[RE] Gate Decorator: Global Filter Pruning Method for Accelerating Deep Convolutional Neural Networks

Xueqing Zhao[1,*]
xueqing.zhao@mail.mcgill.ca

Jinsong Zhang[1,*]
jinsong.zhang@mail.mcgill.ca

Hao Sun[1,*]
hao.sun3@mail.mcgill.ca

[1]Department of Electrical and Computer Engineering
McGill University
Montreal, QC, Canada

[*] Equal contribution

Abstract

Convolutional neural networks (CNN) demonstrated its excellent performance on computer vision related tasks over the past few years. Due to the hardware capacity of computation power and runtime memory on certain devices, application of CNN models are limited. Therefore an efficient method to compress neural network models and accelerate training process is crucially needed. Recently, Zhonghui You et al. proposed a novel global filter pruning algorithm which includes Gate Decorator to solve global filter importance ranking (GFIR) problem and Tick-Tock pruning framework to improve pruning accuracy. In this paper, we aim to reproduce results obtained from Gate Decorator in [1] and investigate how the hyperparameters affect the final accuracy. We tried to reproduce selected experiments from the original paper on CIFAR-10, CIFAR-100, CUB-200 datasets and the produced results are consistent with the original paper except the CUB-200 case. We also figured out that the 'weight decay' and 'batch size' of the fine tuning could have an impact on the final results in ResNet-56 model, while the modification of sparse λ and floating point operations (FLOPs) η did not vary the accuracy too much on VGG-16-M model pruning process.

1 Introduction

Convolutional Neural Networks (CNNs) has been demonstrated as the most promising model to solve the computer vision task, e.g., the ResNet and GoogLeNet achieved a superhuman accuracy on ImageNet [2]. Compared to the earliest CNN architecture, the ResNet [3] or VGGNet [4] have more convolutional layers and more complicated architectures in order to achieve a higher accuracy. However, the number of parameters and computation in model increases frantically as the CNN model goes deeper, which prevent the model to be applied on the resource constrained devices [1]. In order to break through this bottleneck, a lot of pruning methods have been proposed in four aspects, including, quantization, fast convolution, low rank approximation and filter pruning [1]. Among those approaches, one of the latest results showed that the accuracy could remain almost same after pruning the model by reducing 70% FLOPs [1].

Three methods in [1], namely, Gate Decorator, Tick-Tock framework and Group Pruning method, were proposed to solve the following issues in filter pruning: 1) the global filter importance ranking

(GFIR) problem; 2) how to improve the pruning accuracy; 3) the constrained pruning problem, respectively. In Gate Decorator, the Taylor expansion is introduced into the minimization of the loss in pruning process, so that the GFIR can be easily computed during the process of back-propagation. In terms of the Tick-Tock framework, there are two phases (Tick and Tock) proposed to optimize the pruning accuracy. The Tick phase is designed to speed up the pruning process and calculate the importance score of each filter. In Tock phase, we could fine-tune the network to reduce the loss caused by pruning the filter [1]. Furthermore, the Group Pruning applied Gate Batch Normalization (GBN) in the pure shortcut of ResNet in order to solve the misalignment issue, which is also called the constrained pruning problem [4]. In the original experiment, the proposed algorithms were extensively demonstrated using VGG-16 and ResNet on various dataset, including, CIFAR-10, CIFAR-100, CUB-200, ImageNet ILSVRC-12 and PASCAL VOC 2011. The results showed that their model (GBN) could achieve a higher accuracy compared to other pruning algorithms [1].

