One-Cycle Structured Pruning via Stability-Driven Subnetwork Search

Supplementary Material

6. Additional Ablation Studies

6.1. Group Saliency vs Conventional Saliency

Conventionally, filter saliency estimation evaluates each convolutional layer in isolation when ranking filters based on their importance. In this approach, filters are assessed solely based on their individual contributions to the output of their respective layers, without accounting for interdependencies with other components in the network. In contrast, the group saliency used in our paper considers all parameters within a group, including those from interconnected convolutional layers, batch normalization layers, and fully connected layers.

Given the varied dimensions and scales of different group parameters, as described in Sec. 3.2, we apply local normalization before computing the group saliency scores. This step ensures a fair comparison across heterogeneous parameter types and prevents components with inherently larger magnitudes from being overemphasized. As shown in Tab. 6, pruning based on group saliency consistently outperforms conventional saliency estimation, leading to better accuracy-compression trade-offs.

6.2. Analysis of Structured Sparsity Regularization

Regularization in the proposed pruning pipeline helps the stability indicator reach a stable epoch for final pruning. In particular, we utilize Eq. (8) and Eq. (9) from Sec. 3.4 of the main paper for this purpose. While Eq. (8) exhibit a slower sparsity learning, Eq. (9) is designed to speed up this process. Tab. 7 presents accuracy results for comparable FLOPs pruning rates, considering the application of regularization with Eq. (8) and Eq. (9) independently, and in combination. It is important to highlight that the parameter reduction rate is automatically determined based on the pruned architecture for a given FLOPs reduction rate. The accuracy values in the table indicate that better results are achieved when both equations Eq. (8) and Eq. (9) are used together for structured sparsity regularization. Fig. 5 illustrates the progress of sparsity learning, starting from the sparsity learning initiation epoch ($t_{\text{sl-start}}$) up to the epoch of stable pruning (t^*).

Table 6. Performance comparison on CIFAR-10 using OCSPruner with group vs. conventional saliency.

Model	Group Saliency			Conventional Saliency		
	FLOPs (%)	Params (%)	Acc. (%)	FLOPs (%)	Params (%)	Acc. (%)
Vgg16	19.99	7.18	93.87	20.00	3.75	92.94
ResNet32	50.09	57.88	92.94	50.04	38.61	92.60
ResNet56	46.88	47.66	93.85	47.03	43.93	93.48

Table 7. Ablation study on pruning accuracy using Eq. (8) and Eq. (9) independently and in combination for structured sparsity regularization.

Model/ Dataset	Sparsity Learning		FLOPs (%)	Params (%)	Top-1 Acc. (%)	Top-5 Acc. (%)
	Eq. (8)	Eq. (9)				
ResNet32/ CIFAR-10	✓	×	44.69	38.59	92.09	-
	×	✓	44.74	46.94	92.47	-
	✓	✓	44.98	48.28	92.74	-
ResNet18/ ImageNet	✓	×	55.09	55.13	67.89	88.00
	×	✓	55.08	66.55	68.52	88.31
	✓	✓	55.04	67.90	68.66	88.61

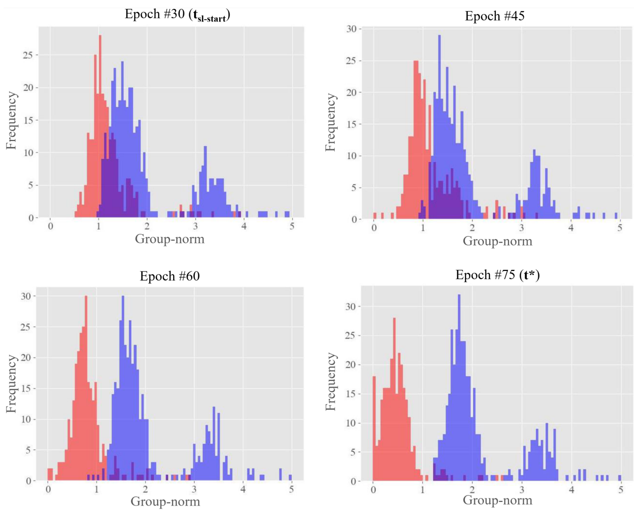

Figure 5. Group-norm distribution of weight parameters while pruning ResNet32 model on CIFAR-10 dataset. Using structured sparsity learning (Eq. (8) and Eq. (9)), the proposed method gradually drives grouped parameters (highlighted in red) toward zero.

