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

Jian-Hao Luo, Jianxin Wu, Weiyao Lin

Presented by Zhuangwei Zhuang

Southern Artificial Intelligence Laboratory

Aug 28, 2017

Content

Background

Background

Deep Neural Network(DNN) is hard to deployed on hardware with the limitation of **computation resources**, **storage**, **battery power**.

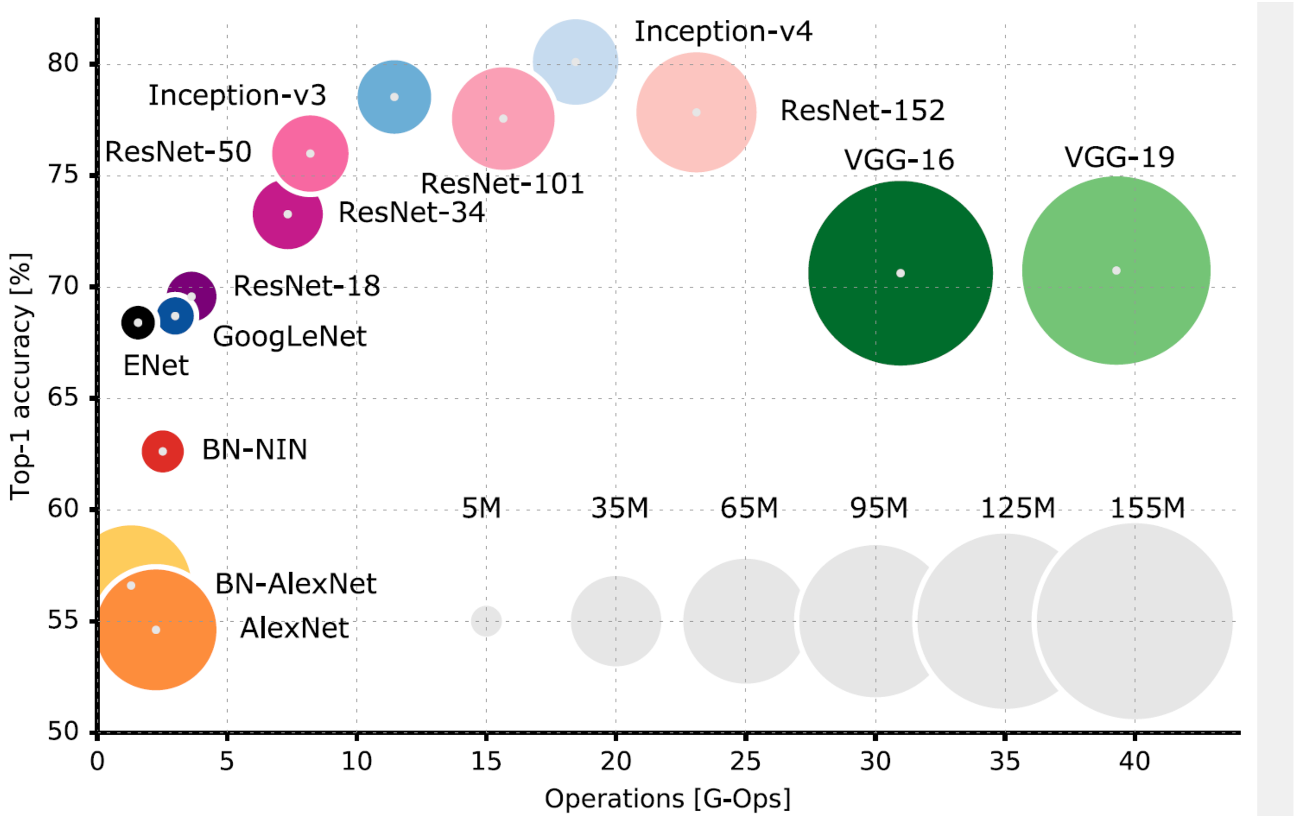


Figure. Performance and model size of different models on ImageNet

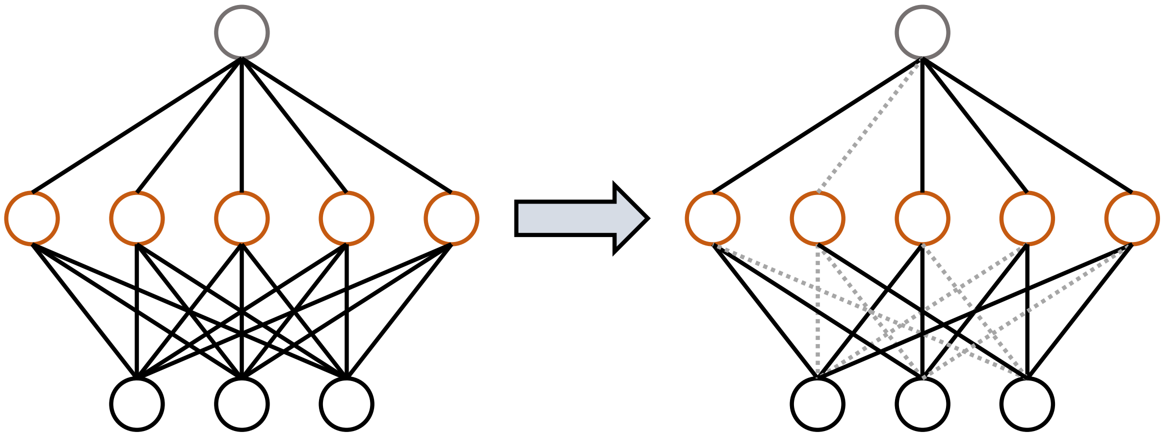
Model Compression

Existing Compression Methods:

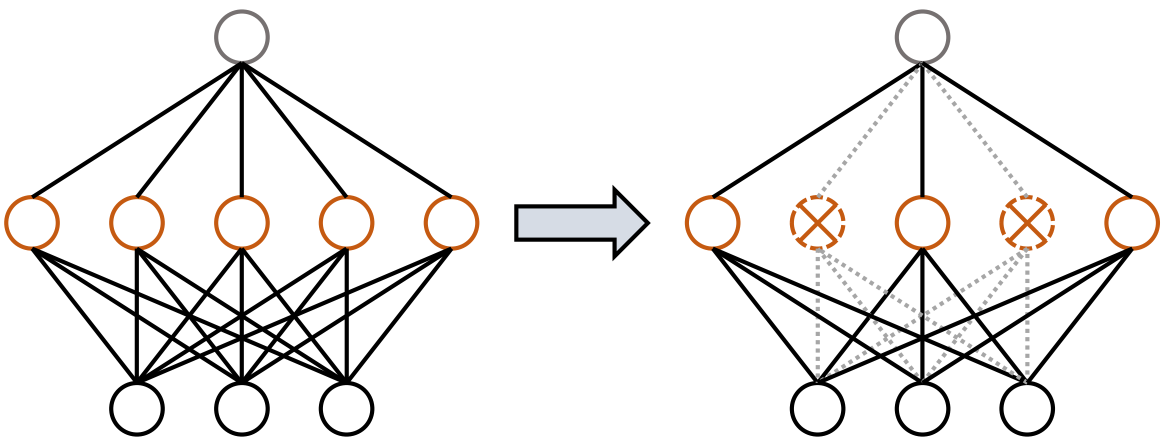
- **Quantization:** convert full-precision weights to low-precision version, e.g. INQ, BWN, TWN.
- **Pruning:** remove less important weights/filters from the model, e.g. Deep Compression, DNS, ThiNet
- **Design new structure:** SqueezeNet, Distilling, ShuffleNet

Pruning Methods

- **Non-structured Pruning:** remove less important **weights**



- **Structured Pruning:** remove less important **filters** from model



Motivation

Problems of Non-structured Pruning

- Need specialized hardware and software for inference
- Ignore cache and memory issues, which leads to limited practical acceleration

Benefits of Structured Pruning

- No change of network structure and can supported by existing deep learning libraries
- Reduce the memory and accelerate inference

Proposed Method

Proposed Method

ThiNet(Thin Net): a filter level pruning compression framework for model compression

