Network Pruning

Jinyang Guo（郭晋阳）
jinyangguo@buaa.edu.cn

Last lecture

- Introduction of generative AI on edge
- On-device generative AI offers many benefits
- Generative AI is happening now on the device
- On-device AI leadership is enabling generative AI to scale
- Hybrid AI is the future

目录

1 Introduction

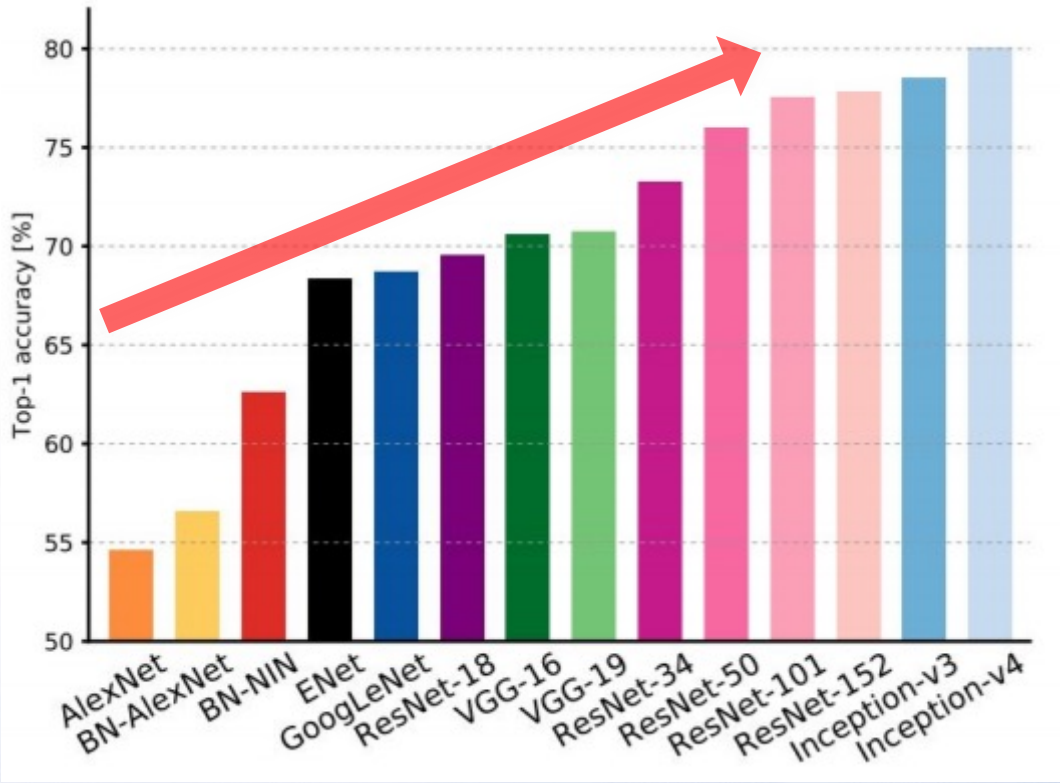
2 Network pruning

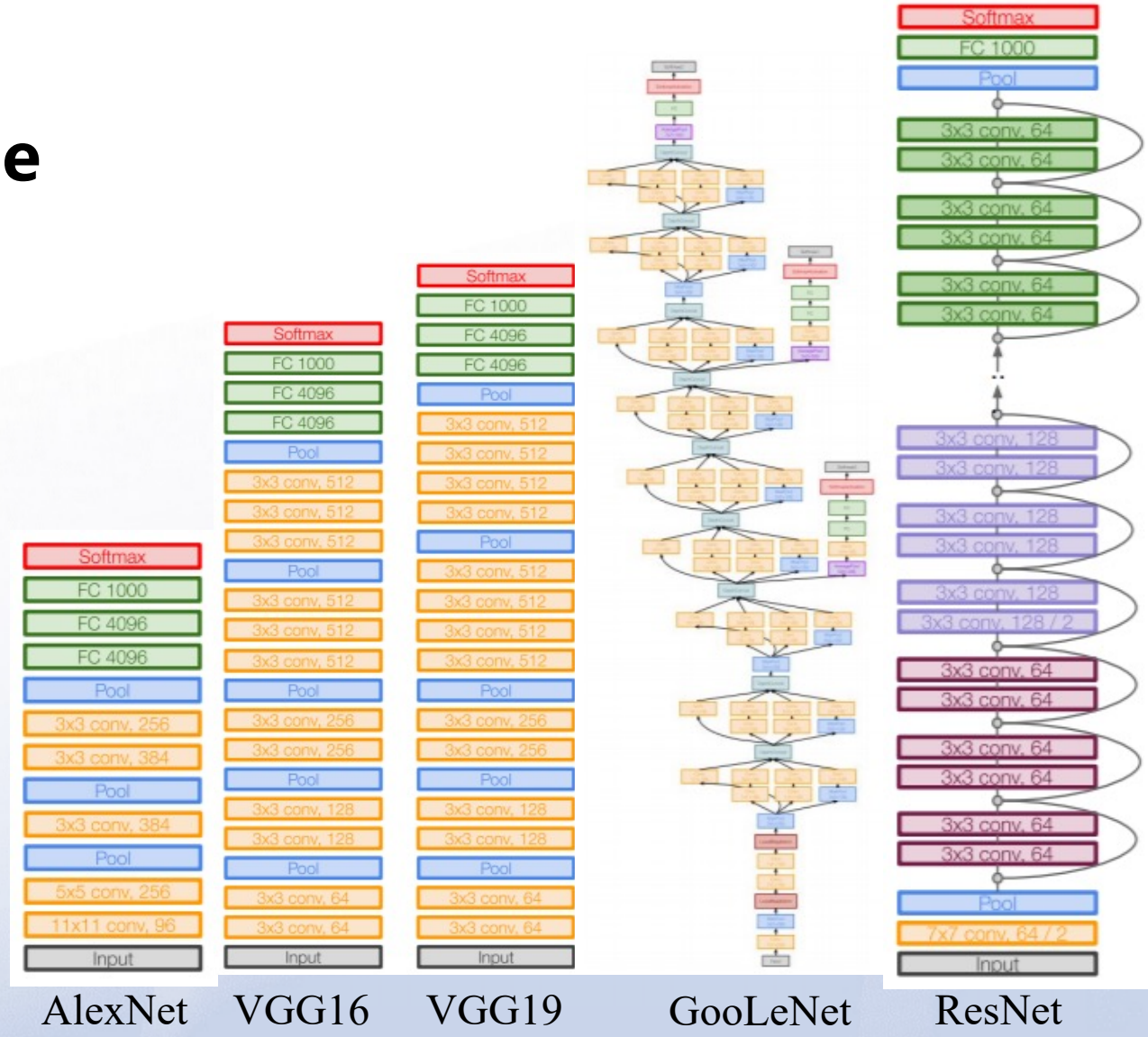
3 Structured pruning

4 Unstructured pruning

Introduction

- Model size grows up quickly
- Large size, better performance

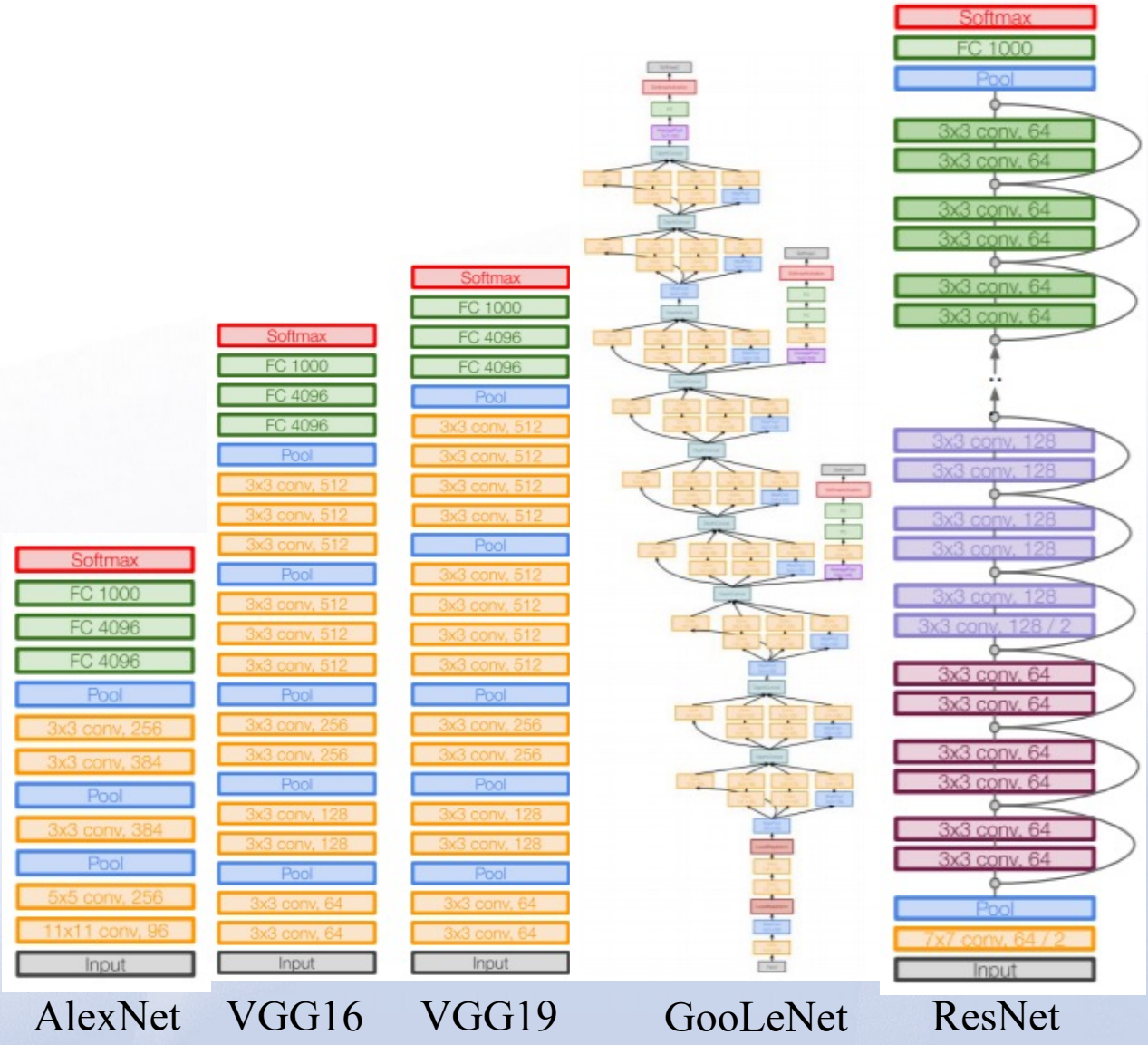




Introduction

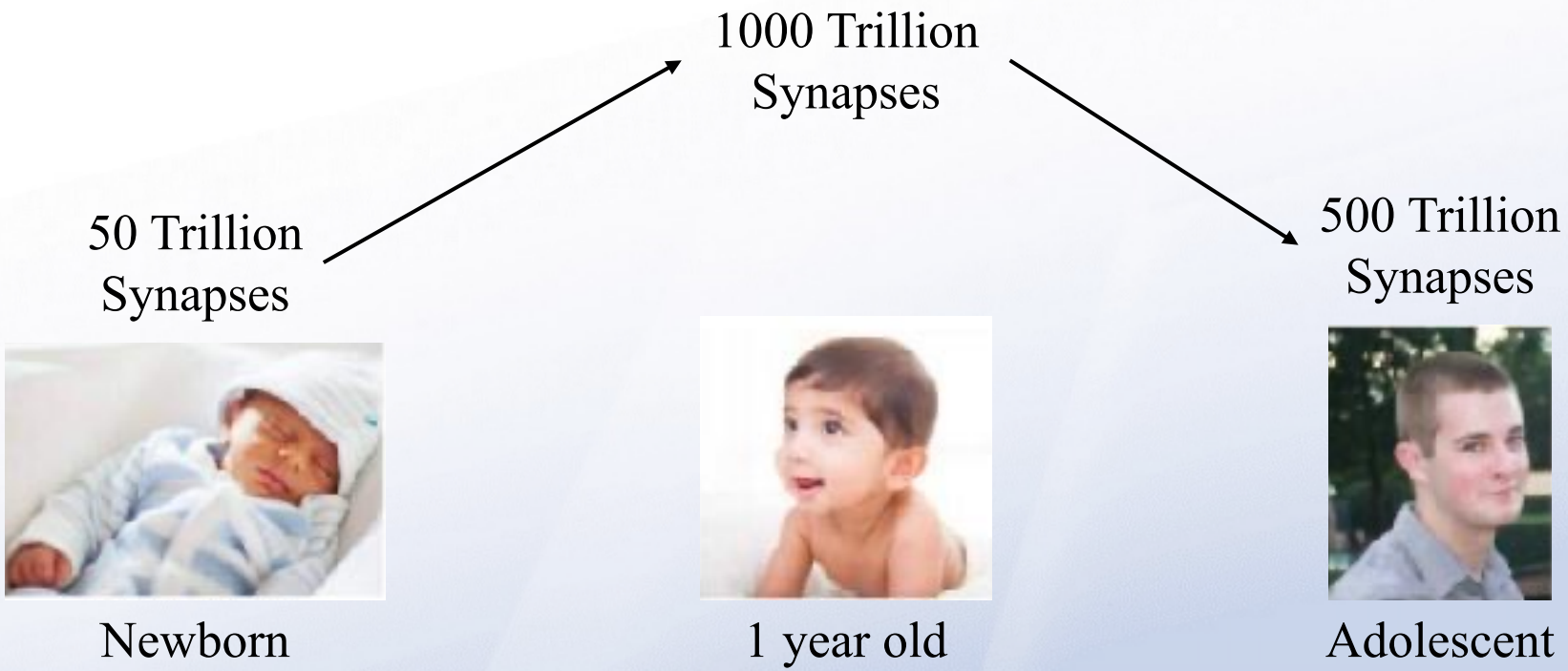
- Computation inefficiency
- Storage inefficiency

Model	Params
AlexNet	60M
VGG16	138M
VGG19	144M
GooLeNet	5M
ResNet50	25M
ResNet101	45M
ResNet152	60M



Introduction

- Human brain is sparse
- Infrequently used synapses would degenerate and eventually disappear during subsequent growth



目录

1 Introduction

2 Network pruning

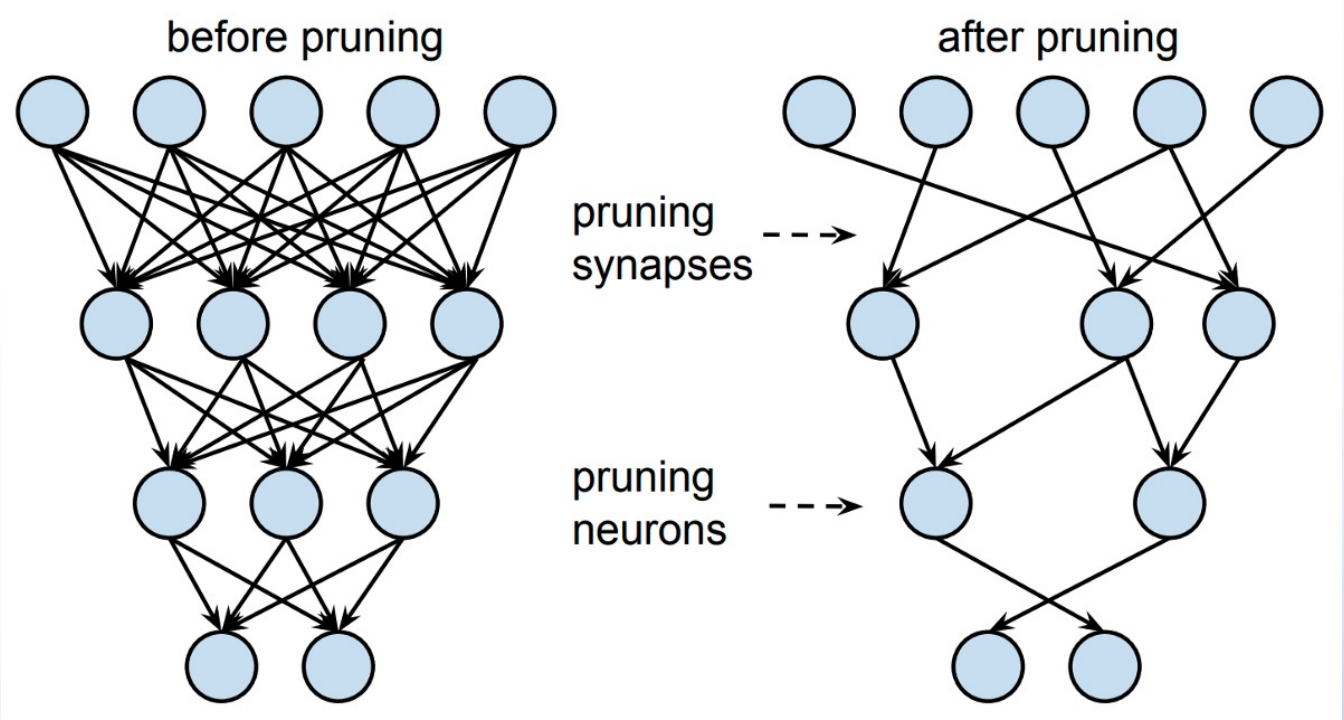
3 Structured pruning

4 Unstructured pruning

Network pruning

