Structured Sparsification with Joint Optimization of Group Convolution and Channel Shuffle

Xin-Yu Zhang, Kai Zhao, Taihong Xiao, Ming-Ming Cheng, Ming-Hsuan Yang

xinyuzhang@mail.nankai.edu.cn

Overview

1. Background
   - Group Convolution and Channel Shuffle
   - Norm-based Filter Pruning
2. Methodology
   - Overview
   - Learning Connectivity with Linear Programming
   - Structured Sparsification
3. Experiments
4. More about Structured Sparsification

Group Convolution

Group convolution (GroupConv) is used for model compression.

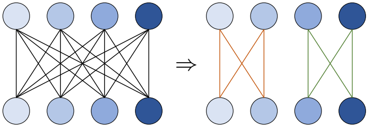


Figure: Vanilla conv → group conv.

Group Convolution

Group convolution (GroupConv) is used for model compression.

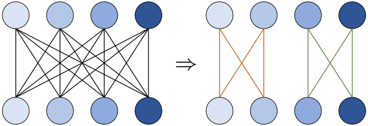


Figure: Vanilla conv → group conv.

Originally, conv3x3s ⇒ GroupConv3x3s (ResNeXts and MobileNets), and conv1x1s become bottleneck.

Group Convolution

Group convolution (GroupConv) is used for model compression.

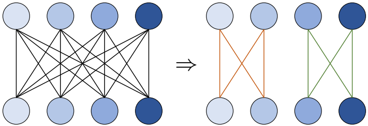


Figure: Vanilla conv → group conv.

Originally, conv3x3s ⇒ GroupConv3x3s (ResNeXts and MobileNets), and conv1x1s become bottleneck.

For conv1x1s ⇒ GroupConv1x1s, the inter-group communication between consecutive GroupConvs?

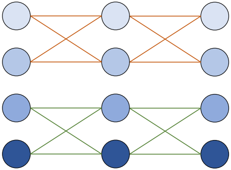


Figure: Consecutive group convs.

Channel Shuffle

ShuffleNet[1]: a *channel shuffle* operation (re-distribute channels from different groups).

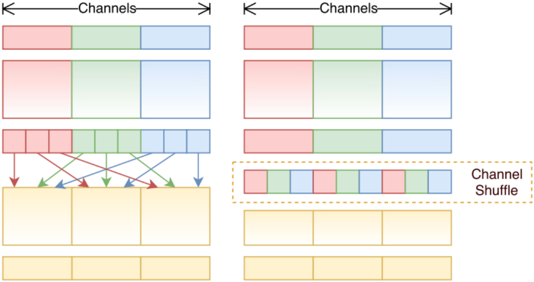


Figure: Channel shuffle in ShuffleNet.

[1] Ma *et al.*, ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices.

Channel Shuffle

ShuffleNet[1]: a *channel shuffle* operation (re-distribute channels from different groups).

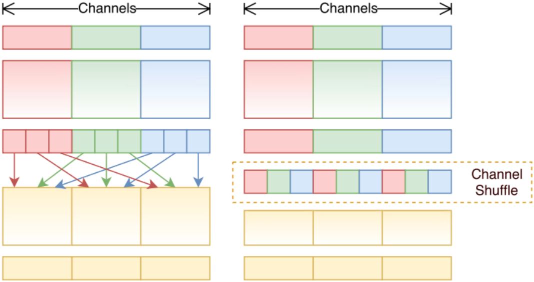


Figure: Channel shuffle in ShuffleNet.

But still a hand-crafted channel shuffle (uniformly distribute).

[1] Ma *et al.*, ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices.

Channel Shuffle

ShuffleNet[1]: a *channel shuffle* operation (re-distribute channels from different groups).

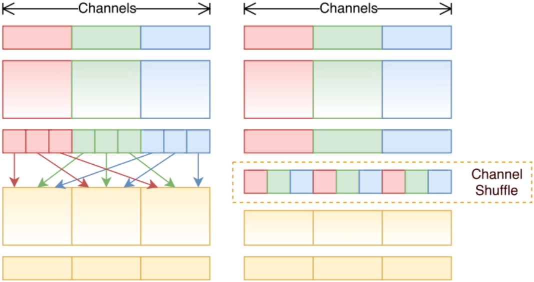


Figure: Channel shuffle in ShuffleNet.

But still a hand-crafted channel shuffle (uniformly distribute).

