Gate Decorator:
Global Filter Pruning Method for Accelerating Deep Convolutional Neural Networks
(NeurIPS 2019)

2020-04-13

hyounguk.shon@kaist.ac.kr

Approaches on CNN Acceleration

- Quantization
  - Reduced weight precision (Xnor-Net)
- Fast convolution
  - Factorized convolution methods (e.g., OctConv, HetConv)
- Low-rank approximation
  - Weight factorization methods (e.g., SVD)
- Filter pruning
  - Layer-by-layer pruning (e.g., ThiNet)
  - Global pruning (e.g., NISP, Gate Decorator)

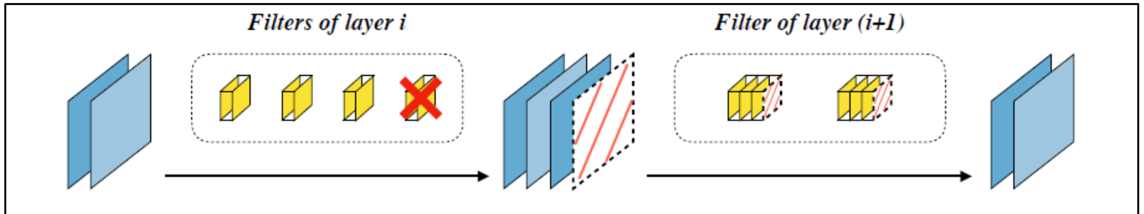


Figure 1: An illustration of filter pruning. The i-th layer has 4 filters (*i.e.* channels). If we remove one of the filters, the corresponding feature map will disappear, and the input of the filters in the $(i+1)$-th layer changes from 4 channels to 3 channels.

Method

Problem Definition

- Global Filter Importance Ranking (GFIR)
  - The key challenge for global pruning is solving the GFIR problem.
  - Our intent is to identify the least important filter k among the model's set of filters K.
- GFIR formulated as an optimization problem:

$$k^* = \arg\min_{k} \left| \mathcal{L}(X, Y; \theta) - \mathcal{L}(X, Y; \theta_k^+) \right| \quad s.t. \ ||k||_0 > 0$$

  - i.e., search for the kernel that brings minimum change in loss when after pruned.
  - Naïve search algorithm demands infeasible computational cost.

Overview of the paper

1. Gate Decorator
    - Multiply each feature map by a trainable scaling factor ϕ.
    - This enables an efficient importance metric for the search problem.
    - Gate decorators serve for the temporary purpose of pruning.
2. Tick-Tock pruning framework
    - An iterative pruning process to boost pruning accuracy.
3. Grouped pruning policy
    - Resolves pruning constraints caused by shortcut connections.

- A quick look on performance:
    - CIFAR-10 / ResNet-56: 70% FLOPs reduction without noticeable loss in accuracy. (SOTA)
    - ImageNet / ResNet-50: 40% FLOPs reduction while *increasing* top-1 accuracy.

Gate Decorator – (I)

- Gate Decorator
  1. We reparametrize a filter by introducing a gate variable ϕ.
     As such, pruning a filter is equivalent to setting ϕ to zero.

     $\phi \in \mathbb{R}$ and use $\hat{z} = \phi z$

     $$\Delta \mathcal{L}_\Omega(\phi) = |\mathcal{L}_\Omega(\phi) - \mathcal{L}_\Omega(0)|$$

  2. Assuming $|\phi| \ll 1$, we approximate $\mathcal{L}_\Omega(\phi)$ by Taylor expansion.
     This approximation allows it to exploit backpropagation.

     $$\begin{aligned} \mathcal{L}_\Omega(0) &= \sum_{p=0}^{P} \frac{\mathcal{L}_\Omega^{(p)}(\phi)}{p!}(0-\phi)^p + R_P(\phi) \\ &= \mathcal{L}_\Omega(\phi) - \phi \nabla_\phi \mathcal{L}_\Omega + R_1(\phi) \end{aligned}$$

     $$\Delta \mathcal{L}_\Omega(\phi) = |\phi \nabla_\phi \mathcal{L}_\Omega - R_1(\phi)| \approx |\phi \nabla_\phi \mathcal{L}_\Omega| = \left| \frac{\delta \mathcal{L}}{\delta \phi} \phi \right|$$

  3. We evaluate important scores on the filters over the dataset D.

     $$\Theta(\phi_i) = \sum_{(X,Y)\in\mathcal{D}} \left| \frac{\delta \mathcal{L}(X,Y;\theta)}{\delta \phi_i} \phi_i \right|$$

  - Before pruning, we reparametrize convolution/batchnorm.
    After pruning we turn them back to vanilla layers.

