深度神经网络加速和压缩

年度进展报告(2017)

程 健

jcheng@nlpr.ia.ac.cn

中国科学院自动化研究所

模式识别国家重点实验室

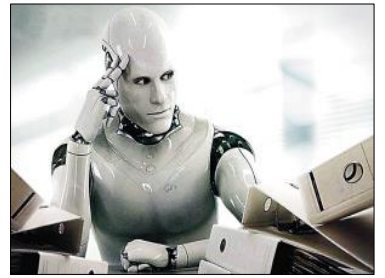
2018.4.22

深度神经网络计算

Year	Model	Layers	Parameter	FLOPs	ImageNet Top-5 error
2012	AlexNet	5+3	60M	725M	16.4%
2013	Clarifai	5+3	60M	—	11.7%
2014	MSRA	5+3	200M	—	8.06%
2014	VGG-19	16+3	143M	19.6B	7.32%
2014	GoogLeNet	22	6.8M	1.566B	6.67%
2015	ResNet	152	19.4M	11.3B	3.57%

深度神经网络计算

移动设备：算不好



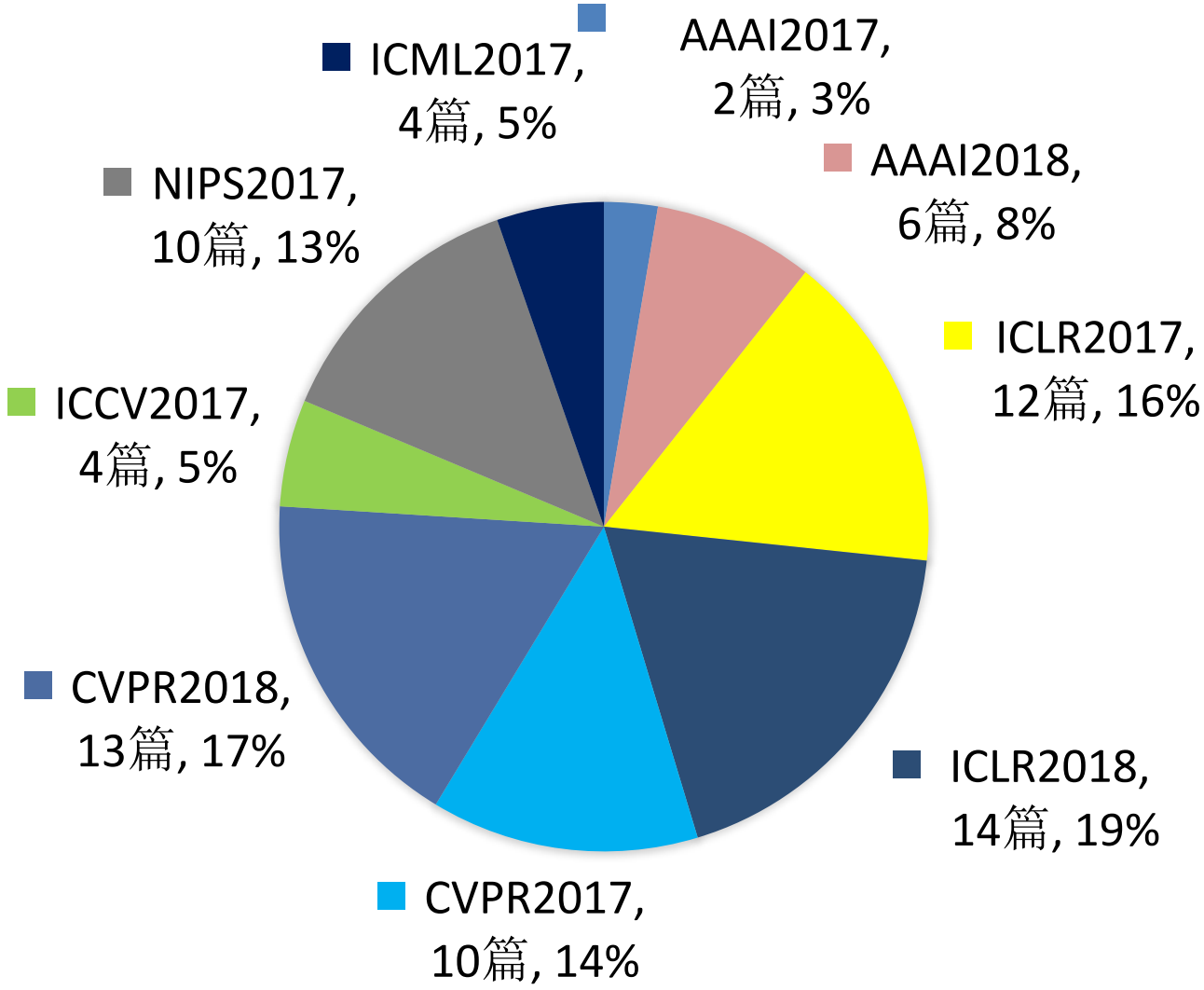
穿戴设备：算不了

数据中心：算不起

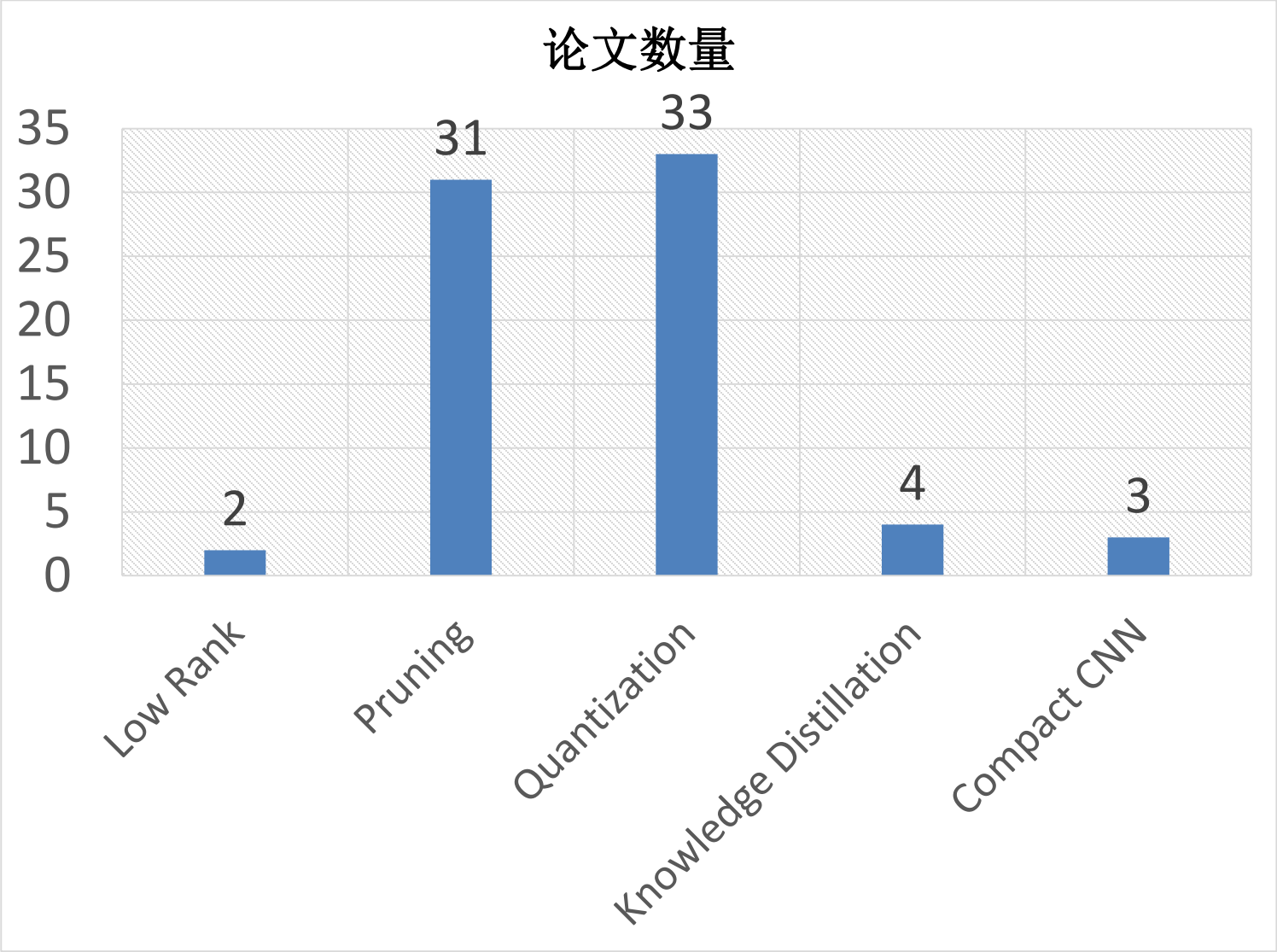
模型加速与压缩方法分类

- Low-Rank
- Pruning
- Quantization
- Knowledge Distillation
- Compact Network Design

会议论文分布


AAAI2017, 2篇, 3%
AAAI2018, 6篇, 8%
ICLR2017, 12篇, 16%
ICLR2018, 14篇, 19%
CVPR2017, 10篇, 14%
CVPR2018, 13篇, 17%
ICCV2017, 4篇, 5%
NIPS2017, 10篇, 13%
ICML2017, 4篇, 5%

会议论文分布



- Quantization becomes popular
  - *efficient training using quantization*
  - *low-bit representation*
  - *binary convolutional neural networks*
- Pruning is still a hot topic
  - *small accuracy drop*
  - *efficient structured pruning*
- Few low-rank based method
  - *tensor decomposition is not efficient for current network structure*

Low-Rank

Previous low-rank based methods:

