电子科技大学
University of Electronic Science and Technology of China

An Efficient Network Architecture: Before and After Training

Wei Han

Data Mining Lab,
Big Data Research Center, UESTC
Email：weihan@std.uestc.edu.cn

Outline

After Training

- Magnitude based
- Activation based
- Reconstruction based
- Influence based

Before Training

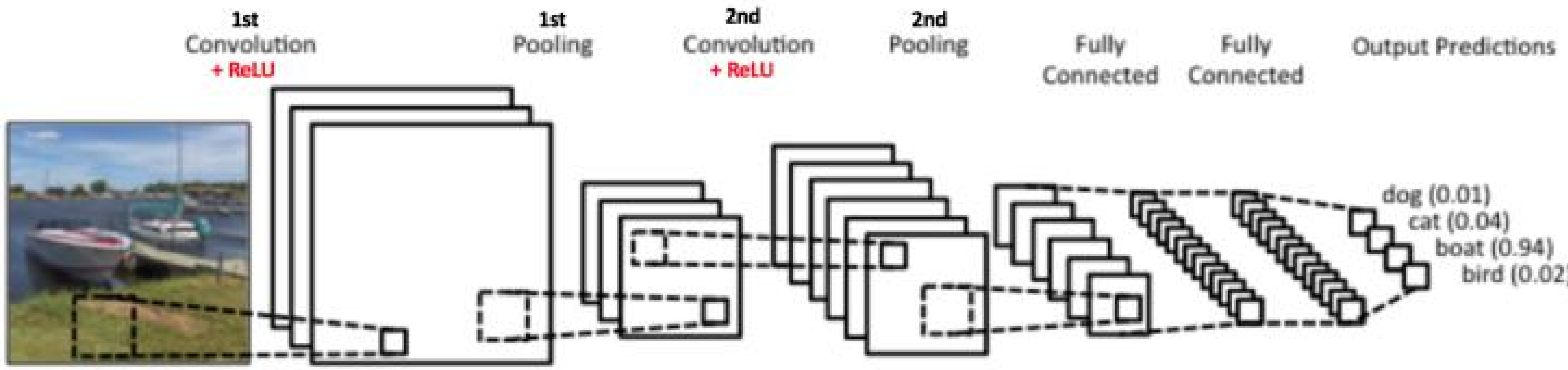
- ResNeXt
- SENet
- MobileNet
- ShuffleNet

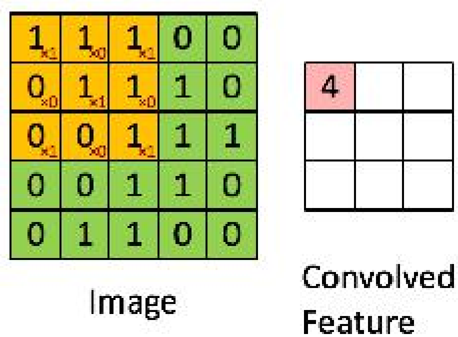
Motivation

- Towards practical deployment
  - Huge & Slow
- Towards interpretability
  - Reduce complexity
- Towards performance
  - Improve information flow

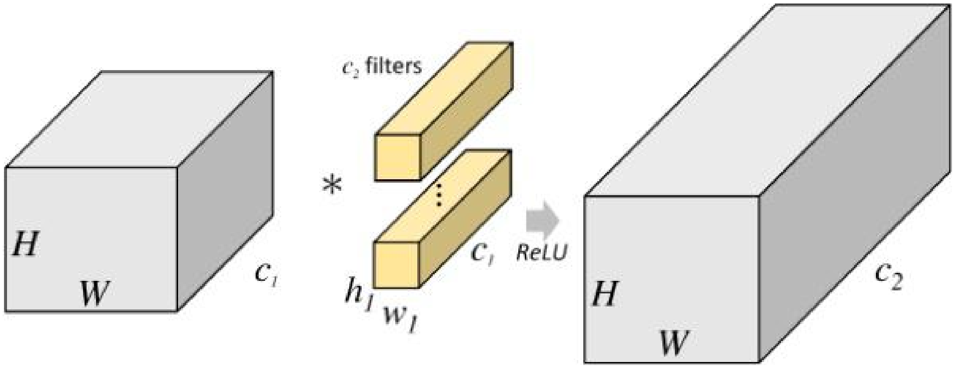
Preliminary

■ Neural Network Model





■ Neural Network Model

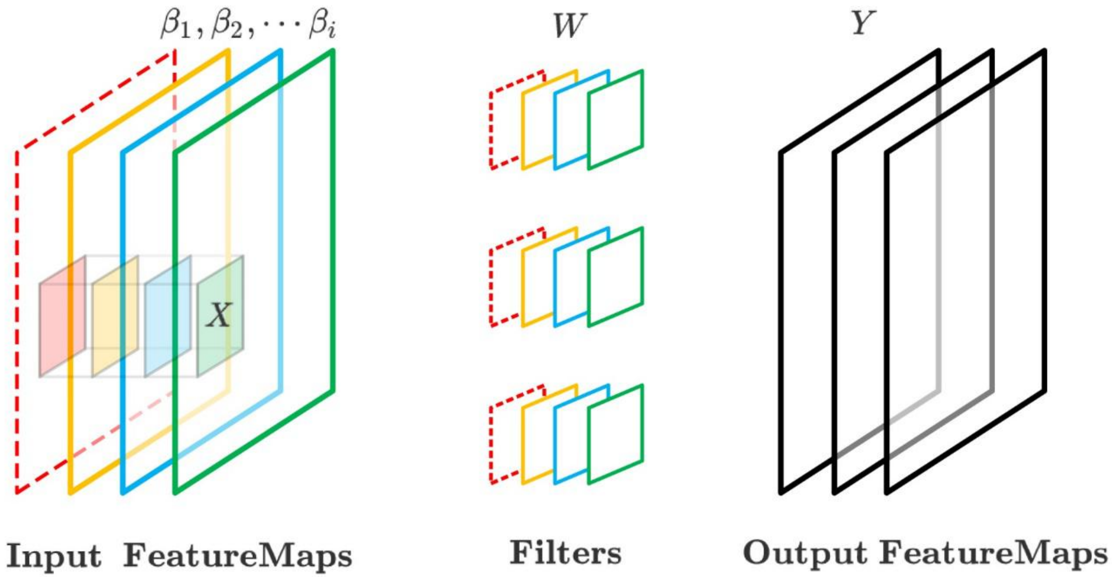


- Params: $(h_l \times w_l \times C_1) \times C_2$
- Flops: $(h_l \times w_l \times C_1 \times C_2) \times (H_{out} \times W_{out})$
- A **fully connection** between C_1 and C_2

After Training

■ Pruning

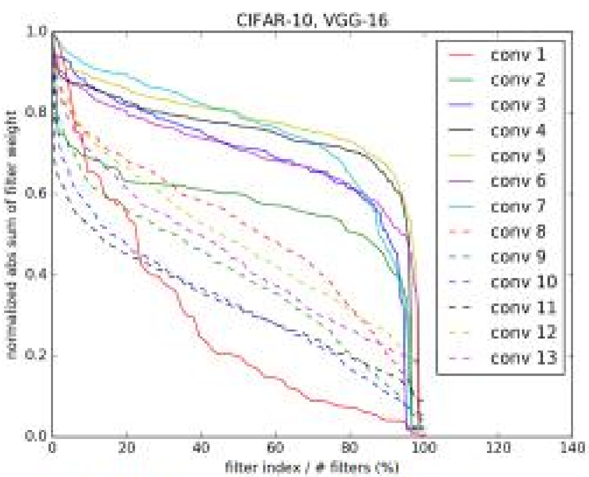
- Build a efficient (effective) network
- By exploiting the weight importance
- Re-training



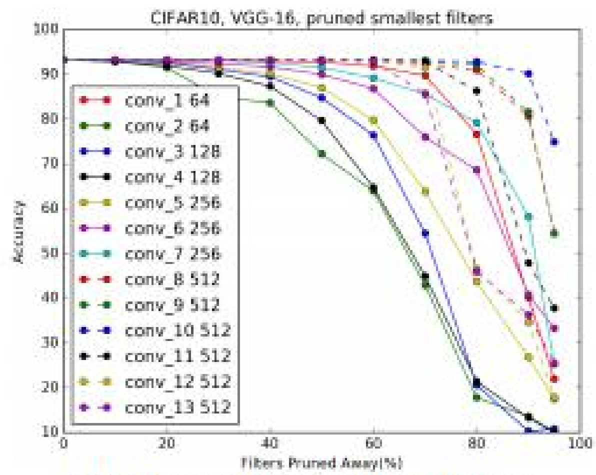
After Training

- **Magnitude based**
  - Small norm less important

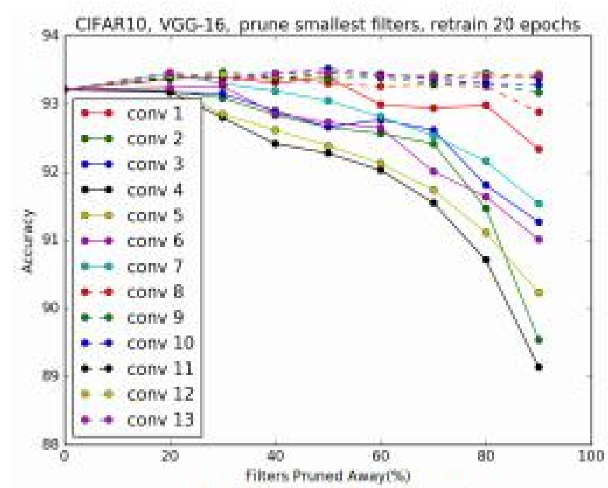
Pruning filters for efficient convnets. *ICLR, 2017*.