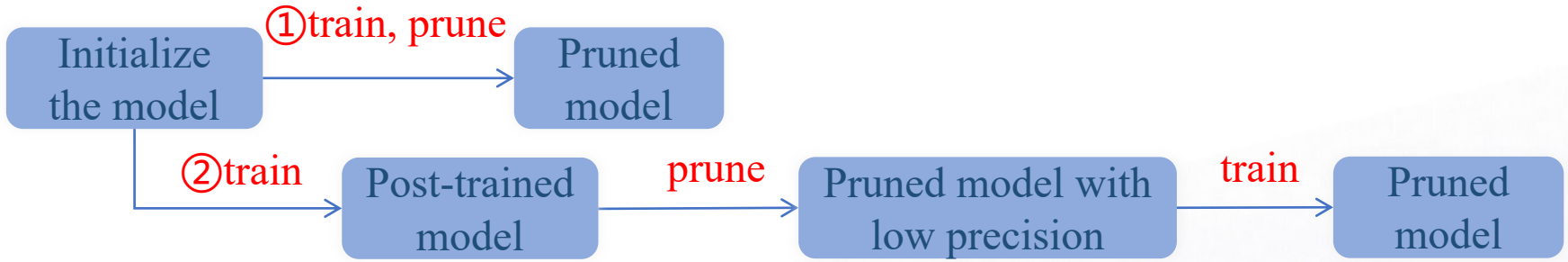
Neural network pruning/sparsity: remove unimportant weights/neurons

Core problem: how to determine if a weight/neuron is unimportant

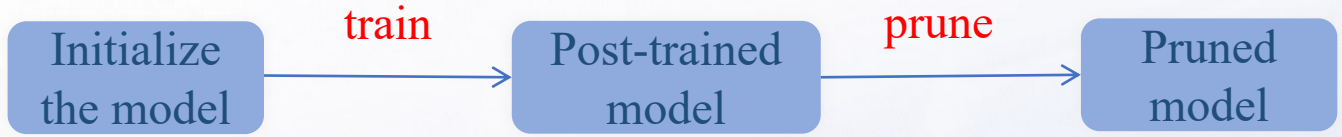


Network pruning

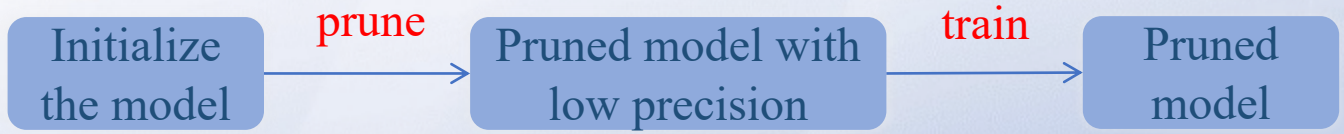
Training-based Pruning



Post-training Pruning

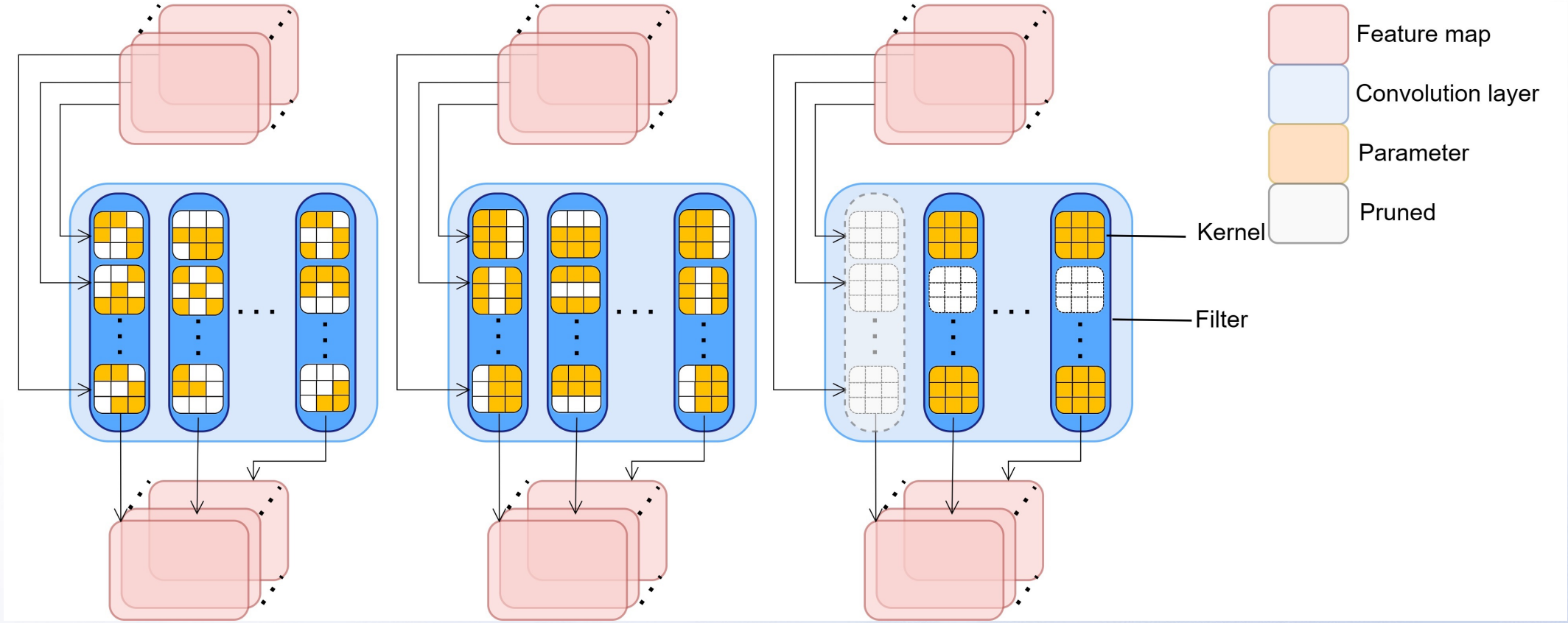


Pre-training Pruning



Network pruning

- Unstructured Pruning vs Semi-structured Pruning vs Structured Pruning



Unstructured Pruning

Semi-structured Pruning

Structured Pruning

Network pruning

Network pruning has attracted extensive research attention since 2016

2016

compress DNN by pruning firstly

2023

LLM pruning

Optimal Brain Damage

Yann Le Cun, John S. Denker and Sara A. Solla
AT&T Bell Laboratories, Holmdel, N. J. 07733

PRUNING FILTERS FOR EFFICIENT CONVNETS

Hao Li[*] University of Maryland haoli@cs.umd.edu
Asim Kadav NEC Labs America asim@nec-labs.com
Igor Durdanovic NEC Labs America igord@nec-labs.com
Hanan Samet[†] University of Maryland hjs@cs.umd.edu
Hans Peter Graf NEC Labs America hpg@nec-labs.com