- SVD
  - *Zhang et al., "Accelerating Very Deep Convolutional Networks for Classification and Detection". IEEE TPAMI 2016.*
- CP decomposition
  - *Lebedev et al., "Speeding-up Convolutional Neural Networks Using Fine-tuned CP-Decomposition". ICLR 2015.*
- Tucker decomposition
  - *Kim et al., "Compression of Deep Convolutional Neural Networks for Fast and Low Power Mobile Applications". ICLR 2016.*
- Tensor Train Decomposition
  - Novikov *et al., "Tensorizing Neural Networks". NIPS 2016.*
- Block Term Decomposition
  - *Wang et al., "Accelerating Convolutional Neural Networks for Mobile Applications". ACM MM 2016.*

Low-Rank

Recent low-rank based methods:

- Tensor Ring (TR) factorizations
  - *Wang et al., "*Wide Compression: Tensor Ring Nets". CVPR2018
- Block Term Decomposition For RNN
  - *Ye et al., "Learning Compact Recurrent Neural Networks with Block-Term Tensor Decomposition ". CVPR2018.*

Why low-rank is not popular anymore?

- Low-rank approximation is not efficient for those 1x1 convolutions
- 3x3 convolutions in bottleneck structure have less computation complexity
- Depthwise convolution or grouped 1x1 convolution is already quite fast.

Pruning

Recent progress in pruning :

- Structured Pruning
  - *Yoon et al. “Combined Group and Exclusive Sparsity for Deep Neural Networks”. ICML2017*
  - *Ren et al. “SBNet: Sparse Blocks Network for Fast Inference”. CVPR2018*
- Filter Pruning
  - *Luo et al. “Thinet: A filter level pruning method for deep neural network compression”. ICCV2017*
  - *Liu et al., “Learning efficient convolutional networks through network slimming”. ICCV2017*
  - *He et al. “Channel Pruning for Accelerating Very Deep Neural Networks”. ICCV2017*
- Gradient Pruning
  - *Sun et al. “meProp: Sparsified Back Propagation for Accelerated Deep Learning with Reduced Overfitting”. ICML2017*
- Fine-grained Pruning in a Bayesian View
  - *Molchanov et al. “Variational Dropout Sparsifies Deep Neural Networks”. ICML2017*

Structured Pruning

Previous group pruning methods mainly use the group sparsity， Yoon et al. use both group sparsity and exclusive sparsity

- *Group Sparsity: Impose sparsity regularization on grouped features to prune columns of weight matrix.*

$$\Omega(\boldsymbol{W}^{(l)}) = \sum_g \|\boldsymbol{W}_g^{(l)}\|_2 = \sum_g \sqrt{\sum_i {w_{g,i}^{(l)}}^2}$$

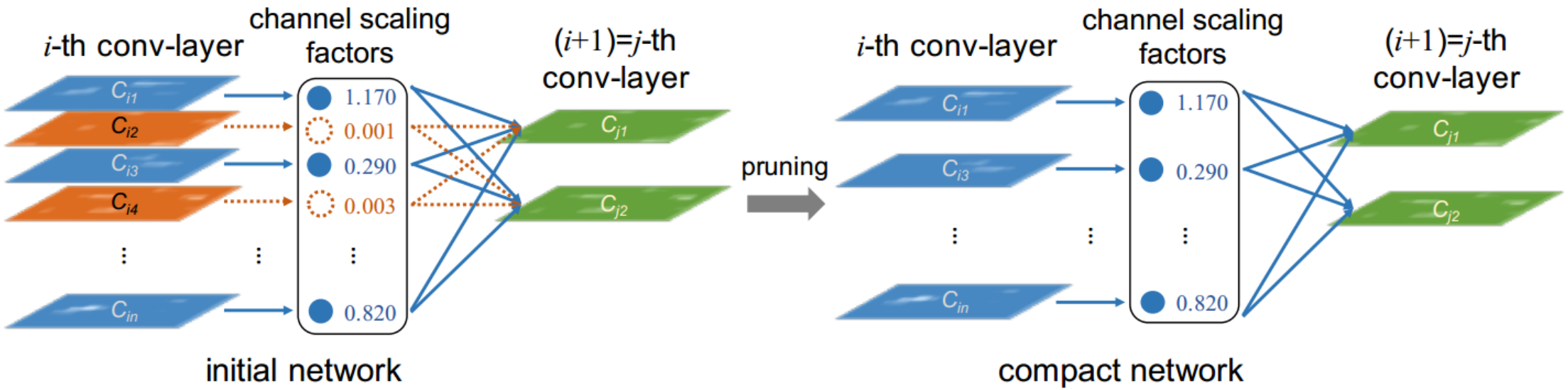
- *Exclusive Sparsity:* promotes competition for features between different weights to learn effective filters

$$\Omega(\boldsymbol{W}^{(l)}) = \frac{1}{2}\sum_g \|\boldsymbol{W}_g^{(l)}\|_1^2 = \frac{1}{2}\sum_g \left(\sum_i |w_{g,i}^{(l)}|\right)^2$$

Yoon et al. “Combined Group and Exclusive Sparsity for Deep Neural Networks”. ICML2017

Network Slimming

- *Associate a scaling factor with each channel*
- *Impose sparsity regularization on these scaling factors*
- *Prune those channels with small scaling factors*