We propose a *learnable channel shuffle* mechanism which unifies the norm-based pruning criteria and the learning of channel permutation.

[1] Ma *et al.*, ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices.

Norm-based Filter Pruning

Filter Pruning: prune unimportant filters w/o performance degradation.

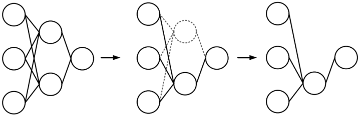


Figure: Filter pruning.

[2] Liu *et al.*, Learning Efficient Convolutional Networks through Network Slimming.

Norm-based Filter Pruning

Filter Pruning: prune unimportant filters w/o performance degradation.

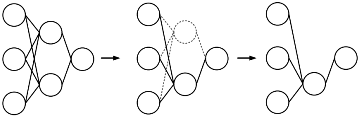


Figure: Filter pruning.

In particular, weight norm $\Rightarrow$ indicator of filter importance.

E.g., Network Slimming[2]: prune according to batch-norm scaling factor.

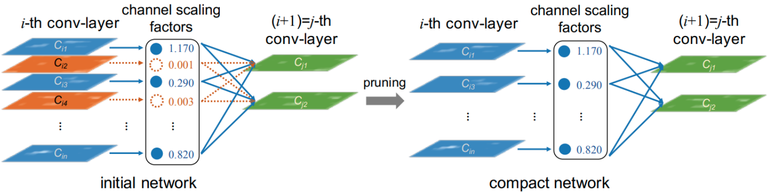


Figure: Network Slimming.

[2] Liu *et al.*, Learning Efficient Convolutional Networks through Network Slimming.

Filter Pruning: prune unimportant filters w/o performance degradation.

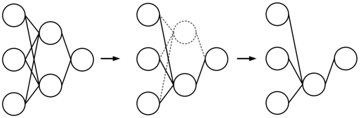


Figure: Filter pruning.

In particular, weight norm $\Rightarrow$ indicator of filter importance.
E.g., Network Slimming[2]: prune according to batch-norm scaling factor.

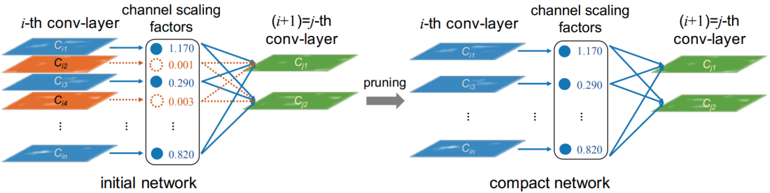


Figure: Network Slimming.

Besides, L_1 regularization (LASSO) $\rightarrow$ batch-norm scaling factors.

[2] Liu *et al.*, Learning Efficient Convolutional Networks through Network Slimming.

Norm-based Filter Pruning

However, problems of filter pruning:

(i) pruning has to deal with special network structures;

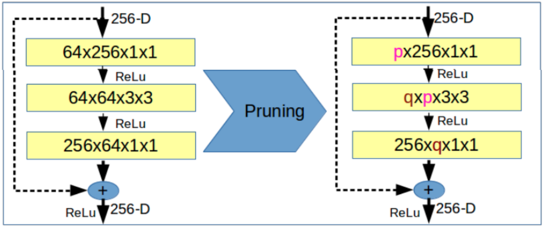


Figure: Pruning residual connection[3].

[3] Singh *et al.*. Play and Prune: Adaptive Filter Pruning for Deep Model Compression.

Norm-based Filter Pruning

However, problems of filter pruning:

(i) pruning has to deal with special network structures;

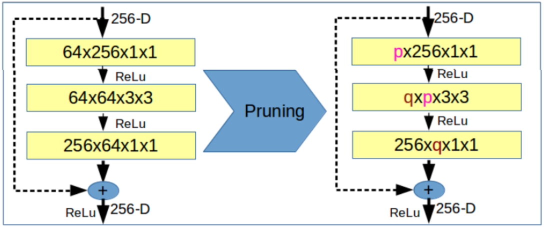


Figure: Pruning residual connection[3].

(ii) pruning cannot achieve a high compression rate w/o degradation;

[3] Singh *et al.*. Play and Prune: Adaptive Filter Pruning for Deep Model Compression.