7. Training and Pruning Details

7.1. Hyper-parameter Configuration

Tab. 8 provides the training and pruning hyperparameters used for experiments on the CIFAR and ImageNet datasets. For the CIFAR dataset, the same parameters are utilized regardless of the model used. However, for the ImageNet dataset, slightly different training settings are employed for the ResNet50 and MobileNetV2 models.

Table 8. Training and pruning hyper-parameters for OCSPruner.

Task	Parameters	Dataset	
		CIFAR	ImageNet
Training	batch size	128	256
	epochs	300	ResNet50: 130 MobileNetV2: 150
	optimizer	SGD	SGD
	momentum	0.9	0.9
	learning rate	0.1	ResNet50: 0.1 MobileNetV2: 0.05
	learning rate decay	MultiStepLR (90, 180, 240, 270; 0.2)	Cosine
	weight decay	5e-4	4e-5
Pruning	threshold for $t_{\text{sl-start}}$ (τ)	-	1e-4
	$t_{\text{sl-start}}$	30	auto (Eq. (4))
	threshold for t^* (ϵ)	1e-3	1e-3
	reg. penalty initial value (λ_0)	1e-4	1e-4
	reg. penalty increment value (σ)	1e-4	1e-4
	λ increment epoch interval (Δt)	1	2

Table 9. The comparison of OCSPruner overall cost versus baseline model training cost.

Dataset	Model	Baseline Model Training Time	OCSPruner Total Time
CIFAR-10	VGG16	0 h 25 min	0 h 27 min
	ResNet56	0 h 25 min	0 h 26 min
CIFAR-100	VGG19	0 h 26 min	0 h 30 min
ImageNet	ResNet50	37 h 55 min	27 h 6 min
	MobileNetV2	38 h 20 min	31 h 57 min

7.2. Computational Cost Analysis

We performed algorithm complexity analysis on a PC with a 64-bit Ubuntu 20.04.6 LTS operating system, driven by an AMD Ryzen 9 7950X 16-Core Processor, 62 GB RAM, and two NVIDIA GeForce RTX 4090 GPUs. Notably, all experiments utilized a single GPU except for ResNet50 pruning on ImageNet, where both GPUs were used. Tab. 9 presents OCSPruner's time complexity, highlighting its efficiency relative to baseline training cost, particularly for large-scale datasets where training time is crucial. While the OCSPruner cost for CIFAR-10/100 datasets slightly surpasses the base network training cost, it is crucial to acknowledge that network slimming offers negligible advantages in small-scale dataset training time, given that the pruning process cost is already factored into the overall calculation.

8. Inference Efficiency on Edge Devices

The primary aim of structurally pruning baseline models is to make them suitable for deployment on low-powered edge devices, resulting in reduced model size and inference time. To validate this, we conducted experiments using the NVIDIA Jetson Xavier NX board to evaluate the inference time of our pruned models on such a low-powered device. In our evaluations, as summarized in Tab. 10, we use a batch size of 128 for the CIFAR-10/100 datasets and a batch size of 16 for the ImageNet dataset. We conducted 200 inference runs in each test case and averaged the latency time. For example, pruning the ResNet50 model FLOPs by 67% on the ImageNet dataset results in a practical speedup of 2.45$\times$ compared to the baseline model's latency. However, the theoretical speedup is expected to be 3.03$\times$. Similarly, in practice, we can see that an 89% FLOPs reduction in the VGG19 model on the CIFAR-100 dataset leads to a 4.22$\times$ speedup in inference latency, whereas the theoretical speedup is around 8.5$\times$.

9. Applying OCSPruner to ViT & DeiT Models

The OCSPruner is also used to prune Vision Transformer (ViT) [1] architecture, which has gained significant attention in computer vision. We integrate OCSPruner into the widely-used DeiT training framework for image classification by applying data augmentation techniques similar to DeiT [4]. The training uses the AdamW optimizer with a cosine learning rate scheduler over 300 epochs, a batch size of 1024, a weight decay of 0.05, and an initial learning rate of 5e-4. For ViT, a warm-up period of 10 epochs with a warm-up learning rate of 0.001$\times$lr is used, while for DeiT, pruning is followed by training with a 5-epoch warm-up at a warm-up learning rate of 0.1$\times$lr. Remarkably, our method achieved a 30-40% reduction in FLOPs while training from scratch on ImageNet. Despite this substantial reduction in FLOPs, the pruned models retained performance comparable to the original counterparts, and in the case of DeiT, the pruned model even showed improved performance.

