Ensemble Knowledge Guided Sub-network Search and Fine-tuning for Filter Pruning

Seunghyun Lee[0000−0001−7139−1764] and Byung Cheol Song[0000−0001−8742−3433]

Inha University, Incheon, Republic of Korea
lsh910703@gmail.com, bcsong@inha.ac.kr

1 Details of the proposed filter importance scoring algorithm

The proposed filter importance scoring (FIS) follows the general Taylor expansion-based methods [15,24]. Filter pruning aims to minimize $\Delta\mathcal{L}$, i.e., the difference in the target loss function when the pre-trained network Θ_0 and a filter θ are removed. As described in this paper, to calculate the importance score of θ, we use f^θ, i.e., the input feature map of the next layer which is generated by θ. For convenience in terms of notation, we redefine the target loss with f^θ removed as $\mathcal{L}(f^\theta)$. Here, $\Delta\mathcal{L}$ is expressed as

$$\Delta\mathcal{L}(f^\theta) = \left|\mathcal{L}(f^\theta) - \mathcal{L}(0)\right| \tag{1}$$

Here, if $\mathcal{L}(0)$ is expanded using Taylor series, it is expressed as follows.

$$\begin{aligned}\mathcal{L}(0) &= \textstyle\sum_{n=0}^{\infty} \frac{\mathcal{L}^{(n)}(f^\theta)}{n!}(0 - f^\theta)^n \\ &= \mathcal{L}(f^\theta) - f^\theta \nabla_{f^\theta}\mathcal{L}(f^\theta) + R_1\end{aligned} \tag{2}$$

where R_1 is the Lagrange remainder. Next, Eq. (1) and Eq. (2) are combined. At this time, if R_1 is omitted to reduce the computational burden, it is expressed by the following formula.

$$\begin{aligned}\Delta\mathcal{L}(f^\theta) &= \left|\mathcal{L}(f^\theta) - \mathcal{L}(f^\theta) + f^\theta \nabla_{f^\theta}\mathcal{L}(f^\theta) - R_1\right| \\ &\approx \left|f^\theta \nabla_{f^\theta}\mathcal{L}(f^\theta)\right|\end{aligned} \tag{3}$$

The proposed method uses the reward $\mathcal{R}(\Theta_i, \cdot)$ instead of the target loss function. As a result, the proposed filter importance score $\mathcal{S}(\theta)$ for θ is expressed by

$$\mathcal{S}(\theta) = \left|\frac{\partial\mathcal{R}(\Theta_i, \cdot)}{\partial f^\theta} f^\theta\right| \tag{4}$$

Method	Acc.	FLOPs ↓	Param ↓
EKG	94.26	25.06	8.84
	94.28	30.07	10.83
	94.19	35.05	16.25
	94.25	40.06	19.45
	94.15	45.00	22.62
	94.09	50.12	27.40
	93.92	55.11	33.69
	93.89	60.02	41.60
	93.69	65.11	46.52
	93.39	70.04	53.14

Table 1. Performance comparison with several existing techniques. The baseline in this experiment is ResNet-56. Here, Acc is the accuracy, and FLOPs ↓ and Param ↓ are the reduction rates of FLOPs and the number of parameters, respectively.

2 Experimental configurations

This section describes the configurations set up for various experiments in this paper. The kernels of convolutional layers and fully-connected (FC) layers were initialized by Xavier initializer [5] and He initializer [7], respectively. As hyper-parameters of batch normalization [10], α and ϵ were set to 0.9 and 1×10^{-5}. The L_2-regularizer was applied to all trained parameters. Also, to train all networks, the stochastic gradient descent (SGD) [11] algorithm was used, and Nesterov accelerated gradient [16] was applied. Here, momentum was set to 0.9.

2.1 CIFAR

CIFAR10 and CIFAR100 use the same configuration as they have no difference except for the number of classes.

Pre-trained network: ResNet-56 [7] and MobileNet-v2 [19] are learned for 200 epochs. Here, the batch size is 128. The initial learning rate is 0.1, which is reduced by 0.2 times at 60, 120, and 160 epochs, respectively. The weight of L_2-regularization is set to 5×10^{-4} for ResNet-56 and 4×10^{-5} for MobileNet-v2, respectively. Each data is normalized on a channel basis. Also, for data augmentation, random horizontal flipping and 32×32 random cropping with padding of 4 pixel-thickness are used.