Norm-based Filter Pruning

However, problems of filter pruning:

(i) pruning has to deal with special network structures;

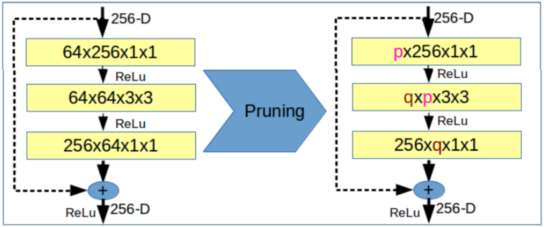


Figure: Pruning residual connection[3].

(ii) pruning cannot achieve a high compression rate w/o degradation;

In this work, we generalize the norm-based pruning criteria to the problem of converting vanilla convolutions into GroupConvs.

[3] Singh *et al.*. Play and Prune: Adaptive Filter Pruning for Deep Model Compression.

Overview

As an alternative to filter pruning, we compress the model by learning a group structure and a channel shuffle pattern jointly.

Overview

As an alternative to filter pruning, we compress the model by learning a group structure and a channel shuffle pattern jointly.

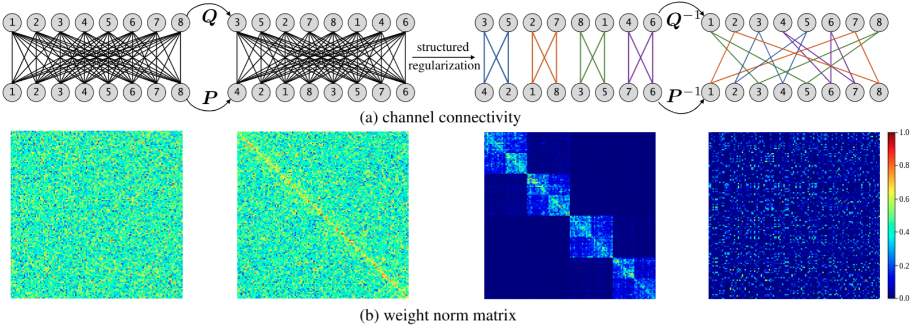


Figure: Overview of the proposed *structured sparsification*.

Overview

As an alternative to filter pruning, we compress the model by learning a group structure and a channel shuffle pattern jointly.

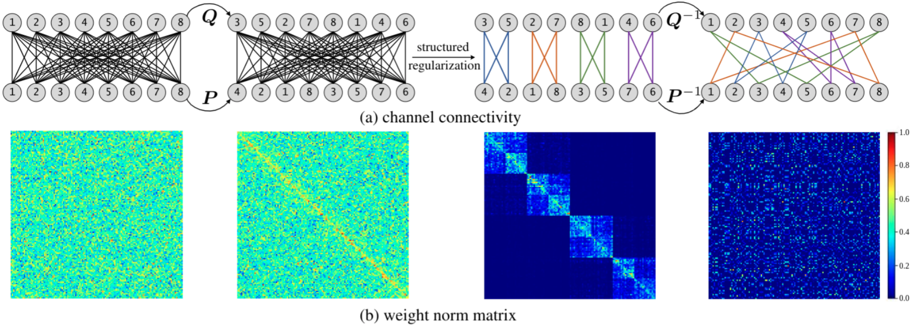


Figure: Overview of the proposed *structured sparsification*.

Challenges.

(i) How to define a suitable channel shuffle? (under what criteria?)

As an alternative to filter pruning, we compress the model by learning a group structure and a channel shuffle pattern jointly.

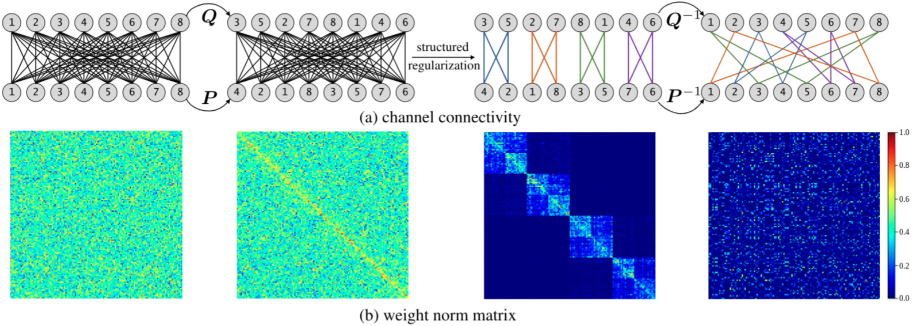


Figure: Overview of the proposed *structured sparsification*.

Challenges.

(i) How to define a suitable channel shuffle? (under what criteria?)