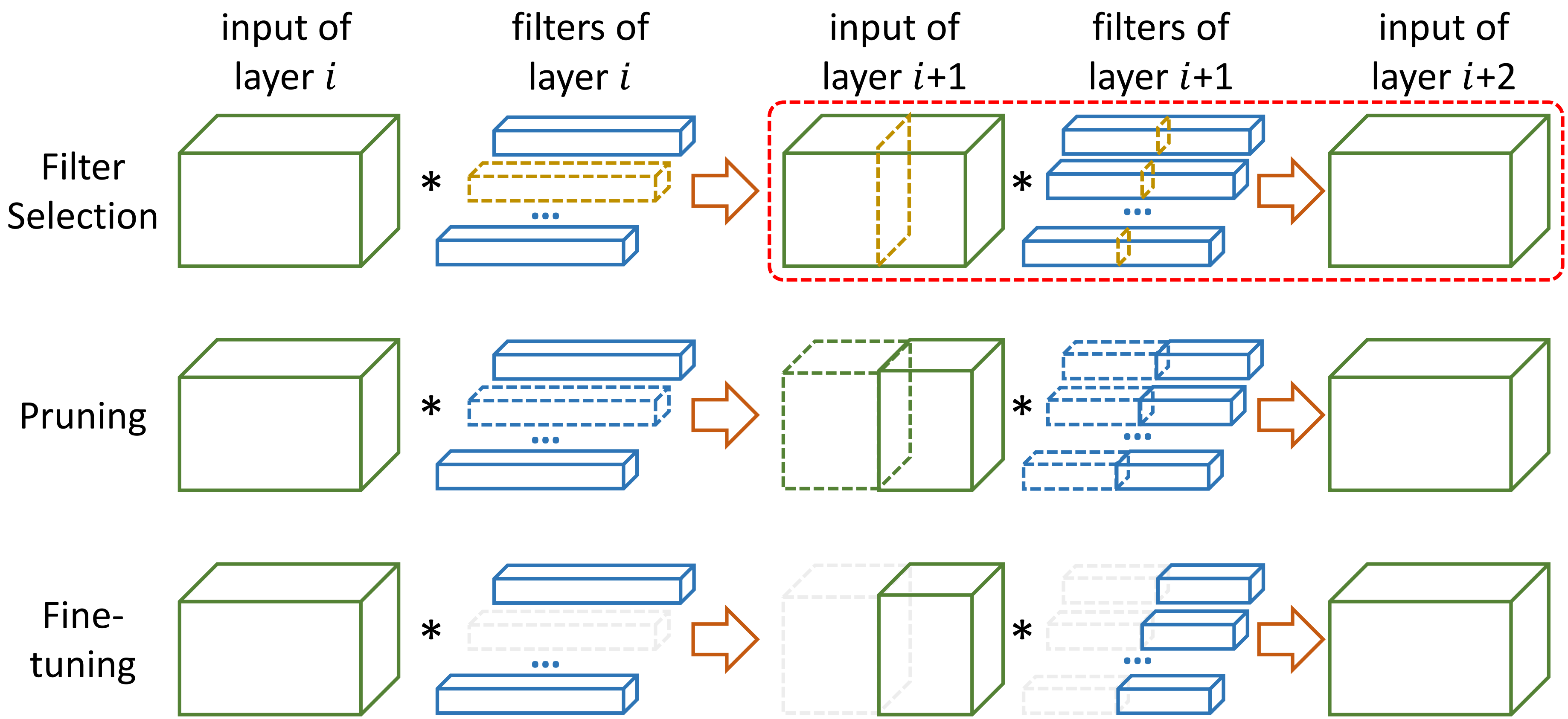
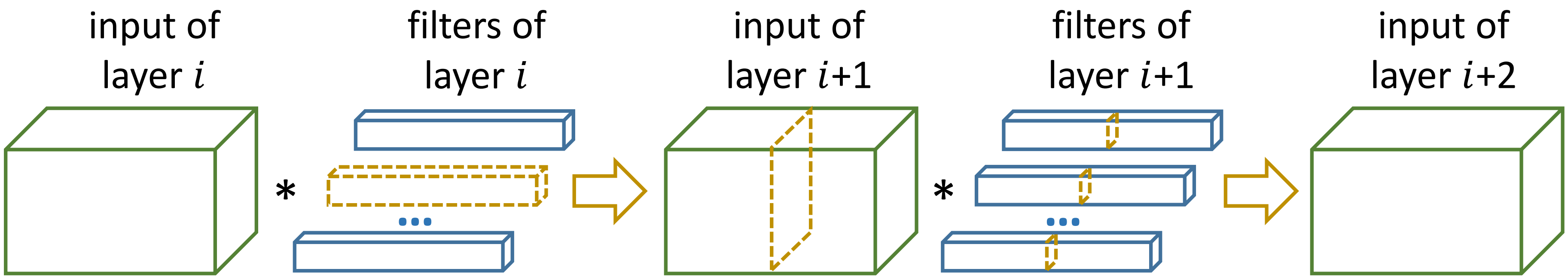


