Group Fisher Pruning for Practical Network Compression (Appendix)

Liyang Liu [*1] Shilong Zhang [*2] Zhanghui Kuang [3] Aojun Zhou [3] Jing-Hao Xue [4]
Xinjiang Wang [3] Yimin Chen [3] Wenming Yang [1] Qingmin Liao [1] Wayne Zhang [2 3 5]

Abstract

In this appendix, first we provide the implementation details used to prune classification and detection networks, and conduct ablation studies on the hyper-parameters including pruning interval and prune/fine-tune learning rate. Then we give results of different network structures about the relationship between relative FLOPs/memory reduction and speedup, which is general and motivates us to normalize the channel importance by its memory reduction. Finally we present visualizations of pruned networks to show our method can automatically find the proper pruning ratio for each layer and prunes coupled channels in complicated structures simultaneously.

1. Implementation Details

Classification. To prune classification networks, we first follow the standard practices in (He et al., 2016; Xie et al., 2017; Sandler et al., 2018; Radosavovic et al., 2020) (such as data augmentation, input size, weight initialization, learning rate scheduling, weight decay, optimizer, momentum and training iterations) to train the dense model. Starting from it, we accumulate the Fisher information importance s_i for $d = 25$ iterations and prune the least important channel (with the smallest memory-normalized importance $s_i/\triangle M$). Then we set the accumulated importance for each channel to 0 and fine-tune the pruned model with lr $= 0.004$, meanwhile we accumulate the importance for another $d = 25$ steps. As the model has converged before pruning, we adopt a small learning rate to update the model weights after pruning each channel. Next the pruning and fine-tuning process recur. In the pruning procedure, we set the masks of the pruned channels to 0. After the FLOPs of the pruned model arrives at the desired quantity, we physically discard all channels with 0 masks and fine-tune the model once with lr $= 0.1$ and parameters inherited from the pruned model.

model	train (h)	prune (h)	fine-tune (h)
ResNet-50	28.2	15.1	16.2
RetinaNet	10.3	6.2	7.0

Table 1: Time for pruning networks to 50% FLOPs.

The time cost for training a dense ResNet-50, pruning it to 50% FLOPs remained and fine-tuning to regain accuracy is shown in Tab. 1. We find that the total time of pruning and fine-tuning is approximately equal to that of training a dense model, showing the efficiency of our method. During deployment, we measure the inference time of the pruned model on an NVIDIA 2080 Ti GPU with batchsize $= 64$ following (Radosavovic et al., 2020), the reported time is averaged across 100 batches with the first 100 batches discarded where the time is inaccurate.

Because of page limit, in the main submission results of comparing with SoTAs on pruning classification networks are concise. Here we present a detailed comparison with state-of-the-art channel pruning methods in Tab. 2. Under similar parameters and FLOPs, models pruned by our method has advantages over previous ones on both absolute accuracy and accuracy drop. Besides, our method achieves more speedup thanks to coupled pruning and memory normalization. For the reduction of parameters and FLOPs, we report the relative values for some methods to follow the original references. Only a few methods measure the actual inference time and speedup on GPUs which we report.

Detection. The dense models for detection are trained with the standard settings from MMDetection (Chen et al., 2019). The pruning interval $d = 10$ and fine-tuning lr $= 0.002$ during pruning. After the pruning stage we fine-tune the model with pruned channels discarded using lr $= 0.01$ to recover its accuracy. The time of each step to prune RetinaNet (Lin et al., 2017b) can also be found in Tab. 1, which shows our method is efficient. Similar to pruning classification networks, the inference time is measured on an NVIDIA 2080 Ti GPU across 100 batches, except that batchsize $= 24$ is

[*]Equal contribution [1]Shenzhen International Graduate School/Department of Electronic Engineering, Tsinghua University, Beijing, China [2]Shanghai AI Laboratory, Shanghai, China [3]SenseTime Research, Hong Kong, China [4]Department of Statistical Science, University College London, London, United Kingdom [5]Qing Yuan Research Institute, Shanghai, China. Correspondence to: Wenming Yang <yang.wenming@sz.tsinghua.edu.cn>.