(a) Filters are ranked by s_j



(b) Prune the smallest filters

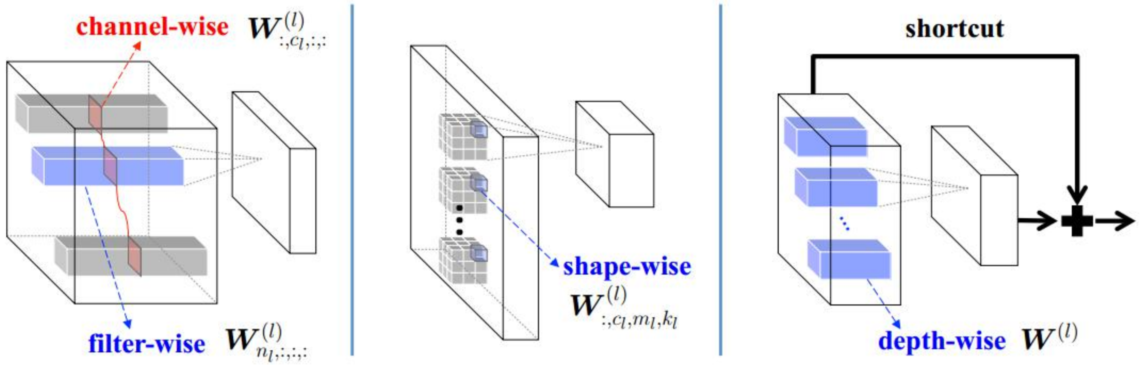


(c) Prune and retrain

After Training

- Sparsity
  - Group Lasso $$R_g(w) = \sum_{g=1}^{G} \|w^{(g)}\|_g$$

Learning structured sparsity in deep neural networks. *NIPS*, 2016.



$$E(W) = E_D(W) + \lambda_n \sum_{l=1}^{L} \left(\sum_{n_l=1}^{N_l} \|W^{(l)}_{n_l,:,:,:}\|_g \right) + \lambda_c \sum_{l=1}^{L} \left(\sum_{c_l=1}^{C_l} \|W^{(l)}_{:,c_l,:,:}\|_g \right)$$

$$E(W) = E_D(W) + \lambda_s \sum_{l=1}^{L} \left(\sum_{c_l=1}^{C_l} \sum_{m_l=1}^{M_l} \sum_{k_l=1}^{K_l} \right) \|W^{(l)}_{:,c_l,m_l,k_l}\|_g$$

$$E(W) = E_D(W) + \lambda_d \sum_{l=1}^{L} \|W^{(l)}\|_g$$

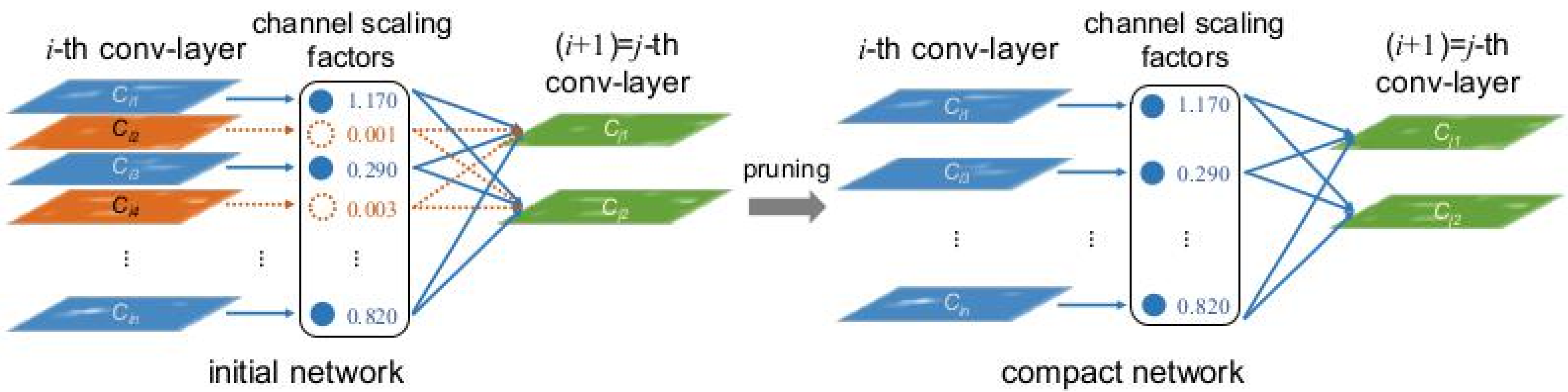
After Training

- Sparsity
  - L0 regularization

Training skinny deep neural networks with iterative hard thresholding methods. *arXiv,* 2016.

  - BN (channel-wise scaling factor)

Learning efficient convolutional networks through network slimming. *ICCV,* 2017.



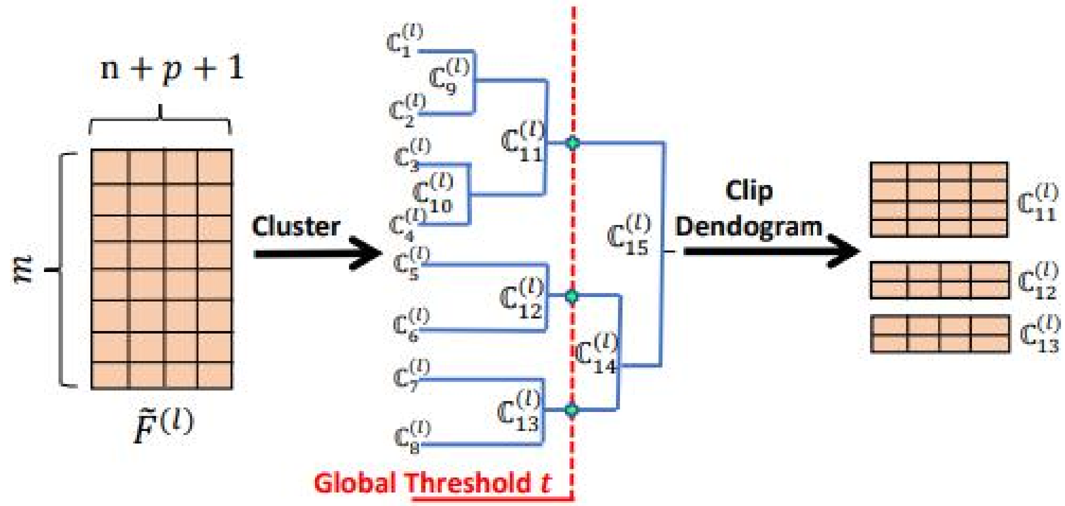
$$\hat{z} = \frac{z_{in} - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}; \quad z_{out} = \gamma \hat{z} + \beta$$

$$L = \sum_{(x,y)} l(f(x, W), y) + \lambda \sum_{\gamma \in \Gamma} g(\gamma)$$

After Training

- **Magnitude based**
  - Cluster (incoming and outgoing weights)

SCSP: Spectral Clustering Filter Pruning with Soft Self-adaption Manners. *arxiv,* 2018.



After Training

- **Activation based**
  - Zero count

Network Trimming: A Data-Driven Neuron Pruning Approach towards Efficient Deep Architectures. *arXiv*, 2016.

$$APoZ_c^{(i)} = APoZ(O_c^{(i)}) = \frac{\sum_k^N \sum_j^M f(O_{c,j}^{(i)}(k) = 0)}{N \times M}$$

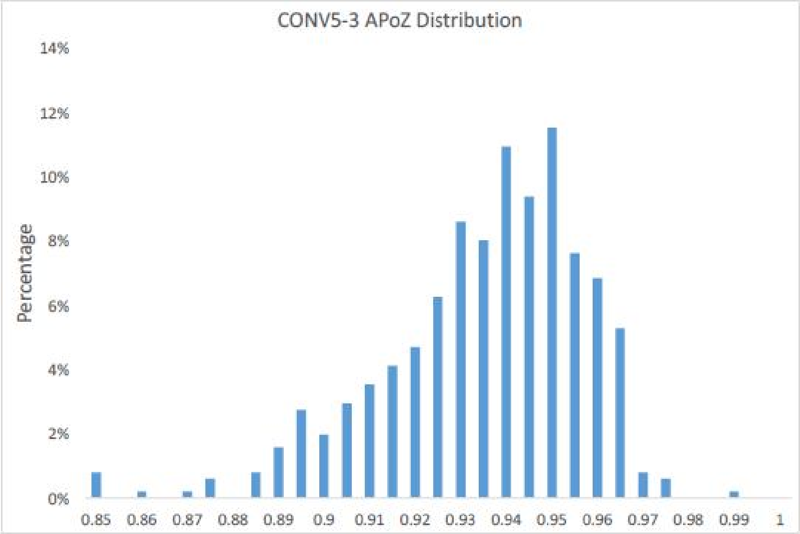


Figure 1: CONV5-3 APoZ Distribution

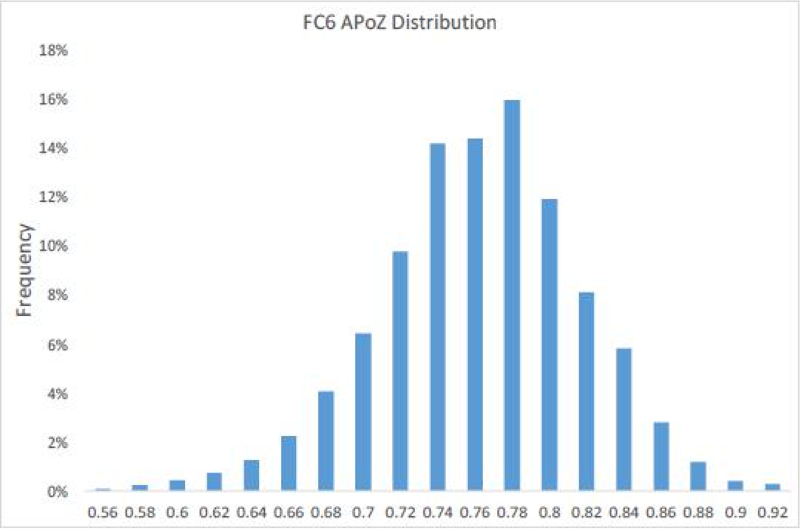
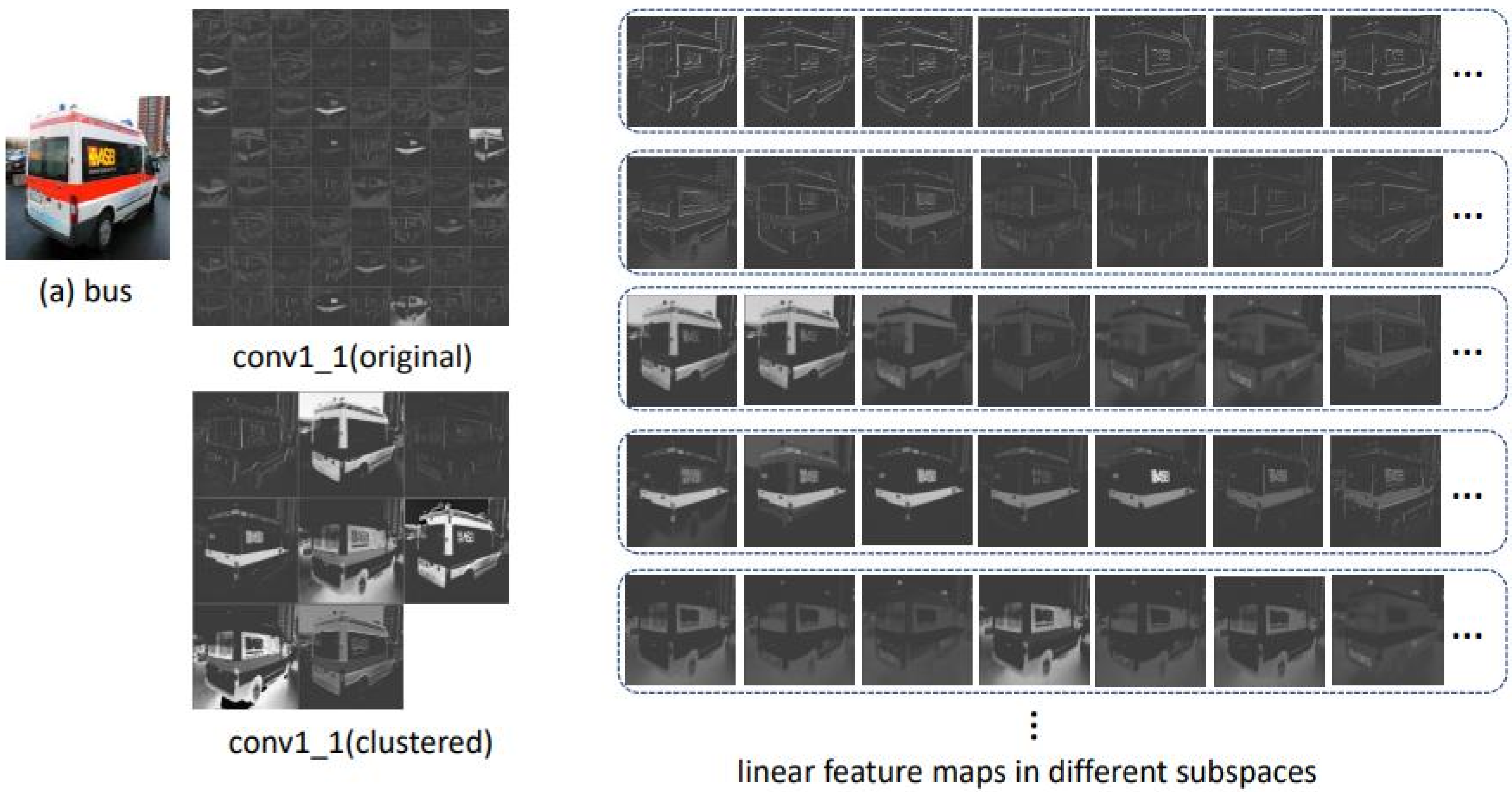


Figure 2: FC6 APoZ Distribution

After Training

- **Activation based**
  - Cluster (similarity upon feature maps)

Exploring Linear Relationship in Feature Map Subspace for ConvNets Compression. *arxiv,* 2018.