Gate Decorator – (II)

- Gated Convolution (GC)
  1. Initialize ϕ, W.
  2. Reparametrize the convolution layer.
  3. Merge ϕ into weights after pruning.

$$\phi := \frac{\|W\|_F}{ck^2} \qquad Y = \phi(X \otimes W) \qquad W := \phi W$$
$$W := \frac{W}{\phi}$$

- Gated Batch Normalization (GBN)
  - If a conv layer is followed by a BN, gate decorator is applied to BN instead.
  1. Initialize $\vec{\phi}, \beta, \gamma$.
  2. Reparametrize the BN layer.
  3. Merge $\vec{\phi}$ into weights after pruning.

$$\vec{\phi} := \gamma$$
$$\beta := \frac{\beta}{\gamma} \qquad \hat{z} = \frac{z_{in} - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}}; \quad z_{out} = \vec{\phi}(\gamma\hat{z} + \beta),\ \vec{\phi} \in \mathbb{R}^c \qquad \gamma := \vec{\phi}$$
$$\gamma := 1 \qquad \beta := \beta\vec{\phi}$$

Tick-Tock pruning framework

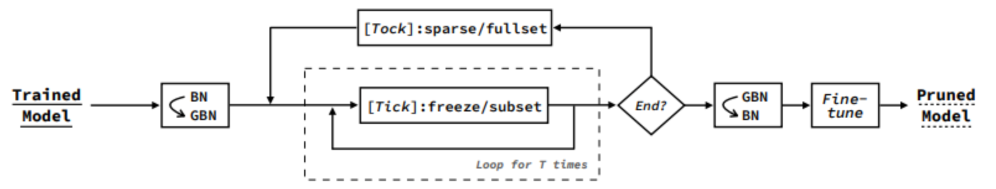


Figure 2: An illustration of the Tick-Tock pruning framework. The Tick phase is executed on a subset of the training data, and the convolution kernels are set to non-updatable. The Tock uses the full training data and adds the sparse constraint of ϕ to the loss function.

- "Tick" phase
  - Calculates importance scores, prunes filters, fixes the internal covariate shift.
  1. Train the network on a small subset of the dataset for just one epoch.
     We update only ϕ and the output layer to avoid overfitting.
  2. Compute importance score Θ, and remove a portion of the least important filters.
  3. Repeat the process for T times.
- "Tock" phase
  - Fine-tune the network to reduce the accumulation of errors caused by pruning.
  - Sparsity constraint on ϕ is included to training objective.

$$\mathcal{L}_{tock} = \mathcal{L} + \lambda \sum_{\phi \in \Phi} |\phi|$$

- Fine-tuning to the training set.
  - Fine-tuning trains more epochs than a Tock step.
  - Fine-tune does not include the sparsity constraint to the loss function.

Group Pruning policy

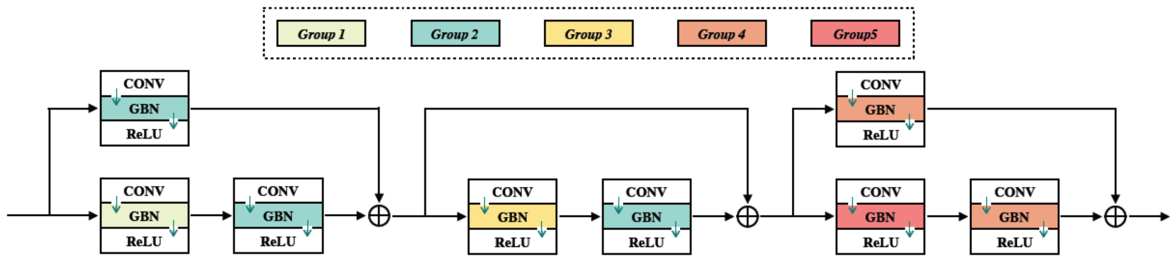


Figure 3: An illustration of Group Pruning. GBNs with the same color belong to the same group.