DEEP COMPRESSION: COMPRESSING DEEP NEURAL NETWORKS WITH PRUNING, TRAINED QUANTIZATION AND HUFFMAN CODING

Song Han Stanford University, Stanford, CA 94305, USA songhan@stanford.edu
Huizi Mao Tsinghua University, Beijing, 100084, China mhz12@mails.tsinghua.edu.cn
William J. Dally Stanford University, Stanford, CA 94305, USA NVIDIA, Santa Clara, CA 95050, USA dally@stanford.edu

A SIMPLE AND EFFECTIVE PRUNING APPROACH FOR LARGE LANGUAGE MODELS

Mingjie Sun[1*] **Zhuang Liu**[2*] **Anna Bair**[1] **J. Zico Kolter**[1,3]
[1]Carnegie Mellon University [2]Meta AI Research [3]Bosch Center for AI

1990

the first pruning method

2017

the first structured pruning method

…
many other methods:
He et al.: AMC
Luo et al.: ThiNet
Liu et al.: Slimming

目录

1 Introduction

2 Network pruning

3 Structured pruning

4 Unstructured pruning

Structured pruning

Learning both Weights and Connections for Efficient Neural Networks

Song Han
Stanford University
songhan@stanford.edu

Jeff Pool
NVIDIA
jpool@nvidia.com

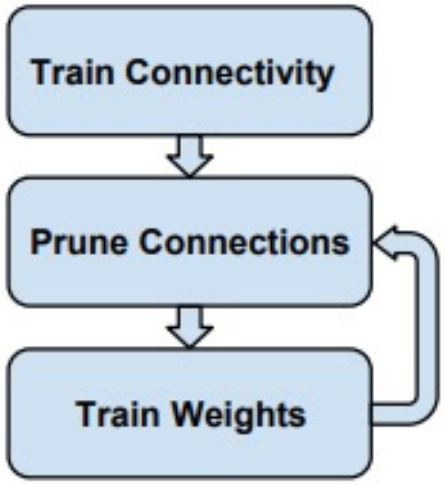
John Tran
NVIDIA
johntran@nvidia.com

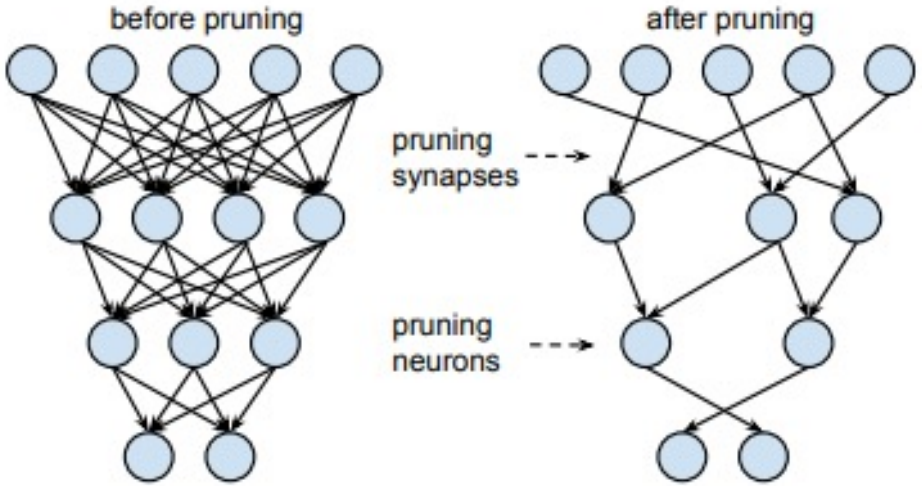
William J. Dally
Stanford University
NVIDIA
dally@stanford.edu

Structured pruning

Problem: How to determine the importance of each neuron

Method: Determine by L2-norm





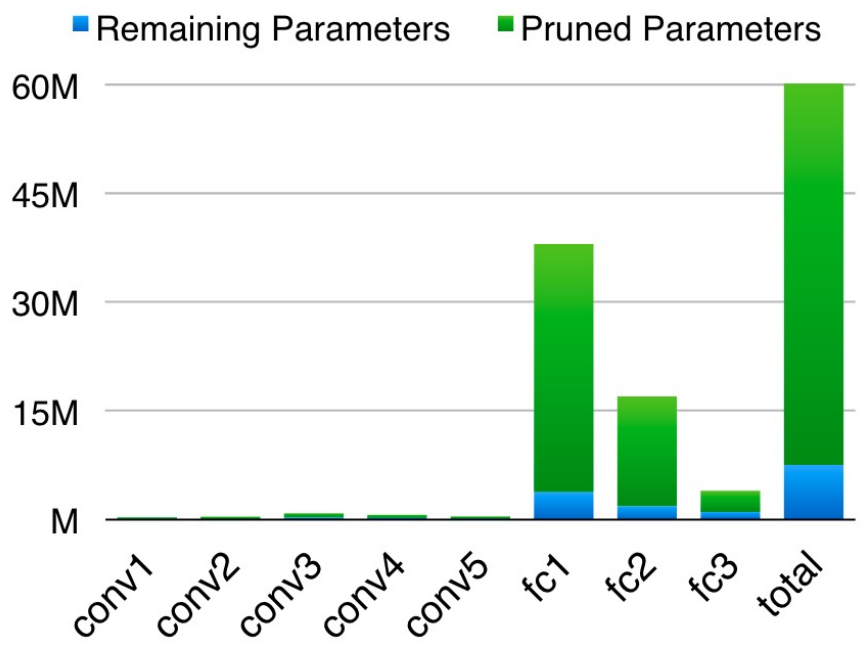
Sparse the network **Iteratively**

Rank the neuron based on **L2-norm**
Remove neuron with **small importance**

Structured pruning

Result: No performance drop under low sparsity ratio

Layer	Weights	FLOP	Act%	Weights%	FLOP%
conv1	35K	211M	88%	84%	84%
conv2	307K	448M	52%	38%	33%
conv3	885K	299M	37%	35%	18%
conv4	663K	224M	40%	37%	14%
conv5	442K	150M	34%	37%	14%
fc1	38M	75M	36%	9%	3%
fc2	17M	34M	40%	9%	3%
fc3	4M	8M	100%	25%	10%
Total	61M	1.5B	54%	**11%**	**30%**



9x parameter compression for AlexNet
3x computational compression

Largely reduce the number of parameter in fully-connected layer

Structured pruning

PRUNING FILTERS FOR EFFICIENT CONVNETS

Hao Li*
University of Maryland
haoli@cs.umd.edu

Asim Kadav
NEC Labs America
asim@nec-labs.com

Igor Durdanovic
NEC Labs America
igord@nec-labs.com

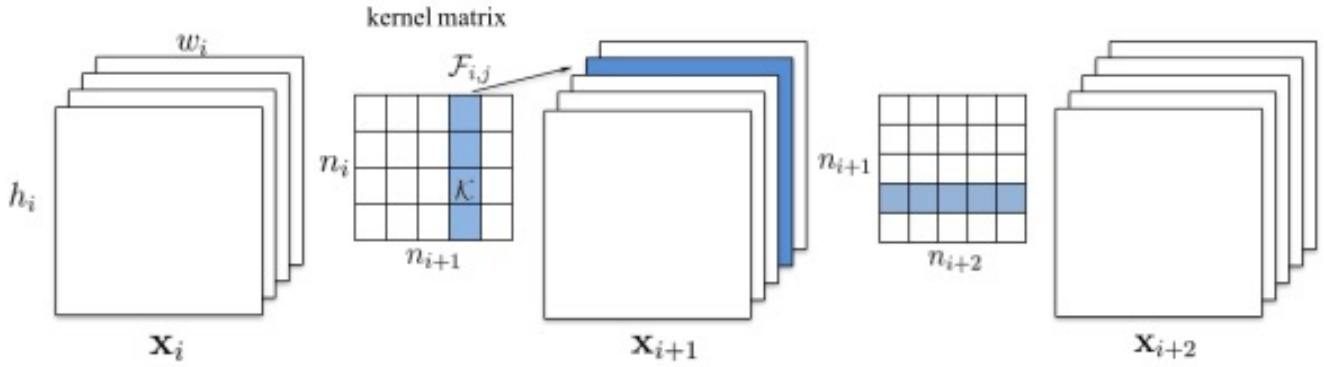
Hanan Samet†
University of Maryland
hjs@cs.umd.edu

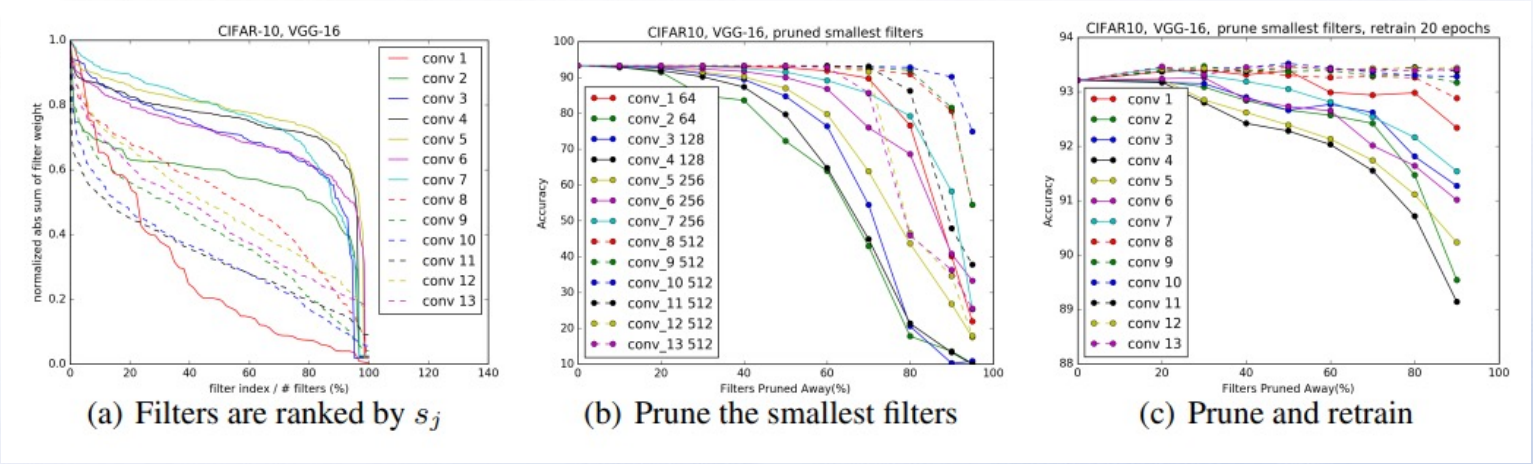
Hans Peter Graf
NEC Labs America
hpg@nec-labs.com

Structured pruning

Problem: how to reduce the computation costs in the convolutional layers

Method: prune filters that have small effect on the output accuracy





(a) Filters are ranked by s_j (b) Prune the smallest filters (c) Prune and retrain

Prune filters with the smallest sum values (L1-norm) and their corresponding feature maps

Structured pruning

Result: computation costs are reduced significantly

Model	Error(%)	FLOP	Pruned %	Parameters	Pruned %
VGG-16	6.75	3.13×10^8		1.5×10^7	
VGG-16-pruned-A	**6.60**	2.06×10^8	34.2%	5.4×10^6	64.0%
VGG-16-pruned-A scratch-train	6.88				
ResNet-56	6.96	1.25×10^8		8.5×10^5	
ResNet-56-pruned-A	6.90	1.12×10^8	10.4%	7.7×10^5	9.4%
ResNet-56-pruned-B	**6.94**	9.09×10^7	27.6%	7.3×10^5	13.7%
ResNet-56-pruned-B scratch-train	8.69				
ResNet-110	6.47	2.53×10^8		1.72×10^6	
ResNet-110-pruned-A	**6.45**	2.13×10^8	15.9%	1.68×10^6	2.3%
ResNet-110-pruned-B	6.70	1.55×10^8	38.6%	1.16×10^6	32.4%
ResNet-110-pruned-B scratch-train	7.06				
ResNet-34	26.77	3.64×10^9		2.16×10^7	
ResNet-34-pruned-A	27.44	3.08×10^9	15.5%	1.99×10^7	7.6%
ResNet-34-pruned-B	27.83	2.76×10^9	24.2%	1.93×10^7	10.8%
ResNet-34-pruned-C	27.52	3.37×10^9	7.5%	2.01×10^7	7.2%

about 30% reduction in FLOP for VGG and ResNets without significant loss in the original accuracy