After Training

- Activation based
  - Entropy based

An Entropy-based Pruning Method for CNN Compression. *arXiv*, 2017.

$$H_j = -\sum_{i=1}^{m} p_i \log p_i .$$

After Training

- Reconstruction based
    - Greedy Algorithm

ThiNet: A Filter Level Pruning Method for Deep Neural Network Compression. *ICCV*, 2017.

$$\arg\min_{S} \sum_{i=1}^{m} \left(\hat{y}_i - \sum_{j \in S} \hat{\mathbf{x}}_{i,j} \right)^2$$
$$\text{s.t.} \quad |S| = C \times r, \quad S \subset \{1, 2, \ldots, C\}.$$

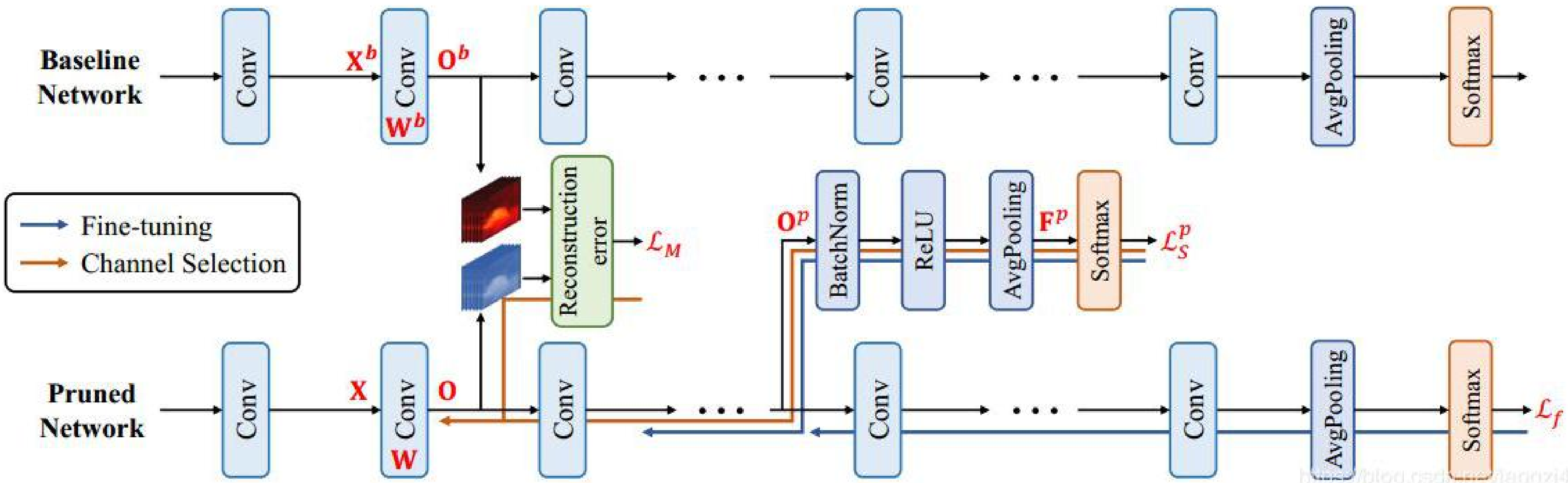
- LASSO regression

Channel pruning for accelerating very deep neural networks. *ICCV*, 2017.

$$\arg\min_{\beta, \mathrm{W}} \frac{1}{2N} \left\| \mathrm{Y} - \sum_{i=1}^{c} \beta_i \mathrm{X_i} \mathrm{W_i}^{\top} \right\|_F^2 + \lambda \|\beta\|_1$$
$$\text{subject to } \|\beta\|_0 \leq c', \forall i \ \|\mathrm{W_i}\|_F = 1$$

■ Reconstruction based

Discrimination-aware Channel Pruning for Deep Neural Networks. *NIPS*, 2018.



$$\mathbf{F}^p(\mathbf{W}) = \text{AvgPooling}(\text{ReLU}(\text{BN}(\mathbf{O}^p))),$$

$$\mathcal{L}_S^p(\mathbf{W}) = -\frac{1}{N}\left[\sum_{i=1}^{N}\sum_{t=1}^{m} I\{y^{(i)} = t\} \log \frac{e^{\boldsymbol{\theta}_t^\top \mathbf{F}^{(p,i)}}}{\sum_{k=1}^{m} e^{\boldsymbol{\theta}_k^\top \mathbf{F}^{(p,i)}}}\right],$$

After Training

- **Influence based**
  - Taylor expansion

Pruning Convolutional Neural Networks for Resource Efficient Transfer Learning. *ICLR*, 2016.

$$\min_{W'} |C(D|W') - C(D|W)| \quad s.t. \quad ||W'||_0 \leq B$$

$$|\Delta C(h_i)| = |C(D, h_i = 0) - C(D, h_i)|$$

$$C(D, h_i = 0) = C(D, h_i) - \frac{\partial C}{\partial h_i} h_i + R_1(h_i = 0)$$

$$|\Delta C(h_i)| = |C(D, h_i = 0) - C(D, h_i)| = |\frac{\partial C}{\partial h_i} h_i|$$

$$\Theta_{TE}(z_l^{(k)}) = |\frac{1}{M} \sum_m \frac{\partial C}{\partial z_{l,m}^{(k)}} z_{l,m}^{(k)}|$$

After Training

- **Influence based**
  - Taylor expansion

Collaborative Channel Pruning for Deep Networks. *ICML,* 2019.

$$\mathcal{L}(\boldsymbol{\beta}, \mathbf{W}) \approx \mathcal{L}(\mathbf{W}) + \mathbf{g}^T\mathbf{v} + \frac{1}{2}\mathbf{v}^T\mathbf{H}\mathbf{v}$$

$$\mathbf{v} = \operatorname{vec}(\boldsymbol{\beta} \odot \mathbf{W} - \mathbf{W}) \qquad \mathbf{g} = \nabla\mathcal{L}(\mathbf{w}),\ \mathbf{H} = \nabla^2\mathcal{L}(\mathbf{w})$$

$$\mathcal{L}(\boldsymbol{\beta}, \bar{\mathbf{W}}) \approx \mathcal{L}(\bar{\mathbf{W}}) + \sum_{i=1}^{c_o}(\beta_i - 1)\,\bar{\mathbf{g}}_i^T\bar{\mathbf{w}}_i + \frac{1}{2}\sum_{i,j=1}^{c_o}(\beta_i - 1)(\beta_j - 1)\,\bar{\mathbf{w}}_i^T\bar{\mathbf{H}}_{ij}\bar{\mathbf{w}}_j$$

$$u_i = \bar{\mathbf{g}}_i^T\bar{\mathbf{w}}_i \quad ,\ \forall i$$

$$s_{ij} = \frac{1}{2}\bar{\mathbf{w}}_i^T\bar{\mathbf{H}}_{ij}\bar{\mathbf{w}}_j,\ \forall i, j$$

$$\min \sum_{i=1}^{c_o} u_i(\beta_i - 1) + \sum_{i,j=1}^{c_o} s_{ij}(\beta_i - 1)(\beta_j - 1)$$

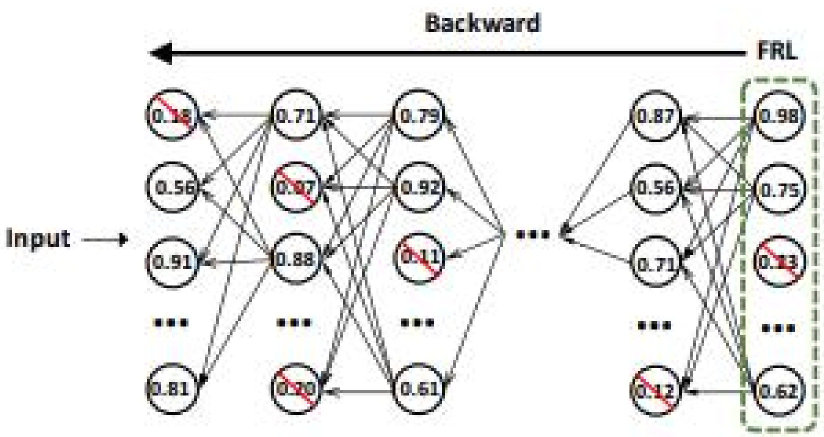
$$\text{s.t. } \|\boldsymbol{\beta}\|_0 = p,\ \beta_i \in \{0, 1\},\ \forall i$$

After Training

- **Influence based**
  - Score Propagation

NISP: Pruning Networks using Neuron Importance Score Propagation. *CVPR*, 2018.

$$f^{(l)}(\mathbf{x}) = \sigma^{(l)}(\mathbf{w}^{(l)}\mathbf{x} + \mathbf{b}^{(l)}), \quad G^{(i,j)} = f^{(j)} \circ G^{(i,j-1)}$$



$$\mathcal{F}(\mathbf{s}_l^*|\mathbf{x}, \mathbf{s}_n; F) = \langle\, \mathbf{s}_n, \ |F(\mathbf{x}) - F(\mathbf{s}_l^* \odot \mathbf{x})| \,\rangle,$$

Lipschitz Continuity:

$$|\sigma^{(k)}(\mathbf{x}) - \sigma^{(k)}(\mathbf{y})| \leq C_\sigma^{(k)} |\mathbf{x} - \mathbf{y}|.$$

$$|f^{(k)}(\mathbf{x}) - f^{(k)}(\mathbf{y})| \leq C_\sigma^{(k)} |\mathbf{w}^{(k)}| \cdot |\mathbf{x} - \mathbf{y}|,$$

$$|G^{(i,j)}(\mathbf{x}) - G^{(i,j)}(\mathbf{y})| \leq C_\sigma^{(j)} |\mathbf{w}^{(j)}| |G^{(i,j-1)}(\mathbf{x}) - G^{(i,j-1)}(\mathbf{y})|$$