- The constrained pruning problem
  - Pruning filters in a residual block may result in misaligned feature maps.
  - Earlier approaches include:
    - bypassing such layers and only prune internal layers of residual blocks. (limits pruning ratio)
    - inserting a sampler before the first conv layer in each res block and leave the last conv layer unpruned. (add new structures to the network)
- Grouped Pruning policy
  - The idea is to prune a group of constrained filters at a time.
  - Importance score of a group is the sum of individuals.

$$\Theta(\phi_j^G) = \sum_{g \in G} \Theta(\phi_j^g)$$

Experiments

Implementation Details

- Datasets
  - Classification: CIFAR-10, CIFAR-100, CUB-200, ILSVRC-12
  - Semantic Segmentation: PASCAL VOC 2011 + SBD
- Baseline Models
  - Classification: VGGNet, ResNet
  - Semantic Segmentation: FCN
- Tick-Tock settings
  - Pruning ratio per Tick step (T=10)
    - ResNet: prune 0.2%
    - VGG: prune 1% per Tick step
  - Subset of training set for Tick
    - ImageNet: 100 images per class
    - CIFAR and CUB: the whole training set

Overall Comparisons – (I)

- ResNet-56 on the CIFAR-10
  - Achieved SOTA pruning ratio without noticeable loss in accuracy.

Metric	Li et al. [25]	NISP [52]	DCP-A [56]	CP [15]	AMC [14]	C-SGD [6]	**GBN-40**	**GBN-30**
FLOPs ↓%	27.6	43.6	47.1	50.0	50.0	60.8	**60.1**	**70.3**
Params ↓%	13.7	42.6	70.3	-	-	-	**53.5**	**66.7**
Accuracy ↓%	-0.02	0.03	-0.01	1.00	0.90	-0.23	**-0.33**	**0.03**

Table 1: The pruning results of ResNet-56 [11] on CIFAR-10 [20]. The baseline accuracy is 93.1%.

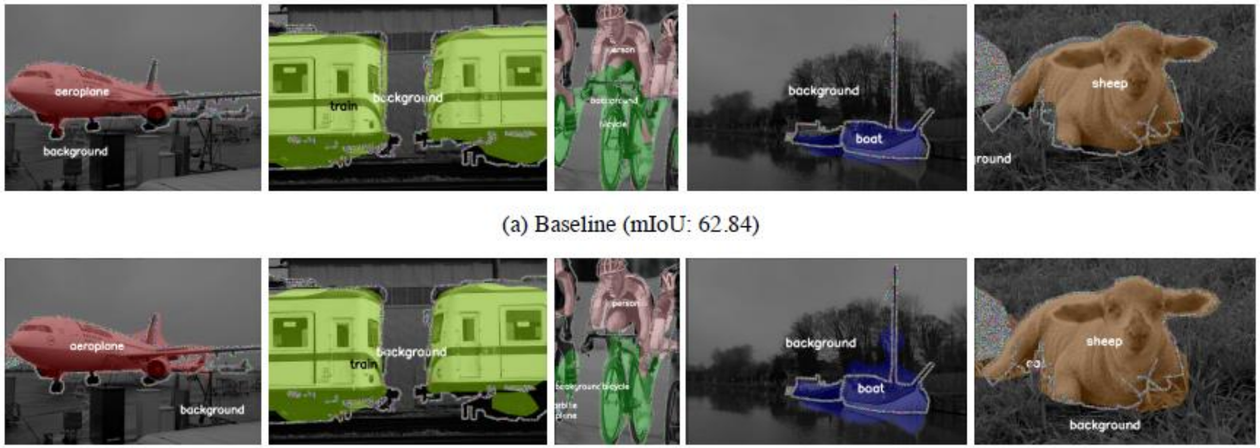
- ResNet-50 on the ILSVRC-12

Table 2: The pruning results of ResNet-50 [11] on the ImageNet [4] dataset. "P.Top-1" and "P.Top-5" denotes the top-1 and top-5 single center crop accuracy of the pruned model on the validation set. "[Top-1] ↓" and "[Top-5] ↓" denotes the decrease in accuracy of the pruned model compared to its unpruned baseline. "Global" identifies whether the method is a global filter pruning algorithm.

