Supplementary Material

Sara Elkerdawy[1][0000−0002−9607−3225], Mostafa Elhoushi[2], Abhineet Singh[1], Hong Zhang[1], and Nilanjan Ray[1]

[1] Department of Computing Science, University of Alberta, Canada
{elkerdaw, asingh1, hzhang, nray1}@ualberta.ca
[2] Toronto Heterogeneous Compilers Lab, Huawei, Canada

1 CIFAR

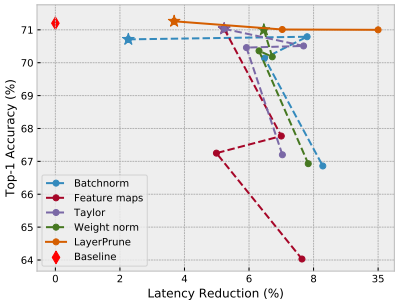
1.1 Training setup

We follow standard hyperparameters used for fine-tuning [1–3]: 30 epoch learning rate $1e^{-3}$ on SGD optimizer. That is except comparison with Chen et al. [4] as the authors train the pruned model with the standard hyperparameters used for training from scratch: 160 epoch with initial learning rate 0.1 and decays on epoch [81, 122] by 0.1.
Layer pruning: For layer pruning, we calculate layer importance as explained in the paper using a one-shot pass over the training set.
Filter pruning: For filter pruning, we prune total 500 and 100 filters in VGG19 and ResNet56 respectively for global-based filter importance criteria such as weight norm, Taylor, Feature maps. We follow the same iterative pruning hyperparameter setup as Taylor [1]. We prune 100 filters each 10 minibatches. Other pruning methods, we report results using their published code with default setup setting such as slimming [5] and ECC[6].

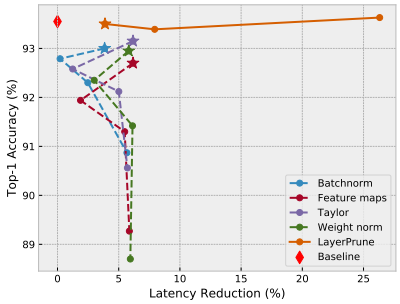
1.2 Ablation

Number of filters pruned. We show accuracy degradation on aggressive filter pruning and the achieved latency reduction compared to LayerPrune. **Fig. 1** shows filter pruning under different number of filters pruned (i.e 100:400) and latency reduction on GPU 1080Ti on batch size=64. Dots are connected based on ascending order of number of filters pruned. It is apparent that pruning more filters doesn't necessarily decrease latency and the relationship between pruned filters and latency reduction is non-linear. In CIFAR-100, latency reduction $\approx$ 8% results in large drop in accuracy from 71.2% to 67%. It is worth noting that LayerPrune is able to achieve up to 35% latency reduction with accuracy 71%. Similarly on CIFAR-10, pruning 50% of the filters can only achieve around 5% latency reduction.
As for VGG19, the maximally achieved pruning latency reduction is 20% to maintain the accuracy from baseline; while LayerPrune finds better models than baseline and filter pruned methods. On comparison with the random experiment

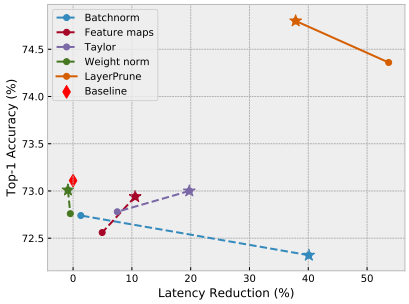
shown in Section (4.2) in main paper, filter pruning methods hover around baseline accuracy and fails to discover other regularized models compared to layer pruning.



(a) CIFAR-100/ResNet56



(b) CIFAR-10/ResNet56



(c) CIFAR-100/VGG19-BN

Fig. 1: Latency reduction of different filter pruning methods under different pruning ratios. Star in each method indicates the lowest pruning ratio (starting point). Dots are connected based on ascending order of number of filters pruned.