Proceedings of the 38th International Conference on Machine Learning, PMLR 139, 2021. Copyright 2021 by the author(s).

	method	T1(%)	B1(%)	T5(%)	B5(%)	P(M)	F(G)	S(×)	△T1	△T5
Res50	ThiNet (Luo et al., 2017)	74.03	75.30	92.11	92.20	33.72%↓	36.79%↓	1.13	1.27	0.09
	SSS (Huang & Wang, 2018)	75.44	76.12	92.61	92.86	25.30	3.47	-	0.68	0.25
	IE (Molchanov et al., 2019)	76.43	76.18	-	-	22.60	3.27	-	-0.25	-
	Meta (Liu et al., 2019)	76.20	76.60	-	-	-	3.0	-	0.40	-
	GBN (You et al., 2019)	76.19	75.85	92.83	92.67	31.83%↓	40.54%↓	-	-0.34	-0.16
	Ours	76.95	76.79	93.51	93.40	23.82	3.06	1.30	-0.16	-0.11
	ThiNet (Luo et al., 2017)	72.03	75.30	90.99	92.20	51.56%↓	55.83%↓	1.27	3.27	1.21
	CP (He et al., 2017)	75.06	76.13	90.80	92.20	-	50.00%↓	-	1.07	1.4
	NISP (Yu et al., 2018)	0.89 ↓	-	-	-	43.82%↓	44.01%↓	-	0.89	-
	SFP (He et al., 2018a)	74.61	76.15	92.06	92.87	-	41.80%↓	1.43	1.54	0.81
	GDP (Lin et al., 2018)	72.61	75.13	91.05	92.30	-	2.24	1.24	2.52	1.25
	SSS (Huang & Wang, 2018)	71.82	76.12	90.79	92.86	15.60	2.33	-	4.30	2.07
	DCP (Zhuang et al., 2018)	74.95	76.01	92.32	92.93	-	55.76%↓	-	1.06	0.61
	AOFP (Ding et al., 2019b)	75.11	75.34	92.28	92.56	-	56.73%↓	-	0.23	0.28
	FPGM (He et al., 2019)	74.83	76.15	92.32	92.87	-	53.50%↓	1.62	1.32	0.55
	IE (Molchanov et al., 2019)	74.50	76.18	-	-	14.20	2.25	-	1.68	-
	C-SGD (Ding et al., 2019a)	74.54	75.33	92.09	92.56	-	46.24%↓	-	0.79	0.47
	Meta (Liu et al., 2019)	75.40	76.60	-	-	-	2.0	-	1.20	-
	GBN (You et al., 2019)	75.18	75.85	92.41	92.67	53.40%↓	55.06%↓	-	0.67	0.26
	LFPC (He et al., 2020)	74.46	76.15	92.04	92.87	-	60.80%↓	-	1.69	0.83
	HRank (Lin et al., 2020)	74.98	76.15	92.33	92.87	16.15	2.30	-	1.17	0.54
	Ours	76.42	76.79	93.07	93.40	19.42	2.04	1.79	0.37	0.33
	ThiNet (Luo et al., 2017)	68.17	75.30	88.86	92.20	66.12%↓	71.50%↓	1.35	7.13	3.34
	GDP (Lin et al., 2018)	70.93	75.13	90.14	92.30	-	1.57	-	4.20	2.16
	IE (Molchanov et al., 2019)	71.69	76.18	-	-	7.90	1.34	-	4.49	-
	Meta (Liu et al., 2019)	73.40	76.60	-	-	-	1.0	-	3.20	-
	CURL (Luo & Wu, 2020)	73.39	76.15	91.46	92.87	6.67	1.11	-	2.76	1.41
	Ours	73.94	76.79	91.71	93.40	12.36	1.02	2.94	2.85	1.69
Res101	ISTA (Ye et al., 2018)	75.27	76.40	-	-	23.6	4.47	-	1.13	-
	SFP (He et al., 2018a)	77.51	77.37	93.71	93.56	-	42.20%↓	-	-0.14	-0.15
	SSS (Huang & Wang, 2018)	75.44	76.40	-	-	-	3.47	-	0.96	-
	AOFP (Ding et al., 2019b)	76.40	76.63	93.07	93.29	-	50.19%↓	-	0.23	0.22
	FPGM (He et al., 2019)	77.32	77.37	93.56	93.56	-	42.20%↓	-	0.05	0
	IE (Molchanov et al., 2019)	77.35	77.37	-	-	31.2	4.70	-	0.02	-
	Ours	78.33	78.29	94.10	94.02	28.02	3.90	1.50	-0.04	-0.08
MBv2	AMC (He et al., 2018b)	70.80	71.80	-	-	-	0.22	-	1.00	-
	Meta (Liu et al., 2019)	72.70	74.70	-	-	-	0.29	-	2.00	-
	Ours	73.42	75.74	91.56	92.61	3.31	0.29	1.84	2.32	1.05
	Meta (Liu et al., 2019)	68.20	74.70	-	-	-	0.14	-	6.50	-
	Ours	69.16	75.74	89.06	92.61	1.81	0.15	1.79	6.58	3.55
	Meta (Liu et al., 2019)	65.00	74.70	-	-	-	0.11	-	9.70	-
	Ours	65.94	75.74	85.96	92.61	1.32	0.10	1.82	9.80	6.65
NeXt50	SSS (Huang & Wang, 2018)	74.98	77.57	92.50	93.68	10.70	2.43	-	2.59	1.18
	Ours	77.53	77.97	93.64	93.89	18.05	2.11	1.57	0.44	0.25

