ThiNet: A Filter Level Pruning Method for Deep Neural Network Compression

Jian-Hao Luo[1], Jianxin Wu[1], and Weiyao Lin[2]

[1]National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China

[2]Shanghai Jiao Tong University, Shanghai, China

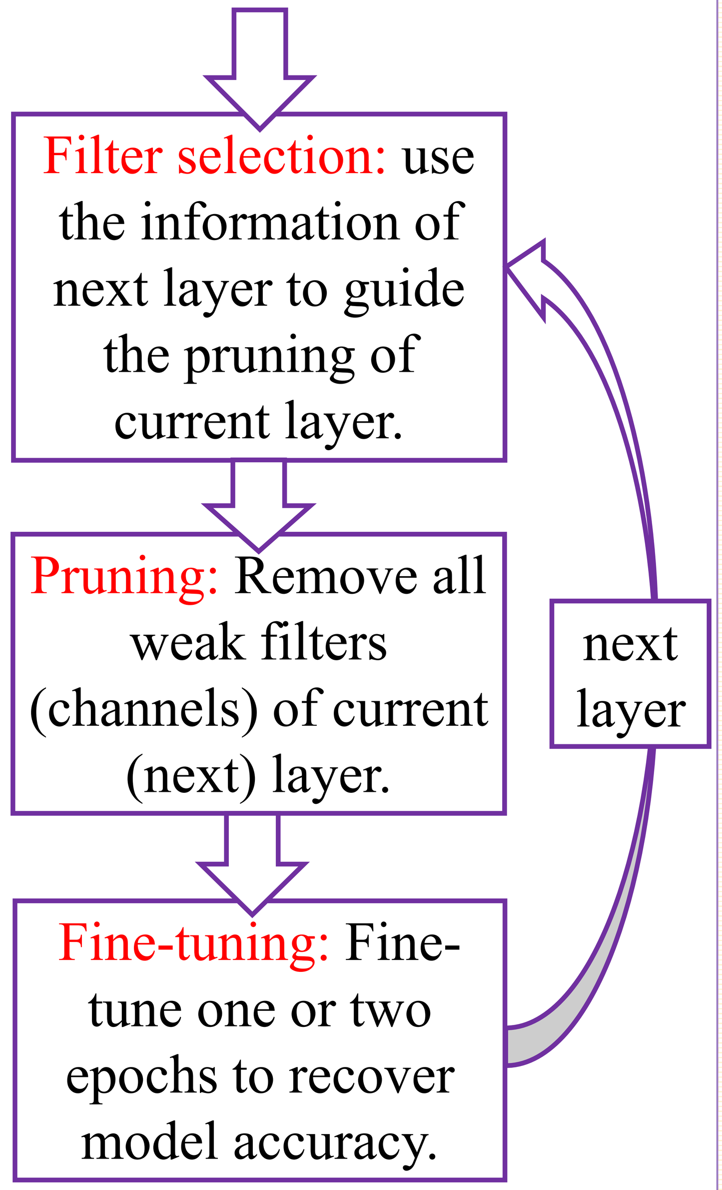
1. Introduction

Advantages of ThiNet

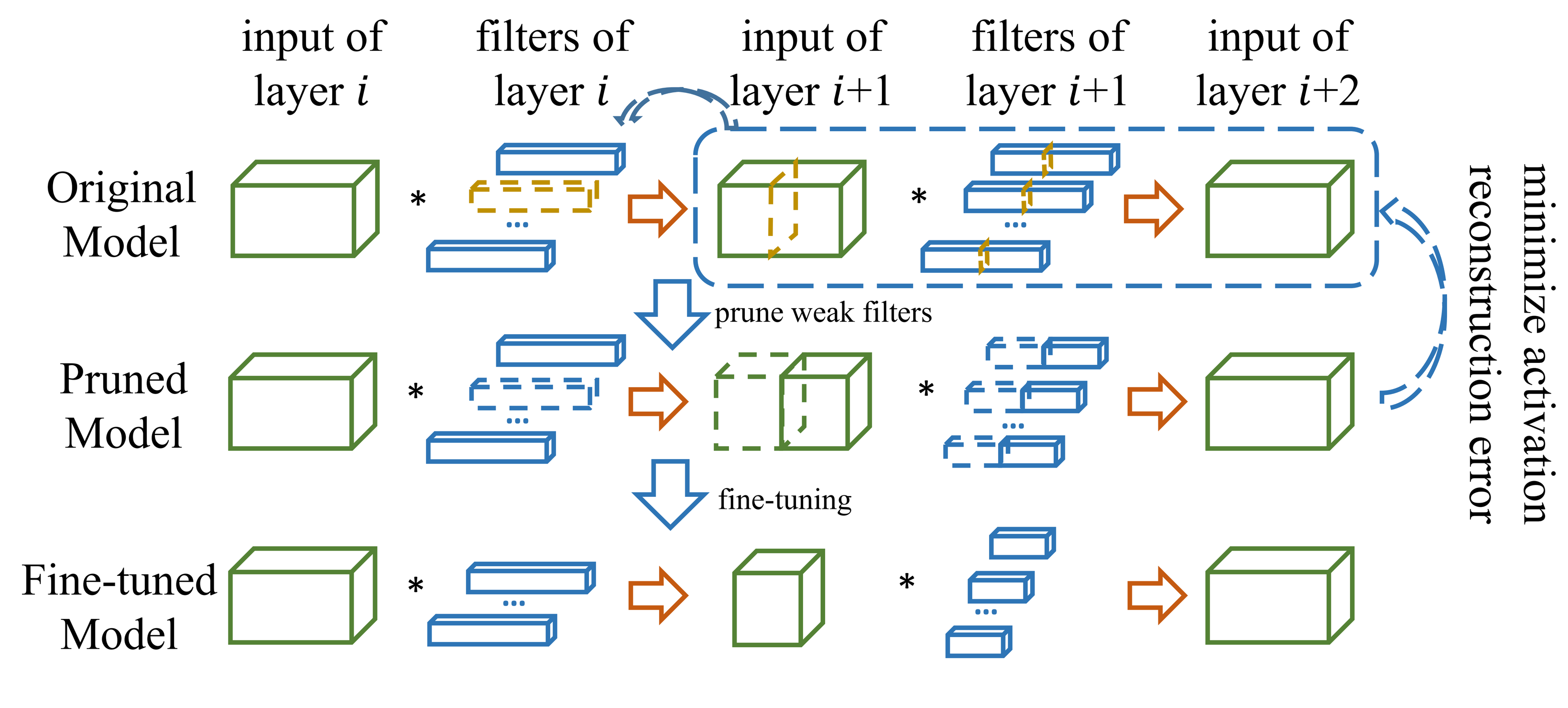
- ✓We focus on filter level pruning, which do not change structure;
- ✓We achieve acceleration and compression simultaneously;