One-shot vs iterative. We also conducted experiments on one-shot vs iterative filter pruning to be comparable with our one-shot LayerPrune pruning step. Our reported results on iterative filter pruning follows the same setup used in literature [1], that is prune 10% each pruning iteration. **Table 1** shows results of iterative vs one-shot (pruning total number of filters at once). Consistent with [1, 2], iterative pruning (i.e re-evaluating criterion of filter after each prune) gives a slightly better accuracy. That shows that it is mandatory for filter pruning to be iterative.

Dataset/Model	Pruning ratio	Criterion	Iterative	One-shot
CIFAR100/ResNet56 (71.20%)	20%	Weight Norm	70.18	70.00
		Feature Maps	67.77	67.7
		Taylor	70.51	70.01
		Batchnorm	70.79	70.36
		Median	**70.34**	70.00
	30%	Weight Norm	70.36	69.1
		Feature Maps	67.25	66.34
		Taylor	70.46	68.27
		Batchnorm	70.14	69.4
		Median	**70.25**	68.68
CIFAR10/ResNet56 (93.55%)	20%	Weight Norm	92.35	92.31
		Feature Maps	91.94	91.9
		Taylor	92.88	92.8
		Batchnorm	92.79	92.76
		Median	**92.57**	92.53

Table 1: Evaluation of iterative and one-shot filter pruning. Baseline accuracy indicates in parentheses.

To analyze the sensitivity of ranking by imprinting on layer pruned models, we calculated Spearman's rank-order correlation between layer ranking by one-shot and layer ranking by re-calculating ranks iteratively after each pruning step. **Table 2** shows accuracy of one-shot and iterative layer pruning and their ranking correlation. The Spearman column indicates high positive relationship between both ranking methods demonstrating the robustness of ranking by imprinting. We observed the difference in ranking is between similarly important layers and this explains why accuracy isn't significantly affected even as correlation decreases, and it shows the sufficiency of one-shot rank estimation with imprinting.

	CIFAR-10 ResNet-56 (93.55%)			CIFAR-100 ResNet-56 (71.2%)		
N pruned	One-shot (%)	Iterative (%)	Spearman	One-shot (%)	Iterative (%)	Spearman
1	93.32	93.32	0.99	71.10	71.10	0.96
2	93.28	93.31	0.97	70.93	70.94	0.97
3	93.17	93.15	0.96	70.88	70.86	0.94
4	93.10	93.03	0.96	70.64	70.71	0.91

Table 2: Spearman rank correlation between one-shot and iterative ranking with imprinting.