Table 2: Compare with SoTAs on ImageNet. The **column** "T1/T5" represents top-1/5 accuracy of the pruned model on the validation set where ↓ shows the accuracy drop compared with the unpruned model. "B1/B5" shows the top-1/5 accuracy of the unpruned base model. "P/F" shows the number of Params/FLOPs of the pruned model, where ↓ elements show the relative Params/FLOPs reduction compared with the unpruned model. "S" denotes the actual speedup of the pruned model on GPUs. "△T1/△T5" denotes the top-1/5 accuracy drop after pruning.

parameter	classification, Res50		detection, RetinaNet	
	value	Top1 (%)	value	mAP (%)
prune step d	5	76.36	5	36.5
	10	76.28	10	36.5
	25	76.42	30	36.5
	40	76.37	40	36.3
prune lr	0.002	76.32	0.0005	36.7
	0.004	76.42	0.001	36.6
	0.04	76.40	0.002	36.5
	0.08	76.59	0.005	36.5
fine-tune lr	0.01	75.86	0.001	35.3
	0.02	76.25	0.0025	35.6
	0.04	76.16	0.005	36.1
	0.08	76.29	0.0075	36.5
	0.1	76.42	0.01	36.5
	0.2	76.33	0.0125	36.5
	0.4	75.96	0.015	36.6

Table 3: Ablation studies on hyper-parameters when pruning networks to 50% FLOPs.

smaller (since input image size of detection is much larger than classification) and the reported time for detection has been divided by the batchsize.

Ablations. In Tab. 3 we provide ablation studies on pruning step d, pruning and fine-tuning learning rate for pruning ResNet-50 and RetinaNet to 50% FLOPs retained. As shown our method is rather robust against the hyper-parameters on both classification and detection.

2. Speedup with Different Normalizations

We find the actual speedup on GPUs is more linearly correlated with the reduction of memory than the reduction of FLOPs, here we show it is generally applicable to different network structures. Assume the FLOPs/memory/inference time of the dense model is $C/M/T$, and that of the pruned model in the pruning procedure is $C'/M'/T'$, we study the relationship between relative speedup $1 - \frac{T'}{T}$ and relative FLOPs/memory reduction $1 - \frac{C'}{C}/1 - \frac{M'}{M}$. In Fig. 1 the status of the pruned model during the pruning process are plotted. When we normalize importances by $\triangle C$ (reduction of FLOPs) of channels, the relative memory reduction is approximately linearly correlated with relative speedup, which grows much slower with the increase of relative FLOPs reduction. It motivates us to normalize importances by $\triangle M$ (reduction of memory). If we prune by the memory-normalized importance $s_i / \triangle M$, the relationship between speedup and memory reduction stays approximately linear, but FLOPs reduction becomes more proportional to relative speedup. It verifies the effectiveness of normalizing by memory, through it we can estimate the actual speedup with the reduction of FLOPs/memory more accurately than normalizing by FLOPs, and thus we are able to achieve higher speedup than previous methods under the same FLOPs.