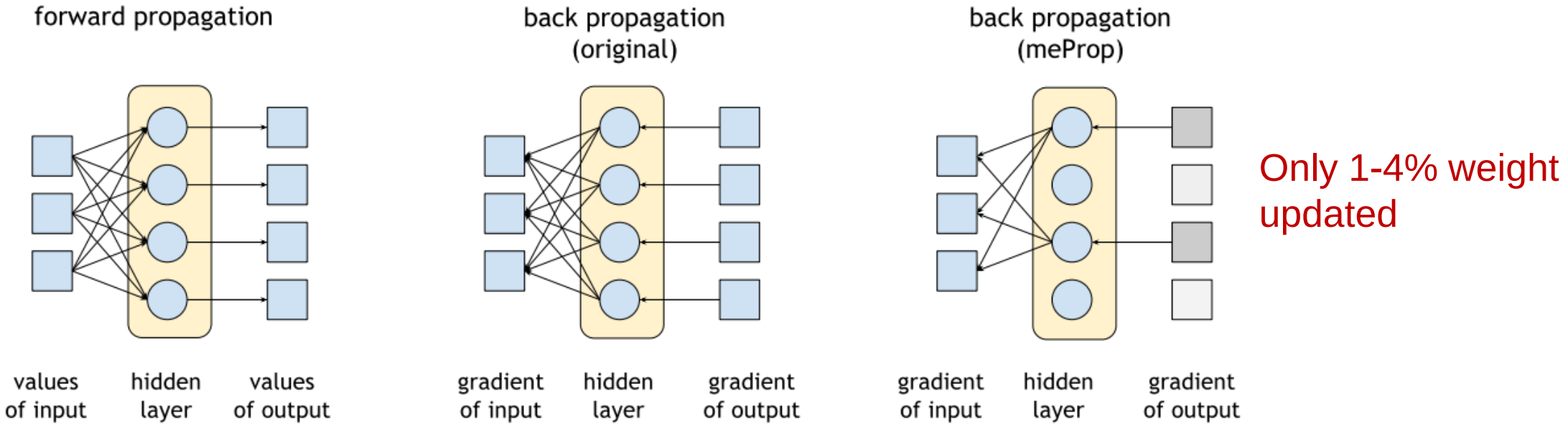


Liu et al., "Learning efficient convolutional networks through network slimming". ICCV2017

meProp

Previous pruning methods are mainly to prune weights, meProp is proposed to prune gradients to speed up training phase.

- Keep top-*k gradients of neural nodes , and prune rest of them*
- *Since some neural nodes' gradients are zeros, back propagation can be accelerated.*



Only 1-4% weight updated

Sun et al. "meProp: Sparsified Back Propagation for Accelerated Deep Learning with Reduced Overfitting". ICML2017

Quantization

- Low-bit Quantization
  - *Leng et al. "Extremely Low Bit Neural Network: Squeeze the Last Bit Out with ADMM". AAAI2018*
  - *Hu et al. "From Hashing to CNNs: Training Binary Weight Networks via Hashing". AAAI2018*
  - *Wang et al. "A General Two-Step Quantization Approach for Low-bit Neural Networks with High Accuracy". CVPR2018*
- Quantization for general training acceleration
  - *Köster et al. "Flexpoint: An Adaptive Numerical Format for Efficient Training of Deep Neural Networks". NIPS2017*
- Gradient Quantization for distributed training
  - *Alistarh et al. "QSGD: Communication-Efficient SGD via Gradient Quantization and Encoding". NIPS2017*
  - *Wen et al. "TernGrad: Ternary Gradients to Reduce Communication in Distributed Deep Learning". NIPS2017*

Low-bit Quantization via ADMM

Since quantization function is non-differentiable, most low-bit quantization method approximate the gradients of weights by “Straight-through” estimator. Recently Leng et al. proposed to train low-bit networks via ADMM.

They formulate neural networks with low-bit constraint on W as follows:

$$\min_{W} \quad f(W) + I_{\mathcal{C}}(W)$$

$$\text{where } I_{\mathcal{C}}(W) = 0 \text{ if } W \in \mathcal{C}, \text{ otherwise } I_{\mathcal{C}}(W) = +\infty.$$

By introducing an auxiliary variable G

$$\begin{aligned} \min_{W,G} \quad & f(W) + I_{\mathcal{C}}(G) \\ \text{s.t.} \quad & W = G \end{aligned}$$

Leng et al. “Extremely Low Bit Neural Network: Squeeze the Last Bit Out with ADMM”. AAAI2018

Binary Weight Network via Hashing

- Solve the binary weight via hashing
- Minimize the quantization error of inner-product similarity:

$$\min \ L(B) = ||X^T W - X^T B||_F^2$$
$$s.t. \ \ B \in \{+1, -1\}^{C \times N}$$