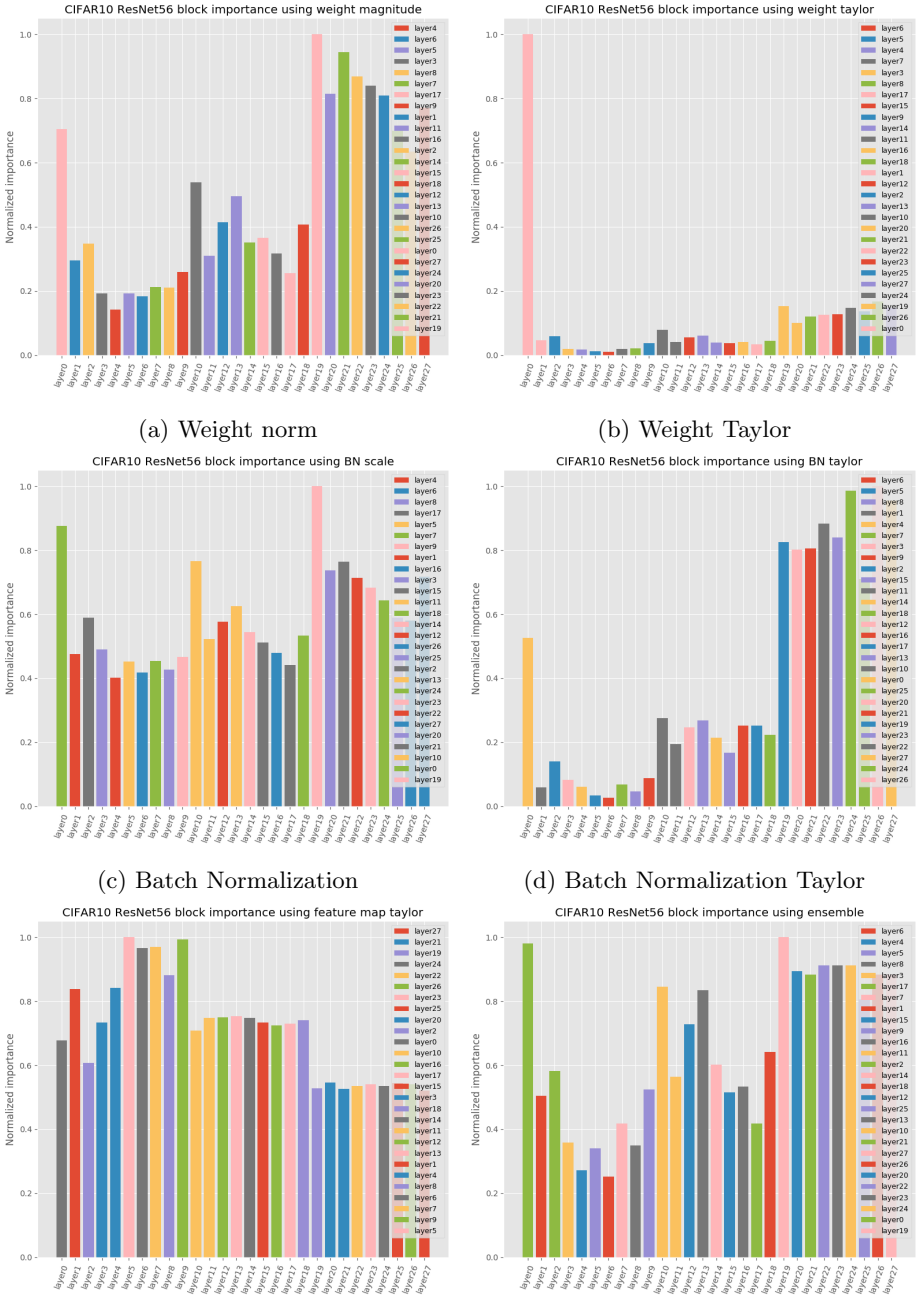
1.3 Architectures

Architectures for results on VGG19-BN are presented in **Table 3**. All layer pruning methods mostly agree on removing same layers. While in filter pruning methods, as minimum number of filters are required per layer, the early layers are pruned as well and thus hurting accuracy.

Method	Accuracy	Architecture
VGG19 (baseline)	73.11	[64, 64, 'M', 128, 128, 'M', 256, 256, 256, 256, 'M', 512, 512, 512, 512, 'M', 512, 512, 512, 512, 'M']
Weight norm [7]	73.01	[47, 64, 'M', 127, 128, 'M', 256, 256, 256, 256, 'M', 512, 508, 494, 472, 'M', 502, 512, 499, 509, 'M']
ECC [6]	72.71	[50, 23, 'M', 128, 128, 'M', 254, 254, 254, 254, 'M', 508, 311, 164, 131, 'M', 158, 319, 509, 64, 'M']
Layer pruning$_2$	73.60	[64, 64, 'M', 128, 128, 'M', 256, 256, 256, 256, 'M', 512, 512, 512, 512, 'M', **0, 0,** 512, 512,'M']
Layer pruning$_5$	74.80	[64, 64, 'M', 128, 128, 'M', 256, 256, 256, 256, 'M', 512, 512, **0, 0,** 'M', **0, 0, 0,** 512, 'M']
Slimming [5]	72.32	[42, 64, 'M', 125, 128, 'M', 255, 256, 255, 256, 'M', 433, 291, 82, 46, 'M', 45, 44, 62, 367, 'M']
Layer pruning$_2$	73.60	[64, 64, 'M', 128, 128, 'M', 256, 256, 256, 256, 'M', 512, 512, 512, 512, 'M', **0, 0,** 512, 512,'M']
Layer pruning$_5$	74.80	[64, 64, 'M', 128, 128, 'M', 256, 256, 256, 256, 'M', 512, 512, **0, 0,** 'M', **0, 0, 0,** 512, 'M']
Taylor [1]	72.61	[61, 64, 'M', 127, 128,'M', 256, 256, 256, 256,'M', 512, 505, 383, 205,'M', 109, 118, 422,482, 'M']
Layer pruning$_2$	73.60	[64, 64, 'M', 128, 128, 'M', 256, 256, 256, 256, 'M', 512, 512, 512, **0,** 'M', **0,** 512, 512, 512,'M']
Layer pruning$_5$	74.80	[64, 64, 'M', 128, 128, 'M', 256, 256, 256, 256, 'M', 512, 512, **0, 0**, 'M', **0, 0, 0,** 512, 'M']

