Structured Sparsification with Joint Optimization of Group Convolution and Channel Shuffle

Xin-Yu Zhang[1], Kai Zhao[1], Taihong Xiao[2], Ming-Ming Cheng[1], Ming-Hsuan Yang[2]

[1]Nankai University, [2]University of California, Merced

arXiv: https://arxiv.org/abs/2002.08127

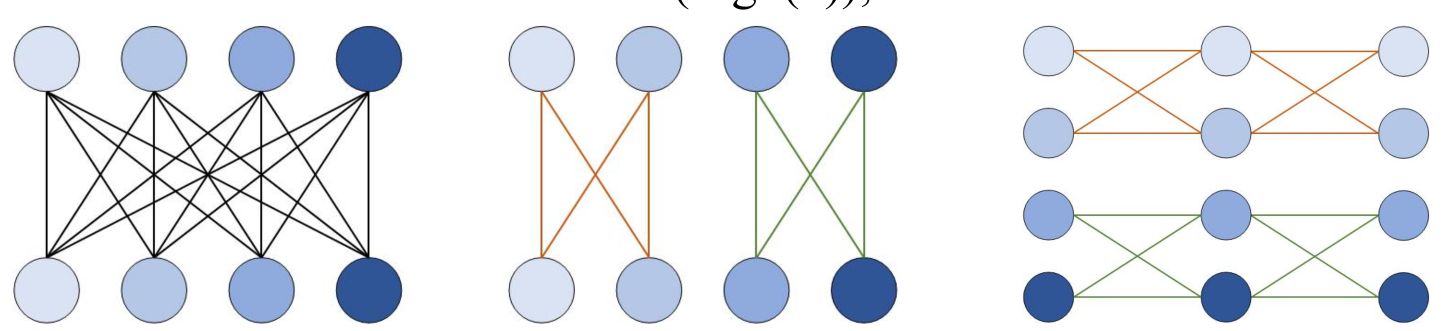
Github: https://github.com/Sakura03/StrucSpars

Email: xinyuzhang@mail.nankai.edu.cn

Background

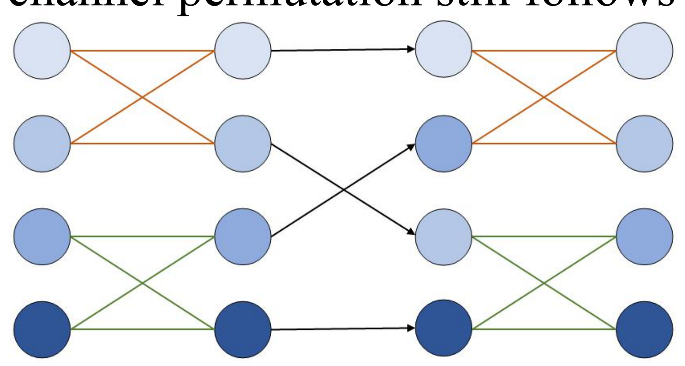
Group convolution (GroupConv) for model compression.

- GroupConv has less parameters and lower complexity (Fig. (a));
- However, if multiple GroupConvs are stacked sequentially, the inter-group communication will be eliminated (Fig. (b));



(a) Vanilla Conv → GroupConv. (b) Consecutive GroupConvs.

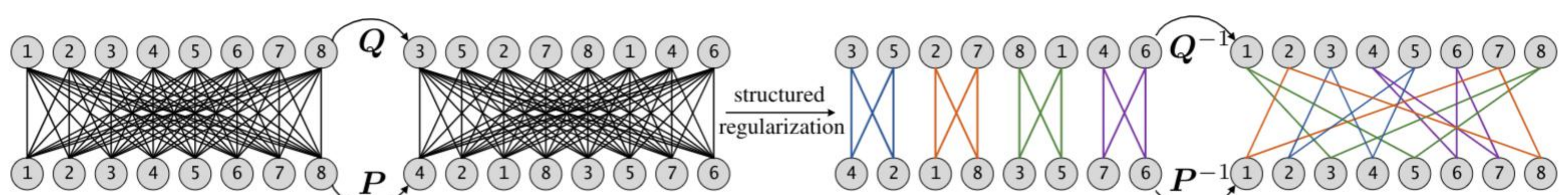
- ShuffleNet introduces a *channel shuffle* operation to re-arrange channels for each group, but the channel permutation still follows a pre-defined scheme (Fig. (c));