(ii) How to *structurally* sparsify the convolutional weights?

Learning Connectivity — Formulation

In general,

$$\text{weight norm matrix of GroupConv} \Rightarrow \text{block-diagonal matrix;}$$
$$\text{channel shuffle} \Rightarrow \text{row/column permutation of weight norm matrix.}$$

Learning Connectivity — Formulation

In general,

$$\text{weight norm matrix of GroupConv} \Rightarrow \text{block-diagonal matrix;}$$
$$\text{channel shuffle} \Rightarrow \text{row/column permutation of weight norm matrix.}$$

In practice, weight norm matrix $\rightarrow$ block-diagonal only by channel shuffle? Not impossible!

Learning Connectivity — Formulation

In general,

weight norm matrix of GroupConv $\Rightarrow$ block-diagonal matrix;
channel shuffle $\Rightarrow$ row/column permutation of weight norm matrix.

In practice, weight norm matrix $\rightarrow$ block-diagonal only by channel shuffle? Not impossible!

Therefore, aim of channel shuffle: permute weight norm matrix to make it "*as block-diagonal as possible*". Formally,

$$\begin{aligned} &\min_{\boldsymbol{P},\boldsymbol{Q}} \ \boldsymbol{PSQ} \otimes \boldsymbol{R} \\ &\text{s.t. } \boldsymbol{P} \in \mathcal{P}^{C^{out}} \ \textit{and} \ \ \boldsymbol{Q} \in \mathcal{P}^{C^{in}}, \end{aligned} \tag{1}$$

where $\boldsymbol{S} \in \mathbb{R}^{C^{out} \times C^{in}}$ is the weight norm matrix, $\boldsymbol{R}$ is a cost matrix, and $\mathcal{P}^N$ is the set of $N \times N$ permutation matrices.

Learning Connectivity — Algorithm

NP-hard problem? Two relaxations:

(a) alternative update of $\boldsymbol{P}$ and $\boldsymbol{Q}$ (*coordinate descent*);

Learning Connectivity — Algorithm

NP-hard problem? Two relaxations:

(a) alternative update of $\boldsymbol{P}$ and $\boldsymbol{Q}$ (*coordinate descent*);

(b) feasible region $\mathcal{P}^N \to$ convex hull, *i.e., Birkhoff polytope*:

$$\mathcal{B}^N = \{\boldsymbol{X} \in \mathbb{R}_+^{N \times N} : \boldsymbol{X}1_N = 1_N,\ \boldsymbol{X}^\top 1_N = 1_N\}. \tag{2}$$

Learning Connectivity — Algorithm

NP-hard problem? Two relaxations:

(a) alternative update of $\boldsymbol{P}$ and $\boldsymbol{Q}$ (*coordinate descent*);

(b) feasible region $\mathcal{P}^N \rightarrow$ convex hull, *i.e., Birkhoff polytope*:

$$\mathcal{B}^N = \{\boldsymbol{X} \in \mathbb{R}_+^{N\times N} : \boldsymbol{X}1_N = 1_N,\ \boldsymbol{X}^\top 1_N = 1_N\}. \tag{2}$$

When updating $\boldsymbol{P}$,

$$\begin{aligned} \min_{\boldsymbol{P}} \ & \boldsymbol{P} \otimes \boldsymbol{R}\boldsymbol{Q}^\top \boldsymbol{S}^\top \\ \text{s.t. } & \boldsymbol{P} \in \mathcal{B}^{C^{out}}. \end{aligned} \tag{3}$$

NP-hard problem? Two relaxations:

(a) alternative update of $\boldsymbol{P}$ and $\boldsymbol{Q}$ (*coordinate descent*);

(b) feasible region $\mathcal{P}^N \rightarrow$ convex hull, *i.e., Birkhoff polytope*:

$$\mathcal{B}^N = \{\boldsymbol{X} \in \mathbb{R}_+^{N\times N} : \boldsymbol{X}1_N = 1_N,\ \boldsymbol{X}^\top 1_N = 1_N\}. \tag{2}$$

When updating $\boldsymbol{P}$,

$$\begin{aligned} &\min_{\boldsymbol{P}}\ \boldsymbol{P} \otimes \boldsymbol{R}\boldsymbol{Q}^\top \boldsymbol{S}^\top \\ &\text{s.t. } \boldsymbol{P} \in \mathcal{B}^{C^{out}}. \end{aligned} \tag{3}$$

In (3), the objective function is linear in $\boldsymbol{P}$ and the feasible region $\mathcal{B}^N$ is a simplex. Therefore, linear programming (LP), solved by network simplex method.