Structured pruning

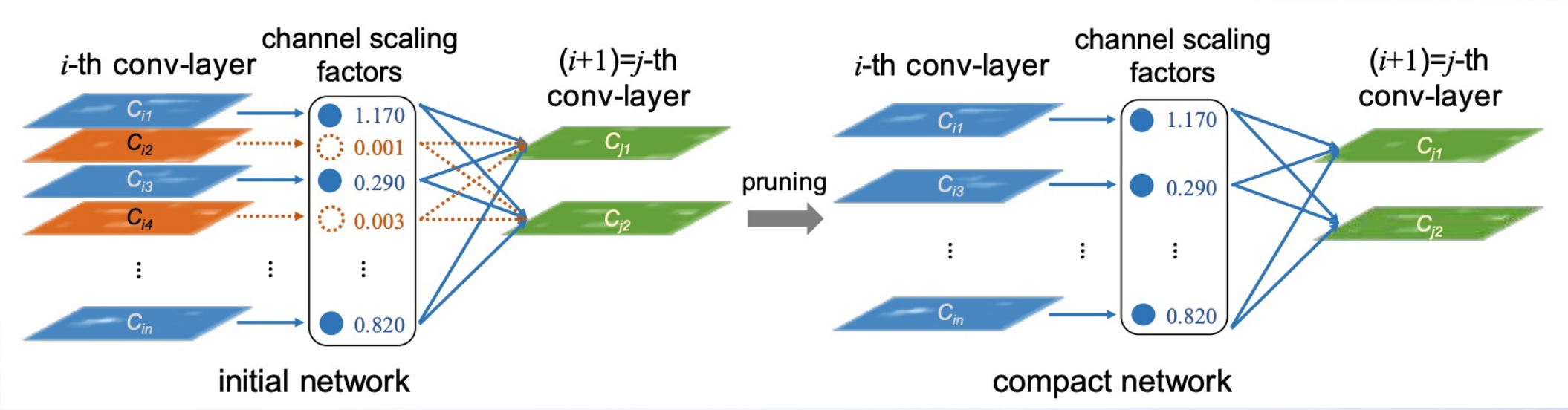
Learning Efficient Convolutional Networks through Network Slimming

Zhuang Liu[1*] Jianguo Li[2] Zhiqiang Shen[3] Gao Huang[4] Shoumeng Yan[2] Changshui Zhang[1]
[1]CSAI, TNList, Tsinghua University [2]Intel Labs China [3]Fudan University [4]Cornell University
{liuzhuangthu, zhiqiangshen0214}@gmail.com, {jianguo.li, shoumeng.yan}@intel.com,
gh349@cornell.edu, zcs@mail.tsinghua.edu.cn

Structured pruning

Problem: How to learn a sparse network through training

Method: Add the channel scaling factors



Channel-level pruning

$$L = L_{softmax} + \lambda \sum ||g||$$

Sparsity regularization imposed on the scaling factors

Structured pruning

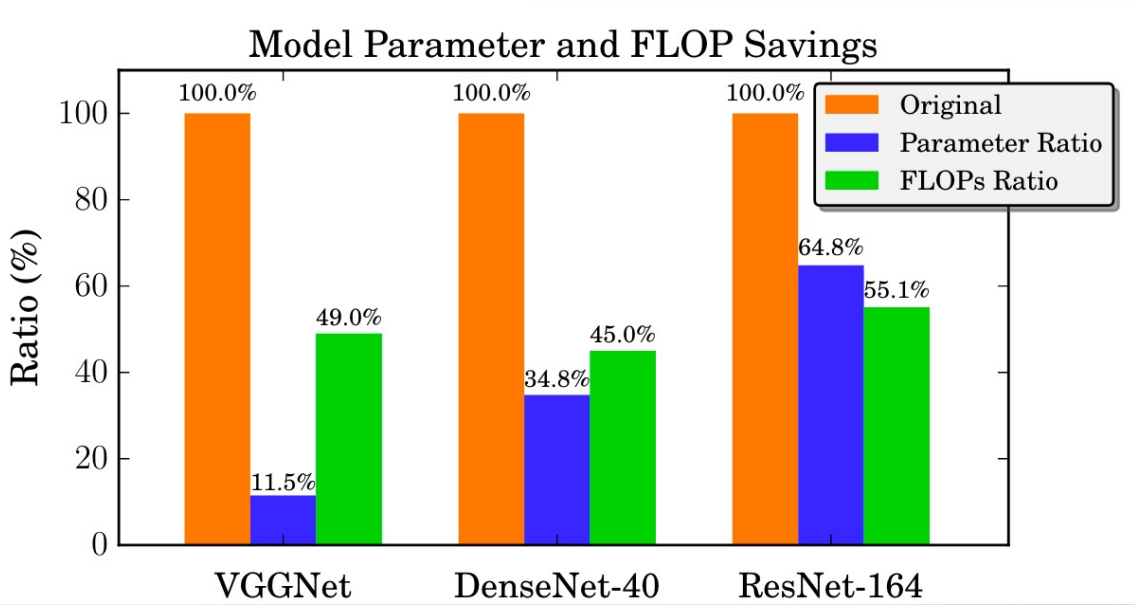
Result: Nearly **no performance drop**, achieving an acceleration effect

(a) Test Errors on CIFAR-10

Model	Test error (%)	Parameters	Pruned	FLOPs	Pruned
VGGNet (Baseline)	6.34	20.04M	-	7.97×10^8	-
VGGNet (70% Pruned)	**6.20**	2.30M	88.5%	3.91×10^8	51.0%
DenseNet-40 (Baseline)	6.11	1.02M	-	5.33×10^8	-
DenseNet-40 (40% Pruned)	**5.19**	0.66M	35.7%	3.81×10^8	28.4%
DenseNet-40 (70% Pruned)	5.65	0.35M	65.2%	2.40×10^8	55.0%
ResNet-164 (Baseline)	5.42	1.70M	-	4.99×10^8	-
ResNet-164 (40% Pruned)	**5.08**	1.44M	14.9%	3.81×10^8	23.7%
ResNet-164 (60% Pruned)	5.27	1.10M	35.2%	2.75×10^8	44.9%

(b) Test Errors on CIFAR-100

Model	Test error (%)	Parameters	Pruned	FLOPs	Pruned
VGGNet (Baseline)	26.74	20.08M	-	7.97×10^8	-
VGGNet (50% Pruned)	**26.52**	5.00M	75.1%	5.01×10^8	37.1%
DenseNet-40 (Baseline)	25.36	1.06M	-	5.33×10^8	-
DenseNet-40 (40% Pruned)	**25.28**	0.66M	37.5%	3.71×10^8	30.3%
DenseNet-40 (60% Pruned)	25.72	0.46M	54.6%	2.81×10^8	47.1%
ResNet-164 (Baseline)	23.37	1.73M	-	5.00×10^8	-
ResNet-164 (40% Pruned)	**22.87**	1.46M	15.5%	3.33×10^8	33.3%
ResNet-164 (60% Pruned)	23.91	1.21M	29.7%	2.47×10^8	50.6%



50% computational compression

10x parameter compression, only **0.14%** accuracy drop

Structured pruning

Channel Pruning for Accelerating Very Deep Neural Networks

Yihui He*
Xi'an Jiaotong University
Xi'an, 710049, China
heyihui@stu.xjtu.edu.cn

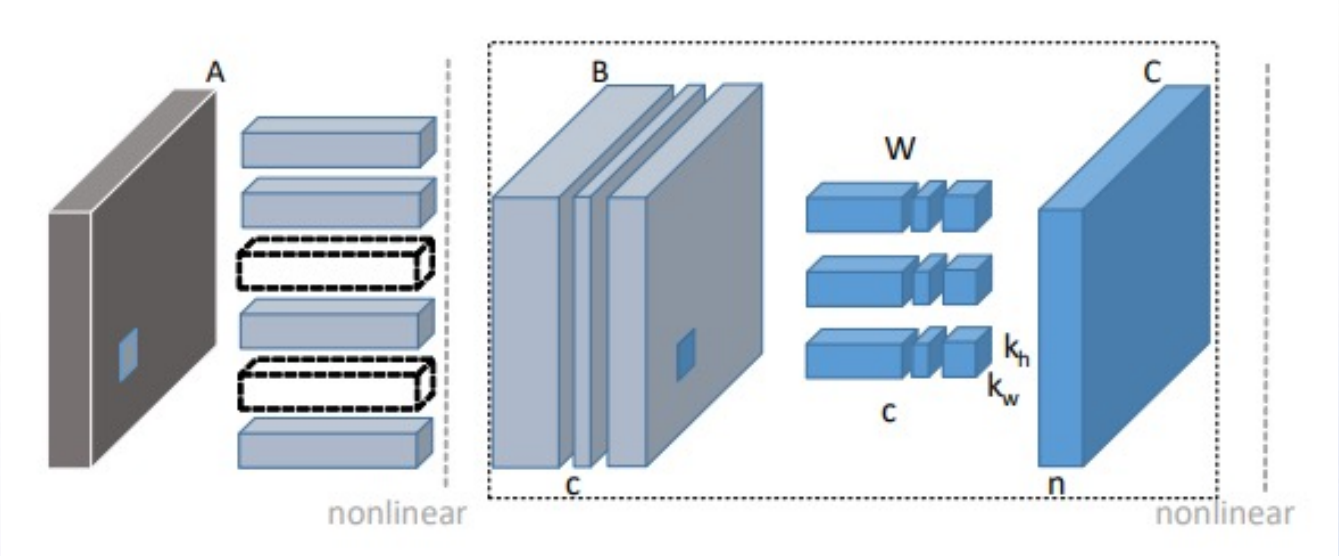
Xiangyu Zhang
Megvii Inc.
Beijing, 100190, China
zhangxiangyu@megvii.com

Jian Sun
Megvii Inc.
Beijing, 100190, China
sunjian@megvii.com

Structured pruning

Problem: how to select the channels to remove

Method: minimize reconstruction error on the output feature maps



reduce the number of channels of feature map B,
while minimizing the reconstruction error on feature map C

Structured pruning

Result: inference efficient while maintaining accuracy

Increase of top-5 error (1-view, 89.9%)		
Solution	4×	5×
Asym. 3D [53]	0.9	2.0
Asym. 3D (fine-tuned) [53]	0.3	1.0
Our 3C	0.7	1.3
Our 3C (fine-tuned)	**0.0**	**0.3**

Solution	Increased err.
Ours	8.0
Ours (enhanced)	4.0
Ours (enhanced, fine-tuned)	1.4

Solution	Increased err.
Filter pruning [31] (our impl.)	92.8
Filter pruning [31] (fine-tuned, our impl.)	4.3
Ours	2.9
Ours (fine-tuned)	**1.0**

