Supplementary: Pushing the Efficiency Limit Using Structured Sparse Convolutions

Vinay Kumar Verma[1*], Nikhil Mehta[1*], Shijing Si[3], Ricardo Henao[1], Lawrence Carin[1,2]
[1]Duke University [2]KAUST Saudi Arabia [3]SEF, Shanghai International Studies University

1. Experimental Details

To show the efficacy of the proposed model we leverage three standard architectures VGG-19 [5], ResNet-32 [1] and ResNet-50 [1]. We use these architecture on the small scale (CIFAR-10), medium scale (CIFAR-100 and Tiny-ImageNet) and large scale (ImageNet) datasets. We evaluated VGG-19 and ResNet-32 architecture over the CIFAR-100, CIFAR-10 and Tiny-ImageNet dataset. Similar to GraSP [6], we doubled the number of filters on each layer for the ResNet-32 architecture. We used the ResNet-50 architecture for the ImageNet dataset. In our approach, the pruning ratio is controlled by changing the hyperparameters p and g. We use the expression for reduction in the number of parameters (given in equation (10) of the main text) as the guiding principle to create candidate values for hyperparameters p and g. From these candidate values, we decide p and g by testing the model on validation data (10% of the training data). Similar to other pruning approaches [4, 2, 6, 3], we empirically observed that the initial layers are more important compared to the deeper layers, hence, high pruning on the initial layers showed significant degradation of model's performance. Moreover, the initial layers have fewer number of parameters compared to the deep layers, and pruning these layers do not help increase the overall pruning ratio.

1.1. CIFAR10 on VGG-19/ResNet-32 Architecture

Following [6], we use a modified low-resolution version of VGG-19 [5] architecture for the CIFAR dataset. It contains 3×3 convolutional filter across five blocks having [64,128,256,512,512] number of filters. The deeper layers lead to most of the compression as they contain more number of the parameters compared to layers closer to the input. We define the $\alpha\%$ winning ticket as the architecture with only $\alpha\%$ parameters of the standard original architecture. On the VGG-19 architecture (CIFAR-10 dataset) for the 10% winning ticket, layers with number of filters [64,128,256,512] have $g = [1, 2, 4, 16]$ and $p = [1, 2, 2, 2]$. In particular, we use the same set of g and p for layers with the same filter size, for instance in the 10% winning ticket, each layer of size 128 has $g = 2$ and $p = 2$. Similarly for the 5%, we have $g = [1, 1, 4, 16], p = [1, 2, 4, 16]$, and for 2% winning ticket, $g = [1, 4, 8, 64]$ and $p = [1, 4, 8, 25]$. For the ResNet-32, architecture we have three blocks of filter with $[32, 64, 128]$ number of filter. Table 1 summarizes the hyperparameters p and g for three values of α for ResNet-32.

We trained the VGG-19 and ResNet-32 model with the SGD optimizer for the 250 epoch with initial learning rate of 0.1 and weight decay 0.0001. The learning rate is decreased by a factor $0.1\times$ at the epoch 100, 150, and 200. For all the experiment a batch size of 128 is used.

Winning Ticket	p	g	Acc.
10%	$[2, 4, 15]$	$[2, 4, 4]$	93.3
5%	$[2, 8, 64]$	$[2, 8, 32]$	91.7
2%	$[16, 32, 64]$	$[8, 32, 64]$	88.4

Table 1. The value of p and g for the ResNet-32 architecture on the CIFAR-10 dataset.

1.2. CIFAR100 on VGG-19/ResNet-32 Architecture

The hyperparameters p and g used for the CIFAR-10 dataset cannot be directly used for CIFAR-100 since the last fully connected layer significantly increases the model size. Therefore for the optimal performance we tune the p and g value on the validation data. On the VGG-19 architecture we use $g = [1, 1, 2, 10]$ and $p = [1, 2, 2, 15]$ for the 10% winning ticket. Similarly, for the 5% and 2% winning ticket we have $g = [1, 1, 3, 30], p = [1, 2, 4, 30]$ and $g = [1, 4, 8, 64], p = [1, 4, 8, 25]$ respectively.

For the ResNet-32 architecture, we have three blocks of $[32, 64, 128]$ number of filters. The hyperparameters p and g are given in table 2.

We trained the VGG-19 and ResNet-32 model with the SGD optimizer for the 250 epoch with initial learning rate of 0.1 and weight decay 0.0001. The learning rate is decreased by a factor $0.1\times$ at the epoch 100, 150, and 200. For all the experiment a batch size of 128 is used.

[*]denotes equal contribution. Correspondence to vinayugc@gmail.com, nick.mehta@duke.edu.