3. Pruned Network Visualization

Classification. In Fig. 3 we compare the remained channels after pruning ResNet-50 to different FLOPs (4G/3G/2G/1G), using memory (left column) and FLOPs (right column) as importance normalizations, respectively (4G is FLOPs for the dense model). The pruned results of all layers are shown in the first row, and those of 1st/2nd/3rd convolutional layers in residual blocks are shown in the following rows for clarity. It shows that normalizing by memory prunes more channels in early stages, but normalizing by FLOPs prunes rather uniformly in different stages.

Detection. Remained channels in the pruned feature pyramid network (Lin et al., 2017a) of RetinaNet are shown in Fig. 2 (a). As expected, the coupled channels are pruned simultaneously and verify that our method can deal with sophisticated structures. We also compare the pruned channels in classification and regression heads of RetinaNet in Fig. 2 (b), where we find that the classification branch retains more capacity than regression. This is because region classification is a more difficult task than box regression since it needs to discriminate dozens of classes, while regressing bounding box requires less capacity as it can share a certain amount of knowledge across classes.

References

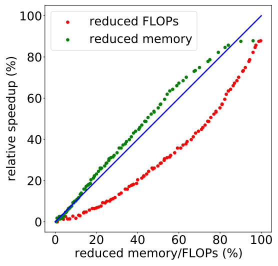
Chen, K., Wang, J., Pang, J., Cao, Y., Xiong, Y., Li, X., Sun, S., Feng, W., Liu, Z., Xu, J., Zhang, Z., Cheng, D., Zhu, C., Cheng, T., Zhao, Q., Li, B., Lu, X., Zhu, R., Wu, Y., Dai, J., Wang, J., Shi, J., Ouyang, W., Loy, C. C., and Lin, D. MMDetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*, 2019. 1

Ding, X., Ding, G., Guo, Y., and Han, J. Centripetal sgd for pruning very deep convolutional networks with complicated structure. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4943–4953, 2019a. 2

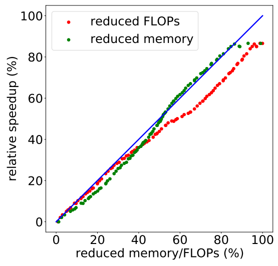
Ding, X., Ding, G., Guo, Y., Han, J., and Yan, C. Approximated oracle filter pruning for destructive cnn width optimization. In *International Conference on Machine Learning*, 2019b. 2

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016. 1

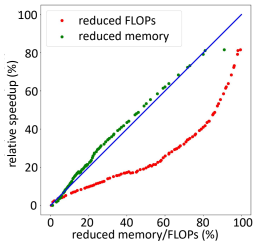
He, Y., Zhang, X., and Sun, J. Channel pruning for accelerating very deep neural networks. In *Proceedings of the*



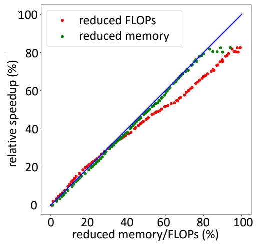
(a) ResNet-50, by FLOPs.



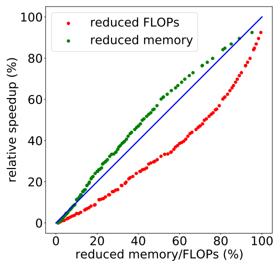
(b) ResNet-50, by memory.



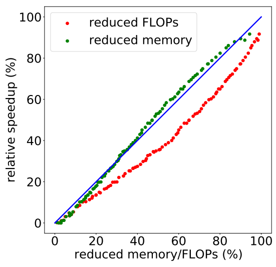
(c) MBv2-2.0×, by FLOPs.



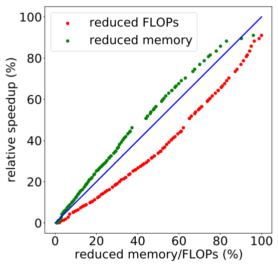
(d) MBv2-2.0×, by memory.