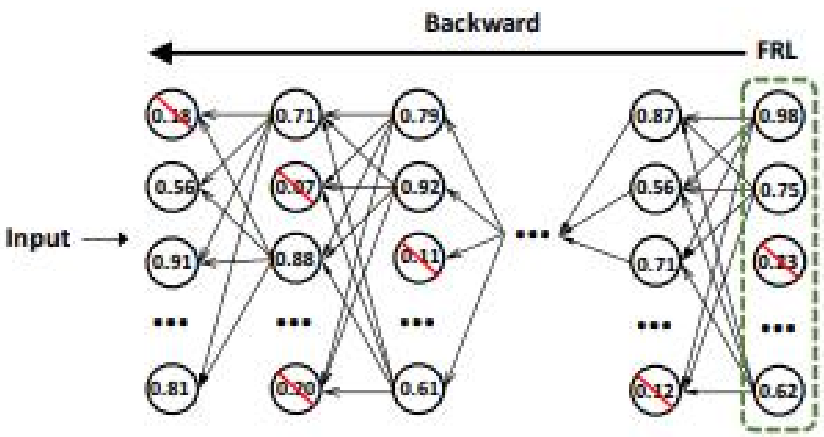
$$\mathbf{x} = \mathbf{x}_l^{(m)}, \mathbf{y} = \mathbf{s}_l^* \odot \mathbf{x}_l^{(m)}, i = l+1$$

$$|G^{(l+1,n)}(\mathbf{x}_l^{(m)}) - G^{(l+1,n)}(\mathbf{s}_l^* \odot \mathbf{x}_l^{(m)})| \leq C_\Sigma^{(l+1,n)} \mathbf{W}^{(l+1,n)} |\mathbf{x}_l^{(m)} - \mathbf{s}_l^* \odot \mathbf{x}_l^{(m)}|$$

After Training

- **Influence based**
  - Score Propagation

NISP: Pruning Networks using Neuron Importance Score Propagation. *CVPR*, 2018.



$$\sum_{m=1}^{M} \mathcal{F}(\mathbf{s}_l^*|\mathbf{x}_l^{(m)}, \mathbf{s}_n; F^{(l+1)}) \le C \sum_i r_{l,i}(1 - s_{l,i}^*)$$

$$C = C_\Sigma^{(l+1,n)} C_x \qquad \mathbf{r}_l = \mathbf{W}^{(l+1,n)\intercal}\mathbf{s}_n$$

$$\arg\min_{\mathbf{s}_l^*} \sum_i r_{l,i}(1 - s_{l,i}^*) \Leftrightarrow \arg\max_{\mathbf{s}_l^*} \sum_i s_{l,i}^* r_{l,i}$$

So:

$$\mathbf{s}_k = |\mathbf{w}^{(k+1)}|^\intercal |\mathbf{w}^{(k+2)}|^\intercal \cdots |\mathbf{w}^{(n)}|^\intercal \mathbf{s}_n$$

$$\mathbf{s}_k = |\mathbf{w}^{(k+1)}|^\intercal \mathbf{s}_{k+1}$$

After Training

- Influence based
  - Weight probe

SNIP: Single-shot Network Pruning based on Connection Sensitivity. *ICLR*, 2018.

$$\min_{\mathbf{w}} L(\mathbf{w}; \mathcal{D}) = \min_{\mathbf{w}} \frac{1}{n} \sum_{i=1}^{n} \ell(\mathbf{w}; (\mathbf{x}_i, \mathbf{y}_i)) ,$$
$$\text{s.t.} \quad \mathbf{w} \in \mathbb{R}^m, \quad \|\mathbf{w}\|_0 \leq \kappa .$$

$$\min_{\mathbf{c},\mathbf{w}} L(\mathbf{c} \odot \mathbf{w}; \mathcal{D}) = \min_{\mathbf{c},\mathbf{w}} \frac{1}{n} \sum_{i=1}^{n} \ell(\mathbf{c} \odot \mathbf{w}; (\mathbf{x}_i, \mathbf{y}_i)) ,$$
$$\text{s.t.} \quad \mathbf{w} \in \mathbb{R}^m ,$$
$$\mathbf{c} \in \{0,1\}^m, \quad \|\mathbf{c}\|_0 \leq \kappa ,$$

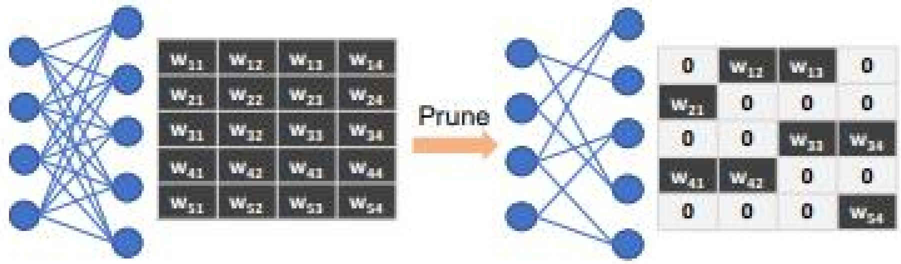
$$\Delta L_j(\mathbf{w}; \mathcal{D}) \approx g_j(\mathbf{w}; \mathcal{D}) = \left.\frac{\partial L(\mathbf{c} \odot \mathbf{w}; \mathcal{D})}{\partial c_j}\right|_{\mathbf{c}=\mathbf{1}} = \lim_{\delta \to 0} \left.\frac{L(\mathbf{c} \odot \mathbf{w}; \mathcal{D}) - L((\mathbf{c} - \delta\, \mathbf{e}_j) \odot \mathbf{w}; \mathcal{D})}{\delta}\right|_{\mathbf{c}=\mathbf{1}}$$

$$s_j = \frac{|g_j(\mathbf{w}; \mathcal{D})|}{\sum_{k=1}^{m} |g_k(\mathbf{w}; \mathcal{D})|}$$

After Training

- Rethinking
  - Sparse Density (irregular pruning)

DARB: A Density-Aware Regular-Block Pruning for Deep Neural Networks. *AAAI*, 2020.



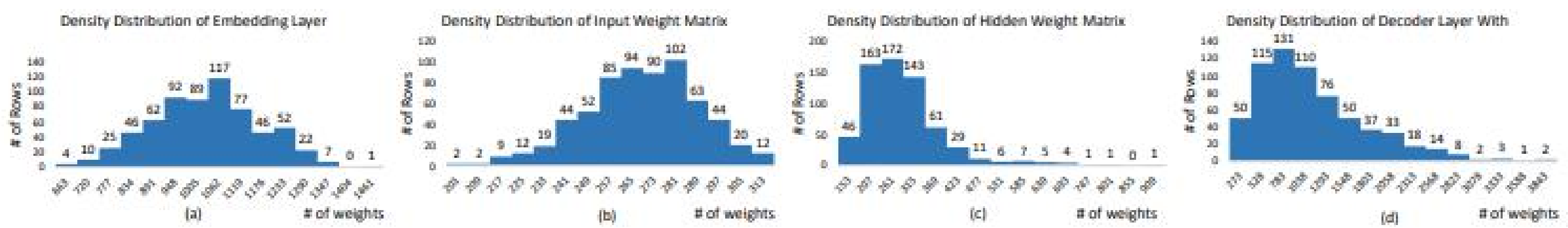
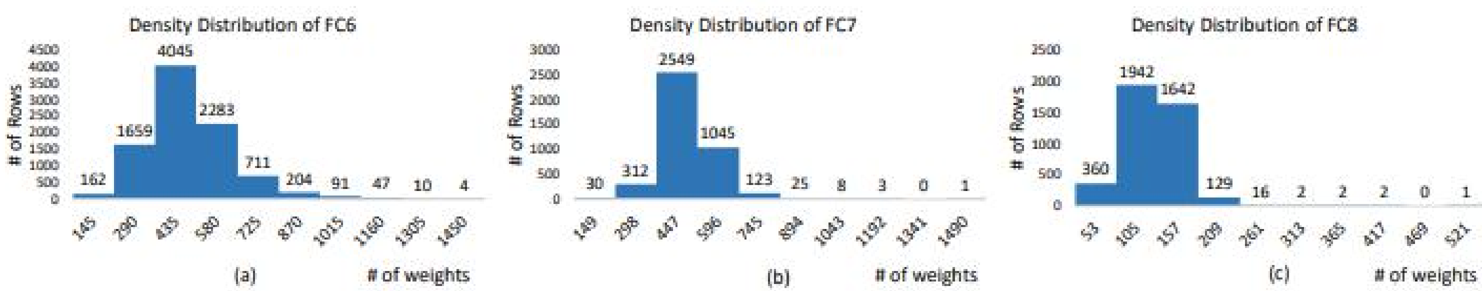


Figure 3: Row density distribution of the four weight components of a medium LSTM with 90% sparsity: (a) embedding layer, (b) input weight matrices, (c) hidden weight matrices, (d) decoder layer.



After Training

■ Rethinking

- Sparse Density (irregular pruning)

DARB: A Density-Aware Regular-Block Pruning for Deep Neural Networks. *AAAI*, 2020.

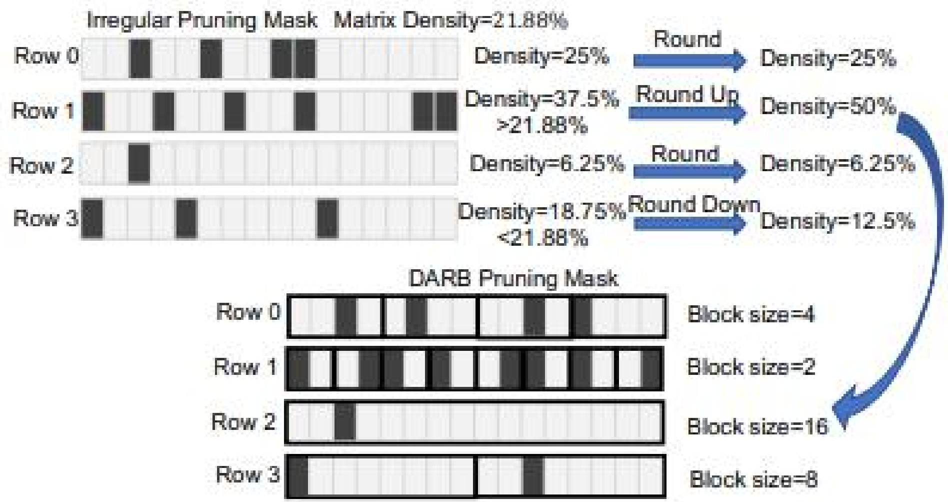


Table 2: The Comparison of Relative Difference of Retained Weights between BMWM and Block Pruning