By a few transformations:

$$\text{Let } S = X^T W, B = g(W), \text{and } h(X) = X, \text{ then:}$$

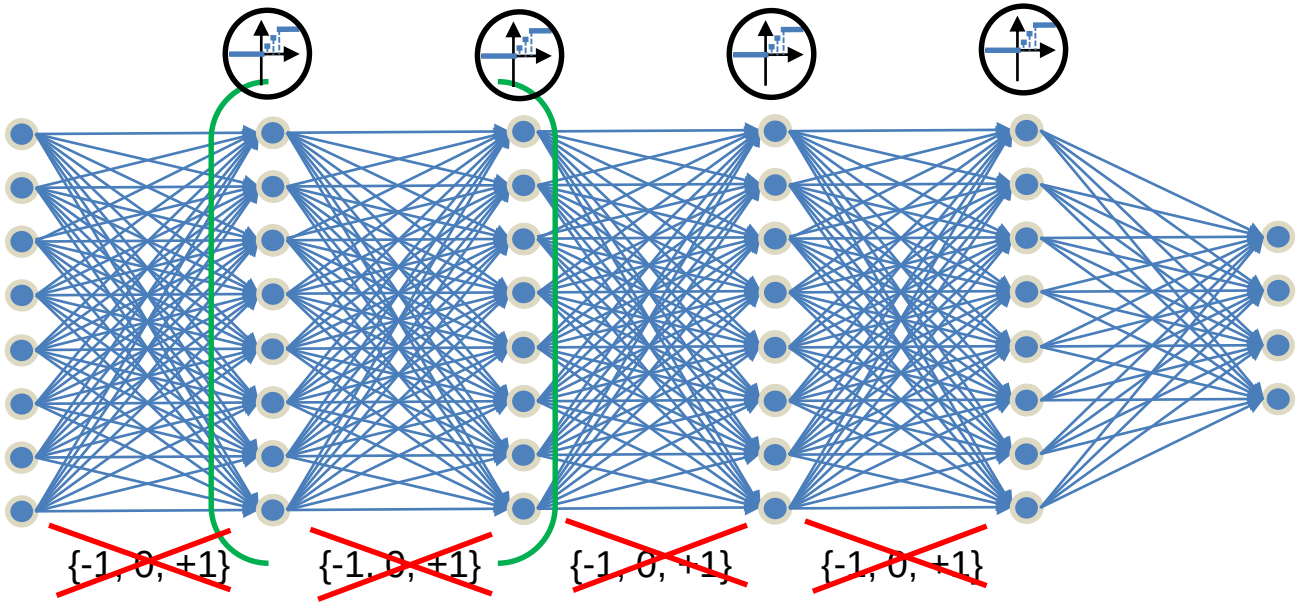
Turns out a Inner Product Preserving Hashing problem:

$$\min ||S - h(X)^T g(W)||_F^2$$

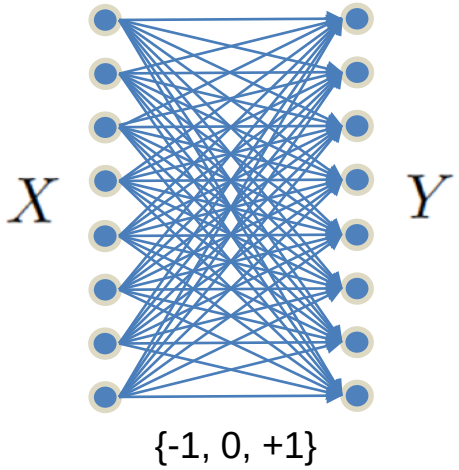
Hu et al. "From Hashing to CNNs: Training Binary Weight Networks via Hashing". AAAI2018