5x speed-up along with only 0.3% increase of error for VGG-16

2x speed-up along with only 1.4% increase of error for ResNet

2x speed-up along with only 1.0% increase of error for Xception

Structured pruning

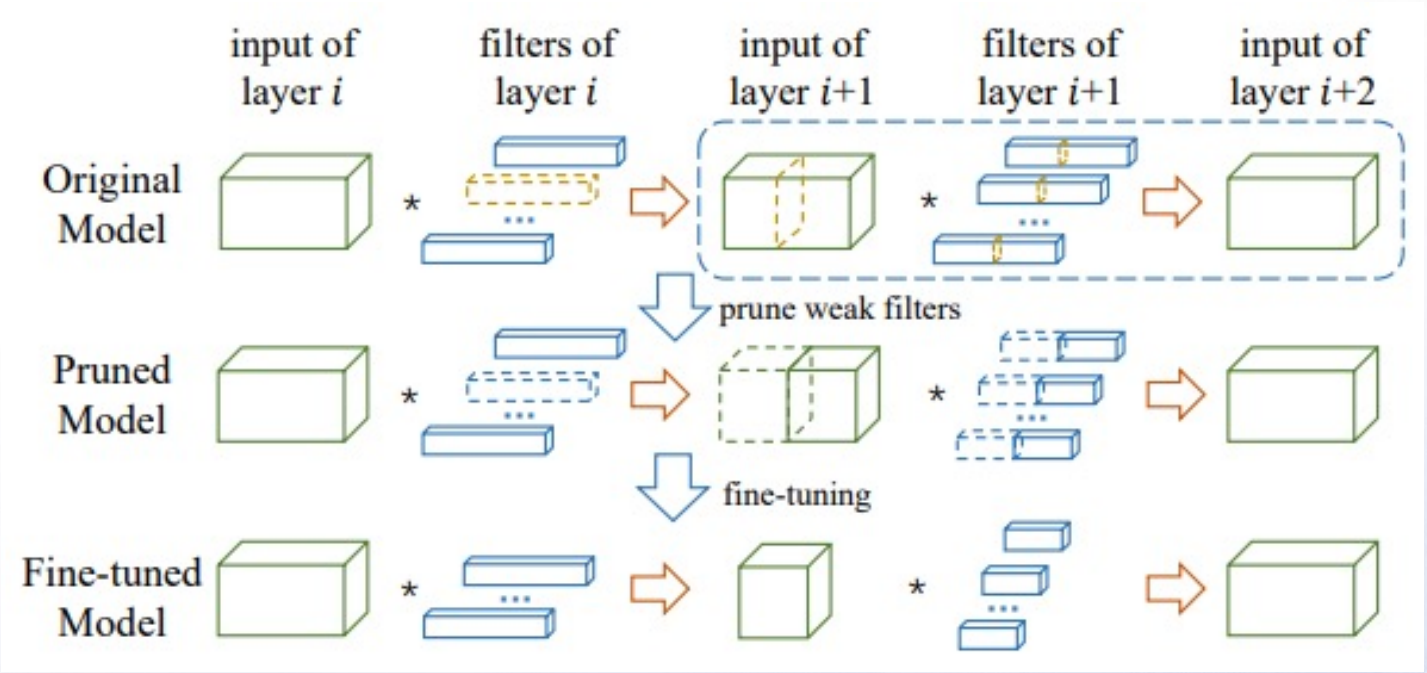
ThiNet: A Filter Level Pruning Method for Deep Neural Network Compression

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
luojh@lamda.nju.edu.cn, wujx2001@nju.edu.cn, wylin@sjtu.edu.cn

Structured pruning

Problem: how to accelerate and compress CNN models in both training and test stages

Method: prune filters based on statistics information computed from its **next layer**



use layer *i+1* to guide the pruning in layer *I*

weak channels in layer (*i+1*)'s input and their corresponding filters in layer *i* would be pruned away

Structured pruning

Result: **has advanced the state-of-the-art**

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

3.31× FLOPs reduction and 16.63× compression on VGG-16, with only 0.52% top-5 accuracy drop

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×

comparison among several state-of-the-art pruning methods

Structured pruning

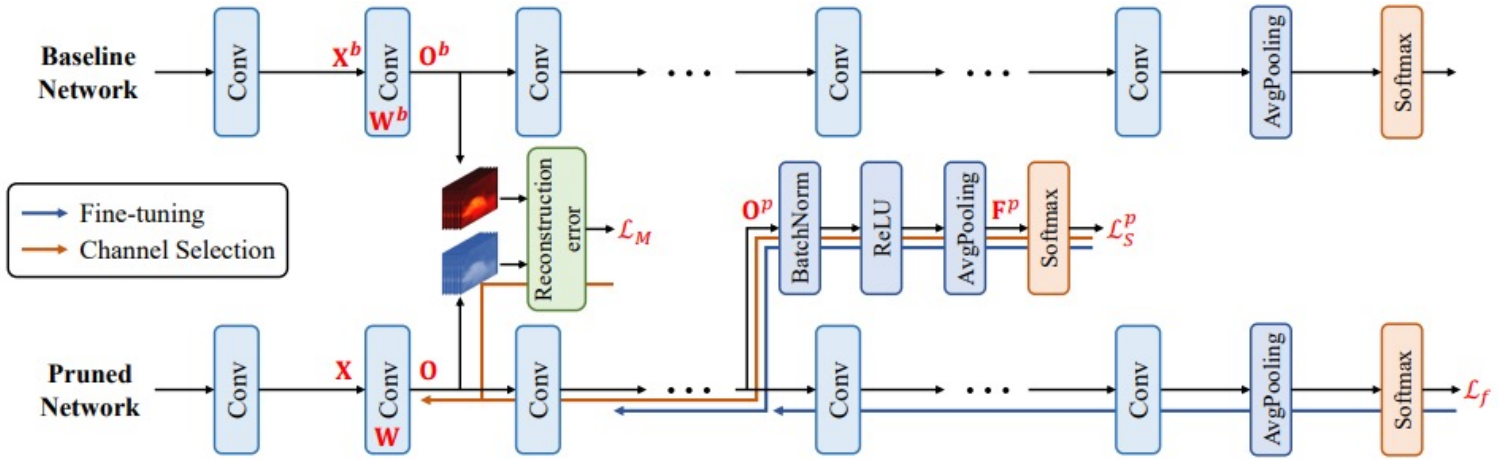
Discrimination-aware Channel Pruning for Deep Neural Networks

Zhuangwei Zhuang[1*], **Mingkui Tan**[1*†], **Bohan Zhuang**[2*], **Jing Liu**[1*], **Yong Guo**[1], **Qingyao Wu**[1], **Junzhou Huang**[3,4], **Jinhui Zhu**[1†]
[1]South China University of Technology, [2]The University of Adelaide,
[3]University of Texas at Arlington, [4]Tencent AI Lab
{z.zhuangwei, seliujing, guo.yong}@mail.scut.edu.cn, jzhuang@uta.edu
{mingkuitan, qyw, csjhzhu}@scut.edu.cn, bohan.zhuang@adelaide.edu.au

Structured pruning

Problem: existing methods ignore the discriminative power of channels

Method: introduce additional discrimination-aware losses



$\mathcal{L}_s^p$ denotes the discrimination-aware loss

$\mathcal{L}_M$ denotes the reconstruction loss

$\mathcal{L}_f$ denotes the final loss

Algorithm 1 Discrimination-aware channel pruning (DCP)

Input: Pre-trained model M, training data $\{\mathbf{x}_i, y_i\}_{i=1}^N$, and parameters $\{\kappa_l\}_{l=1}^L$.
for $p \in \{1, ..., P+1\}$ **do**
  Construct loss $\mathcal{L}_S^p$ to layer L_p as in Figure 1.
  Learn $\boldsymbol{\theta}$ and **Fine-tune** M with $\mathcal{L}_S^p$ and $\mathcal{L}_f$.
  for $l \in \{L_{p-1}+1, ..., L_p\}$ **do**
    Do **Channel Selection** for layer l using Algorithm 2.
  end for
end for

Algorithm 2 Greedy algorithm for channel selection

Input: Training data, model M, parameters κ_l, and ϵ.
Output: Selected channel subset $\mathcal{A}$ and model parameters $\mathbf{W}_{\mathcal{A}}$.
Initialize $\mathcal{A} \leftarrow \emptyset$, and $t = 0$.
while (stopping conditions are not achieved) **do**
  Compute gradients of $\mathcal{L}$ w.r.t. $\mathbf{W}$: $\mathbf{G} = \partial\mathcal{L}/\partial\mathbf{W}$.
  Find the channel $k = \arg\max_{j \notin \mathcal{A}} \{||\mathbf{G}_j||_F\}$.
  Let $\mathcal{A} \leftarrow \mathcal{A} \cup \{k\}$.
  Solve Problem (8) to update $\mathbf{W}_{\mathcal{A}}$.
  Let $t \leftarrow t + 1$.
end while

first fine-tune the pruned model by $\mathcal{L}_s^p$ and $\mathcal{L}_f$

then conduct the channel selection for each layer with $\mathcal{L}_s^p$ and $\mathcal{L}_M$

Structured pruning

Result: outperforms several state-of-the-art methods

Model		ThiNet [28]	CP [14]	Sliming [27]	WM	WM+	Random DCP	DCP	DCP-Adapt
VGGNet (Baseline 6.01%)	#Param. ↓	1.92×	1.92×	8.71×	1.92×	1.92×	1.92×	1.92×	**15.58**×
	#FLOPs ↓	2.00×	2.00×	2.04×	2.00×	2.00×	2.00×	2.00×	**2.86**×
	Err. gap (%)	+0.14	+0.32	+0.19	+0.38	+0.11	+0.14	**-0.17**	**-0.58**
ResNet-56 (Baseline 6.20%)	#Param. ↓	1.97×	-	-	1.97×	1.97×	1.97×	1.97×	**3.37**×
	#FLOPs ↓	1.99×	2×	-	1.99×	1.99×	1.99×	1.99×	1.89×
	Err. gap (%)	+0.82	+1.0	-	+0.56	+0.45	+0.63	**+0.31**	**-0.01**

on CIFAR-10, pruned VGGNet outperforms the pre-trained model by 0.58% in testing error, and obtains 15.58× reduction in model size

Model		ThiNet [28]	CP [14]	WM	WM+	DCP
ResNet-50	#Param. ↓	2.06×	-	2.06×	2.06×	2.06×
	#FLOPs ↓	2.25×	2×	2.25×	2.25×	2.25×
	Top-1 gap (%)	+1.87	-	+2.81	+2.41	**+1.06**
	Top-5 gap (%)	+1.12	+1.40	+1.62	+1.28	**+0.61**

on ILSVRC-12, pruned ResNet-50 with 30% reduction of channels outperforms the baseline model by 0.39% in top-1 accuracy

Structured pruning

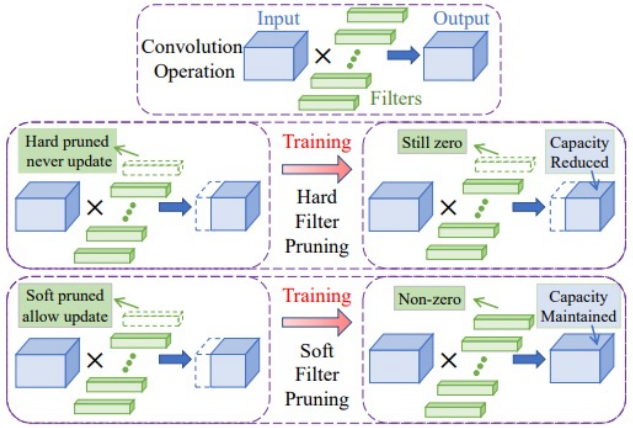
Soft Filter Pruning for Accelerating Deep Convolutional Neural Networks

Yang He[1,2], Guoliang Kang[2], Xuanyi Dong[2], Yanwei Fu[3*], Yi Yang[1,2*]
[1]SUSTech-UTS Joint Centre of CIS, Southern University of Science and Technology
[2]CAI, University of Technology Sydney
[3]The School of Data Science, Fudan University
{yang.he-1, guoliang.kang, xuanyi.dong}@student.uts.edu.au,
yanweifu@fudan.edu.cn, yi.yang@uts.edu.au

Structured pruning

Problem: most previous pruning methods suffer from the problems of the model capacity reduction and the dependence on pre-trained model

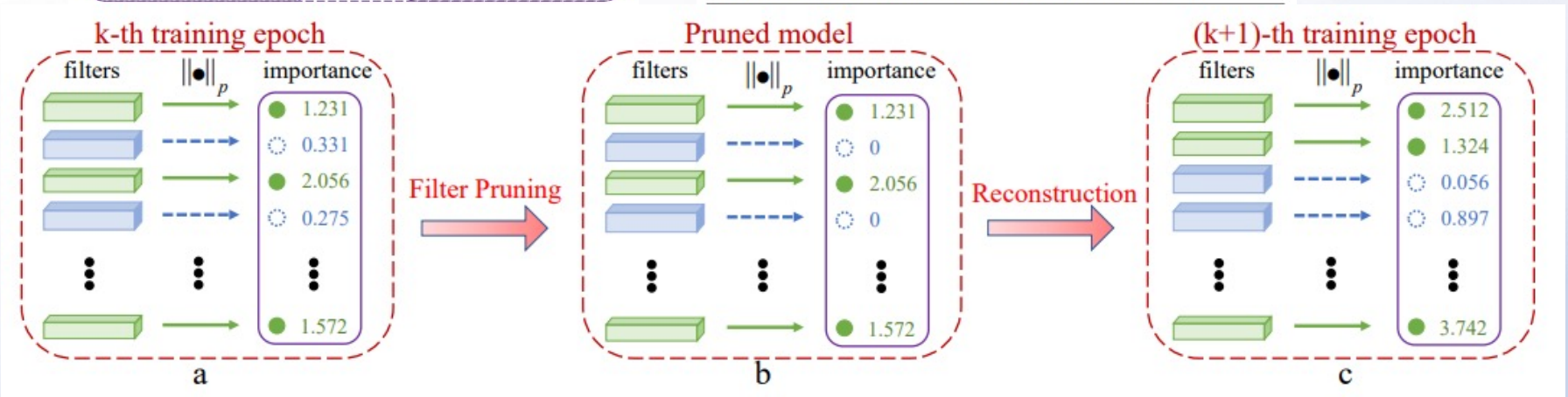
Method: dynamically prune the filters in a soft manner