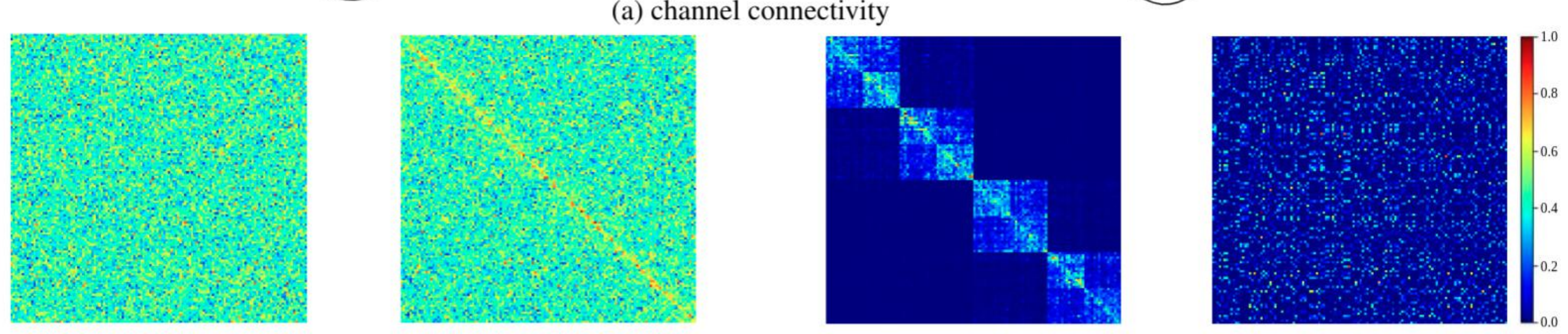
(c) Channel shuffle facilitates inter-group data flow.

- This work proposes a *learnable channel shuffle* mechanism which unifies the norm-based criteria and the learning of channel permutation.

Overview



(a) channel connectivity



(b) weight norm matrix

a) Learn the permutation matrices ($\boldsymbol{P}$, $\boldsymbol{Q}$) and use them to perform channel shuffle;

b) Structurally regularize the convolutional weights to induce a group structure;

c) Remove the sparsified weights and form the GroupConvs;

d) Shuffle back to the original ordering and obtain a *structurally sparse representation of the convolutional weights.*

Chanllenges

① Under what criteria to learn the channel shuffle (⇒ linear programming);

② How to induce a group structure on the convolutional weights (⇒ structured regularization).

Learning of Connectivity

Observations.

- Weight norm matrix of GroupConv → *block-diagonal* matrix;
- Channel shuffle → *row/column permutation* of weight norm matrix.

Formulation.

- Since weight norm matrix cannot be permuted to be an exact block-diagonal one, the aim of channel shuffle is to make it "*as block-diagonal as possible*":

$$\min_{\boldsymbol{P},\boldsymbol{Q}} \boldsymbol{PSQ} \otimes \boldsymbol{R} \quad \text{s.t.} \ \ \boldsymbol{P} \in \mathcal{P}^{C_{\text{out}}} \ \text{ and } \ \boldsymbol{Q} \in \mathcal{P}^{C_{\text{in}}}, \tag{1}$$

where $\boldsymbol{S}$ is the weight norm matrix, *i.e.*, $S_{j,i} = \left\|\boldsymbol{W}_{j,i,:,:}\right\|$, $\boldsymbol{R}$ is a cost matrix, $\mathcal{P}^N$ denotes the set of $N \times N$ permutation matrices, and $\otimes$ denotes the operator of element-wise multiplication and summation over all entries.

Solution.