In this paper, our target is to reproduce the results shown in [1], including, 1) the pruning results of ResNet-56 on CIFAR-10 (Table 1 in [1]); 2) the test results of VGG-16-M model on the CIFAR-100 (Table 3 in [1]); 3) the pruning results of VGG-16-M on CUB-200 (Figure 4 in [1]). We also swept the hyperparameters to make a further investigation on the pruning algorithm and compared the performance of VGG-16-M and pruned ResNet-56 on CIFAR-10. According to our experiments, we achieved consistent results compared to the original paper except the VGG-16-M on CUB-200 dataset. The highest accuracy of the pruned ResNet-56 we realized are 93.46% on GBN-40 (0.36% higher than the original baseline, 93.1%) and 93.00% (0.1% lower than the original baseline) on GBN-30, respectively. By further tuning the hyperparameters, (training batch size in fine tuning and the weight decay parameter in pruning model), we figured out that the accuracy will be decreased when changing the training batch size of fine tuning from 64 to 128 or decreasing the weight decay from 5e-4 to 1e-4. In terms of the VGG-16-M, the highest accuracy is 74.49% with Tick-Tock after finetuning on CIFAR-100, which is 0.11% less than the accuracy in [1]. Compared with the pruned ResNet-56, the VGG-16-M shows a higher accuracy on CIFAR-10. In the case of CUB-200, the optimal baseline we achieved, 70.275%, is around 7.925% lower than the original result, 78%. The result of baseline (42.169%) could be even worse without pre-training on ImageNet. According to the further investigation of the pruning algorithm, the hyperparameter sparse λ and FLOPs η have no obvious impacts on the performance of VGG-16-M pruning process.

2 Related work

Several related filter pruning algorithms have been mentioned and divided into two categories in the original paper [1], including, 1) prune the filters by evaluating their importance [5-8]; 2) train the model under certain restrictions [9-11]. In terms of other methods, the author of [1] mentioned that quantization methods could lead to a significant loss reduction since it reduces the number of different parameter values [12, 13]. In order to solve the accuracy loss issue in quantization methods, a moderate quantification strategy was proposed and could be in the quantization network [14, 15].

3 Dataset and setup

The datasets used by the original paper are CIFAR-10, CIFAR-100, CUB-200, ImageNet, ILSVRC-12 and PASCAL VOC2011. Due to the limited time constrains and computational resources, we decided to reproduce selected experiments on CIFAR-10, CIFAR-100 and CUB-200. CIFAR-10 [16] contains 60000, 32×32 RGB images in 10 different classes, there are 50000 training images and 10000 test images. CIFAR-100 is similar as CIFAR-10 except it has 100 classes containing 600 images each. The 100 classes in the CIFAR-100 are further grouped into 20 super classes [16]. Caltech-UCSD Birds-200-2011 (CUB-200) is another image dataset composes 11788 images of 200 bird species [17]. CIFAR-10 and CIFAR-100 are provided by TensorFlow Dataset, and CUB-200 dataset is downloaded from [17]. Notice that the resolution of each picture in CUB-200 is different, thus a resizing step is required while preprocessing the data.

4 Approach and Results

Only two examples were shown in the original code provided by the authors [18]: 1) the results (accuracy 93.15%) of ResNet - 56 on CIFAR-10 with a FLOPs reduction by 70% (GBN-30); 2) the accuracy (92.07%) of VGG-16 on CIFAR-10 with the FLOPs reduction by 90%.

In our experiment, the pruning of ResNet-56 was verified by merging the 'fine tuning' code and 'pruning' code together, which allow us to conduct a testing of GBN-30 and GBN-40 more convenient. In terms of the VGG-16, we tried to build the model from the scratch and used the pruning code provided by the authors. Note that the networks in section 4.2 and 4.3 are trained with sparse λ and FLOPs η set to 1×10^{-3} and 0 respectively. Due to the limited time constraints, we conducted the experiments in Section 4.5 using VGG-16-M on CIFAR-100.

4.1 ResNet-56 on the CIFAR-10

The accuracy (93.15%) of GBN-30 from the original code is higher than the result of the paper (93.07%). Notice that the default weight decay is 5e-4, which is different to the setting mentioned in [1]. Therefore, we changed the weight decay to 1e-4 according to the original paper and conducted the experiment with different batch sizes during the fine tuning. The performance of different combination is shown in Table 1. Note that GBN-N means that $N\%$ FLOPs are remained. The original results are 93.07% for GBN-30 and 93.41% for GBN-40. We achieved a deviation smaller than 0.1 compared to the original %accuracy (see in Table 1).