Algorithm 1 Algorithm Description of SFP

Input: training data: $\mathbf{X}$, pruning rate: P_i
the model with parameters $\mathbf{W} = \{\mathbf{W}^{(i)}, 0 \le i \le L\}$.
Initialize the model parameter $\mathbf{W}$
for $epoch = 1$; $epoch \le epoch_{max}$; $epoch++$ **do**
  Update the model parameter $\mathbf{W}$ based on $\mathbf{X}$
  for $i = 1$; $i \le L$; $i++$ **do**
    Calculate the ℓ_2-norm for each filter $\|\mathcal{F}_{i,j}\|_2, 1 \le j \le N_{i+1}$
    Zeroize $N_{i+1}P_i$ filters by ℓ_2-norm filter selection
  end for
end for
Obtain the compact model with parameters $\mathbf{W}^*$ from $\mathbf{W}$
Output: The compact model and its parameters $\mathbf{W}^*$

- zeroize filters with small l2-norm
- use training data to update the pruned model
- repeat this process until converged
- finally prune the filters with small l2-norm without updating



Structured pruning

Result: advances the state-of-the-art

Depth	Method	Fine-tune?	Baseline Accu. (%)	Accelerated Accu. (%)	Accu. Drop (%)	FLOPs	Pruned FLOPs(%)
20	[Dong *et al.*, 2017a]	N	91.53	91.43	0.10	3.20E7	20.3
	Ours(10%)	N	**92.20 ± 0.18**	**92.24 ± 0.33**	**-0.04**	3.44E7	15.2
	Ours(20%)	N	**92.20 ± 0.18**	91.20 ± 0.30	1.00	2.87E7	29.3
	Ours(30%)	N	**92.20 ± 0.18**	90.83 ± 0.31	1.37	**2.43E7**	**42.2**
32	[Dong *et al.*, 2017a]	N	92.33	90.74	1.59	4.70E7	31.2
	Ours(10%)	N	**92.63 ± 0.70**	**93.22 ± 0.09**	**-0.59**	5.86E7	14.9
	Ours(20%)	N	**92.63 ± 0.70**	90.63 ± 0.37	0.00	4.90E7	28.8
	Ours(30%)	N	**92.63 ± 0.70**	90.08 ± 0.08	0.55	**4.03E7**	**41.5**
56	[Li *et al.*, 2017]	N	93.04	91.31	1.75	9.09E7	27.6
	[Li *et al.*, 2017]	Y	93.04	93.06	-0.02	9.09E7	27.6
	[He *et al.*, 2017]	N	92.80	90.90	1.90	-	50.0
	[He *et al.*, 2017]	Y	92.80	91.80	1.00	-	50.0
	Ours(10%)	N	**93.59 ± 0.58**	**93.89 ± 0.19**	**-0.30**	1.070E8	14.7
	Ours(20%)	N	**93.59 ± 0.58**	93.47 ± 0.24	0.12	8.98E7	28.4
	Ours(30%)	N	**93.59 ± 0.58**	93.10 ± 0.20	0.49	7.40E7	41.1
	Ours(30%)	Y	**93.59 ± 0.58**	93.78 ± 0.22	-0.19	7.40E7	41.1
	Ours(40%)	N	**93.59 ± 0.58**	92.26 ± 0.31	1.33	**5.94E7**	**52.6**
	Ours(40%)	Y	**93.59 ± 0.58**	93.35 ± 0.31	0.24	**5.94E7**	**52.6**
110	[Li *et al.*, 2017]	N	93.53	92.94	0.61	1.55E8	38.6
	[Li *et al.*, 2017]	Y	93.53	93.30	0.20	1.55E8	38.6
	[Dong *et al.*, 2017a]	N	93.63	93.44	0.19	-	34.2
	Ours(10%)	N	**93.68 ± 0.32**	93.83 ± 0.19	-0.15	2.16E8	14.6
	Ours(20%)	N	**93.68 ± 0.32**	**93.93 ± 0.41**	**-0.25**	1.82E8	28.2
	Ours(30%)	N	**93.68 ± 0.32**	93.38 ± 0.30	0.30	**1.50E8**	**40.8**
	Ours(30%)	Y	**93.68 ± 0.32**	93.86 ± 0.21	-0.18	**1.50E8**	**40.8**

accelerate the inference of ResNet-110 to 40.8% speed-up with only 0.30% accuracy drop without fine-tuning

Structured pruning

AMC: AutoML for Model Compression and Acceleration on Mobile Devices

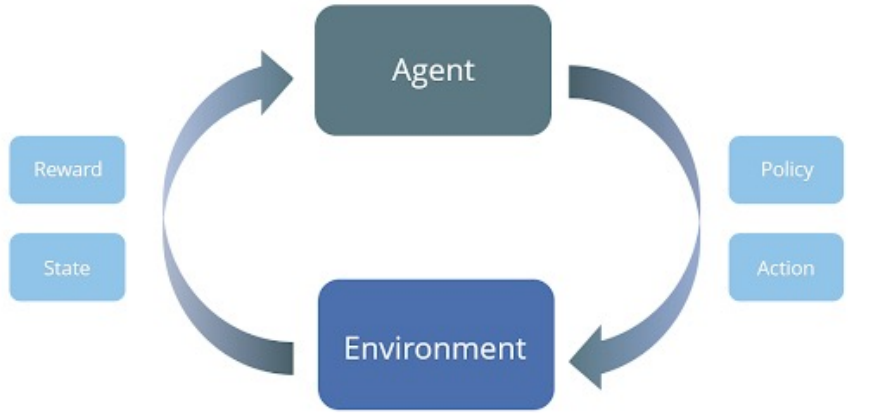
Yihui He[‡*], Ji Lin[†*], Zhijian Liu[†], Hanrui Wang[†], Li-Jia Li[↕], and Song Han[†]
{jilin, songhan}@mit.edu

[†]Massachusetts Institute of Technology
[‡]Carnegie Mellon University
[↕]Google

Structured pruning

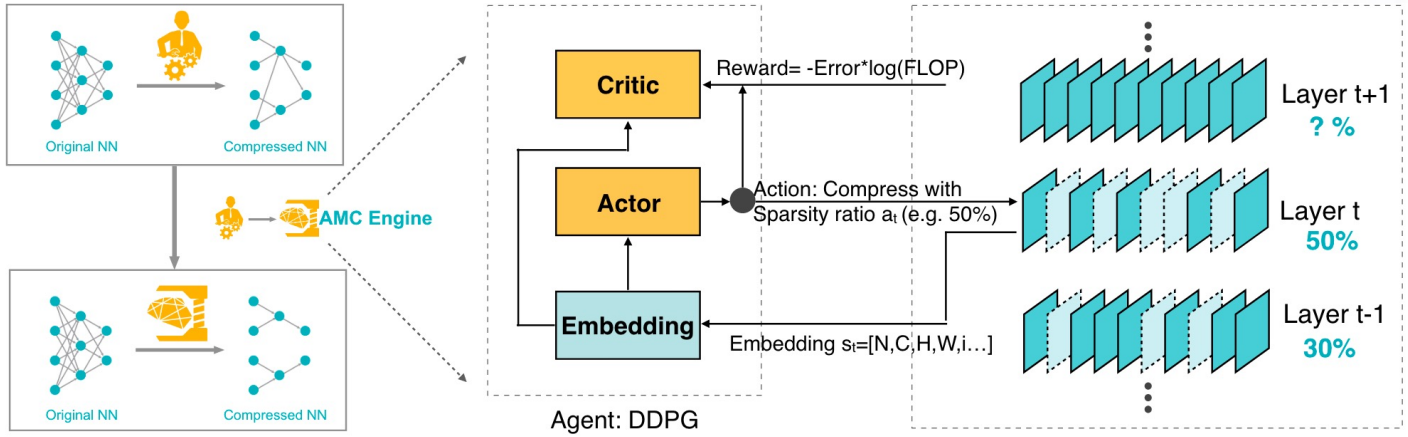
Problem: How to determine the sparsity ratio of every layer

Method: Use the reinforcement learning to **automatically learn** the sparsity ratio of every layer



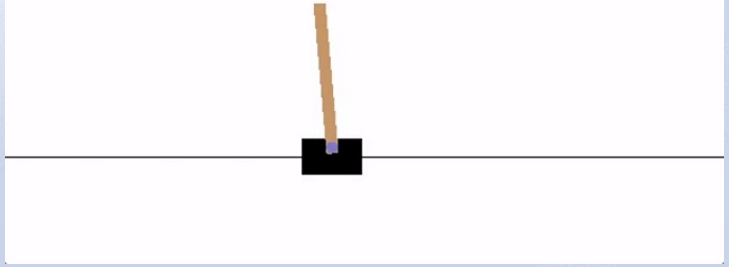
Introduce reinforcement learning

Filter-level pruning



Model Compression by AI:
Automated, Higher Compression Rate, Faster

How many filters should be removed?



Structured pruning

Result: Smaller model size, higher sparsity ratio

Table 2. Pruning policy comparison of Plain-20, ResNets [21] on CIFAR-10 [28]. R_{Err} corresponds to FLOPs-constrained compression with channel pruning, while R_{Param} corresponds to accuracy guaranteed compression with fine-grained pruning. For both shallow network Plain-20 and deeper network ResNets, AMC outperforms *hand-crafted* policies by a large margin. This enables efficient exploration without fine-tuning. Although AMC makes many trials on model architecture, we have separate validation and test dataset. No over-fitting is observed.

Model	Policy	Ratio	Val Acc.	Test Acc.	Acc. after FT.
Plain-20 (90.5%)	deep (handcraft)	50% FLOPs	79.6	79.2	88.3
	shallow (handcraft)		83.2	82.9	89.2
	uniform (handcraft)		84.0	83.9	89.7
	AMC (R_{Err})		**86.4**	**86.0**	**90.2**
ResNet-56 (92.8%)	uniform (handcraft)	50% FLOPs	87.5	87.4	89.8
	deep (handcraft)		88.4	88.4	91.5
	AMC (R_{Err})		**90.2**	**90.1**	**91.9**
ResNet-50 (93.53%)	**AMC (R_{Param})**	60% Params	**93.64**	**93.55**	-

50% compression for CIFAR10 with higher accuracy

Table 3. Learning-based model compression (AMC) outperforms rule-based model compression. Rule-based heuristics are suboptimal. (for reference, the baseline top-1 accuracy of VGG-16 is 70.5%, MobileNet is 70.6%, MobileNet-V2 is 71.8%).

	policy	FLOPs	ΔAcc %
VGG-16	FP (handcraft) [31]	20%	-14.6
	RNP (handcraft) [33]		-3.58
	SPP (handcraft) [49]		-2.3
	CP (handcraft) [22]		-1.7
	AMC (ours)		**-1.4**
MobileNet	uniform (0.75-224) [23]	56%	-2.5
	AMC (ours)	50%	**-0.4**
	uniform (0.75-192) [23]	41%	-3.7
	AMC (ours)	40%	**-1.7**
MobileNet-V2	uniform (0.75-224) [44]	70%	-2.0
	AMC (ours)		**-1.0**