Method	Global	P. Top-1	[Top-1] ↓	P. Top-5	[Top-5] ↓	FLOPs ↓%	Param ↓%
ThiNet-70 [30]	✗	72.04	0.84	90.67	0.47	36.75	33.72
SFP [12]	✗	74.61	1.54	92.06	0.81	41.80	-
GBN-60	✓	**76.19**	**-0.31**	**92.83**	**-0.16**	**40.54**	**31.83**
NISP [52]	✓	-	0.89	-	-	44.01	43.82
FPGM [13]	✗	74.83	1.32	92.32	0.55	53.50	-
ThiNet-50 [30]	✗	71.01	1.87	90.02	1.12	55.76	51.56
DCP [56]	✗	74.95	1.06	92.32	0.61	55.76	51.45
GDP [26]	✓	71.89	3.24	90.71	1.59	51.30	-
GBN-50	✓	**75.18**	**0.67**	**92.41**	**0.26**	**55.06**	**53.40**

Overall Comparisons – (II)

- FCN-32s on the PASCAL VOC 2011
    - 27% FLOPs reduction, 73% parameters reduction while maintaining mIOU.



(a) Baseline (mIoU: 62.84)

(b) Pruned (mIoU: 62.88)

Figure 6: The segmentation instances.

More Explorations – (I)

- Effectiveness on GFIR against other global pruning methods:
  - Slim method ranks filters by the magnitude of its factors, which is insufficient according to our analysis.
  - GBN outperforms PCNN by a large margin.

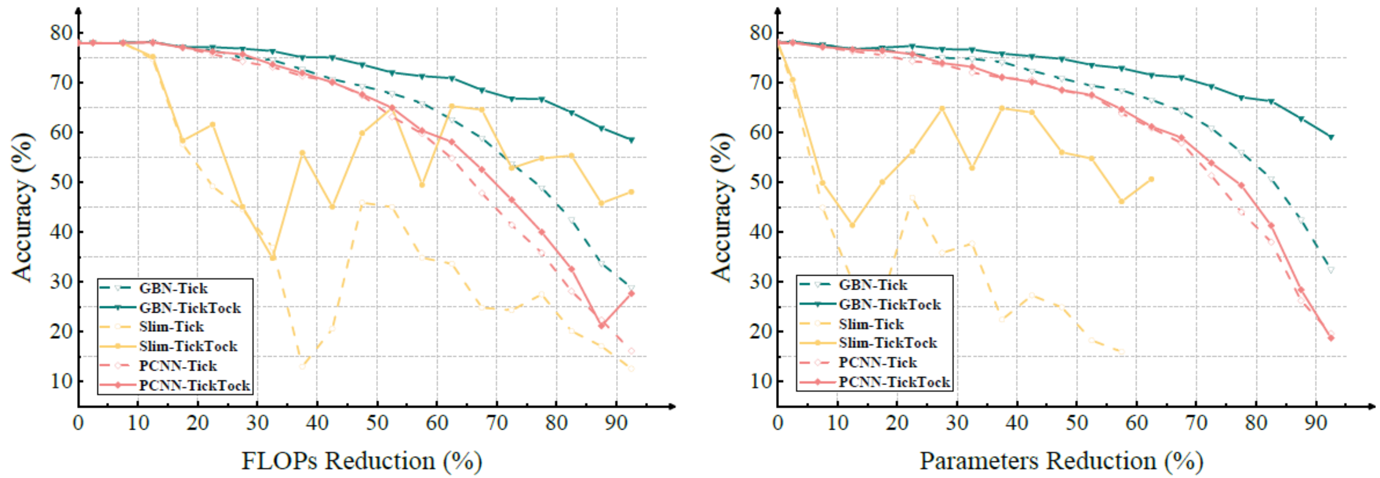


Figure 4: The pruning results of VGG-16-M [22] on the CUB-200 [45] dataset. The reported results are the model test accuracy before the Fine-tune phase. Slim [28] and PCNN [33] are compared.