Table 3: Architectures of different pruning methods on VGG19-BN CIFAR-100. x in Layer pruning$_x$ indicates number of layers removed. Number of filters per layer is shown where 0 indicates removed layers and 'M' indicates max pooling operation.

ResNet56 has 3 groups of 9 basicblocks where each basicblock has two 3x3 convolution layer. We show block importance based on each criterion for CIFAR-10 in Fig. 2 and CIFAR-100 in **Fig. 3**. Weight magnitude, Batch Normalization and Taylor magnitude criteria have similar block ordering that focus more on the early layers. On the other hand, feature maps criterion is more biased to pruning the deeper layers. This stems from the fact that as we go deeper, feature maps tend to be sparser and so their importance calculated using Taylor on feature maps [8] will lead to a bias and failure in deeper models. Ensemble selects layers that are constantly voted as not important (e.g CIFAR-10 blocks 6,4,5), however, it is sensitive to individual errors. For example in CIFAR-10, ensemble prioritizes pruning block17 over block7 even when the latter has lower ranks in most of the criteria but the large ranking gap in one criterion, that is feature maps criterion, resulted in block17 to have lower rank.



(a) Weight norm

(b) Weight Taylor

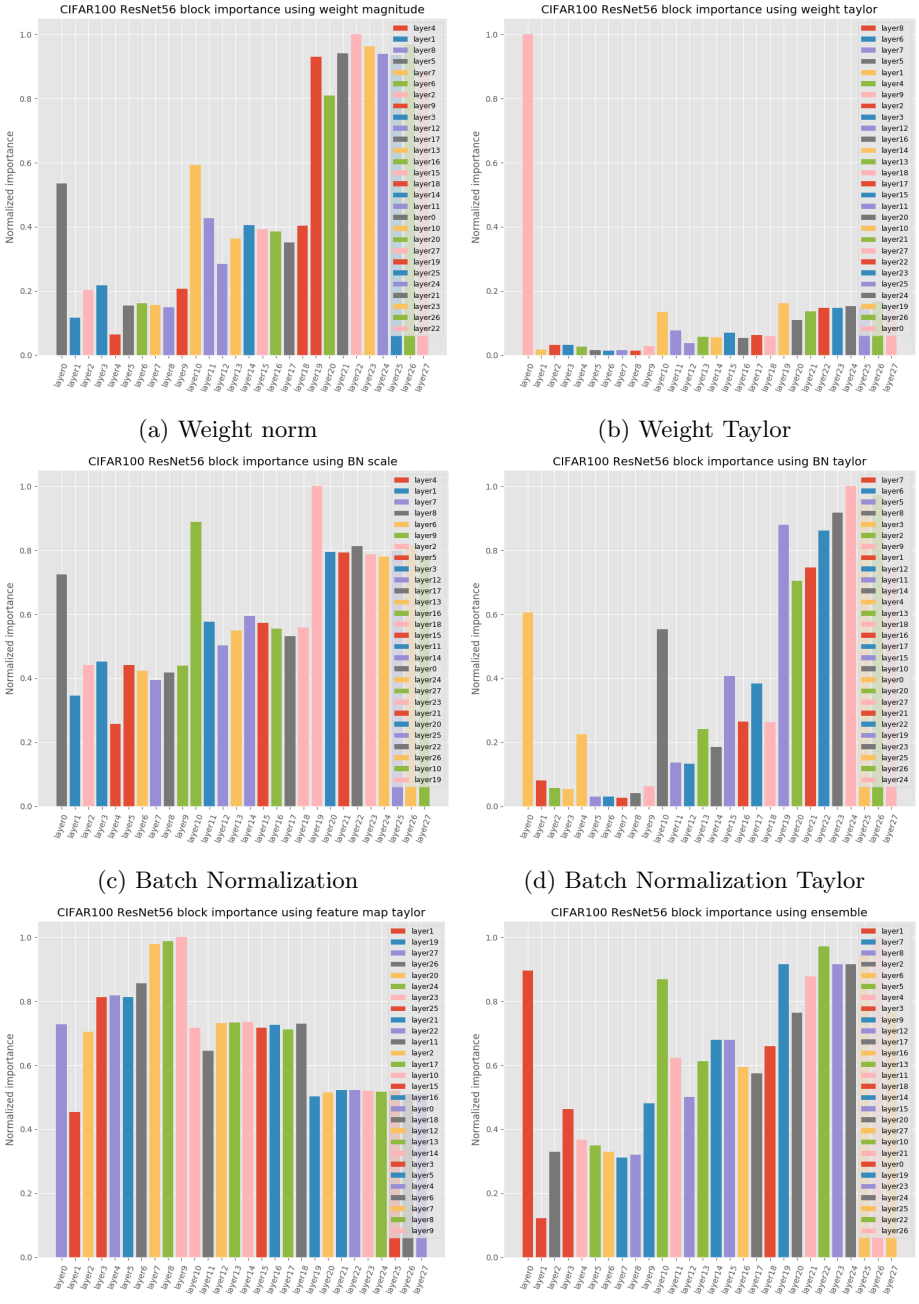
(c) Batch Normalization

(d) Batch Normalization Taylor

(e) Feature maps

(f) Ensemble