Figure. Illustration of ThiNet

Proposed Method

Framework of ThiNet

- **Filter selection:** use layer i+1to guide the pruning in layer i
- **Pruning:** prune weak channels in layer i+1 and the related filters in layer i
- **Fine-tuning:** reduce loss of accuracy



Filter Selection

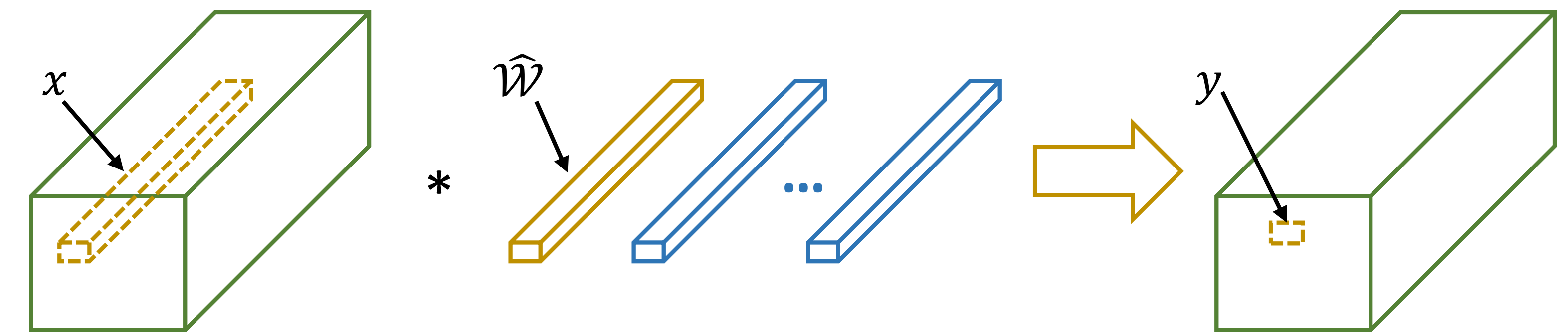
- Convolution operation can be computed as follows:

$$y = \sum_{c=1}^{C} \sum_{k_1=1}^{K} \sum_{k_2=1}^{K} \widehat{\mathcal{W}}_{c,k_1,k_2} \times x_{c,k_1,k_2} + b \qquad (1)$$

y is the element sampled from input of layer i+2

$\widehat{\mathcal{W}} \in \mathbb{R}^{C \times K \times K}$ is the corresponding filter

$x \in \mathbb{R}^{C \times K \times K}$ is the sliding window



Filter Selection

- Define:

$$\hat{x} = \sum_{k_1}^{K} \sum_{k-2}^{K} \widehat{\mathcal{W}}_{c,k_1,k_2} \times x_{c,k_1,k_2} \qquad (2)$$

- Then:

$$\hat{y} = \sum_{c=1}^{C} \hat{x}_c \quad \text{where } \hat{y} = y - b \qquad (3)$$