Table 1: The accuracy of GBN-30 and GBN-40 using ResNet-56

Accuracy (GBN-30 / GBN-40)	Finetuning Batch Size 64	Finetuning Batch Size 128
Weight Decay: 1×10^{-4}	92.55% / 93.46%	91.97% / $92.90\%\%$
Weight Decay: 5×10^{-4}	93.00% / 93.30%	———————

According to Table 1, these two hyperparameters could have a significant impact on the performance of the proposed filter pruning methods. It turns out that the batch size of 64 and the weight decay of 5e-4 could yield out a higher accuracy. We achieved the highest accuracy of GBN-30 and GBN-40 around 93.00% and 93.46%, respectively. According to the results (see Table 1), a higher accuracy in GBN-40 and an accuracy deterioration in GBN-30 compared to the original baseline (93.1%) were observed, which is consistent to the original paper. We also achieved a higher accuracy than the original paper for GBN-40, while the results of GBN-30 are worse. So, the experiments show that the performance of the proposed filter pruning method on ResNet could be further improved by optimizing the hyperparameters.

4.2 ResNet-56 and VGG-16-M on the CIFAR-10

In addition, we compared the performance of VGG-16-M and ResNet-56 models with GBN on CIFAR-10. Table 2 shows the corresponding pruning results. The test accuracy of the ResNet-56 is approximately 0.3% lower than the VGG-16-M model. This is primarily because the parameters for ResNet-56 is not optimized due to the limited computational resources and training time. Moreover, ResNet-56 may be too complicated model for such a relatively simple classification task. Compared with the baseline model with accuracy 93.10%, ResNet-56 after GBN reduced 60% FLOPs with a accuracy loss smaller than 0.5%.

Table 2: The comparison between VGG-16-M and ResNet-56 before fine tuning

Accuracy	20% FLOPs $\downarrow$	40% FLOPs $\downarrow$	60% FLOPs $\downarrow$	70% FLOPs $\downarrow$
VGG-16-M	93.64%	93.53%	93.37%	93.13%
ResNet-56	93.28%	93.27%	92.64%	91.76%

4.3 VGG-16-M model on CIFAR-100

Table 3 shows the accuracy before/after fine-tuning for VGG-16-M model on CIFAR-100 dataset. Note that these results are captured before fine tuning. Compared with the unpruned baseline model (73.2% accuracy), the pruned model (GBN with Tick-Tock) with 40% FLOPs reduction improves accuracy by 1.2%. When reducing 60% FLOPs, the pruned model still able to achieve the similar accuracy as the unpruned baseline model. Overall, these results are consistent with the original paper.

Table 3: The pruning results of VGG-16 on the CIFAR-100 dataset

	GBN with Tick-Tock	
FLOPS ↓	Before fine tuning	After fine tuning
40%	72.64%	74.49%
60%	70.91%	73.13%
80%	67.50%	71.14%

4.4 VGG-16-M model on CUB-200

The first paragraph of section 4.3 in [1] compared the proposed algorithm with other two global filter pruning methods, and the pruning results were plotted in Figure 4. We tried to reproduce the curve as well.

Table 4: Comparison of two baseline training process

	Resize	Pre-trained on ImageNet	Epochs	Test accuracy
baseline 1	64×64	False	160	42.169%
baseline 2	224×224	True	90	70.275%

We modified the codes provided by the author, adding a loader for the CUB-200 dataset [17]. First of all, we attempted to use a pre-trained VGG-16 model as described in the paper. This model has already been trained on ImageNet. In preprocessing, we resize the image in each channel to 224*224, which is the size lower bound of these Pytorch pre-trained modules. Other settings are exactly the same as stated in the original paper: the batch size is 64, and the learning rate is set to 3e-3 at first and divided by 3 every 30 epochs, with 90 epochs in total.