Fig. 2: Plots of block importance using different layer criterion on CIFAR-10 ResNet56. Legend on each sub-plot shows sorted blocks in ascending order based on importance.



(a) Weight norm (b) Weight Taylor

(c) Batch Normalization (d) Batch Normalization Taylor

(e) Feature maps (f) Ensemble

Fig. 3: Plots of block importance using different layer criterion on CIFAR-100 ResNet56. Legend on each sub-plot shows sorted blocks in ascending order based on importance.

2 ImageNet

2.1 Training setup

Filter pruning. We follow the same setup as Taylor [1] for global-based filter pruning, we prune 100 filters each 30 minibatches for 10 iterations. For methods like ECC [6], slimming [5], SSS [3] and HRank [9], we report using their default hyperparameter either reported in their papers or using their code. We fine-tune using the same setup as previously mentioned in CIFAR.

2.2 Ablation

Aggressive layer pruning. In this section we show results on small light-weight models with accuracy drop from baseline to show the effectiveness of LayerPrune under high pruning ratio. We compare pruned models from ResNet50 and ResNet34 with handcrafted small variants of ResNet. Results are presneted in **Table 4**. We achieve 1.44% better accuracy on similar latency as ResNet34 (74.74 vs 73.30). In addition, we match ResNet34 accuracy (73.39 vs 73.30) with 1.2x speedup. Similarly, we match ResNet18 latency with 0.5% higher accuracy.