Learning Connectivity — Discussion

By LP theory, one solution of a LP problem $\rightarrow$ vertex of feasible region.

[4] Birkhoff, Three Observations on Linear Algebra.

By LP theory, one solution of a LP problem $\rightarrow$ vertex of feasible region.

By Birkhoff-von Neumann theorem[4], vertices of Birkhoff polytope $\rightarrow$ permutation matrices.

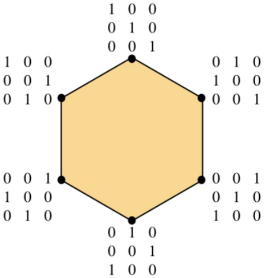


Figure: Vertices of Birkhoff polytope.

[4] Birkhoff, Three Observations on Linear Algebra.

Learning Connectivity — Discussion

By LP theory, one solution of a LP problem $\rightarrow$ vertex of feasible region.

By Birkhoff-von Neumann theorem[4], vertices of Birkhoff polytope $\rightarrow$ permutation matrices.

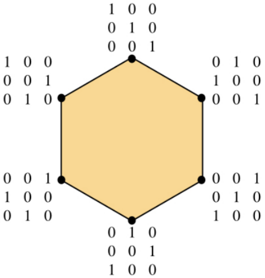


Figure: Vertices of Birkhoff polytope.

Therefore, relaxed feasible region $\mathcal{B}^N$ naturally reduced to $\mathcal{P}^N$.

[4] Birkhoff, Three Observations on Linear Algebra.

Structured Sparsification

Despite channel shuffle, the group structure cannot be formed naturally. Therefore, still need *structured regularization* of certain form.

Structured Sparsification

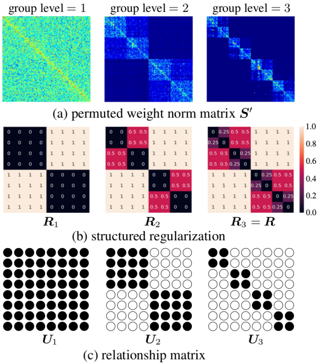
Despite channel shuffle, the group structure cannot be formed naturally. Therefore, still need *structured regularization* of certain form.

Structured L_1 regularization.

$$\mathcal{L}_{\text{reg}} = \boldsymbol{S}' \otimes \boldsymbol{R}_g, \tag{4}$$

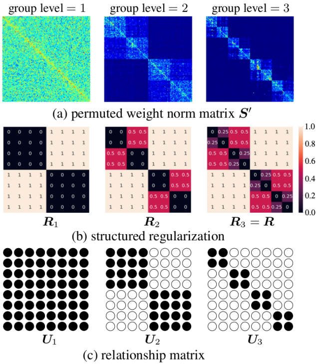
where $\boldsymbol{S}' = \boldsymbol{PSQ}$ permuted weight norm matrix, and $\boldsymbol{R}_g$ shown on the right.

Highlights: (a) LASSO, (b) hierarchical penalty.



Structured Sparsification

Despite channel shuffle, the group structure cannot be formed naturally. Therefore, still need *structured regularization* of certain form.



Structured L_1 regularization.

$$\mathcal{L}_{\text{reg}} = \boldsymbol{S}' \otimes \boldsymbol{R}_g, \qquad (4)$$

where $\boldsymbol{S}' = \boldsymbol{PSQ}$ permuted weight norm matrix, and $\boldsymbol{R}_g$ shown on the right.

Highlights: (a) LASSO, (b) hierarchical penalty.

Grouping Criteria.

$$g = \max\{g : \boldsymbol{S}' \otimes \boldsymbol{U}_g \geq p \sum_{i,j} S_{i,j},\ g = 1, 2, \cdots\}, \qquad (5)$$

where $\boldsymbol{U}_g$ is the relationship matrix.

Training Pipeline

Algorithm 1 Training Pipeline.

```
Initially update P and Q.
for t := 1 to #epochs do
    Train with structured regularization;
    Update P and Q;
    Determine the current group level g by the grouping criteria;
    Update the structured sparsification matrices (R_g);
    Adjust regularization coefficient (refer to paper).
end for
```

Experiments

Performance on ImageNet against two prior works, *i.e.*, Slimming[5] and Taylor[6] (refer to paper for full comparison).