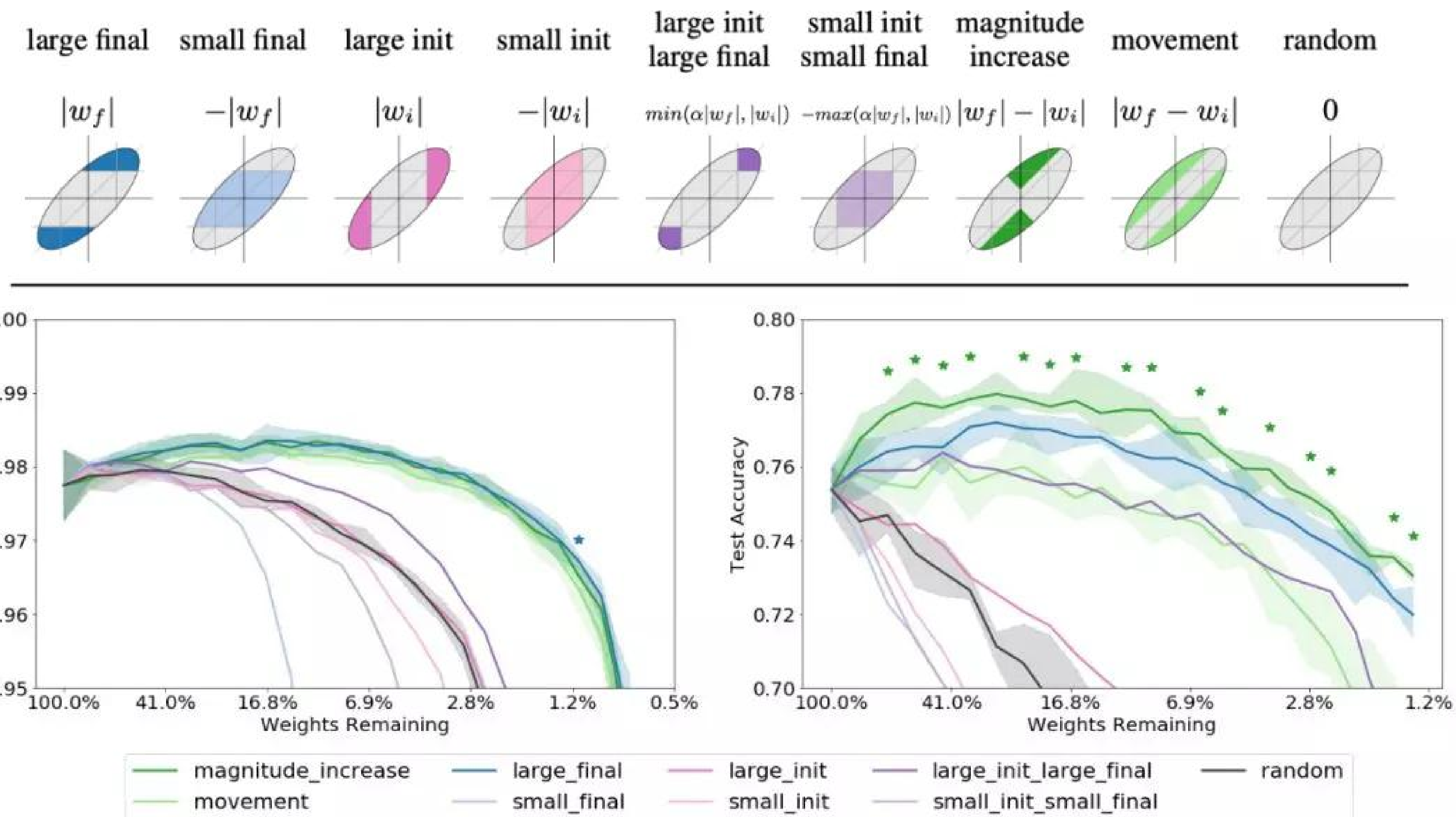
Layer	BMWM to Irregular Pruning	Block Pruning 4×4 to Irregular Pruning
Embedding	8.8%	45.3%
LSTM1	4.3%	31.1%
LSTM2	4.3%	31.1%
Decoder	8.9%	45.2%

$$\frac{|\bar{W}_{irr} - \bar{W}_{bmwm}|}{\bar{W}_{irr}}$$

After Training

- Decode Lottery Ticket

Deconstructing Lottery Tickets: Zeros, Signs, and the Supermask. *NIPS*, 2019.

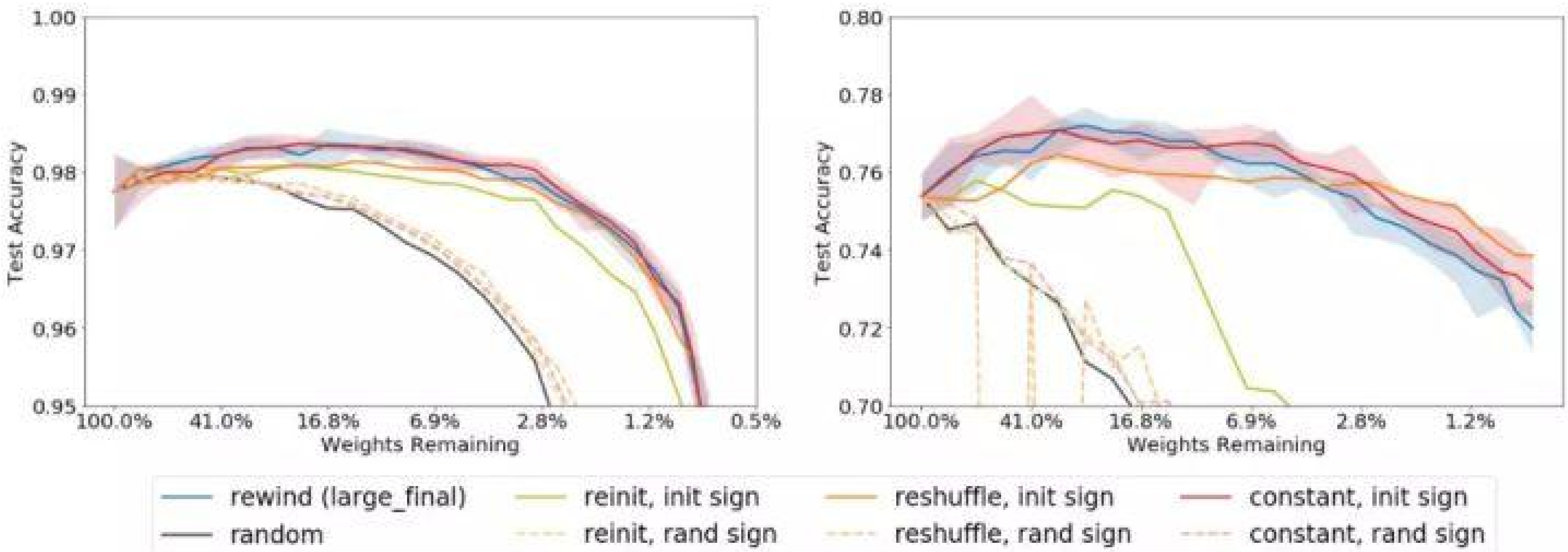


After Training

- Decode Lottery Ticket

Deconstructing Lottery Tickets: Zeros, Signs, and the Supermask. *NIPS*, 2019.

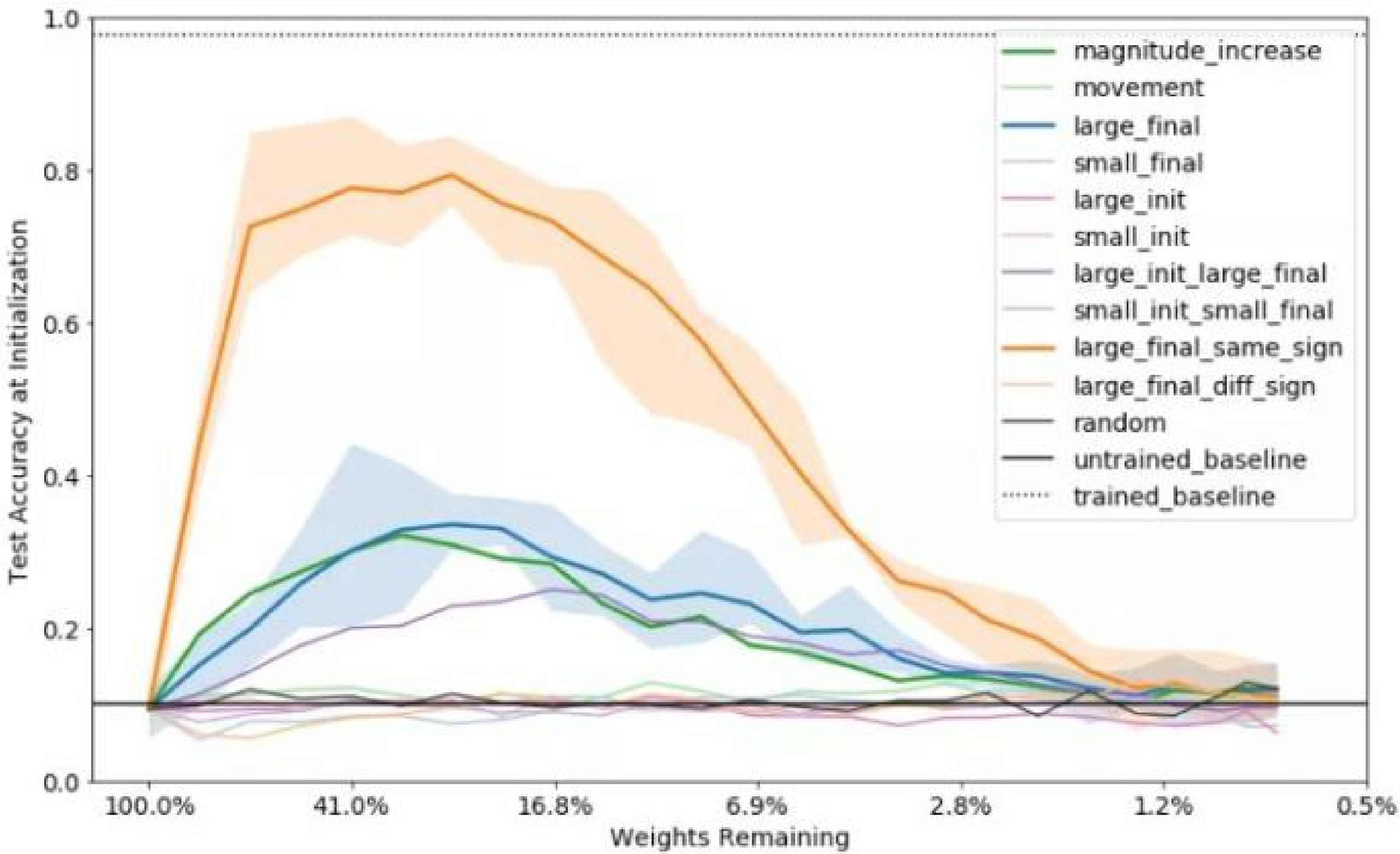
- Reinit：基于原始的初始化分布来初始化保留的权重
- Reshuffle：基于保留权重的原始分布进行初始化
- Constant：将保留的权重设为正或负的常数，即每层原初始值的标准差



After Training

- Decode Lottery Ticket

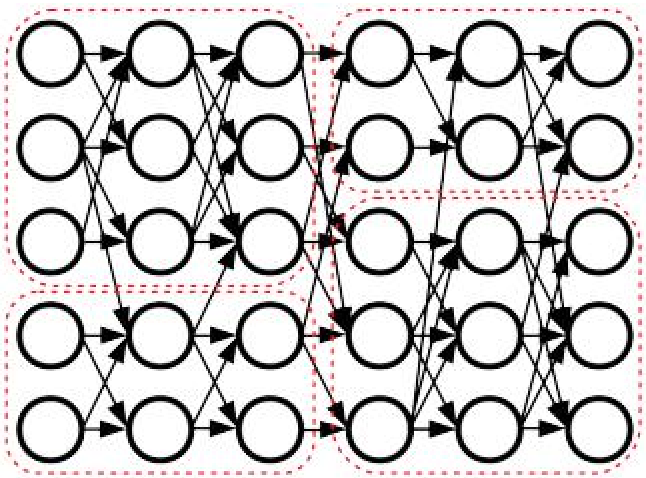
Deconstructing Lottery Tickets: Zeros, Signs, and the Supermask. *NIPS*, 2019.



