloci, T. Duerig *et al.*, "The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale," *arXiv preprint arXiv:1811.00982*, 2018.

[20] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

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

[22] S. Srinivas and R. V. Babu, "Data-free parameter pruning for deep neural networks," *Computer Science*, pp. 2830–2838, 2015.