- If we can find a subset $S \subset \{1, 2, \ldots, C\}$ and $\hat{y} = \sum_{c \in S} \hat{x}_c$, then $\hat{x}_{c \notin S}$ can be removed without changing the result

Greedy Method

- Given a set of m training examples $\{(\hat{x}_i, \hat{y}_i)\}$, the channel selection problem can be solved as optimization problem:

$$\arg\min_{S} \sum_{i=1}^{m} \left(\hat{y}_i - \sum_{j \in S} \hat{x}_{i,j} \right)^2 \tag{4}$$

$$s.t.\ |S| = C \times r, S \subset \{1,2,\ldots,C\}$$

- Let T be the subset of removed channels, then:

$$\arg\min_{T} \sum_{i=1}^{m} \left(\sum_{j \in T} \hat{x}_{i,j} \right)^2 \tag{5}$$

$$s.t.\ |T| = C \times (1-r), T \subset \{1,2,\ldots,C\}$$

Greedy Method

- Use greedy method to solve the optimization problem

A greedy algorithm for minimizing Eq. (5)

Input: Training set $\{(\hat{x}_i, \hat{y}_i\ )\}$ and compression rate r
Output: The subset of removed channels T
$T \leftarrow \emptyset; I \leftarrow \{1,2, \dots, C\};$
while $|T| < C\ (1 - r)$ **do**
  $min\_value \leftarrow\ +\infty;$
  for each item $i \in I$ **do**
    $tmpT \leftarrow T \cup \{i\};$
    compute $value$ from Eq. (5) using $tmpT$;
    if $value < min\_value$ then
      $min\_value \leftarrow\ value; min\_i \leftarrow I;$
    end if
  end for
  move min_i from I to T
end while

Minimize the Reconstruction Error

- Minimize the reconstruction error by weighting the channels:

$$\hat{w} = \arg\min_{w} \sum_{i=1}^{m} (\hat{y}_i - w^T \hat{x}_i^*)^2 \tag{6}$$

where $\hat{x}_i^*$ indicates the training samples after channel selection

- Eq. (6) can be solved by the ordinary least squares approach:

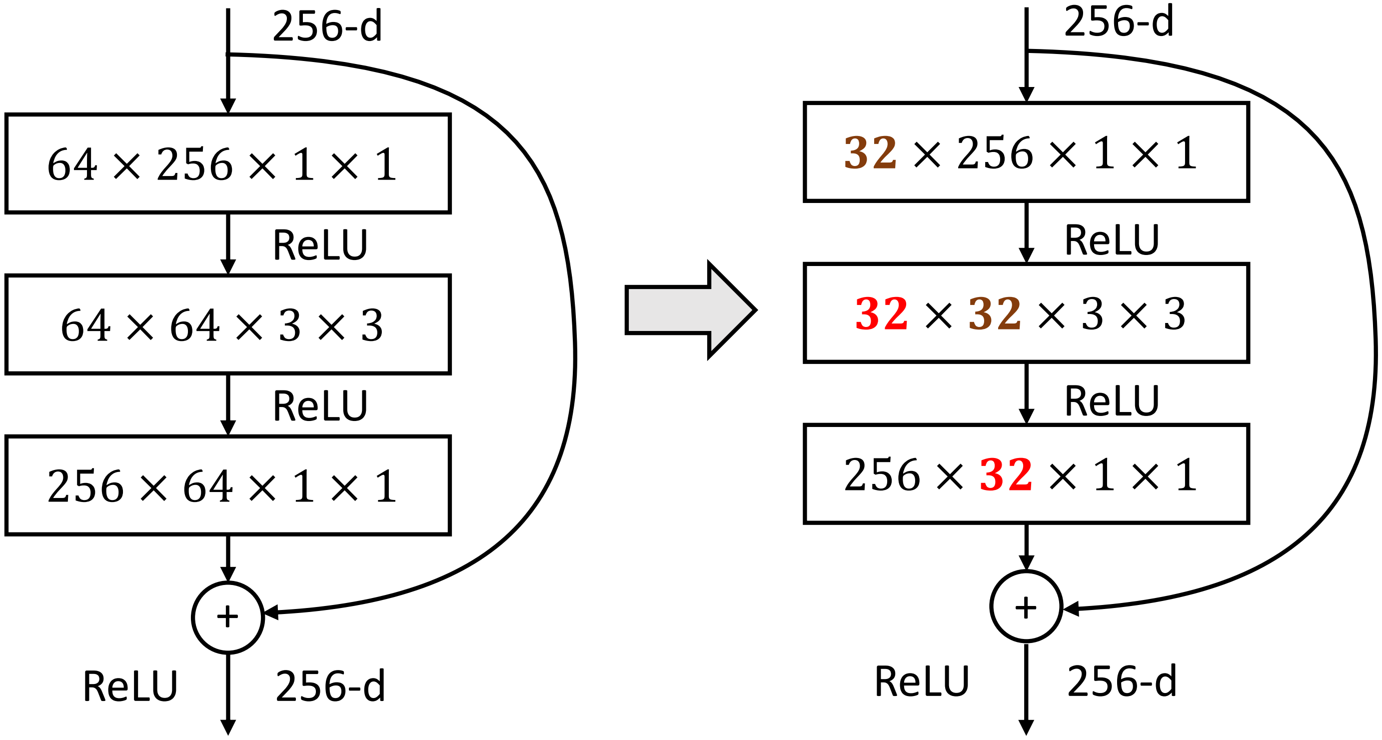
$$\hat{w} = (X^T X)^{-1} X^T y \tag{7}$$