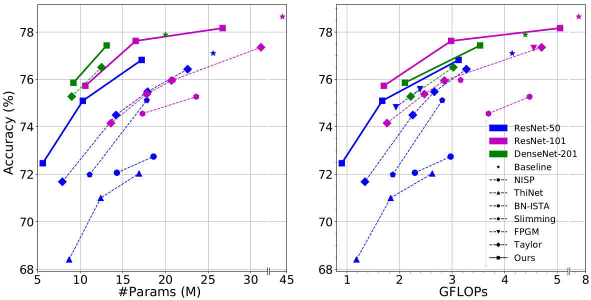
Methods	#Params.(10^6) ↓	GFLOPs ↓	Acc.(%) ↑
	ResNet-50		
Baseline	25.6	4.14	77.10
Slimming-20%	17.8	2.81	75.12
Taylor-19%	17.9	2.66	75.48
StrucSpars-35%	**17.2**	3.12	**76.82**
Taylor-28%	14.2	2.25	74.50
StrucSpars-65%	**10.3**	**1.67**	**75.10**
Taylor-44%	7.9	1.34	71.69
Slimming-50%	11.1	1.87	71.99
StrucSpars-85%	**5.6**	**0.90**	**72.47**
	ResNet-101		
Baseline	44.5	7.87	78.64
Taylor-25%	31.2	4.70	77.35
StrucSpars-40%	**26.7**	5.05	**78.16**
Taylor-45%	20.7	**2.85**	75.95
Slimming-50%	20.9	3.16	75.97
StrucSpars-65%	**16.5**	2.98	**77.62**
Taylor-60%	13.6	1.76	74.16
StrucSpars-80%	**10.6**	**1.70**	**75.73**
	DenseNet-201		
Baseline	20.0	4.39	77.88
Taylor-40%	**12.5**	**3.02**	76.51
StrucSpars-38%	13.1	3.53	**77.43**
Taylor-64%	**9.0**	2.21	75.28
StrucSpars-60%	9.2	**2.10**	**75.86**

[5] Liu *et al.*, Learning Efficient Convolutional Networks through Network Slimming.

[6] Molchanov *et al.*, Importance Estimation for Neural Network Pruning.

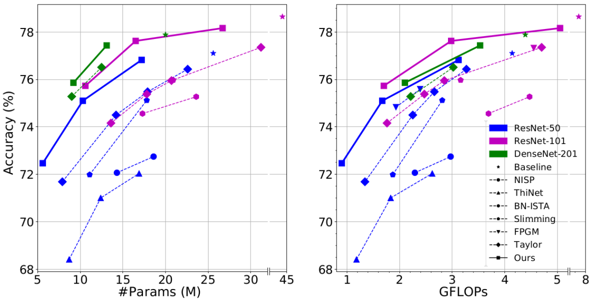
Ablation Studies

Accuracy vs.Complexity.



Ablation Studies

Accuracy vs.Complexity.



Wall-time acceleration.

Model	GFLOPs	Avg. Runtime (ms)	FPS
ResNet-50	4.14	80.2	12.4
StrucSpars-35%	3.12	68.2	14.7
StrucSpars-65%	1.67	61.3	16.3
StrucSpars-85%	0.90	53.5	18.7

Ablation Studies

Channel shuffle mechanism. We empirically compare the following five settings:

(i) FINETUNE: train → compress → finetune pipeline;

(ii) FROMSCRATCH: learned channel shuffle, but train from scratch;

(iii) SHUFFLENET: hand-crafted channel shuffle as in ShuffleNet;

(iv) RANDOM: random channel shuffle (*i.e.*, random permutation);

(v) NOSHUFFLE: no channel shuffle.

Config.	**ResNet-50**-65%		**ResNet-101**-65%	
Acc.	Top-1	Top-5	Top-1	Top-5
FINETUNE	75.10	92.52	77.62	93.72
FROMSCRATCH	75.02	92.46	77.14	93.53
SHUFFLENET	74.97	92.41	76.91	93.38
RANDOM	69.45	89.45	73.16	91.44
NOSHUFFLE	73.30	91.39	75.31	92.64

Further information

Refer to our paper[7] for limitations and future perspectives:

(i) **Data-Driven Structured Sparsification**;

(ii) **Progressive Sparsification Solution**;

(iii) **Combination with Filter Pruning**.

The source codes are available:
`https://github.com/Sakura03/StrucSpars`.

[7] `https://arxiv.org/abs/2002.08127`

Thanks!