- ✓Pruned model can be further compressed via other methods;

Main idea

- ✓We formally establish filter pruning as an optimization problem.
- ✓Selection criterion: minimize the reconstruction error.
- ✓Selection + Reconstruction

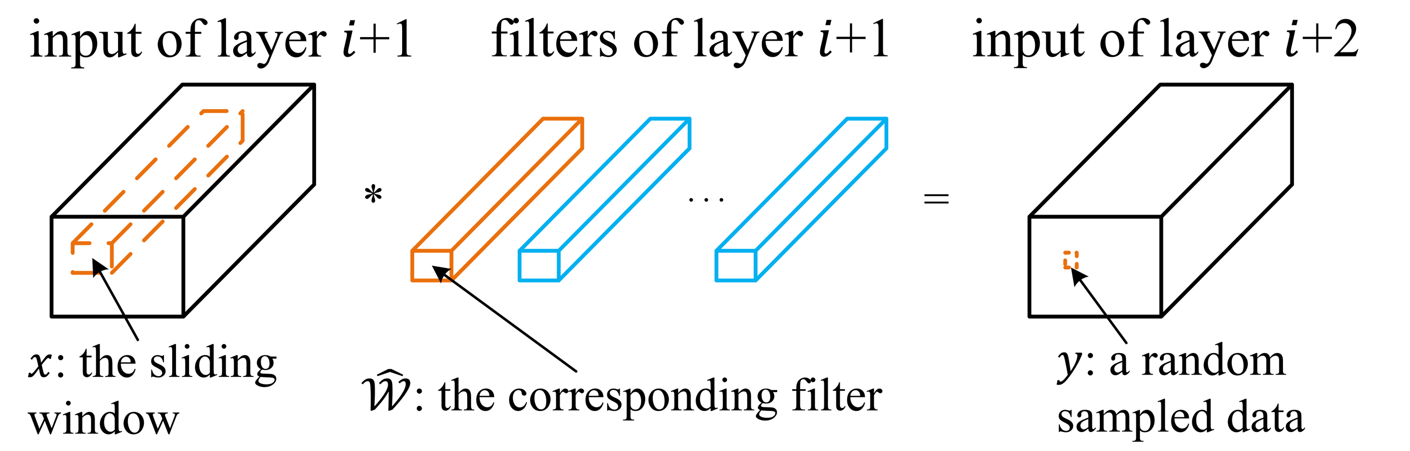


2. ThiNet Framework



- ➢ We focus on the next layer (dotted box part) to determine which filters can be pruned (highlighted in yellow).
- ➢ These weak filters and channels could be removed safely.
- ➢ We then fine-tune the pruned model and move to next layer.