Model	Accuracy	FPS (1080Ti)	FPS (Xavier)
ResNet50	76.14	129	62
LayerPrune$_6$-ResNet50	**74.74**	214	108
LayerPrune$_7$-ResNet50	74.31	239	114
LayerPrune$_8$-ResNet50	73.39	**248**	**122**
ResNet34	73.30	206	105
LayerPrune$_8$-ResNet34	**70.32**	364	168
LayerPrune$_9$-ResNet34	69.00	**405**	**181**
ResNet18	69.76	360	169

Table 4: Accuracy with small handcrafted ResNets on similar frames per second.

Training from scratch Traning from scratch for ImageNet is done for 90 epochs with 0.1 initial lr, 0.1 lr decay each 30 epochs. Fine-tuning is done for 30 epochs with 1e$-$3 initial lr, 0.1 lr decay each 10 epoch. In **Table 5**, we compare our LayerPrune models trained from scratch and fine-tuned. Fine-tuned models consistently outperform training from scratch of the same pruned architecture.

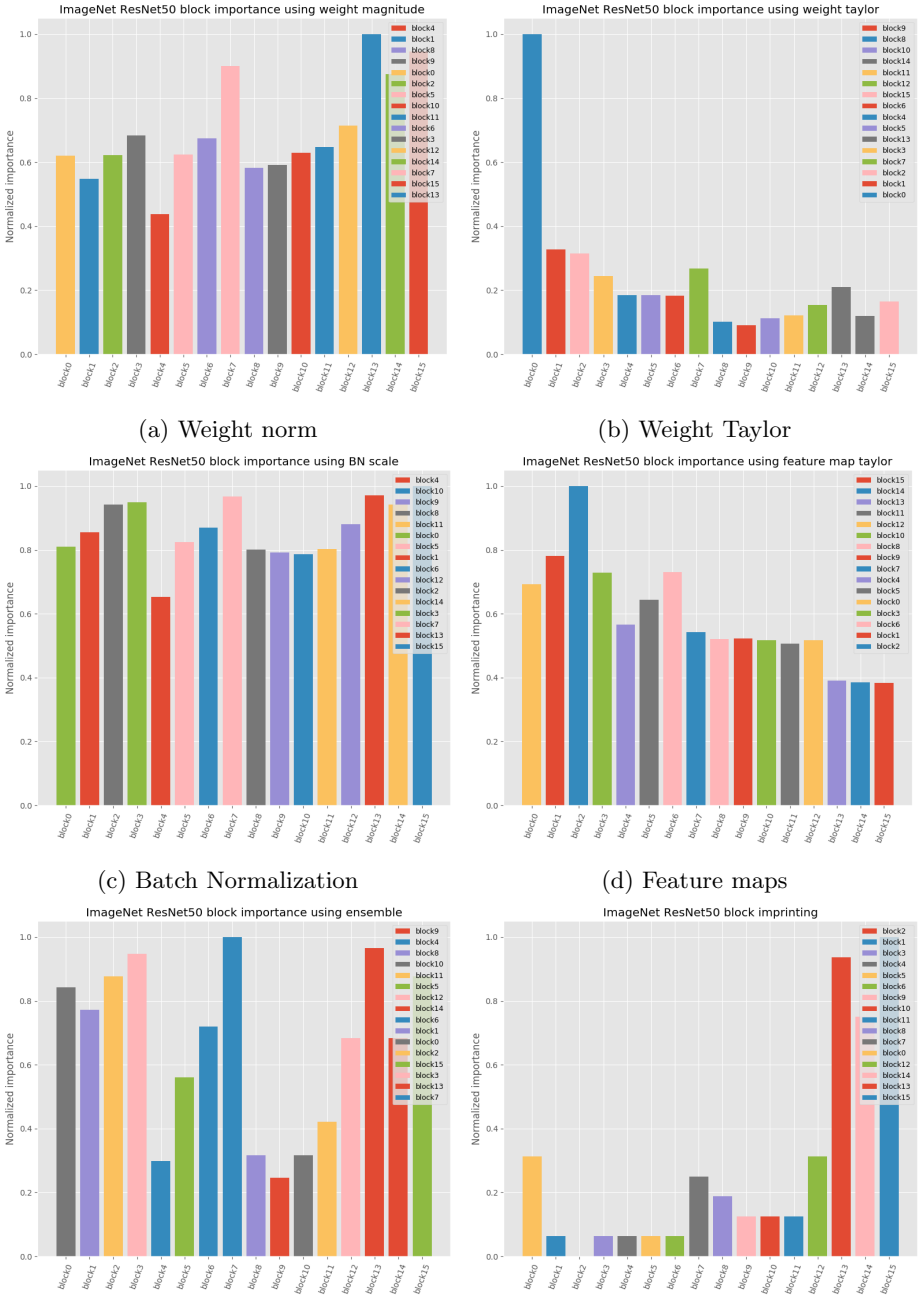
Training speed End-to-end optimization filter pruning methods such as slimming require training from scratch with sparsity inducing terms in the training. This requires 90 epochs in ImageNet. All methods, including ours, fine-tuned for 30 epochs. Hence, our layer-pruning is 4 times faster than these methods.
For iterative filter pruning methods we observed an average 1.9x speedup in the fine-tuning phase in layer-pruned models compared to fine-tuning phase in filter pruning. The training is conducted on 4 x V100 GPUs.