Two-Step Quantization

Stage-1



Stage-2



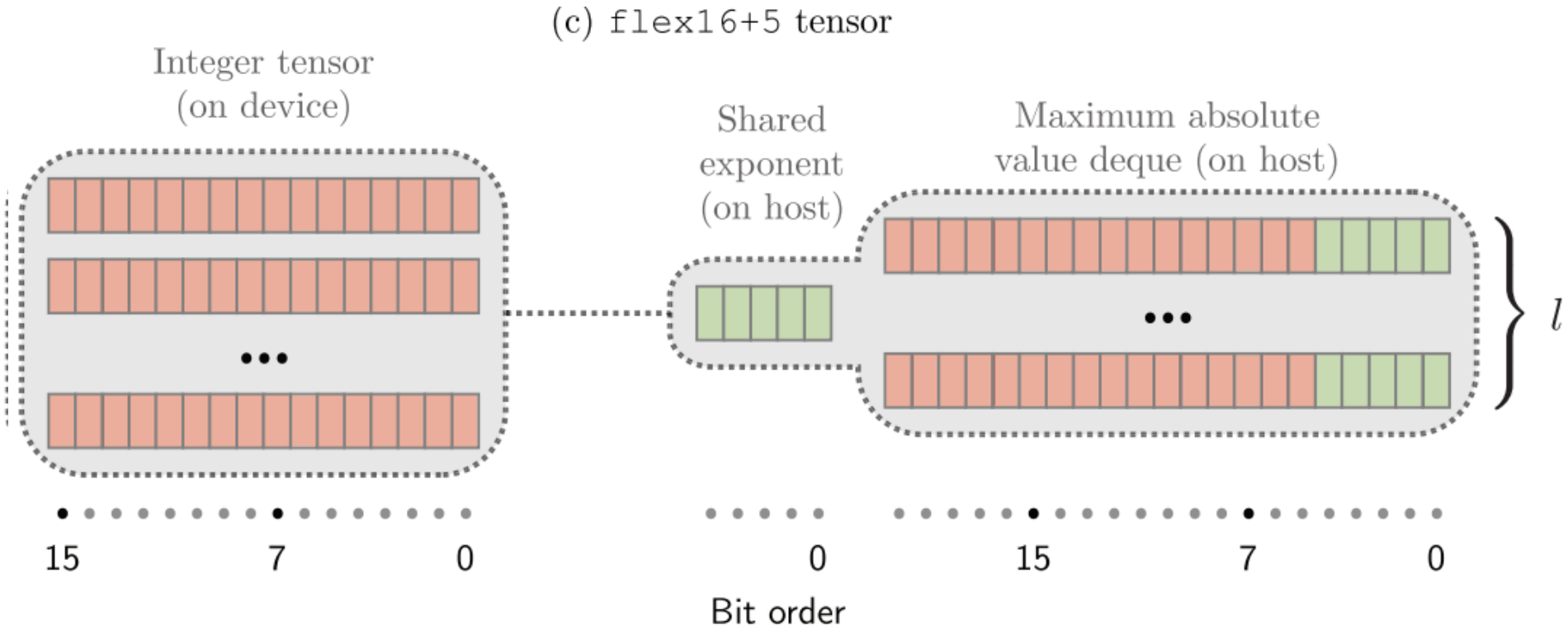
$$\begin{aligned} &\underset{\Lambda,\hat{W}}{\text{minimize}} \quad \| Y - Q_\epsilon(\Lambda \hat{W} X) \|_F^2 \\ =&\underset{\{\alpha_i\},\{\hat{w}_i^T\}}{\text{minimize}} \quad \sum_i \| y_i^T - Q_\epsilon(\alpha_i \hat{w}_i^T X) \|_2^2 \end{aligned}$$

- Quantize both weight and actition is diffifult
- Inspired by Two-Step Hashing, Two-Step quantization method first quantizes the activation, then quantize the weights

Wang et al.“A General Two-Step Quantization Approach for Low-bit Neural Networks with High Accuracy”.CVPR2018

Flexpoint

- Previous network quantization methods mostly focus on the inference phase, Flexpoint is an effective method for training.
- Core idea is exponent sharing, data in the same tensor share the same exponent.



Köster et al. "Flexpoint: An Adaptive Numerical Format for Efficient Training of Deep Neural Networks". NIPS2017

Gradient Quantization for Distributed Deep Learning

- Distributed SGD is fast, but it is limited by communication cost
- gradients communication between servers and workers is expensive.
- Reduce the communication cost by gradient quantization

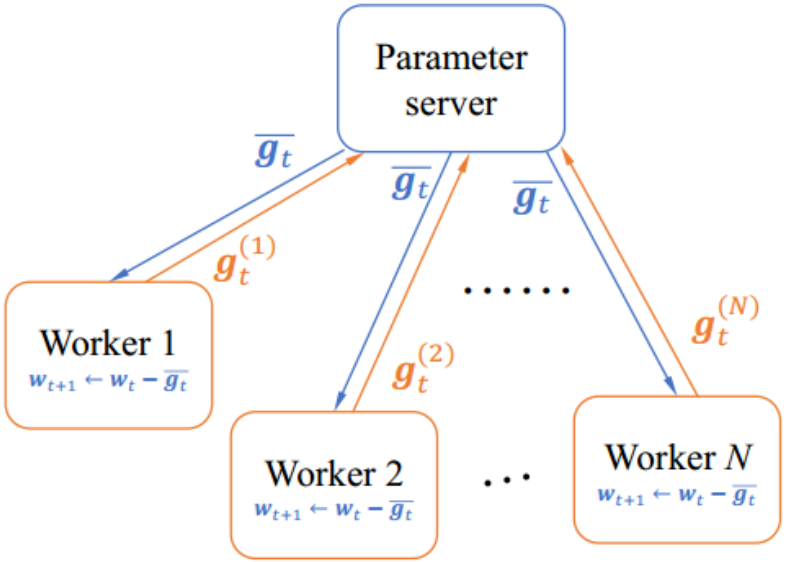


Figure 1: Distributed SGD with data parallelism.

Algorithm 1 *TernGrad*: distributed SGD training using ternary gradients.