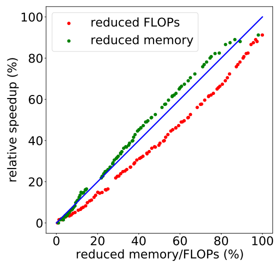
(e) ResNeXt-50, by FLOPs.



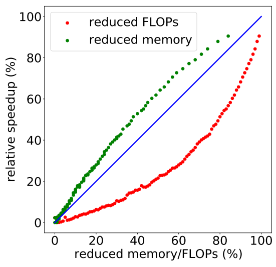
(f) ResNeXt-50, by memory.



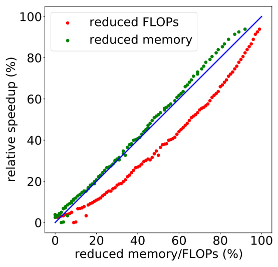
(g) NetX-3.2G, by FLOPs.



(h) NetX-3.2G, by memory.

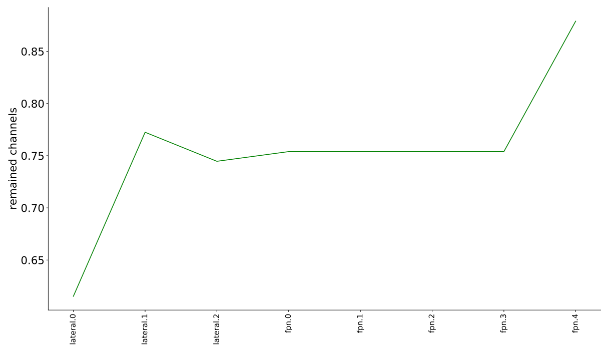


(i) RetinaNet, by FLOPs.

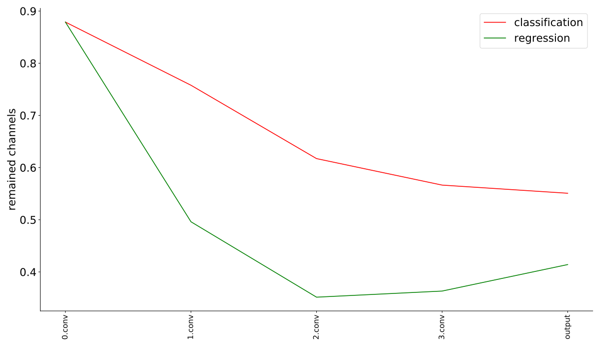


(j) RetinaNet, by memory.

Figure 1: The comparison between normalizing importances by FLOPs and memory under different network structures. States of the model (reduced FLOPs/memory and relative speedup) during the pruning process are shown. In each figure, two points on one horizontal line indicate the same model. NetX represents RegNetX.

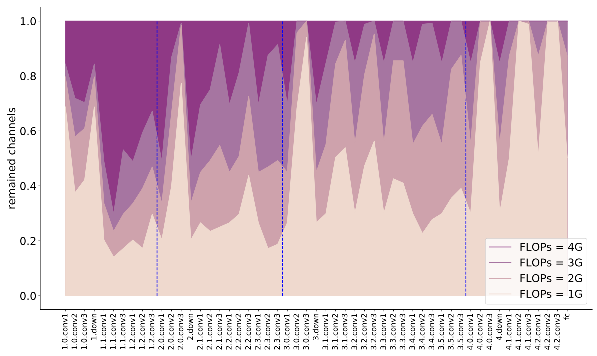


(a) Pruned feature pyramid network.

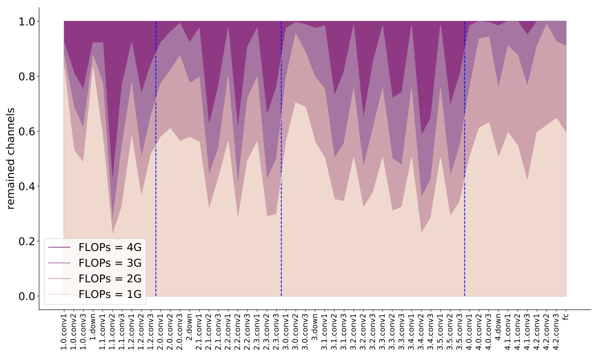


(b) Pruned classification and regression head.