Sub-network search phase: In the sub-network search phase, we divide the training dataset $\mathcal{D}^{train}$ into the training subset $\mathcal{D}^{subset}$ and the validation set $\mathcal{D}^{val}$. $\mathcal{D}^{subset}$ and $\mathcal{D}^{val}$ are configured by sampling 256 and 32 data per class so that there is no duplicate. Here, data augmentation is not used. In addition, batch normalization layers are set as training mode, and batch statistics are calculated and used each time.

Prior to the sub-network search phase, the pre-trained network is fine-tuned for 1 epoch with $\mathcal{X}^{subset}$. Here, the learning rate is set to 1×10^{-3}. The batch size for fine-tuning and evaluation is 256, and the ratio of the filters with low

ResNet	Method	Top-1 (diff.)	Top-5 (diff.)	FLOPs ↓	GPU hour
50	DSA [17]	75.1 (-0.92)	92.45 (-0.41)	40.5	675*
	FPGM [12]	75.59 (-0.56)	92.63 (-0.24)	42.7	198*
	BNP [14]	75.51 (-1.01)	92.43 (-0.66)	45.6	360
	GBN [24]	76.19 (+0.31)	92.83 (-0.16)	41.0	530*
	TAS [3]	76.20 (-1.26)	92.06 (-0.81)	44.1	532*
	SRR-GR [22]	75.76 (-0.37)	92.67 (-0.19)	45.3	N/A
	NPPM [4]	75.96 (-0.19)	92.75 (-0.12)	56.2	N/A
	ResRep [2]	76.15 (-0.00)	92.89 (+0.02)	54.9	427
	EKG	76.43 (-0.02)	93.13 (-0.02)	45.0	315
		75.93 (-0.52)	92.82 (-0.33)	55.0	310
	EKG-BYOL	76.60 (-0.40)	93.23 (-0.31)	55.0	315
	Autoslim [25]	75.6	N/A	51.6	480
	DMCP [6]	76.20	N/A	46.7	N/A
	CafeNet [20]	76.90	93.3	52.0	1000*
	BCNet [21]	76.90	93.3	52.0	1000*

Table 2. Comparison with various pruning techniques for ResNet-50 trained on ImageNet. * indicates the estimated value.

scores r is 0.2. In the last iteration, r is adjusted so that the FLOPs reduction rate is as close as possible to the given target pruning rate τ.

Memory bank building phase: The number of teachers K to build a memory bank is 5. The settings of all other hyper-parameters are the same as those for evaluating in the sub-network search phase.

Fine-tuning by ensemble knowledge transfer: ResNet-56 and MobileNet-v2 are trained only for 100 epochs, which is a half of a pre-trained network. Here, the batch size is 128. The initial learning rate is 1×10^{-2}, which decreases by 0.2 times at 30, 60, and 80 epochs, respectively. The weight of L_2-regularization is 5×10^{-4}. For additional data augmentation, random brightness, contrast, and saturation distortion are used. the temperature

2.2 ImageNet-2012

Since the basic training configuration of ImageNet-2012 is the same as that of CIFAR, this section describes only hyper-parameters.

Sub-network search phase: $\mathcal{D}^{subset}$ and $\mathcal{D}^{val}$ are configured by sampling 256 and 32 data per class so that there is no duplicate. Prior to the sub-network search, the pre-trained network is fine-tuned for 1 epoch with $\mathcal{D}^{subset}$. Here, the learning rate is 1×10^{-3}. The batch size for fine-tuning and evaluation is 256, and the ratio of the filters with low scores r is set to 0.1 for ResNet-18 and 0.2 for ResNet-34 and -50.

Memory bank building phase: The number of teachers K to build a memory bank is 5. The settings of all other hyper-parameters are the same as those for evaluating in the sub-network search phase.

Contrastive knowledge transfer: ResNet-18, -34 -50 are trained for 100 epochs. Here, the batch size is 256. The initial learning rate is 0.1 and decreases by 0.1 times at 30, 60, and 90 epochs. The weight of L_2-regularization is set to 1×10^{-4}. We employ random brightness, contrast, saturation, and lighting distortion for additional data augmentation.