- Since problem (1) is NP-hard, we adopt two techniques to make it solvable:
  a) the set $\mathcal{P}^N$ of permutation matrices is relaxed to its convex hull --- the set of *doubly-stochastic matrices*, *i.e.*, the *Birkhoff polytope*:
  $$\mathcal{B}^N = \left\{\boldsymbol{X} \in \mathbb{R}_+^{N\times N} : \boldsymbol{X}\,\mathbf{1}_N = \mathbf{1}_N,\ \boldsymbol{X}^T\mathbf{1}_N = \mathbf{1}_N\right\};$$
  b) the variables $\boldsymbol{P}$ and $\boldsymbol{Q}$ in (1) are updated alternatively, namely, the *coordinate descent* algorithm is used;
- With the two tricks, problem (1) is adapted as (when updating $\boldsymbol{P}$):

$$\min_{\boldsymbol{P}} \boldsymbol{P} \otimes \boldsymbol{RQ}^T\boldsymbol{S}^T \quad \text{s.t.} \ \ \boldsymbol{P} \in \mathcal{B}^{C_{\text{out}}}; \tag{2}$$

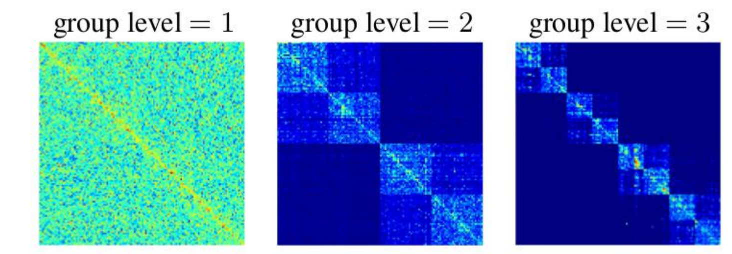
- In (2), the objective function is linear in $\boldsymbol{P}$, and $\mathcal{B}^N$ is a *simplex*, so (2) is essentially a *linear program*, which can be solved by the *network simplex* method.

Discussion.

- By linear programming theory and Birkhoff's theorem, at least one solution of (2) is exactly a permutation matrix;
- Therefore, feasible region $\mathcal{B}^N$ is naturally reduced to $\mathcal{P}^N$.

Structured Regularization

- Despite channel shuffle, the group structure cannot be formed spontaneously.



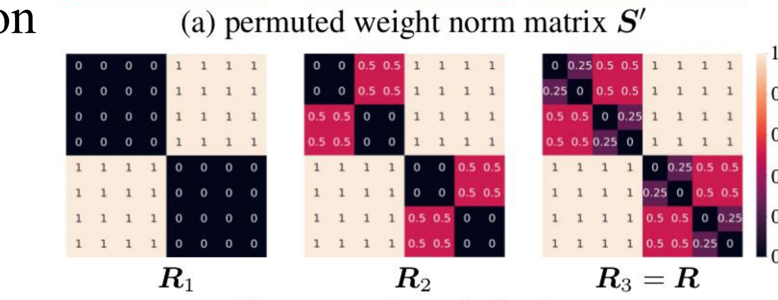
(a) permuted weight norm matrix S'

Structured Regularization.

- We impose *structured L_1 regularization* on the permuted weight norm matrix:

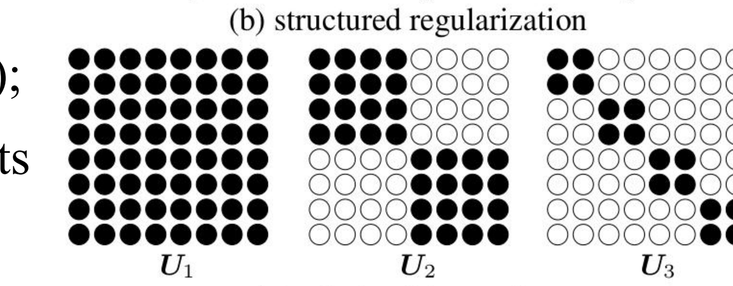
$$\mathcal{L}_{\text{reg}} = \boldsymbol{S}' \otimes \boldsymbol{R}_g,$$

where $\boldsymbol{S}' = \boldsymbol{PSQ}$ is the permuted weight norm matrix (Fig. (a)), $\boldsymbol{R}_g$ denotes the structured regularization matrix (Fig. (b));



(b) structured regularization

- Refer to our paper for details of $\boldsymbol{R}_g$ and its relationship to the cost matrix $\boldsymbol{R}$.