Worker : $i = 1, ..., N$

1 Input $\boldsymbol{z}_t^{(i)}$, a part of a mini-batch of training samples $\boldsymbol{z}_t$

2 Compute gradients $\boldsymbol{g}_t^{(i)}$ under $\boldsymbol{z}_t^{(i)}$

3 Ternarize gradients to $\tilde{\boldsymbol{g}}_t^{(i)} = ternarize(\boldsymbol{g}_t^{(i)})$

4 Push ternary $\tilde{\boldsymbol{g}}_t^{(i)}$ to the server

5 Pull averaged gradients $\overline{\boldsymbol{g}_t}$ from the server

6 Update parameters $\boldsymbol{w}_{t+1} \leftarrow \boldsymbol{w}_t - \eta \cdot \overline{\boldsymbol{g}_t}$

Server :

7 Average ternary gradients $\overline{\boldsymbol{g}_t} = \sum_i \tilde{\boldsymbol{g}}_t^{(i)} / N$

Wen et al. "TernGrad: Ternary Gradients to Reduce Communication in Distributed Deep Learning". NIPS2017

Knowledge Distillation

Two key steps:

- Define knowledge
- Define optimization loss which learns knowledge from teacher network

Previous Methods:

- KD (Knowledge Distillation)
  - Hinton et al. “Distilling the knowledge in a neural network”. *arXiv preprint arXiv:1503.02531*.
- FitNets
  - *Romero et al. “*Fitnets: Hints for thin deep nets”.ICLR 2015

Knowledge Distillation

Recent progress in Knowledge Distillation:

- *Yim et al. "A Gift From Knowledge Distillation: Fast Optimization, Network Minimization and Transfer Learning ". CVPR2017*
- *Zagoruyko et al. "Pay More Attention to Attention: Improving the Performance of Convolutional Neural Networks via Attention Transfer". ICLR2017*
- *Chen et al. "Learning Efficient Object Detection Models with Knowledge Distillation". NIPS2017*

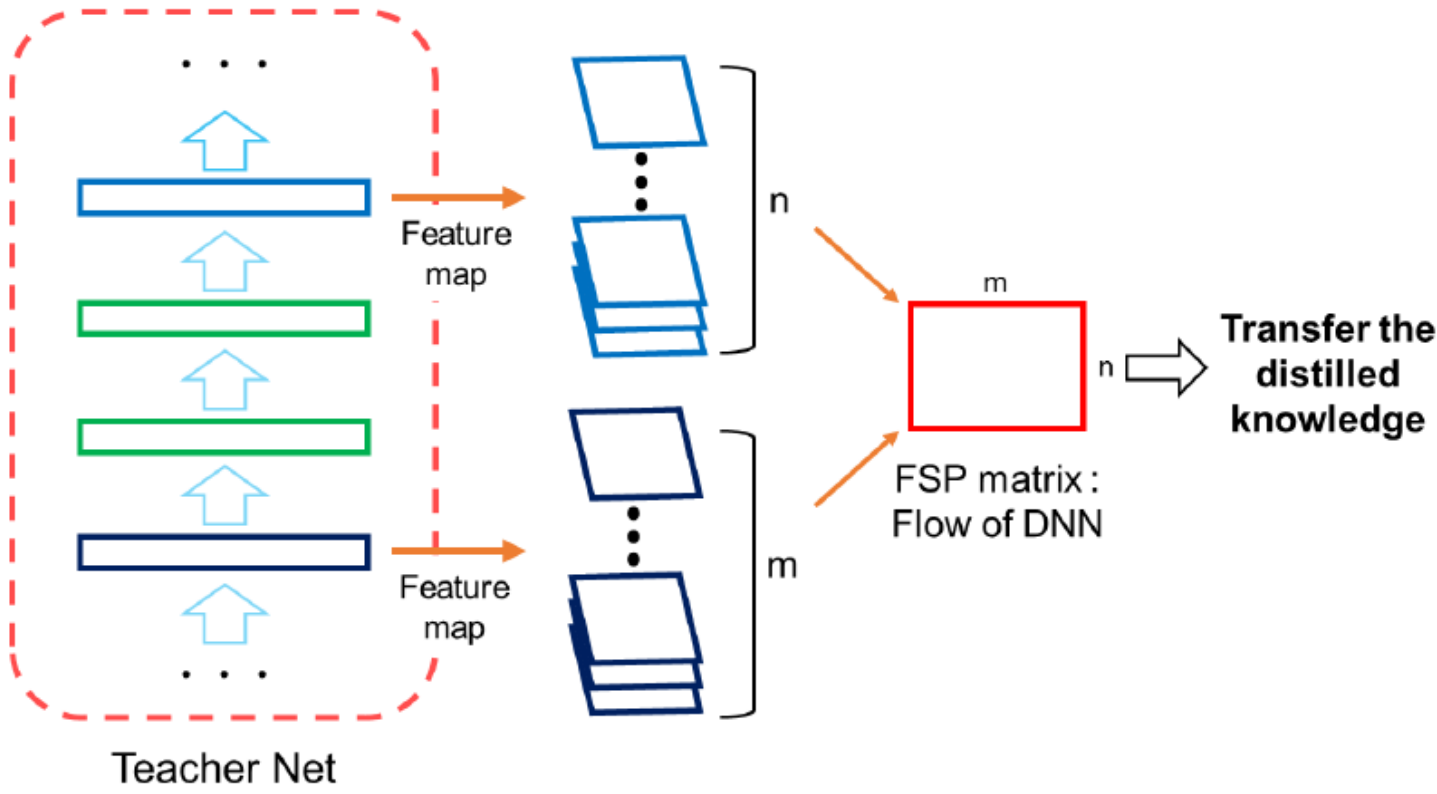
Knowledge Distillation

- Knowledge Definition: (FSP) flow of solution procedure matrix

$$G_{i,j}(x;W) = \sum_{s=1}^{h}\sum_{t=1}^{w} \frac{F^1_{s,t,i}(x;W) \times F^2_{s,t,j}(x;W)}{h \times w}$$

- Loss Definition: L2 loss

$$L_{FSP}(W_t, W_s) = \frac{1}{N}\sum_{x}\sum_{i=1}^{n} \lambda_i \times \|(G^T_i(x;W_t) - G^S_i(x;W_s)\|^2_2,$$



Yim et al. "A Gift From Knowledge Distillation: Fast Optimization, Network Minimization and Transfer Learning ". CVPR2017.