Experimental Results

Pruning Strategy

- **VGG-16:** prune the **first 10** convolutional layers and replace the FC layers with a global average pooling layer

- **ResNet-50:** prune the **first two** convolutional layers



Comparison of Existing Methods

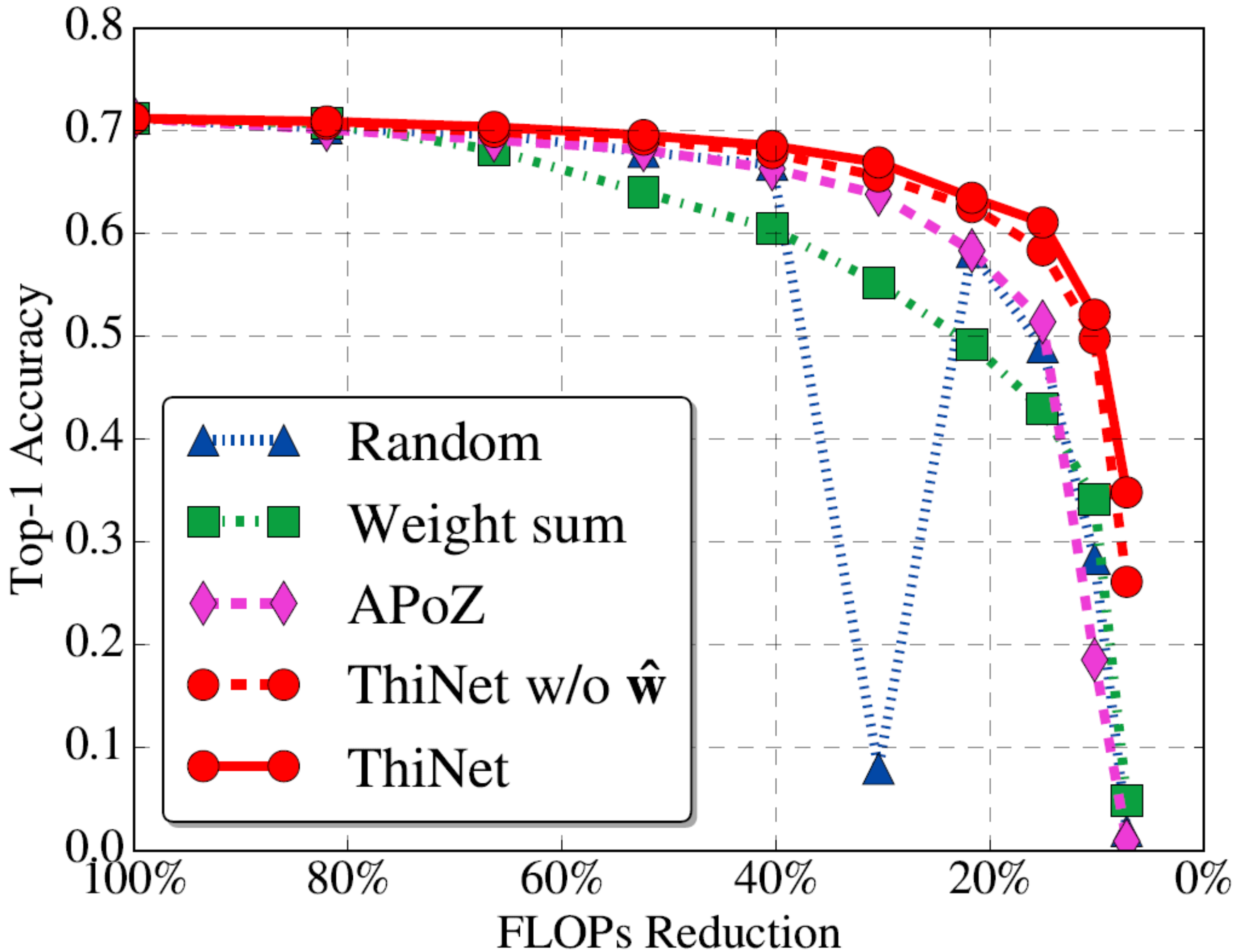


Figure. Comparison of different channel selection methods, using VGG-16-GAP on CUB-200

VGG-16 on ImageNet

- **ThiNet-Conv:** prune **50%** of the **first 10** convolutional layers
- **ThiNet-GAP:** replace the **FC layers** with a global average pooling (GAP) layer based on ThiNet-Conv

Table. Pruning results of VGG-16 on ImageNet

Model	Top-1	Top-5	#Param.	#FLOPs	f./b. (ms)
Original	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(Han et al.)	57.67%	80.39%	1.24M	1.72B	37.30/68.62

VGG-16 on ImageNet

- **ThiNet-WS:** use the weight sum (WS) method for pruning

Table. Comparison of state-of-the-art methods on VGG-16

Method	Top-1	Top-5	#Param.	#FLOPs
APoZ-1 (Hu et al.)	-2.16%	-0.84%	2.04×	≈1×
APoZ-2 (Hu et al.)	+1.81%	+1.25%	2.70×	≈1×
Taylor-1 (Molchanov et al.)	-	-1.44%	≈1×	2.68×
Taylor-2 (Molchanov et al.)	-	-3.94%	≈1×	3.86×
ThiNet-WS (Li et al.)	+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×

ResNet-50 on ImageNet

Table. Performance of pruning ResNet-50 on ImageNet