3. Selection + Reconstruction



convolution: $y = \sum_{c=1}^{C}\sum_{k_1=1}^{K}\sum_{k_2=1}^{K} \widehat{\mathcal{W}}_{c,k_1,k_2} \times x_{c,k_1,k_2} + b \Rightarrow \hat{y} = \sum_{c=1}^{C} \hat{x}_c$

define: $\hat{x}_c = \sum_{k_1=1}^{K}\sum_{k_2=1}^{K} \widehat{\mathcal{W}}_{c,k_1,k_2} \times x_{c,k_1,k_2}$ + $\hat{y} = y - b$

Goal: find a subset $S \subset \{1,2,\dots,C\}$ making the equality hold.

3. Selection + Reconstruction

Given m training examples $\{(\hat{\mathrm{x}}_i, \hat{y}_i)\}$, the original channel selection problem becomes the following optimization problem:

$$\arg\min_S \sum_{i=1}^{m}\Big(\hat{y}_i - \sum_{j\in S}\hat{\mathrm{x}}_{i,j}\Big)^2 \quad \text{s.t. } |S| = C \times r, S \subset \{1,2,\dots,C\}$$

let $S \cup T = \{1,2,\dots,C\}$ and $S \cap T = \emptyset$ $\Rightarrow$

$$\arg\min_T \sum_{i=1}^{m}\Big(\sum_{j\in T}\hat{\mathrm{x}}_{i,j}\Big)^2 \quad \text{s.t. } |T| = C \times (1-r), T \subset \{1,2,\dots,C\}$$

We propose a greedy algorithm to solve the optimization problem:

- Initialize the removed channel set $\mathrm{T} = \emptyset$;
- At each time, we calculate the objective value if we add current channel into T;
- We choose the channel which has smallest objective value, and add it into T;
- Move to next iteration if $|T| < C \times (1-r)$;

Final Step: After selection, we further minimize the reconstruction error:

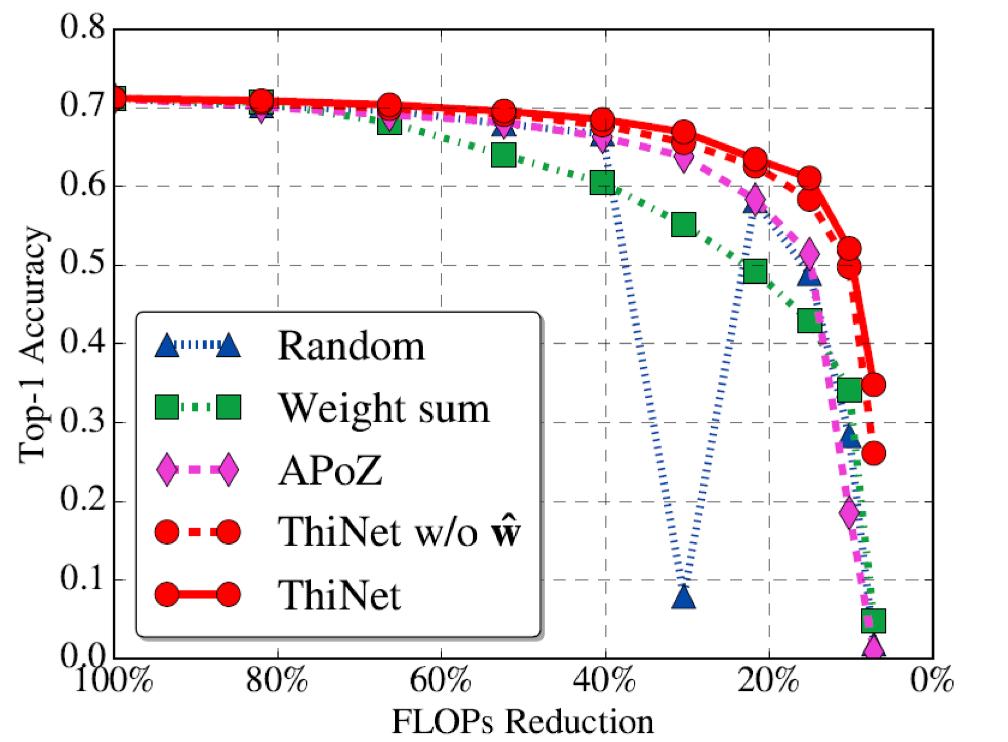
$$\hat{\mathrm{w}} = \arg\min_{\mathrm{w}} \sum_{i=1}^{m}(\hat{y}_i - \mathrm{w}^{\mathrm{T}}\hat{\mathrm{x}}_i^*)^2$$