After Training

- Modular

  Pruned Neural Networks Are Surprisingly Modular. *arxiv*, 2020.



Algorithm 1 Normalized Spectral Clustering [34]

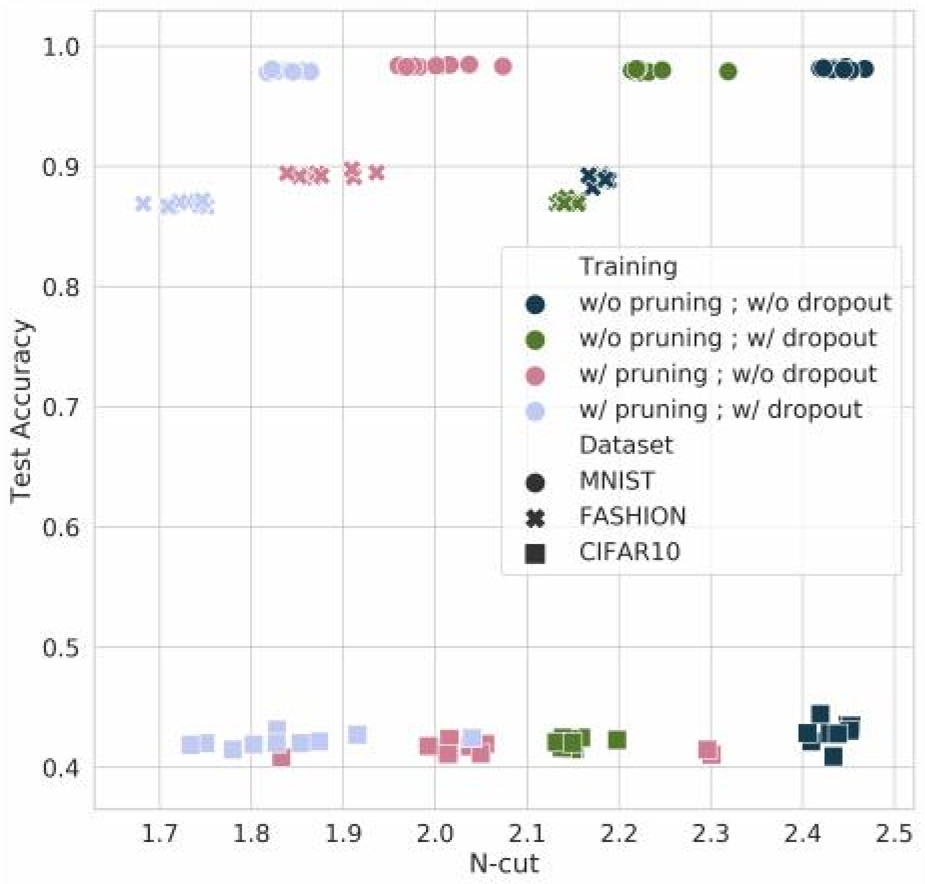
Input: Adjacency matrix A, number k of clusters
Compute the normalized Laplacian L_{norm}
Compute the first k eigenvectors $u_1, \ldots, u_k \in \mathbb{R}^N$ of L_{norm}
Form the matrix $U \in \mathbb{R}^{k \times N}$ whose j^{th} row is $u_j^\top$
For $n \in \{1, \ldots, N\}$, let $y_n \in \mathbb{R}^k$ be the n^{th} column of U
Cluster the points $(y_n)_{n=1}^N$ with the k-means algorithm into clusters $C_1, \ldots, C_k$
Return: Clusters $X_1, \ldots, X_k$ with $X_i = \{n \in \{1, \ldots, N\} \mid y_n \in C_i\}$

After Training

数据挖掘实验室
Data Mining Lab

- Modular

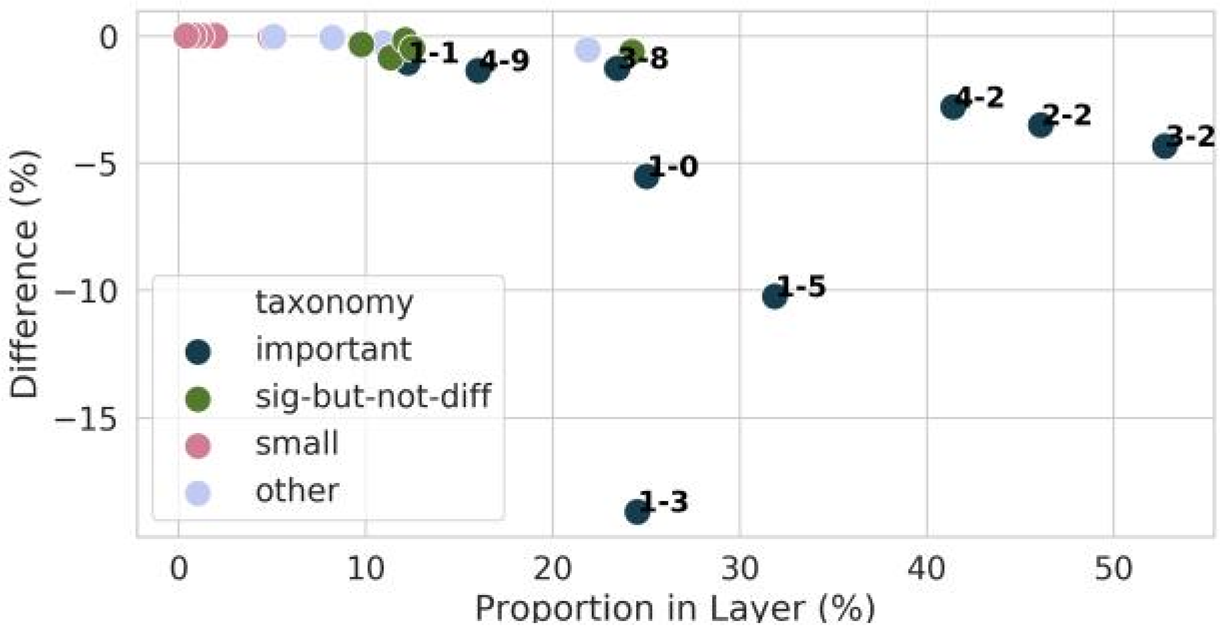
  Pruned Neural Networks Are Surprisingly Modular. *arxiv*, 2020.



After Training

- Modular

  Pruned Neural Networks Are Surprisingly Modular. *arxiv*, 2020.

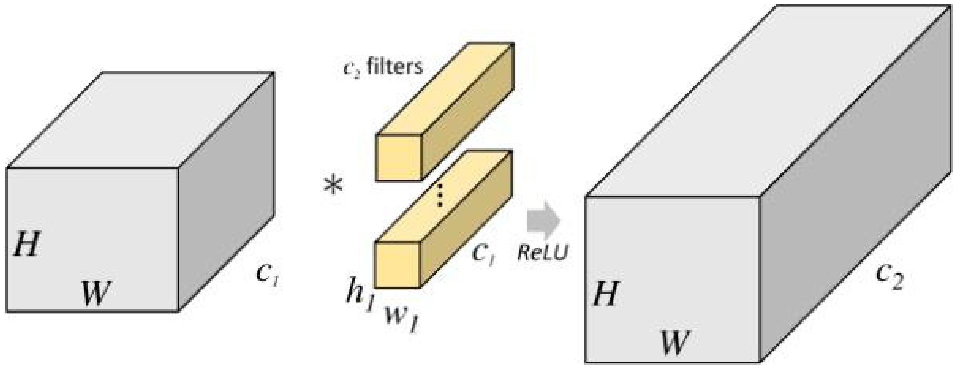


After Training

- Summary
  - Irregular pruning, powerful
  - Structural pruning, efficient
  - Sparsity in training
  - Sign is important for init
  - Cross-Layer Pruning

Before Training

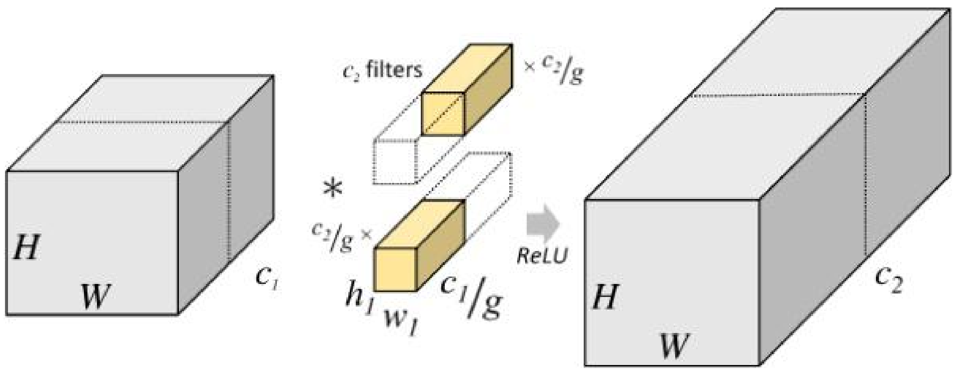
Preliminary



- Params: $(h_l \times w_l \times C_1) \times C_2$
- Flops: $(h_l \times w_l \times C_1 \times C_2) \times (H_{out} \times W_{out})$