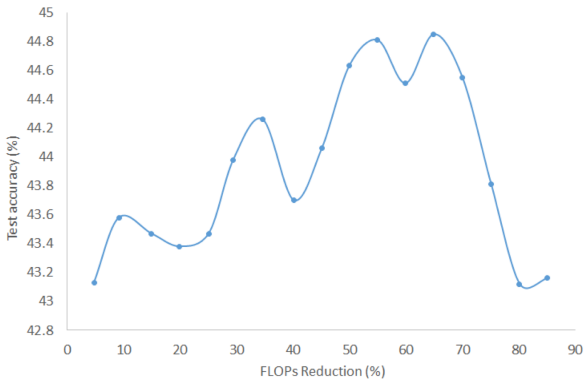

Figure 1: Pruning results of baseline 1 without fine-tuning.

The baseline model after training reached a test set accuracy of 70.275%, which is lower than the reported result (around 78%). The reason causing the difference may be that the resolution is not high enough and we could not find information about the input image size in the paper. Another issue is that we failed to prune this model since it was imported directly from the pre-trained modules in Pytorch (torchvision.models) rather than customized. Thus, the given code for pruning does not work.

Therefore, we tried to build a model from scratch. We defined a standard VGG-16 model that is able to be pruned by the provided codes. Unlike the pre-trained model, we found it's hard for this raw model to converge. The best result we got is with a input size of 64×64, a batch size of 128 and 160 training epochs, with only 42.169% accuracy on test set. Details are described in Table 4. Although the performance of the baseline model does not work as expected, we still conducted a pruning test on it. All the parameters during pruning is set to default (sparse λ = 1e-3, FLOPs $\eta = 0$). We recorded the test accuracy of pruned models without fine-tuning in Figure 1.

4.5 Additional modification to GBN on VGG-16-M model

Experiments were also conducted to explore the importance of parameters such as spare λ and FLOPs η in GBN algorithm using VGG-16-M model on CIFAR-100 dataset.

Varying sparse λ

In the Tock phase, a sparse constraint is added to the loss function to reveal the unimportant filterers. The sparse λ is a constant multiply with the sparse constraint. We have run several experiments with varying sparse λ. From figure 2, the accuracy is not affected by value of sparse λ.

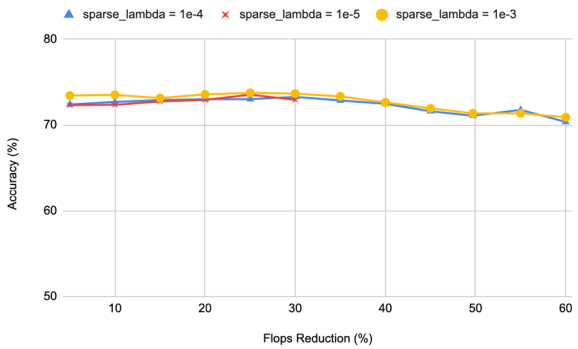

Figure 2: Results with different sparse λ.

Varying FLOPs η

The default value of FLOPs η in GBN is 0. Varying FLOPs η values may yield different accuracy. Thus, we tested our VGG-16-M model on CIFAR-100 with several FLOPs η values. It can be observed from Fig. 2, the testing accuracy with different FLOPs η, 1e-8, 1e-10 and 0 overlapped while the pruning ratio is smaller than 70%. FLOPs η = 1e-10 improves testing accuracy slightly (1%) when 75% FLOPs pruned in comparison with the default FLOPs η = 0.

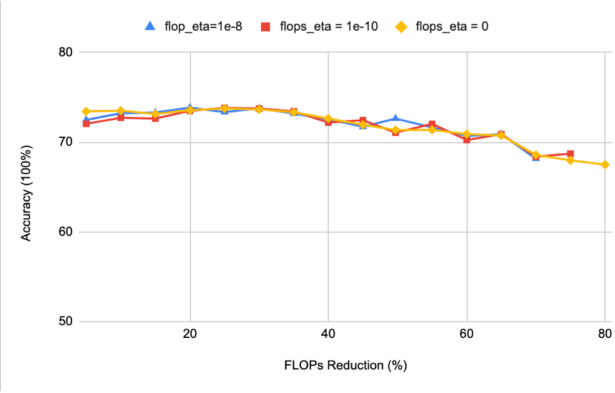

Figure 3: Results with different FLOPs η.