More Explorations – (II)

- Global filters pruning viewed as a NAS process:
  - Baseline model: VGG-16-M, 73.19% test acc on CIFAR-10.
  - Shrunk network: ¼ FLOPs, ½ parameter counts, 1.98% acc drop.
  - Pruned network: ¼ FLOPs, ~1/3 parameter counts, 1.30% acc drop.
  - Redundancy in the deep layer is unnecessary.
  - The middle layers seem to be more important.
  - This demonstrates that the pruning method can be viewed as a task-driven NAS algorithm.

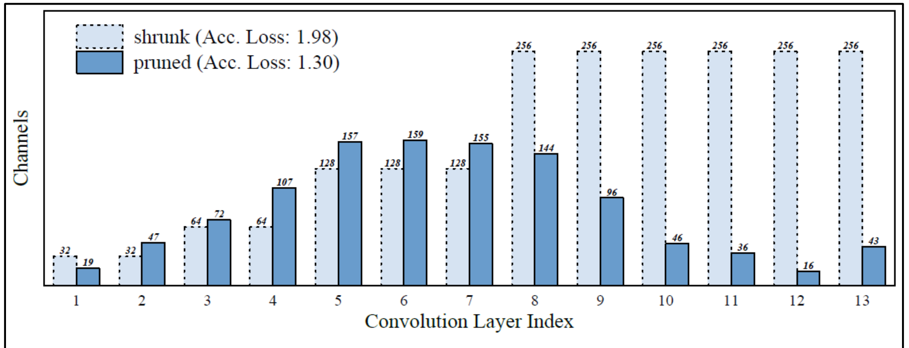


Figure 5: An illustration of two network architectures with the same FLOPs.

More Explorations – (III)

- Effectiveness of the Tick-Tock framework:
  - Iterative pruning processes preserve better accuracy than One-shot pruning.
  - Tick-only and Tick-Tock preserves similar network structure.

FLOPs-Pruned	GBN with One-Shot			GBN with Tick-Only			GBN with Tick-Tock		
	Param	Finetune	Scratch	Param	Finetune	Scratch	Param	Finetune	Scratch
40%	79.3%	71.8	73.1	68.5%	73.0	73.7	69.0%	**74.6**	73.7
60%	92.0%	62.1	68.0	86.2%	71.4	72.9	85.5%	**73.2**	73.0
80%	97.5%	57.7	59.9	95.0%	68.4	69.6	94.7%	**71.2**	69.9

Table 3: The test results of VGG-16-M [22] model on the CIFAR-100 [20] dataset under different pruning schemas. The accuracy of unpruned baseline model is 73.2%. "Param" denotes the percentage of parameters that been removed. "Finetune" represents the test accuracy of the pruned model after fine-tuning. "Scratch" shows the test result of the random initialized model, which has the same architecture as the pruned one. When training the "Scratch" model, we doubled the epochs to 320.

Conclusion

- This paper proposes three components for global filter pruning:
  - The Gate Decorator algorithm to resolve the GFIR problem.
  - The Tick-Tock pruning process to boost pruning accuracy.
  - The Grouped Pruning method to resolve the constrained pruning problem.
- This paper demonstrates:
  - The global filter pruning can be viewed as a task-driven NAS algorithm.
  - Experiments show that the proposed method outperforms several SOTA pruning methods.

References

- Zhonghui You, Kun Yan, Jinmian Ye, Meng Ma, Ping Wang. Gate Decorator: Global Filter Pruning Method for Accelerating Deep Convolutional Neural Networks. *Proceedings of Advances in Neural Information Processing Systems*, 2019
- Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. Pruning convolutional neural networks for resource efficient transfer learning. ICLR, 2017.
- Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learning efficient convolutional networks through network slimming. ICCV, 2017.
- https://medium.com/@nainaakash012/gate-decorator-global-filter-pruning-afc12fcc71c6