which can be solved using least square. Finally, we use $\hat{\mathrm{w}}$ to rescale the selected filter weight:

$$\mathcal{W}_{:,i,:,:} = \hat{\mathrm{w}}_i \mathcal{W}_{:,i,:,:}$$

4. Experiments

4.1 Compared on CUB200



- ➢Weight sum [1]: $S_i = \sum|\mathcal{W}(i,:,:,:)|$;
- ➢APoZ [2]: $\mathrm{S_i} = \frac{1}{|\mathrm{I(i,:,:,:)}|}\sum\sum \mathbb{I}(|\mathrm{I}(i,:,:,:) == 0|)$;
- ➢ThiNet w/o $\hat{\mathrm{w}}$: ThiNet without least square;
- ➢ThiNet: our channel selection method.

[1] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf. Pruning filters for efficient ConvNets. In ICLR, pages 1–13, 2017.

[2] H. Hu, R. Peng, Y.W. Tai, and C. K. Tang. Network trimming: A data-driven neuron pruning approach towards efficient deep architectures. In arXiv preprint arXiv:1607.03250, pages 1–9, 2016.

4.2 Prune VGG16 on ImageNet

Model	Top-1	Top-5	#Param.	#FLOPs[1]	f./b. (ms)
Original[2]	68.34%	88.44%	138.34M	30.94B	189.92/407.56
ThiNet-Conv	69.80%	89.53%	131.44M	9.58B	76.71/152.05
Train from scratch	67.00%	87.45%	131.44M	9.58B	76.71/152.05
ThiNet-GAP	67.34%	87.92%	8.32M	9.34B	71.73/145.51
ThiNet-Tiny	59.34%	81.97%	1.32M	2.01B	29.51/55.83
SqueezeNet[15]	57.67%	80.39%	1.24M	1.72B	37.30/68.62

[1] In this paper, we only consider the FLOPs of convolution operations, which is commonly used for computation complexity comparison.

[2] For a fair comparison, the accuracy of original VGG-16 model is evaluated on resized center-cropped images using pre-trained model as adopted in [10, 14]. The same strategy is also used in ResNet-50.

- ➢ Pruning is much better than training from scratch;
- ➢ VGG-16 model can be further pruned into a very small model with only 5.05MB model size, preserving AlexNet level accuracy;

Method	Top-1 Acc.	Top-5 Acc.	#Param. ↓	#FLOPs ↓
APoZ-1 [14]	-2.16%	-0.84%	2.04×	≈ 1×
APoZ-2 [14]	+1.81%	+1.25%	2.70×	≈ 1×
Taylor-1 [23]	–	-1.44%	≈ 1×	2.68×
Taylor-2 [23]	–	-3.94%	≈ 1×	3.86×
ThiNet-WS [21]	+1.01%	+0.69%	1.05×	3.23×
ThiNet-Conv	+1.46%	+1.09%	1.05×	3.23×
ThiNet-GAP	-1.00%	-0.52%	16.63×	3.31×

4.3 Transfer Learning

Dataset	Strategy	#Param.	#FLOPs	Top-1
CUB-200	VGG-16	135.07M	30.93B	72.30%
	FT & prune	7.91M	9.34B	66.90%
	Train from scratch	7.91M	9.34B	44.27%
	ThiNet-Conv	128.16M	9.58B	70.90%
	ThiNet-GAP	7.91M	9.34B	69.43%
	ThiNet-Tiny	1.12M	2.01B	65.45%
	AlexNet	57.68M	1.44B	57.28%
Indoor-67	VGG-16	134.52M	30.93B	72.46%
	FT & prune	7.84M	9.34B	64.70%
	Train from scratch	7.84M	9.34B	38.81%
	ThiNet-Conv	127.62M	9.57B	72.31%
	ThiNet-GAP	7.84M	9.34B	70.22%
	ThiNet-Tiny	1.08M	2.01B	62.84%
	AlexNet	57.68M	1.44B	59.55%

About Me

Project Page

LAMDA Learning And Mining from DatA