5 Conclusion

In this project, we managed to reproduce a subset of results in the original paper [1] using the open-source codes provided by the author. Firstly, we applied ResNet-56 on CIFAR-10 dataset and achieved the best similarity (accuracy difference < 0.12%) with the original results with weight decay of 5e-4 and fine-tuning batch size of 64. Then we successfully reproduced the performance of VGG-16-M on CIFAR-100. Unfortunately, though we attempted to employ VGG-16-M model on CUB-200 dataset, we failed to get the similar curve as the targeting paper due to uncertainty of some key hyperparameters.

Aside from baseline reproduction, we made a few extensions as well. We compared the performance of two different models: VGG-16-M and ResNet-56 on CIFAR-10 dataset. It turns out that VGG-16-M outperforms ResNet-56 on this task in terms of test accuracy after pruning. Furthermore, we conducted hyperparameter study using VGG-16-M on CIFAR-100 dataset. We swept sparse λ and FLOPs η that are two important values in GBN. It can be inferred from the results that both parameters do not have a strong impact on the pruning process, which facilitates the optimization of GBN. In terms of the future works, a more powerful GPU is needed to speed up the training process and the proposed pruning algorithm could be further optimized by tuning the hyperparameters.

Statement of contribution

Xueqing was in charge of the VGG-16-M model on CIFAR-100 and CIFAR-10 as well as writing the report.

Jinsong worked on VGG-16-M model in CUB-200 dataset and data analysis. He also contributed to write the report.

Hao worked on ResNet-56 and writing the report.

References

[1] Zhonghui You, Kun Yan, Jinmian Ye, Meng Ma, and Ping Wang. "Gate decorator: Global filter pruning method for accelerating deep convolutional neural networks." In Advances in Neural Information Processing Systems, pp. 2130-2141. 2019.

[2] William L. Hamilton. "COMP 551 - Applied Machine Learning", McGill University and Mila, 2019.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 770–778, 2016.

[4] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In International Conference on Learning Representations (ICLR), 2015.

[5] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. In International Conference on Learning Representations (ICLR), 2017.

[6] Hengyuan Hu, Rui Peng, Yu-Wing Tai, and Chi-Keung Tang. Network trimming: A data-driven neuron pruning approach towards efficient deep architectures. arXiv, abs/1607.03250, 2016.

[7] Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In IEEE International Conference on Computer Vision (ICCV), pages 1398–1406, 2017.

[8] Jian-Hao Luo, Jianxin Wu, and Weiyao Lin. Thinet: A filter level pruning method for deep neural network compression. In IEEE International Conference on Computer Vision (ICCV), pages 5068–5076, 2017.

[9] Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learning efficient convolutional networks through network slimming. In IEEE International Conference on Computer Vision (ICCV), pages 2755–2763, 2017.

[10] JianboYe,XinLu,ZheLin,andJamesZ.Wang.Rethinkingthesmaller-norm-less-informativeassumption in channel pruning of convolution layers. In International Conference on Learning Representations (ICLR), 2018.

[11] Xiaohan Ding, Guiguang Ding, Yuchen Guo, and Jungong Han. Centripetal SGD for pruning very deep convolutional networks with complicated structure. arXiv, abs/1904.03837, 2019.

[12] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In European Conference on Computer Vision (ECCV), pages 525–542, 2016.

[13] Matthieu Courbariaux and Yoshua Bengio. Binarynet: Training deep neural networks with weights and activations constrained to +1 or -1. arXiv, abs/1602.02830, 2016.

[14] Aojun Zhou, Anbang Yao, Yiwen Guo, Lin Xu, and Yurong Chen. Incremental network quantization: Towards lossless cnns with low-precision weights. In International Conference on Learning Representations (ICLR), 2017.

[15] Bohan Zhuang, Chunhua Shen, Mingkui Tan, Lingqiao Liu, and Ian D. Reid. Towards effective low-bitwidth convolutional neural networks. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 7920–7928, 2018.

[16] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

[17] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The Caltech-UCSD Birds-200-2011 Dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011.

[18] https://github.com/youzhonghui/gate-decorator-pruning