Estimation of GPU hours: How to estimate GPU hours in Fig. 6 is described here. Most of previous works did not show such costs, but the approximate cost of some methods, i.e., FPGM [8], GBN [24], DAS [18], TAS [3] can be estimated based on the released information. First, in the case of FPGM and GBN, how many epochs are required for pruning and fine-tuning are disclosed. In our environment, multiplying this by the GPU hours required to infer 1 epoch gives us the approximate GPU hours required by both techniques. For example, assuming epoch for forwarding is 0.5, GPU hours of FPGM and GBN are $100 \times 1.98 = 198$ and $((100000/1281167 \times 10 \times 0.5 + 10) \times 20 + 60) \times 1.98 = 530$, respectively. Also, DSA and TAS only disclosed GPU hours for ResNet-18. At this time, if it is assumed that computing time is proportional to FLOPs of ResNet-18 and ResNet-50, approximate GPU hours can be estimated. As a result, GPU hours of DSA and TAS are $300 \times 4.1/1.82 = 675$ and $236 \times 4.1/1.82 = 532$. The authors mentioned in the paper that CafeNet [20] and BCNet [21] require a huge amount of computation to give up part of the algorithms when learning ImageNet. It is actually impossible to run the algorithms on a standard workstation. So, the costs of CafeNet and BCNet were expressed as '1000' symbolically in the paper. These estimates are not exact and may vary depending on the testing environment. However, it is useful enough as an index for estimating the approximate cost.

3 Experimental setting for random sub-networks

This section describes the configuration for the experiment in Figure 2 of this paper. First, the process of generating a random sub-network is as follows. A random sub-network is generated in a similar way to the proposed method's search process. At this time, in order to give randomness, selecting filter importance scores and candidates are randomly determined. In addition, in order to prevent the distribution of sub-networks intensively in low performance areas, we used the L_1-norm-based score for several trials so that a sub-network with higher performance is selected.

Next, as mentioned above, when calculating the validation loss of the random sub-network, the loss is calculated by selecting the batch normalization layer as the training mode. In the case of the condition number, we analyze a 1-directional landscape composed by gradient direction. Here, if the loss landscape is traversed as much as the calculated gradient, the loss diverges, making it difficult to analyze the landscape. Thus, we traversed the 0.1 times boundary of gradients and analyzed the saturated part of the sub-network more intensively. The points where condition numbers were calculated were generated by

Method	Top-1 (diff.)	Top-5 (diff.)	FLOPs ↓
AMC [9]	70.80 (-1.00)	N/A	30.0
MetaPruning [13]	71.20 (-0.80)	N/A	30.7
LeGR [1]	71.40 (-0.40)	N/A	30.0
Greedy Pruning [23]	71.60 (-0.40)	N/A	30.0
NPPM [4]	72.02 (+0.02)	90.26 (-0.12)	29.7
CafeNet [20]	73.3	91.1	30.0
EKG	71.08 (-0.20)	89.88 (-0.14)	30.2

Table 3. Comparison with various techniques for MobileNet-v2 trained on ImageNet

uniformly sampling 21 points at the traverse boundary. And all the condition numbers are averaged and used as an index to evaluate the sub-network.

Finally, the training of each random sub-network follows a plain fine-tuning strategy, and its training configuration is the same as that used when learning CIFAR mentioned above.

4 Numerical values for each plot result

Table 1 and Table 2 show the numerical value of each point at Figure 5 and the bottom of Figure 6 of this paper, respectively.

5 Additional experiments for MobileNet-v2

We additionally provide the experimental results for MobileNet-v2 trained on ImageNet according to the meta-reviewer's advise (see Table 3). Note that we were able to run two experiments on our workstation for about 10 days just after the meta-review was released. Also, since there is a risk of not being able to sufficiently tune hyper-parameters in a right time, we used the same hyper-parameters as ResNet-50. The experimental results show that the performance of the network pruned by EKG is 71.08, which is somewhat worse than the best performance of the other methods. However, the performance degradation is only -0.2%, which is low compared to most of the existing methods. Thus, we can expect that EKG can approach the SOTA performance if a significant hyper-parameter search is allowed.