N Blocks pruned	Fine-tuned	Scratch
1	**76.72**	75.70
2	**76.53**	75.96
3	**76.40**	75.80
4	**75.82**	75.0

Table 5: Accuracy of our ResNet50 pruned models trained from scratch and fine-tuned.

2.3 Architectures

ResNet50 Details of block importance for each criterion is shown in **Fig. 4**. Unlike criteria that depend on filter statistics like weight values or gradients, imprinting asses quality of a block based on accuracy gained. This handles the network's distribution discrepancy in these statistics.



(a) Weight norm

(b) Weight Taylor

(c) Batch Normalization

(d) Feature maps

(e) Ensemble

(f) Imprinting

Fig. 4: Plots of block importance (normalized for visualization) using different layer criterion on ImageNet ResNet50. Legend on each sub-plot shows sorted blocks in ascending order based on importance.

References

1. Molchanov, P., Mallya, A., Tyree, S., Frosio, I., Kautz, J.: Importance estimation for neural network pruning. In: Proceedings of the IEEE CVPR. (2019) 11264–11272
2. Liu, Z., Sun, M., Zhou, T., Huang, G., Darrell, T.: Rethinking the value of network pruning. arXiv preprint arXiv:1810.05270 (2018)
3. Huang, Z., Wang, N.: Data-driven sparse structure selection for deep neural networks. In: Proceedings of the European conference on computer vision (ECCV). (2018) 304–320
4. Chen, S., Zhao, Q.: Shallowing deep networks: Layer-wise pruning based on feature representations. IEEE transactions on pattern analysis and machine intelligence (2018) 3048–3056
5. Liu, Z., Li, J., Shen, Z., Huang, G., Yan, S., Zhang, C.: Learning efficient convolutional networks through network slimming. In: Proceedings of the IEEE ICCV. (2017) 2736–2744
6. Yang, H., Zhu, Y., Liu, J.: Ecc: Platform-independent energy-constrained deep neural network compression via a bilinear regression model. In: Proceedings of the IEEE CVPR. (2019) 11206–11215
7. Han, S., Pool, J., Tran, J., Dally, W.: Learning both weights and connections for efficient neural network. In: Advances in neural information processing systems. (2015) 1135–1143
8. Molchanov, P., Tyree, S., Karras, T., Aila, T., Kautz, J.: Pruning convolutional neural networks for resource efficient transfer learning. arXiv preprint arXiv:1611.06440 **3** (2016)
9. Lin, M., Ji, R., Wang, Y., Zhang, Y., Zhang, B., Tian, Y., Shao, L.: Hrank: Filter pruning using high-rank feature map. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. (2020) 1529–1538