30% sparsity ratio for ImageNet only 1% accuracy drop

目录

1 Introduction

2 Network pruning

3 Structured pruning

4 Unstructured pruning

Unstructured pruning

Post-training deep neural network pruning via layer-wise calibration

Ivan Lazarevich
Intel Corporation
ivan.lazarevich@intel.com

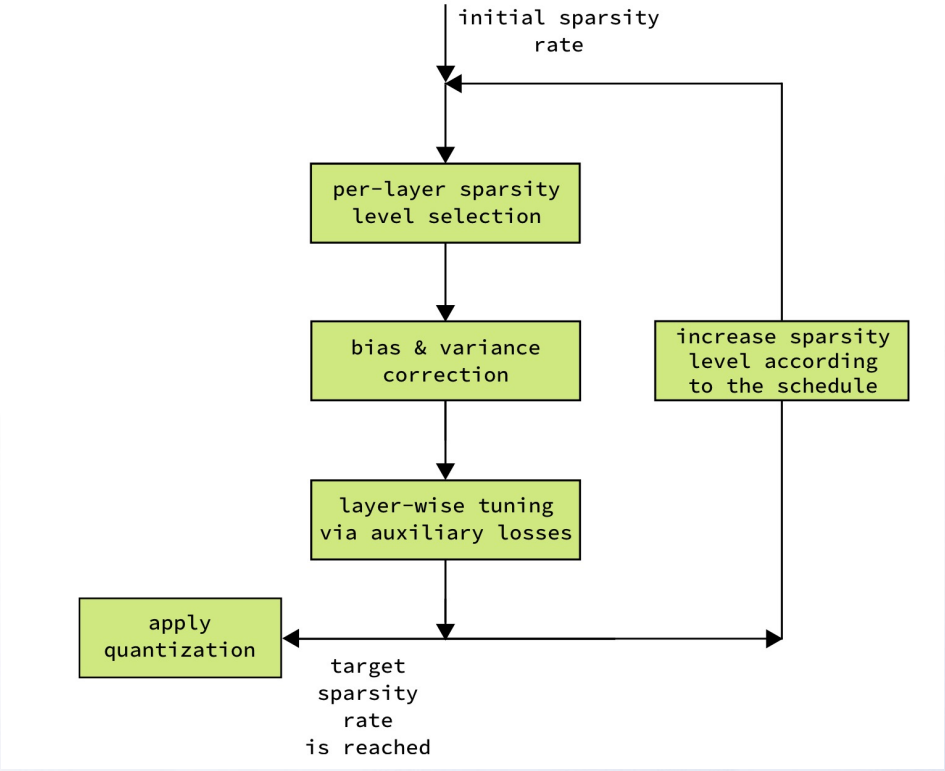
Alexander Kozlov
Intel Corporation
alexander.kozlov@intel.com

Nikita Malinin
Intel Corporation
nikita.malinin@intel.com

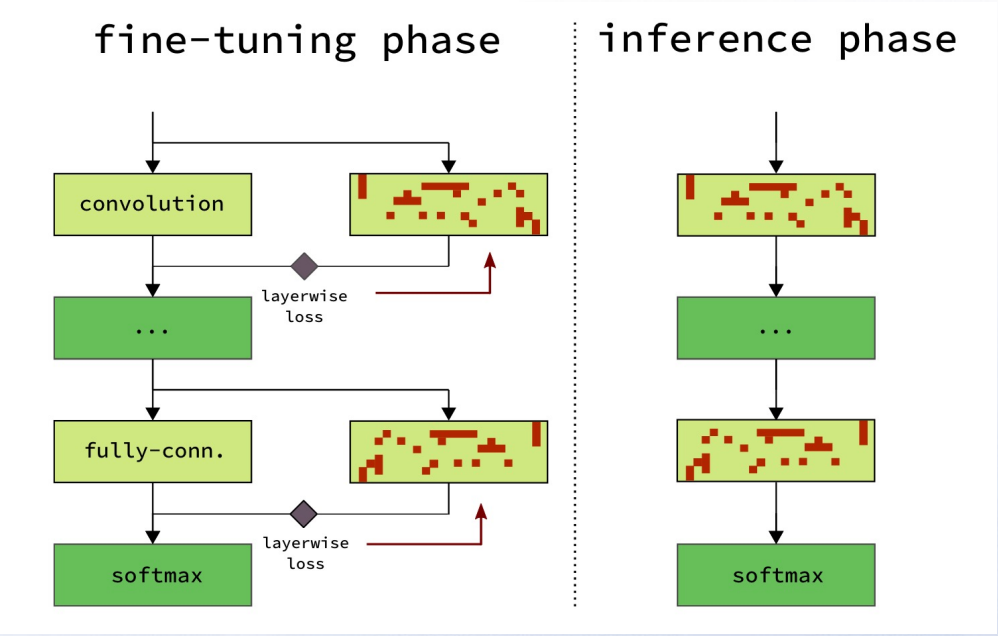
Unstructured pruning

Problem: massive data and computation burdens during model pruning

Method: use a few unlabeled data to calibrate



Post-training sparsity pipeline



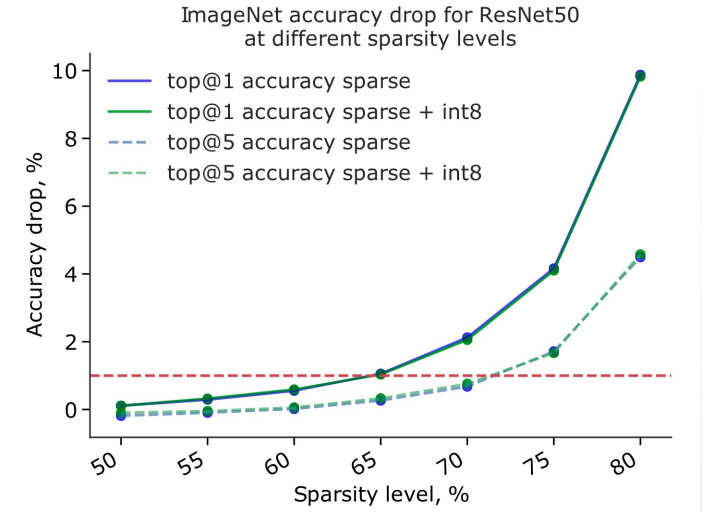
$$\mathrm{L} = \sum_{i \in batch} \left(Y^i_{dense} - f\left(W^S M^S, X^i_{dense} \right) - b^S \right)^2$$

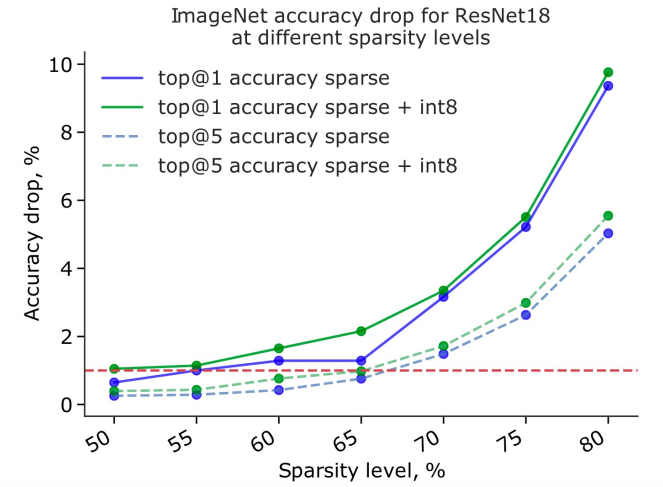
Post-training sparse model calibration

Unstructured pruning

Result: achieving high accuracy without dependence on training

Model	Dataset (acc. metric)	Sparsity rate, %	Compressed model acc.	Absolute acc. drop
ResNet50	ImageNet (top@1 acc.)	65	75.09	1.04
ResNet18	ImageNet (top@1 acc.)	50	68.93	0.81
GoogleNetV4	ImageNet (top@1 acc.)	50	78.96	0.94
MobileNetV2	ImageNet (top@1 acc.)	40	70.29	1.51
MobileNetV1-SSD	VOC07 (mAP)	50	71.53	0.98
TinyYOLOv2	COCO (AP)	50	28.29	0.83
NCF	MovieLens 20M (hit ratio)	70	64.67	0.93
BERT-base	MRPC (acc.)	50	82.50	0.63





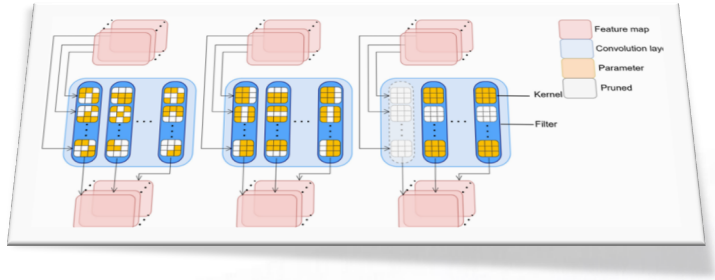
40%-70% sparsity ratio
nearly no performance drop

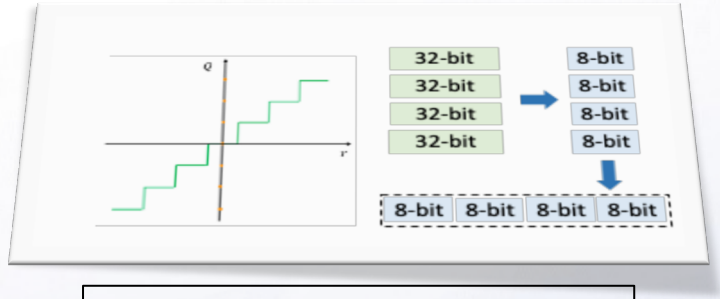
LLM稀疏量化联合压缩

- **问题：单独使用量化方法难以在极端压缩率下保持模型性能，需要考虑稀疏量化联合场景下，对异常值的耦合处理问题**

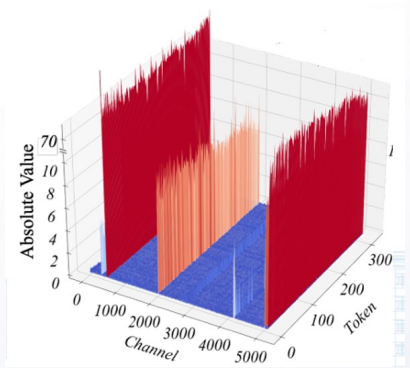
挑战1 稀疏化量化在大模型场景中耦合性差

稀疏倾向于保留绝对值更大的参数



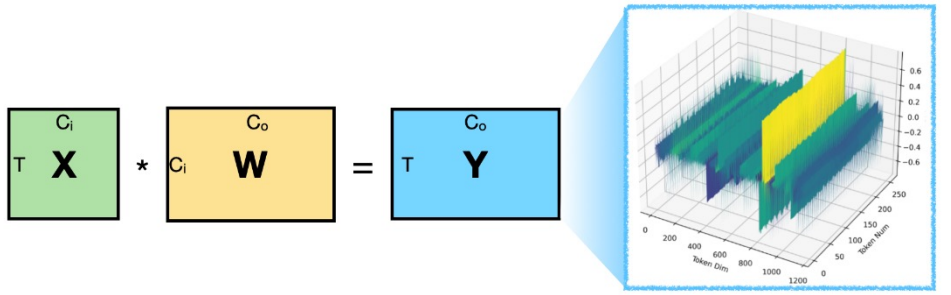


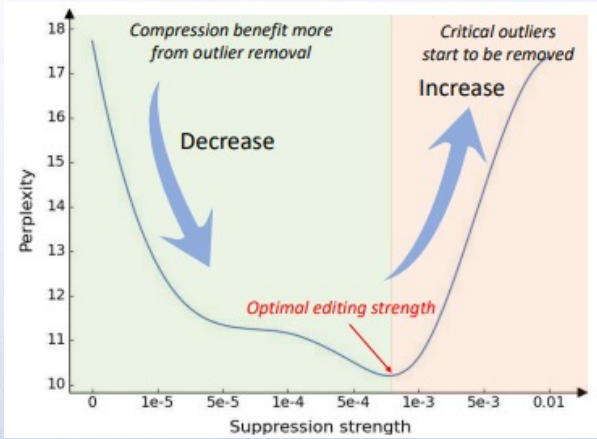
量化偏好变化范围更小的参数



?