10. Extended Applications: Object Detection

In this experiment, we applied a pruned ResNet50 model with OCSPruner as a backbone network for object detection. We use Single Shot multibox Detector (SSD) [3] as our detection network with an input size resolution of 300 × 300. The COCO object detection dataset [2] is used for training and evaluation which contains images of 80 object categories with 2.5 million labeled instances in 328k images. The pruned ResNet50 backbone model trained with OCSPruner on the ImageNet dataset is attached with SSD head and finetuned on the COCO object detection dataset. The model is trained for 130 epochs with a batch size of 32, a learning rate of 0.0026, and a weight decay of 0.0005,

Table 10. Latency analysis comparing baseline and pruned models on various datasets using the NVIDIA Jetson board. The percentage value in the subscript denotes the FLOPs pruning rate.

Dataset	Model	Method	MACs (G)	Params (M)	NVIDIA Jetson Xavier NX		
					Batch Size	Latency (ms)	Speed Up
CIFAR-10	ResNet56	Baseline	0.13	0.86	128	97.2	1.00×
		OCSPruner$_{61\%}$	0.05	0.32	128	63.6	1.54×
	ResNet110	Baseline	0.26	1.73	128	193.0	1.00×
		OCSPruner$_{67\%}$	0.08	0.51	128	110.5	1.75×
	VGG16	Baseline	0.40	14.73	128	90.8	1.00×
		OCSPruner$_{80\%}$	0.08	1.0	128	31.2	2.90×
CIFAR-100	VGG19	Baseline	0.51	20.09	128	114.2	1.00×
		OCSPruner$_{89\%}$	0.06	1.15	128	27.3	4.22×
ImageNet	ResNet50	Baseline	4.12	25.56	16	256.9	1.00×
		OCSPruner$_{43\%}$	2.35	16.47	16	164.4	1.56×
		OCSPruner$_{48\%}$	2.13	15.89	16	154.5	1.66×
		OCSPruner$_{57\%}$	1.78	13.25	16	133.1	1.93×
		OCSPruner$_{67\%}$	1.36	10.41	16	105.3	2.45×

Table 11. Pruning ViT and DeiT models on ImageNet-1k.

Model	Method	Params (M)	FLOPs (G)	Top-1 ACC (%)	Top-5 ACC (%)
ViT-Small	Baseline	22.05	4.61	78.84	95.33
	OCSPruner	12.18	2.72	77.59	93.31
DeiT-Tiny	Baseline	5.72	1.26	72.20	91.10
	OCSPruner	3.82	0.88	72.51	91.15

Table 12. Object Detection results using SSD300 with a pruned ResNet50 backbone on the COCO dataset.

Backbone (%)	FLOPs (G) (%)	Params (M) (%)	Average Precision (AP)	Average Recall (AR)
ResNet50	100	100	26.90	38.10
Pruned ResNet50	74.58	86.63	25.78	37.01
	65.67	81.52	25.30	36.50
	44.06	68.11	23.82	35.44

with the learning rate decayed at epochs 86 and 108, utilizing two NVIDIA GeForce RTX 4090 GPUs.

The performance of SSD300 with a pruned ResNet50 backbone is detailed in Tab. 12. In our experimental setup using the default ResNet50 backbone, we achieved an average precision (AP) of 26.90%. Transitioning to a pruned ResNet50 backbone resulted in a significant reduction in FLOPs compared to parameter reduction rates. By simplifying the backbone architecture, we achieved a notable overall complexity reduction, albeit requiring channel pruning in the SSD head for substantial parameter reduction. It's important to note that our experiments focused solely on varying pruning rates for the backbone while keeping the SSD head unchanged. Remarkably, we observed only a 1.12% drop in AP with a 25.42% reduction in FLOPs. Furthermore, an approximately 3% drop in AP corresponded to a 56% reduction in overall FLOPs.

11. Visualization of Pruned Network Structures

The visualization in Fig. 6 shows the result of pruning across different models, pruning rates, and datasets with OCSPruner. Baseline models often exhibit unnecessary complexity that is typically not required. As illustrated in Fig. 6, when pruning VGG16/19, numerous filters from the later parts of the network are eliminated, whereas when pruning ResNet models, filters are consistently removed from all parts of the network. Yet, the pruned networks continue to perform well. This observation suggests that layers with higher pruning rates capture minimal or redundant information, rendering them suitable for pruning.