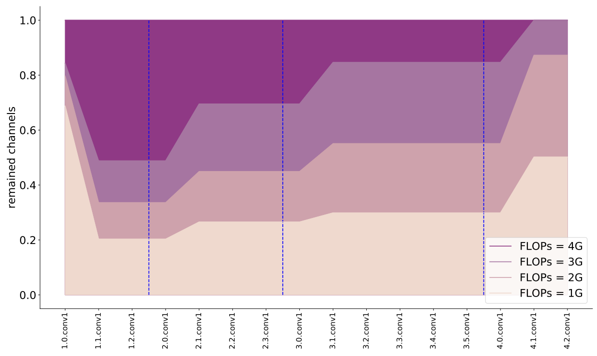
Figure 2: Visualization of channels (percentage) in pruned RetinaNet where 50% FLOPs are remained.



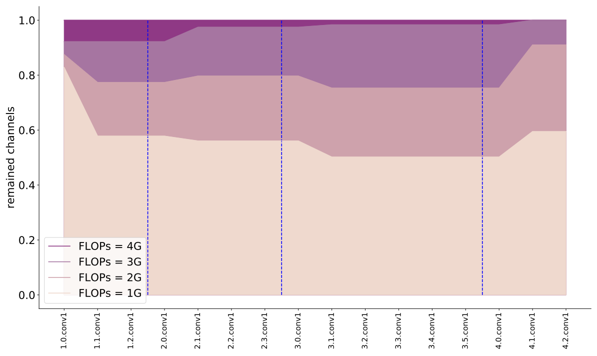
(a) All layers, normalize by memory.



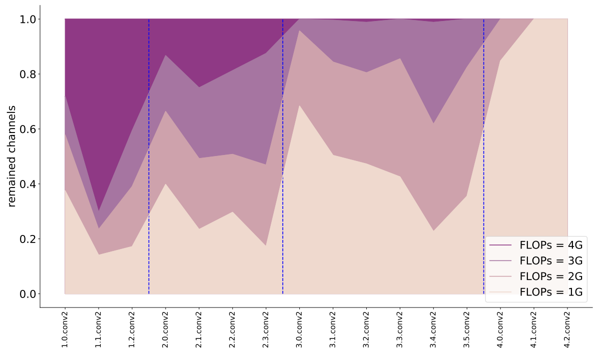
(b) All layers, normalize by FLOPs.



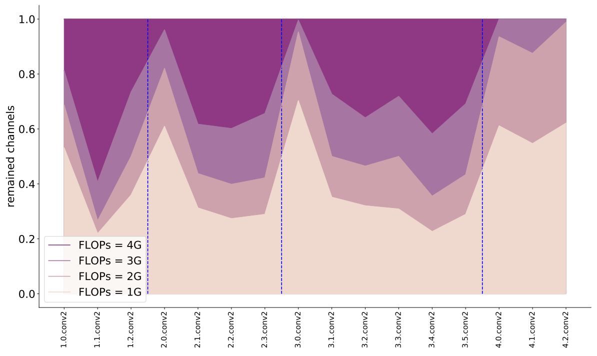
(c) 1st conv layers in bottleneck, normalize by memory.



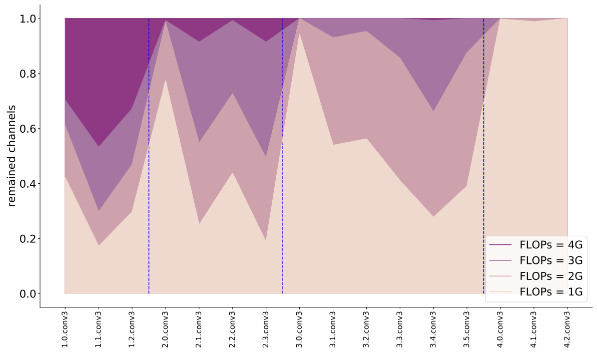
(d) 1st conv layers in bottleneck, normalize by FLOPs.