挑战2 大模型中存在大量异常值





Guo et al., Compressing Large Language Models by Joint Sparsification and Quantization, ICML 2024

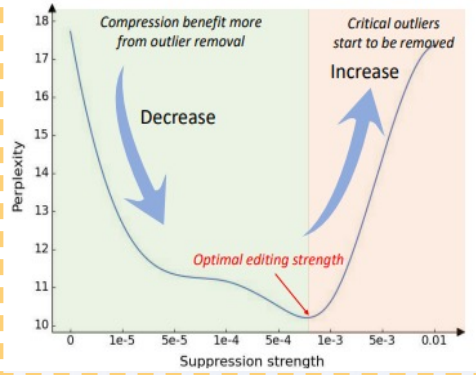
LLM稀疏量化联合压缩

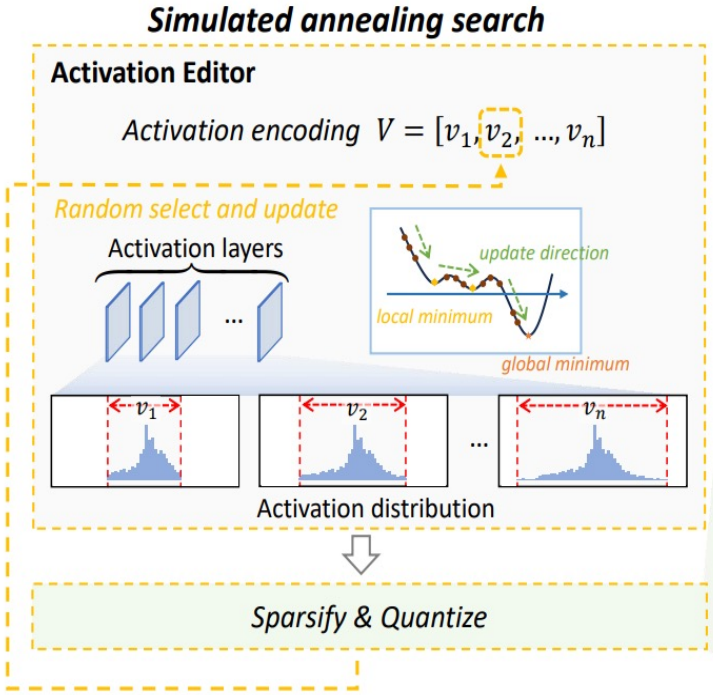
思路：提出联合使用稀疏化与量化，综合利用两种方法的优势

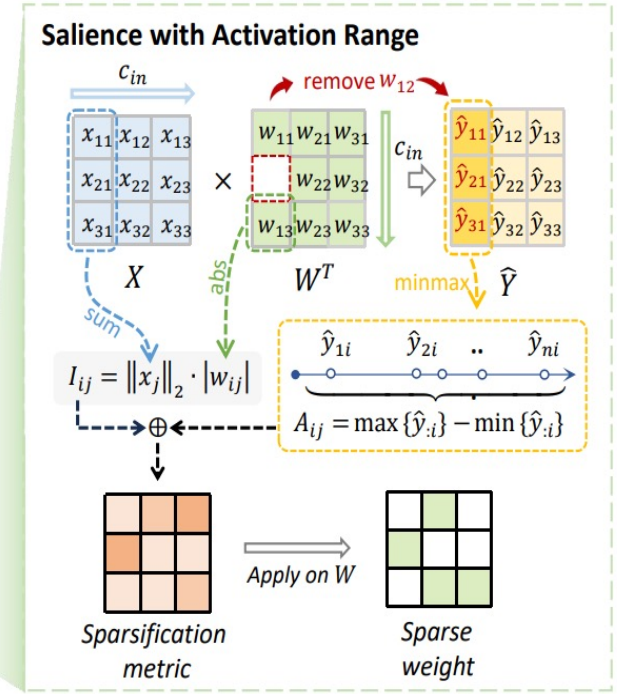
基于搜索的激活编辑器启发式的对异常值进行处理 ← **稀疏量化联合框架** → **激活范围显著性度量**能够平衡量化和剪枝权衡

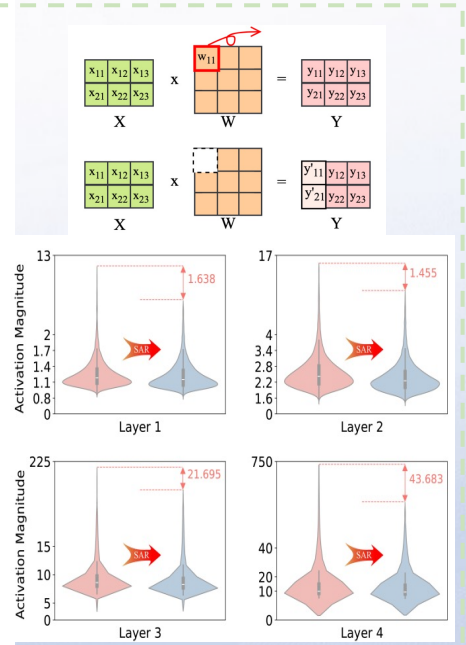
$$\mathbf{F}_{edit} = clamp(\mathbf{F}, LB, HB),$$
$$where \quad LB = r \cdot max(\mathbf{F}) + (1-r) \cdot min(\mathbf{F}),$$
$$and \quad HB = (1-r) \cdot max(\mathbf{F}) + r \cdot min(\mathbf{F}).$$

$$V^* = \arg\max_V \mathscr{P}(COM(LLM; V))$$









Guo et al., Compressing Large Language Models by Joint Sparsification and Quantization, ICML 2024

LLM稀疏量化联合压缩

- **在所有数据集和模型取得最好结果，带来33%推理加速**

零样本生成表现评估

方法	计算量（#MACs）	模型大小	WikiText↓	PIQA	BoolQ	MMLU	HellaSwag	Arc-e	Arc-c	WinoGrande	平均值
LLaMA-7B[6]	14.64T	12.6G	9.42	79.16	74.95	33.0	76.20	63.59	44.71	70.01	63.09
Wanda[12]	3.24T	1.8G	260364.00	50.98	48.41	22.9	26.46	26.85	26.88	50.28	36.11
LLM-Pruner[13]	3.24T	1.8G	2535837.61	49.35	46.91	23.5	26.26	26.77	28.92	49.96	35.95
SparseGPT[33]	3.24T	1.8G	5789.33	51.41	37.98	23.2	27.15	26.01	25.09	48.70	34.22
SmoothQuant[19]	3.24T	4.7G	12.70	76.93	72.60	28.4	71.56	70.16	41.72	66.06	61.06
OmniQuant[1]	3.24T	4.7G	12.49	77.24	73.27	28.5	71.33	69.10	42.11	68.10	61.38
Wanda+SmoothQuant	3.24T	3.5G	12.34	77.97	72.69	28.1	71.84	68.27	41.89	68.43	61.31
SparseGPT+SmoothQuant	3.24T	3.5G	11.98	77.20	73.24	30.2	70.91	68.48	41.21	69.30	61.51
JSQ (Ours)	3.24T	3.5G	11.04	78.72	74.58	30.0	72.33	69.30	43.42	69.58	62.56
LLaMA-13B[6]	26.52T	24.3G	8.21	80.14	77.89	42.1	79.09	74.75	47.70	72.77	67.78
Wanda[12]	4.09T	3.4G	172946.83	50.33	37.83	25.7	26.47	26.81	26.54	49.49	34.74
LLM-Pruner[13]	4.09T	3.4G	934523.91	49.67	38.10	25.0	25.52	25.97	29.10	49.96	34.76
SparseGPT[33]	4.09T	3.4G	1265.67	51.36	38.07	23.2	28.02	27.82	23.46	49.72	34.52
SmoothQuant[19]	4.09T	9.1G	11.65	77.80	70.98	31.9	76.97	68.52	41.72	69.61	62.50
OmniQuant[1]	4.09T	9.1G	10.27	78.80	75.43	36.4	75.31	70.11	43.63	70.32	64.29
Wanda+SmoothQuant	4.09T	6.8G	10.39	78.45	75.66	31.5	75.54	71.21	44.71	69.61	63.81
SparseGPT+SmoothQuant	4.09T	6.8G	10.49	78.40	76.61	32.8	75.45	69.53	44.11	67.96	63.55
JSQ (Ours)	4.09T	6.8G	9.58	80.05	78.16	38.2	75.41	72.93	46.67	72.24	66.24
LLaMA-33B[6]	33.12T	60.6G	6.24	82.21	82.72	43.1	82.62	78.91	52.90	75.77	71.18
Wanda[12]	4.77T	8.5G	313764.56	50.00	42.78	22.8	26.20	26.56	27.39	49.64	35.05
LLM-Pruner[13]	4.77T	8.5G	3533383.20	48.97	39.63	24.2	25.94	26.18	26.96	49.01	34.41
SparseGPT[33]	4.77T	8.5G	461.82	52.34	57.25	22.9	29.10	28.70	22.18	47.99	37.21
SmoothQuant[19]	4.77T	22.7G	11.05	77.04	76.24	36.8	76.86	72.56	50.17	72.45	66.02
OmniQuant[1]	4.77T	22.7G	9.41	79.05	78.20	45.1	78.54	73.37	50.82	75.22	68.61
Wanda+SmoothQuant	4.77T	17.0G	8.00	77.97	80.98	38.2	79.35	74.71	51.11	74.19	68.07
SparseGPT+SmoothQuant	4.77T	17.0G	8.05	77.75	82.78	43.7	79.20	74.37	50.43	74.66	68.98
JSQ (Ours)	4.77T	17.0G	7.68	79.20	81.92	49.3	80.76	76.35	51.92	77.51	70.27
LLaMA2-7B[7]	14.64T	12.6G	8.79	79.11	77.71	41.6	76.01	74.49	46.25	69.06	66.32
Wanda[12]	3.24T	1.8G	100584.54	48.97	37.83	25.5	26.31	26.26	26.96	49.96	34.54
SmoothQuant[19]	3.24T	4.7G	12.22	76.12	72.32	35.2	72.86	70.63	43.69	64.56	62.20
Wanda+SmoothQuant	3.24T	3.5G	10.74	78.35	75.44	34.2	72.91	69.53	43.17	67.17	62.97
JSQ (Ours)	3.24T	3.5G	10.64	79.09	76.10	34.6	73.28	71.23	45.56	69.03	64.13
LLaMA2-13B[7]	26.52T	24.3G	7.90	80.52	80.61	52.1	79.37	77.48	49.06	72.30	70.21
Wanda[12]	4.09T	3.4G	201702.58	49.29	37.83	22.9	25.81	27.06	26.79	51.70	34.48
SmoothQuant[19]	4.09T	9.1G	9.32	77.97	76.42	46.4	75.85	73.78	45.14	67.64	66.17
Wanda+SmoothQuant	4.09T	6.8G	9.79	78.54	78.91	46.7	76.64	73.23	47.61	70.09	67.38
JSQ (Ours)	4.09T	6.8G	8.58	79.25	79.61	48.5	76.32	75.51	46.25	71.84	68.18
ChatGLM3-6B[24]	12.24T	11.6G	10.12	81.45	86.45	62.1	78.01	79.59	53.58	72.61	73.40
Wanda[12]	2.20T	1.6G	49249.03	51.58	49.20	23.0	26.18	25.00	25.94	49.33	35.75
SmoothQuant[19]	2.20T	4.4G	15.23	73.12	75.54	48.7	51.30	60.82	36.60	53.83	57.13
Wanda+SmoothQuant	2.20T	3.3G	12.01	78.67	83.43	57.7	72.42	73.86	46.59	68.35	68.72
JSQ (Ours)	2.20T	3.3G	11.83	79.89	83.89	57.9	73.34	75.10	47.02	69.74	69.55

硬件加速性能

表 6 WikiText 上 2:4 和 4:8 半结构化稀疏性能

方法	计算量（#MACs）	模型大小	WikiText↓
LLaMA-7B	14.64T	12.6G	9.42
Unstructured	3.03T	3.2G	11.45
Structured 2:4	3.03T	3.2G	13.54
Structured 4:8	3.03T	3.2G	12.15
LLaMA2-7B	14.64T	12.6G	8.79
Unstructured	3.03T	3.2G	10.89
Structured 2:4	3.03T	3.2G	13.24
Structured 4:8	3.03T	3.2G	12.07

在相同计算量下，JSQ 框架在几乎所有数据集和模型上取得最好结果、更快推理速度（33%加速）

Guo et al., Compressing Large Language Models by Joint Sparsification and Quantization, ICML 2024

Thanks!