Model	Top-1	Top-5	#Param.	#FLOPs	f./b. (ms)
Original	72.88%	91.14%	25.56M	7.72B	188.27/269.32
ThiNet-70	72.04%	90.67%	16.94M	4.88B	169.38/243.37
ThiNet-50	71.01%	90.02%	12.38M	3.41B	153.60/212.29
ThiNet-30	68.42%	88.30%	8.66M	2.20B	144.45/200.67

Domain Adaptation Ability

Table. Comparison of different methods on CUB-200 and Indoor-67. "FT" denotes "Fine Tune"

Data set	Stategy	#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**	**9.58B**	**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%

Conclusion

Conclusion

Contributions

- Proposed ThiNet, a **filter pruning** framework, to accelerate and compress CNN models
- Formally establish filter pruning as an **optimization problem**
- VGG-16 model can be pruned into **5.05MB**, and shows promising generalization ability on **transfer learning**

Future work

- Prune the projection short-cuts of ResNet
- Explore more on channel selection method

Thank You

Reference

- Luo, J. H., Wu, J., & Lin, W. (2017). ThiNet: A Filter Level Pruning Method for Deep Neural Network Compression. arXiv preprint arXiv:1707.06342.
- 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.
- H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf. Pruning filters for efficient ConvNets. In ICLR, pages 1–13, 2017.