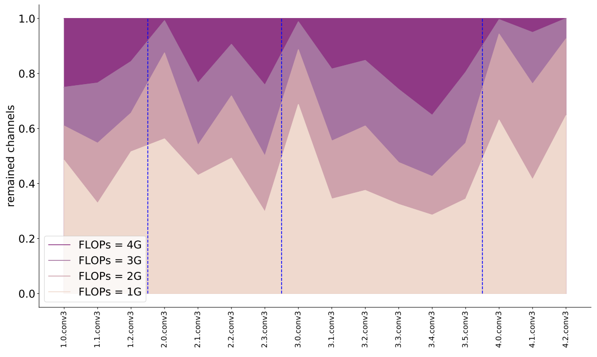
(e) 2nd conv layers in bottleneck, normalize by memory.



(f) 2nd conv layers in bottleneck, normalize by FLOPs.



(g) 3rd conv layers in bottleneck, normalize by memory.



(h) 3rd conv layers in bottleneck, normalize by FLOPs.

Figure 3: Remained input channels (percentage) in pruned ResNet-50 under different FLOPs budgets. The four different colors indicate various amount of FLOPs. The blue dashed lines show that the feature maps are 2× down-sampled.

IEEE International Conference on Computer Vision, pp. 1389–1397, 2017. 2

He, Y., Kang, G., Dong, X., Fu, Y., and Yang, Y. Soft filter pruning for accelerating deep convolutional neural networks. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 2234–2240, 2018a. 2

He, Y., Lin, J., Liu, Z., Wang, H., Li, L.-J., and 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, 2018b. 2

He, Y., Liu, P., Wang, Z., Hu, Z., and 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. 2

He, Y., Ding, Y., Liu, P., Zhu, L., Zhang, H., and Yang, Y. Learning filter pruning criteria for deep convolutional neural networks acceleration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2009–2018, 2020. 2

Huang, Z. and Wang, N. Data-driven sparse structure selection for deep neural networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 304–320, 2018. 2

Lin, M., Ji, R., Wang, Y., Zhang, Y., Zhang, B., Tian, Y., and Shao, L. Hrank: Filter pruning using high-rank feature map. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1529–1538, 2020. 2

Lin, S., Ji, R., Li, Y., Wu, Y., Huang, F., and Zhang, B. Accelerating convolutional networks via global & dynamic filter pruning. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 2425–2432, 2018. 2

Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B., and Belongie, S. Feature pyramid networks for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2117–2125, 2017a. 3

Lin, T.-Y., Goyal, P., Girshick, R., He, K., and Dollár, P. Focal loss for dense object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2980–2988, 2017b. 1

Liu, Z., Mu, H., Zhang, X., Guo, Z., Yang, X., Cheng, K.-T., and Sun, J. Metapruning: Meta learning for automatic neural network channel pruning. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3296–3305, 2019. 2

Luo, J.-H. and Wu, J. Neural network pruning with residual-connections and limited-data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1458–1467, 2020. 2

Luo, J.-H., Wu, J., and Lin, W. Thinet: A filter level pruning method for deep neural network compression. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 5058–5066, 2017. 2

Molchanov, P., Mallya, A., Tyree, S., Frosio, I., and Kautz, J. Importance estimation for neural network pruning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 11264–11272, 2019. 2

Radosavovic, I., Kosaraju, R. P., Girshick, R., He, K., and Dollár, P. Designing network design spaces. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10428–10436, 2020. 1

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., and 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. 1

Xie, S., Girshick, R., Dollár, P., Tu, Z., and He, K. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1492–1500, 2017. 1

Ye, J., Lu, X., Lin, Z., and Wang, J. Z. Rethinking the smaller-norm-less-informative assumption in channel pruning of convolution layers. In *International Conference on Learning Representations (ICLR)*, 2018. 2

You, Z., Yan, K., Ye, J., Ma, M., and 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. 2

Yu, R., Li, A., Chen, C.-F., Lai, J.-H., Morariu, V. I., Han, X., Gao, M., Lin, C.-Y., and Davis, L. S. Nisp: Pruning networks using neuron importance score propagation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 9194–9203, 2018. 2

Zhuang, Z., Tan, M., Zhuang, B., Liu, J., Guo, Y., Wu, Q., Huang, J., and Zhu, J. Discrimination-aware channel pruning for deep neural networks. In *Advances in Neural Information Processing Systems*, pp. 875–886, 2018. 2