Winning Ticket	p	g	Acc.
10%	$[2, 4, 16]$	$[2, 4, 4]$	71.1
5%	$[2, 6, 64]$	$[2, 16, 32]$	67.3
2%	$[16, 32, 64]$	$[16, 64, 64]$	63.0

Table 2. The value of p and g for the ResNet-32 architecture on the CIFAR-100 dataset.

1.3. Tiny-ImageNet on ResNet-32/VGG-19 Architecture

Tiny-ImageNet is a medium scale dataset containing 200 classes of the ImageNet dataset of low resolution. Tiny-ImageNet is trained on the VGG-19 architecture for the 350 epoch with the initial learning rate of 0.1 and weight decay 0.0001. We decrease the learning rate by a factor $0.1\times$ after epochs 150, 250 and 300. We use three pruning ratio 90%, 95% and 98% for the VGG-19 architecture, however for the ResNet-32, pruning ratio 85%, 90% and 95% is used to be consistent with the baselines. For the VGG-19 architecture we use $g = [1, 1, 2, 10], p = [1, 2, 2, 15]$, $g = [1, 1, 3, 35], p = [1, 2, 4, 35]$ and $g = [1, 4, 8, 64], p = [1, 4, 12, 40]$ for α equals 90%, 95% and 98% respectively. Similarly for the ResNet-32 architecture to prune the model 85% we use $g = [1, 2, 11], p = [1, 4, 10]$. Also, for the 90% and 95% pruning ration $g = [2, 4, 16], p = [2, 4, 8]$ and $g = [2, 10, 64], p = [2, 16, 32]$ are used respectively.

1.4. ImageNet on ResNet-50 Architecture

For ImageNet, we use the ResNet-50 architecture with two different values of α i.e. 40% and 20%. The model is trained for 150 epochs similar to GraSP with an initial learning rate of 0.1 and weight decay 0.0001. We decrease the learning rate after every 50 epoch by a factor $0.1\times$. ResNet-50 contains 3×3 convolutional filter and 1×1 convolutional filter. For the $\alpha = 40\%$ winning ticket, we use $g = [1, 1, 4, 8], p = [1, 2, 2, 2]$ on the layer with $[64, 128, 256, 512]$ number of 3×3 filters. In the ResNet-50 most of the layers contain 1×1 filter. We use the $p = [1, 1, 2, 2, 2, 2]$ for layers containing $[64, 128, 256, 512, 1024, 2048]$ number of 1×1 filters. Similarly for $\alpha = 20\%$, we use $g = [1, 1, 4, 6], p = [1, 2, 4, 8]$ on the 3×3 filters and $p = [1, 1, 4, 4, 8, 8]$ for the 1×1 filter.

2. Sensitivity over p and g value

We discussed earlier that our model's pruning can be controlled by the hyperparameters p and g. In this section, we show that the model performance is not extremely sensitive to different values of p and g. We perform a study for different hyperparameters on the ResNet-32 architecture for CIFAR-100 dataset. We use the same winning ticket (10%) for the different candidate values of p and g and evaluate the model's performance. The results is shown in the table-3.

However, if the initial layers (layers close to the input) are pruned excessively, the model performance degrades. For example, see the last row in table 3. This observation is similar to what has been observed in previous works [4, 2, 6, 3].

g	p	accuracy
$[1, 6, 16]$	$[2, 6, 6]$	69.95
$[2, 6, 15]$	$[2, 3, 4]$	69.97
$[2, 4, 16]$	$[2, 4, 5]$	70.31
$[1, 4, 15]$	$[2, 16, 16]$	69.85
$[2, 4, 19]$	$[2, 4, 4]$	70.69
$[4, 8, 8]$	$[4, 8, 9]$	69.45

Table 3. Ablation over the ResNet-32 architecture for the CIFAR-100 dataset. We set the p and g value such that all the model has same winning ticket (10%).

We can conclude that each layer has different importance with respect to pruning; pruning more parameters from the important layers degrade the performance. Therefore, selecting the R_F (equation (10) in the main text) for each layer is important.

References

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[2] Yang He, Guoliang Kang, Xuanyi Dong, Yanwei Fu, and Yi Yang. Soft filter pruning for accelerating deep convolutional neural networks. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, IJCAI'18, pages 2234–2240. AAAI Press, 2018.

[3] Yang He, Ping Liu, Ziwei Wang, Zhilan Hu, and Yi Yang. Filter pruning via geometric median for deep convolutional neural networks acceleration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4340–4349, 2019.

[4] Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1389–1397, 2017.

[5] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *International Conference on Learning Representations*, 2014.

[6] Chaoqi Wang, Zhang, Guodong, and Roger Grosse. Picking winning tickets before training by preserving gradient flow. *ICLR*, 2020.