References

1. Chin, T.W., Ding, R., Zhang, C., Marculescu, D.: Towards efficient model compression via learned global ranking. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 1518–1528 (2020)
2. Ding, X., Hao, T., Tan, J., Liu, J., Han, J., Guo, Y., Ding, G.: Resrep: Lossless cnn pruning via decoupling remembering and forgetting. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 4510–4520 (2021)
3. Dong, X., Yang, Y.: Network pruning via transformable architecture search. In: Advances in Neural Information Processing Systems. pp. 760–771 (2019)

4. Gao, S., Huang, F., Cai, W., Huang, H.: Network pruning via performance maximization. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 9270–9280 (2021)
5. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: Proceedings of the thirteenth international conference on artificial intelligence and statistics. pp. 249–256 (2010)
6. Guo, S., Wang, Y., Li, Q., Yan, J.: Dmcp: Differentiable markov channel pruning for neural networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 1539–1547 (2020)
7. 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)
8. He, Y., Liu, P., Wang, Z., Hu, Z., Yang, Y.: Filter pruning via geometric median for deep convolutional neural networks acceleration. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 4340–4349 (2019)
9. He, Y., Lin, J., Liu, Z., Wang, H., Li, L.J., Han, S.: Amc: Automl for model compression and acceleration on mobile devices. In: Proceedings of the European conference on computer vision (ECCV). pp. 784–800 (2018)
10. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167 (2015)
11. Kiefer, J., Wolfowitz, J., et al.: Stochastic estimation of the maximum of a regression function. The Annals of Mathematical Statistics **23**(3), 462–466 (1952)
12. Lin, M., Ji, R., Zhang, Y., Zhang, B., Wu, Y., Tian, Y.: Channel pruning via automatic structure search. arXiv preprint arXiv:2001.08565 (2020)
13. Liu, Z., Mu, H., Zhang, X., Guo, Z., Yang, X., Cheng, K.T., Sun, J.: Metapruning: Meta learning for automatic neural network channel pruning. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 3296–3305 (2019)
14. Lu, X., Huang, H., Dong, W., Li, X., Shi, G.: Beyond network pruning: a joint search-and-training approach. In: IJCAI. pp. 2583–2590 (2020)
15. Molchanov, P., Tyree, S., Karras, T., Aila, T., Kautz, J.: Pruning convolutional neural networks for resource efficient inference. In: International Conference on Learning Representations (2017)
16. Nesterov, Y.: A method for unconstrained convex minimization problem with the rate of convergence o (1/kˆ 2). In: Doklady AN USSR. vol. 269, pp. 543–547 (1983)
17. Ning, X., Zhao, T., Li, W., Lei, P., Wang, Y., Yang, H.: Dsa: More efficient budgeted pruning via differentiable sparsity allocation. arXiv preprint arXiv:2004.02164 (2020)
18. Ning, X., Zhao, T., Li, W., Lei, P., Wang, Y., Yang, H.: Dsa: More efficient budgeted pruning via differentiable sparsity allocation. In: Proceedings of the European Conference on Computer Vision (ECCV) (2020)
19. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C.: Mobilenetv2: Inverted residuals and linear bottlenecks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 4510–4520 (2018)
20. Su, X., You, S., Huang, T., Wang, F., Qian, C., Zhang, C., Xu, C.: Locally free weight sharing for network width search. In: International Conference on Learning Representations (2021), https://openreview.net/forum?id=S0UdquAnr9k
21. Su, X., You, S., Wang, F., Qian, C., Zhang, C., Xu, C.: Bcnet: Searching for network width with bilaterally coupled network. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 2175–2184 (June 2021)

22. Wang, Z., Li, C., Wang, X.: Convolutional neural network pruning with structural redundancy reduction. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 14913–14922 (2021)
23. Ye, M., Gong, C., Nie, L., Zhou, D., Klivans, A., Liu, Q.: Good subnetworks provably exist: Pruning via greedy forward selection. In: International Conference on Machine Learning. pp. 10820–10830. PMLR (2020)
24. You, Z., Yan, K., Ye, J., Ma, M., Wang, P.: Gate decorator: Global filter pruning method for accelerating deep convolutional neural networks. In: Advances in Neural Information Processing Systems. pp. 2133–2144 (2019)
25. Yu, J., Huang, T.: Autoslim: Towards one-shot architecture search for channel numbers. arXiv preprint arXiv:1903.11728 (2019)