(c) relationship matrix

Grouping Criteria.

- The grouping criteria is as follows:

$$g = \max\left\{\, g : \boldsymbol{S}' \otimes \boldsymbol{U}_g \geq p\, \boldsymbol{S}' \otimes \boldsymbol{U}_1, g = 1,\, 2, \cdots\right\},$$

where $\boldsymbol{U}_g$ is the relationship matrix (Fig. (c)).

Future Perspective

- **Data-Driven Structured Sparsification.**

  Can the weight norm fully represent weight importance to accuracy? How about making the sparsification process be guided by the data loss? We suggest *optimization-based meta-learning* techniques.

- **Progressive Sparsification.**

  Spasified weights are removed progressively during training, leading to *finetune-free* training pipeline.

- **Combination with Filter Pruning.**

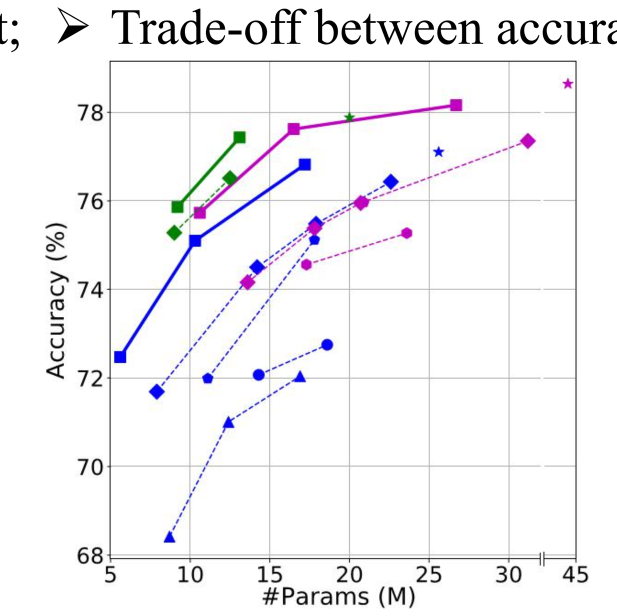
  Filter pruning is beneficial within a strict memory budget. In order to jointly prune filters and learn a group structure, we suggest *group sparsity constraints*.

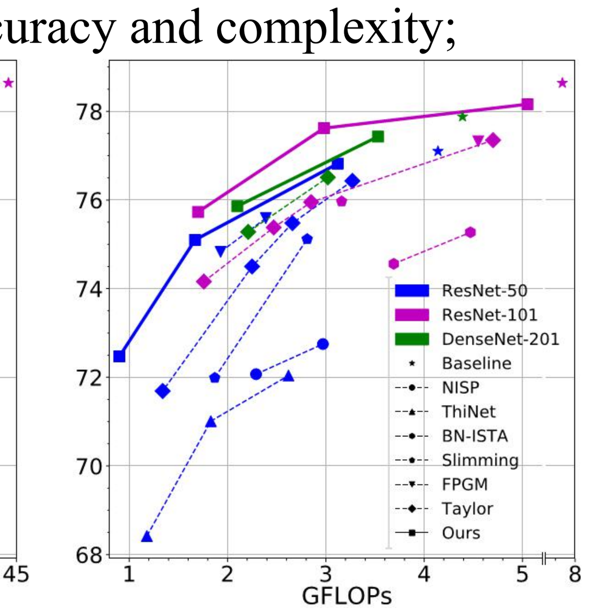
Experiments

- Comparison with SOTA pruning methods on ImageNet;

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**

- Trade-off between accuracy and complexity;





- Impact of channel shuffle mechanism.

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