References

[1] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 2

[2] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. *arXiv preprint arXiv:1405.0312*, 2014. 2

[3] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian

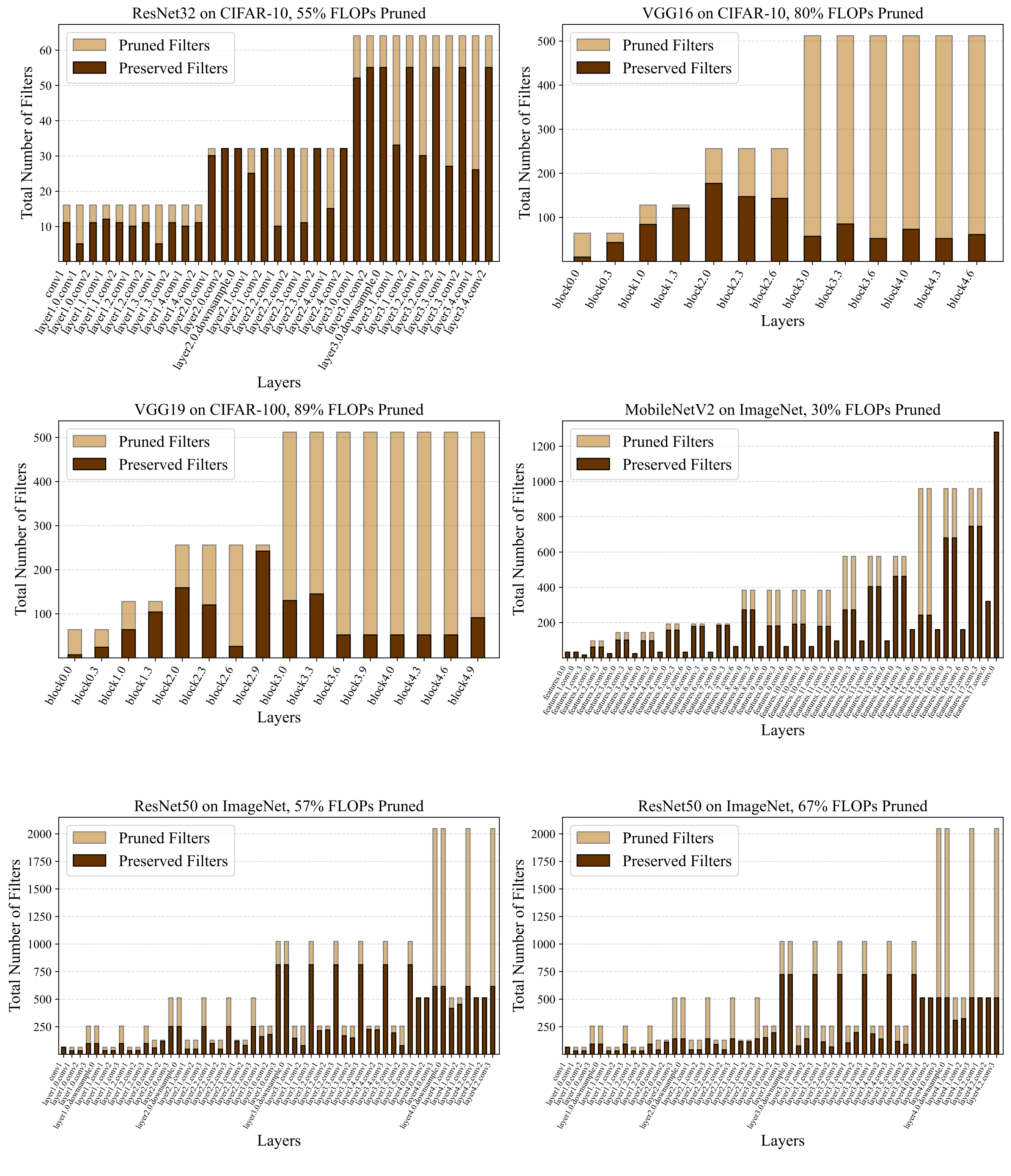

Figure 6. Visualization of pruned models displaying pruned and retained filters across each layer's total number of filters.

Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*, pages 21–37. Springer, 2016. 2

[4] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Herve Jegou. Training data-efficient image transformers amp; distillation through attention. In *International Conference on Machine Learning*, pages 10347–10357, 2021. 2