数据挖掘实验室
Data Mining Lab

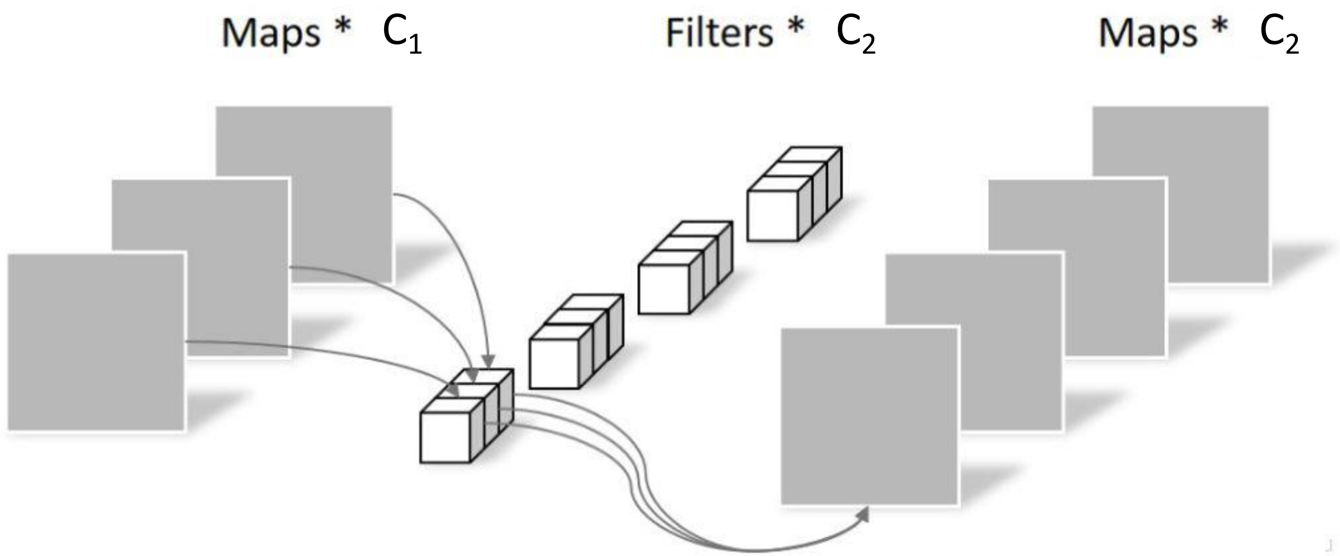
Group Convolution



- Params: $g \times \left(h_l \times w_l \times \frac{C_1}{g}\right) \times \frac{C_2}{g} = h_l \times w_l \times C_1 \times C_2/g$
- Flops: $h_l \times w_l \times C_1 \times C_2 \times H_{out} \times W_{out}/g$

Before Training

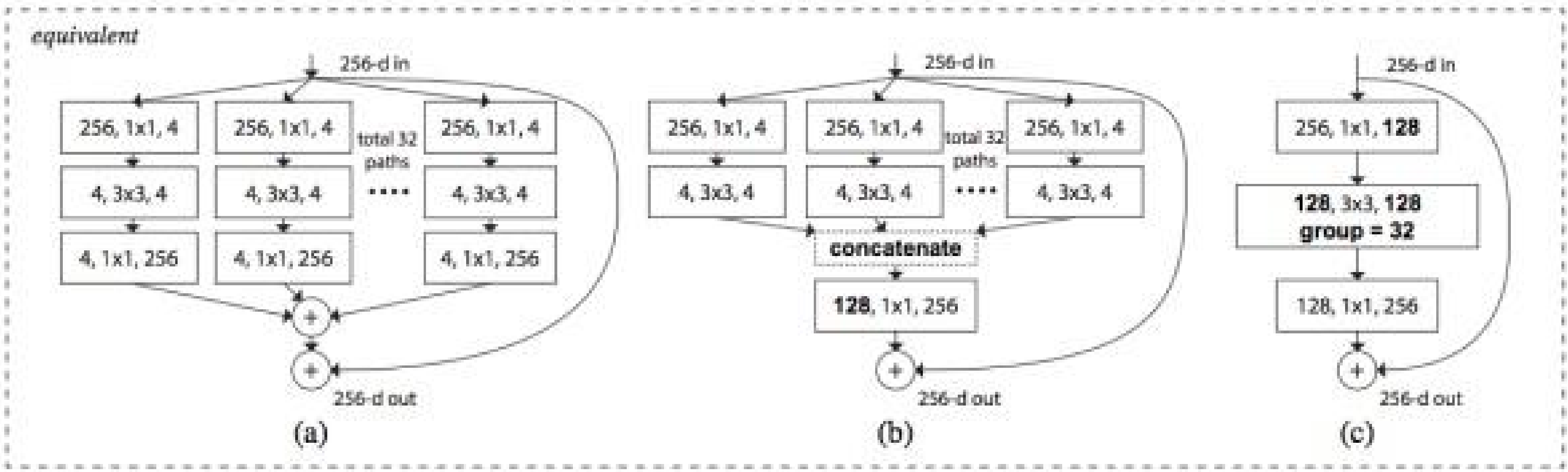
■ 1 × 1 Convolution



- Params: $C_1 \times C_2$
- Flops: $C_1 \times C_2 \times H_{out} \times W_{out}$

Before Training

■ ResNeXt



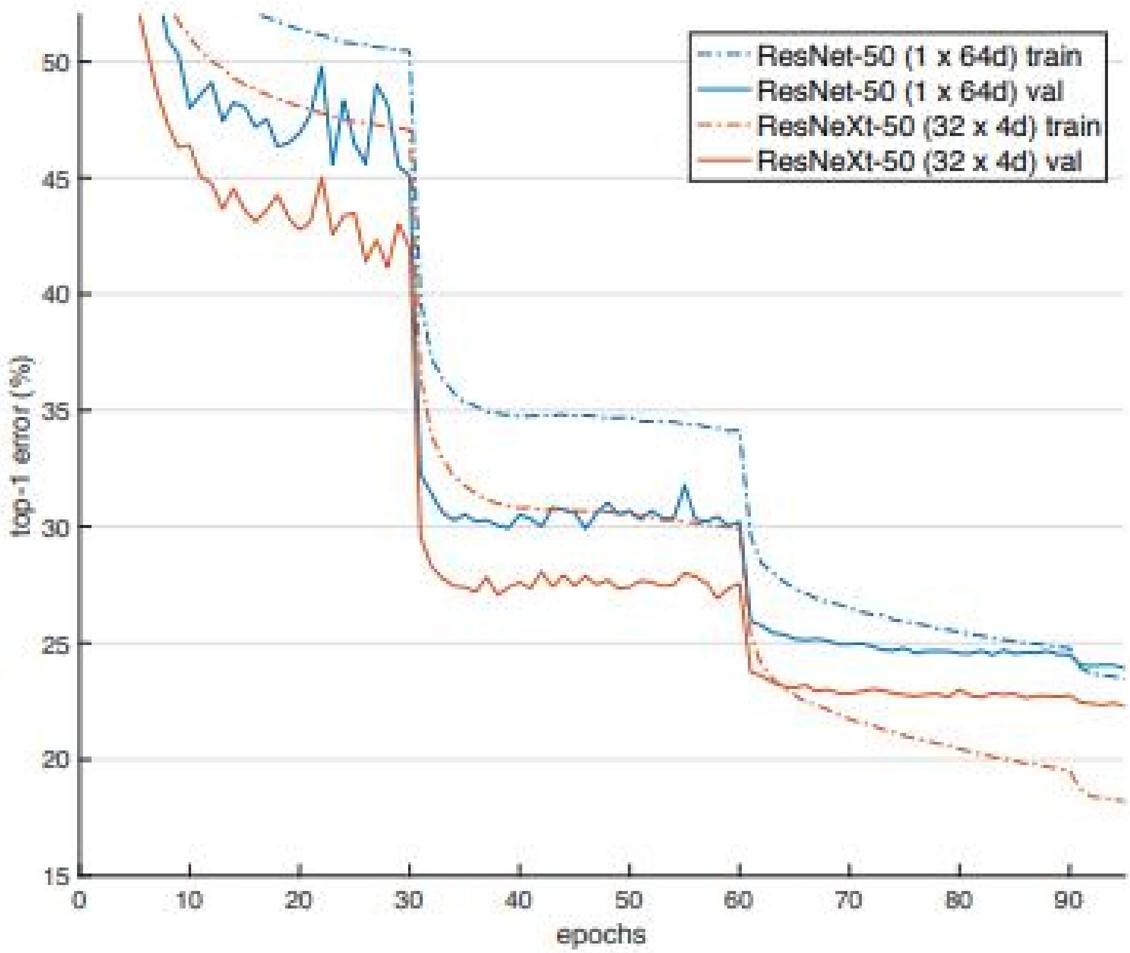
- Group Convolution + 1×1 Convolution
- Concise network structure design

- ResNeXt

stage	output	ResNet-50	ResNeXt-50 (32×4d)
conv1	112×112	7×7, 64, stride 2	7×7, 64, stride 2
conv2	56×56	3×3 max pool, stride 2	3×3 max pool, stride 2
		$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128, C{=}32 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3	28×28	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256, C{=}32 \\ 1\times1, 512 \end{bmatrix}\times4$
conv4	14×14	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512, C{=}32 \\ 1\times1, 1024 \end{bmatrix}\times6$
conv5	7×7	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 1024 \\ 3\times3, 1024, C{=}32 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	global average pool 1000-d fc, softmax	global average pool 1000-d fc, softmax
# params.		**25.5**$\times10^6$	**25.0**$\times10^6$
FLOPs		**4.1**$\times10^9$	**4.2**$\times10^9$

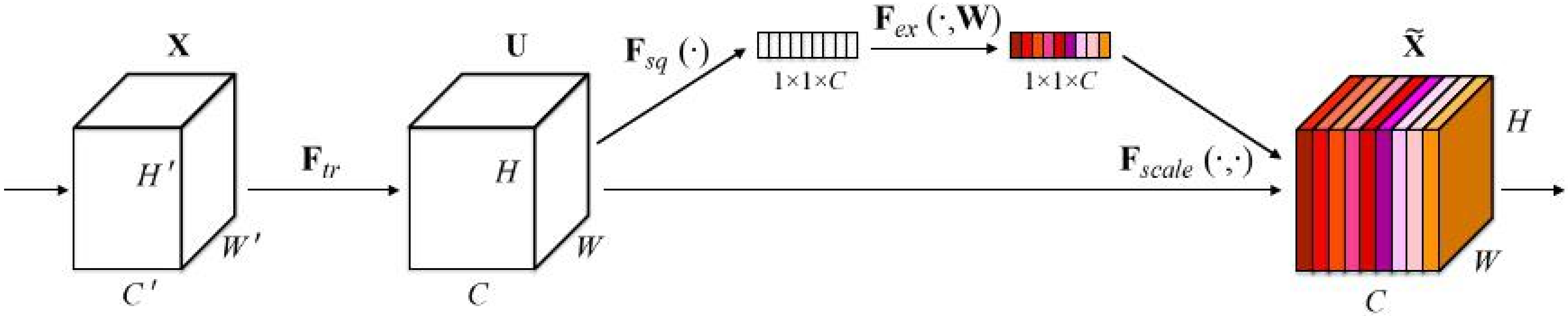
Before Training

- ResNeXt



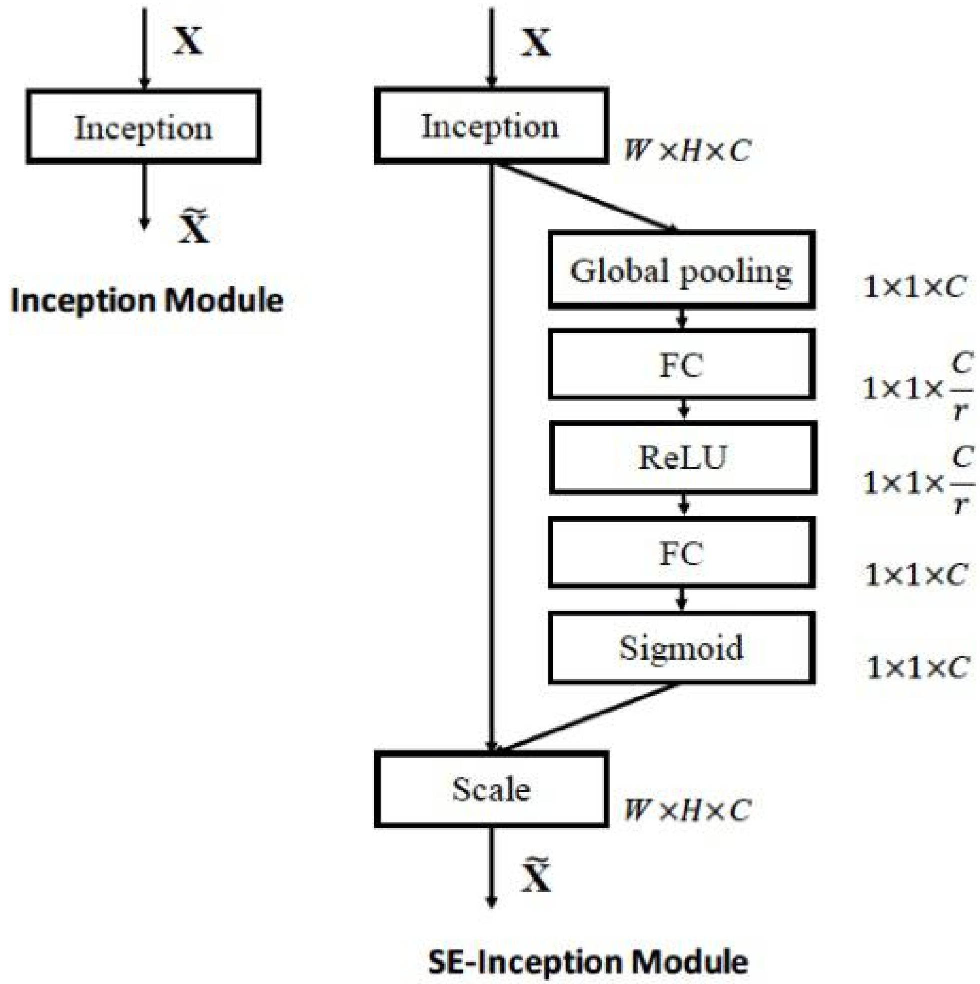
Before Training

■ Squeeze-and-Excitation Networks

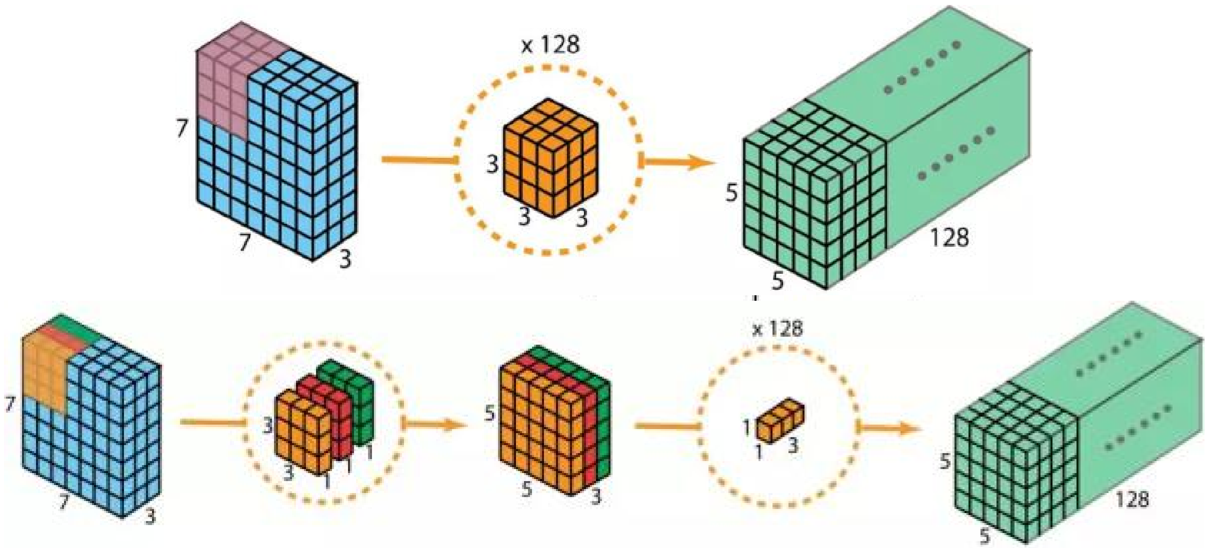


- Explicitly model the relationship between channels
- 2017 ImageNet Champion

- Squeeze-and-Excitation Networks



■ MobileNet V1



$$\frac{h_l \times w_l \times C_1 \times H_{out} \times W_{out} + C_1 \times C_2 \times H_{out} \times W_{out}}{h_l \times w_l \times C_1 \times C_2 \times H_{out} \times W_{out}}$$

$$= \frac{1}{N} + \frac{1}{h_l \times w_l}$$

Before Training

数据挖掘实验室
Data Mining Lab

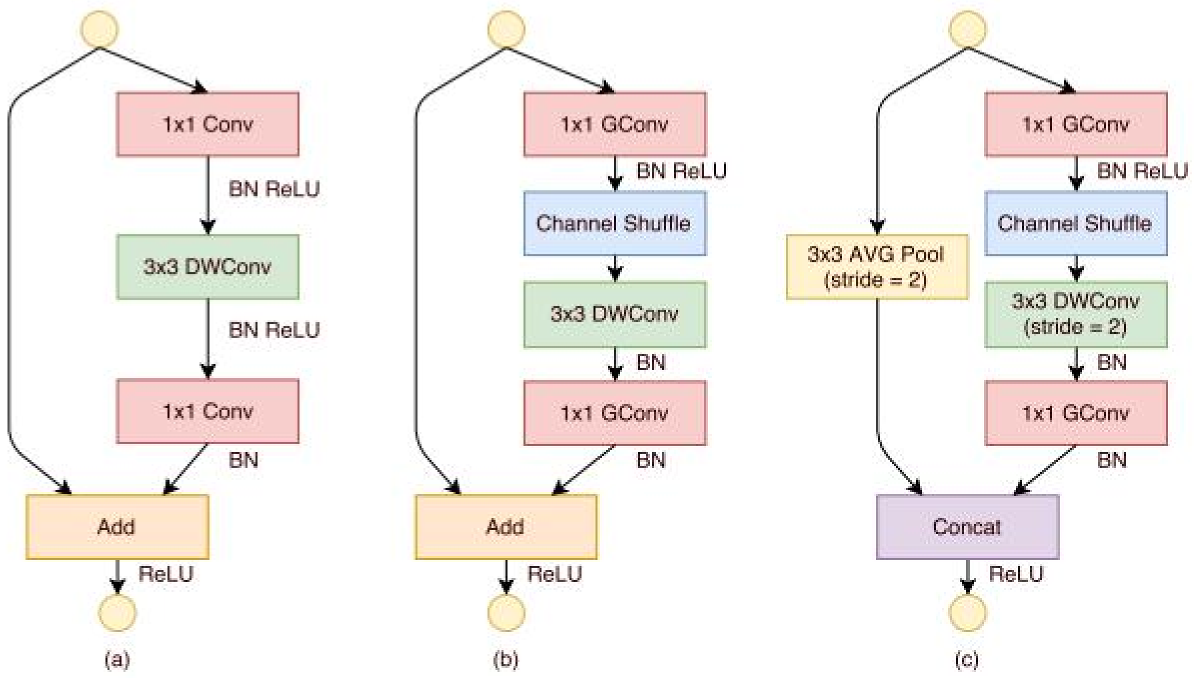
- MobileNet V1
  - Not efficient in practical computation
  - Too many Conv 1×1 operators

Table 2. Resource Per Layer Type

Type	Mult-Adds	Parameters
Conv 1×1	94.86%	74.59%
Conv DW 3×3	3.06%	1.06%
Conv 3×3	1.19%	0.02%
Fully Connected	0.18%	24.33%

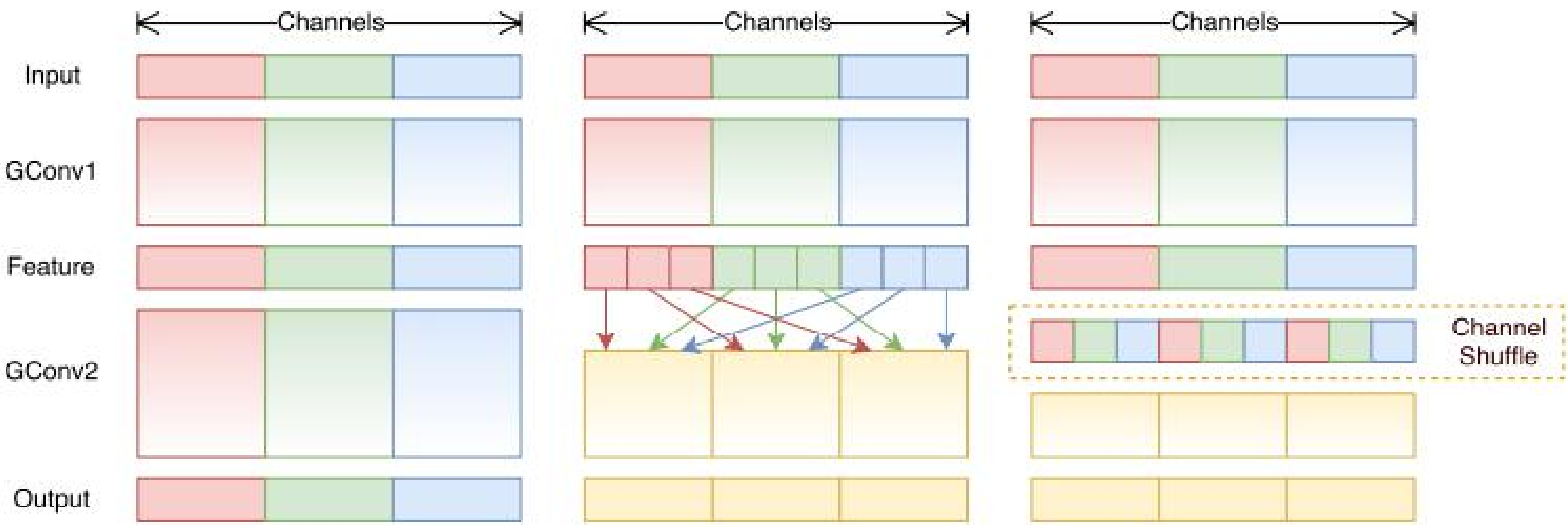
Before Training

- ShuffleNet V1
  - Group & Shuffle (for information interaction)



Before Training

- ShuffleNet V1
  - Group & Shuffle (for information interaction)



■ ShuffleNet V1

- More group
- Less Channel

Model	Complexity (MFLOPs)	Classification error (%)				
		$g = 1$	$g = 2$	$g = 3$	$g = 4$	$g = 8$
ShuffleNet 1×	140	33.6	32.7	32.6	32.8	**32.4**
ShuffleNet 0.5×	38	45.1	44.4	43.2	**41.6**	42.3
ShuffleNet 0.25×	13	57.1	56.8	55.0	54.2	**52.7**

Table 2. Classification error vs. number of groups g (*smaller number represents better performance*)

Summary

数据挖掘实验室
Data Mining Lab

- Do it via channel/filter!
- Ensure information flow!
- Hardware-friendly

Thank you