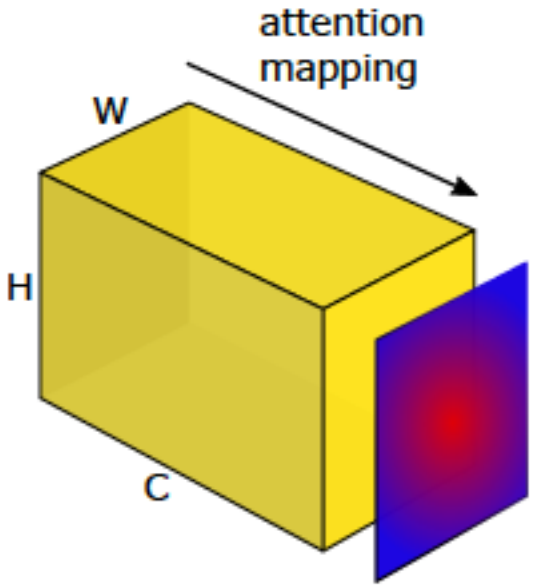
Knowledge Distillation

- Knowledge Definition: activation based attention

$$F_{\text{sum}}(A) = \sum_{i=1}^{C} |A_i|$$

$$F_{\text{sum}}^{p}(A) = \sum_{i=1}^{C} |A_i|^p$$

$$F_{\text{max}}^{p}(A) = \max_{i=1,C} |A_i|^p$$



- Loss Definition: Normalized L2 loss

$$\sum_{j \in \mathcal{I}} \left\| \frac{Q_S^j}{\|Q_S^j\|_2} - \frac{Q_T^j}{\|Q_T^j\|_2} \right\|_p$$

Zagoruyko et al. "Pay More Attention to Attention: Improving the Performance of Convolutional Neural Networks via Attention Transfer". ICLR2017

Knowledge Distillation

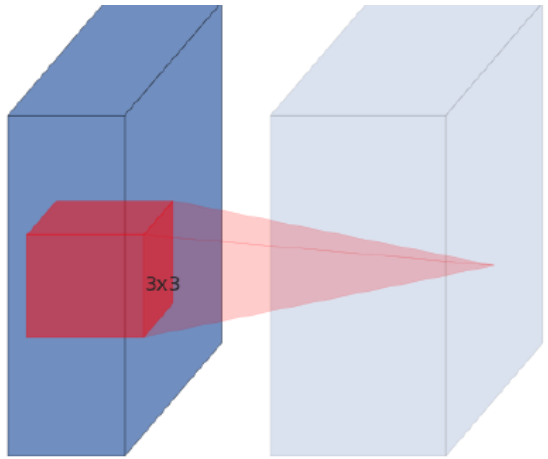
Method	Knowledge	Loss
KD	Soften Softmax output	KL divergence
FitNets	Intermediate activation	L2 loss
Yim et al	FSP(flow of solution procedure)	L2 loss
Zagoruyko et al.	Featuremap Attention	Normalized L2 Loss

Compact Network Design

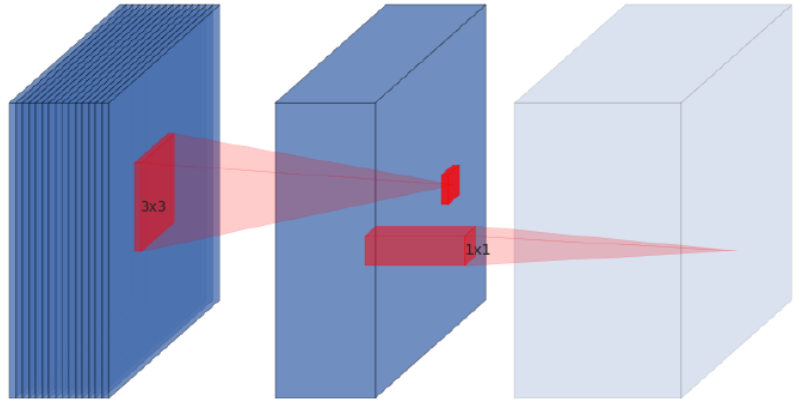
- *Howard et al. "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications". CVPR2017*
- *Sandler et al. "Inverted Residuals and Linear Bottlenecks: Mobile Networks for Classification, Detection and Segmentation". CVPR2018*
- *Zhang et al. ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices. CVPR2018*

MobileNet

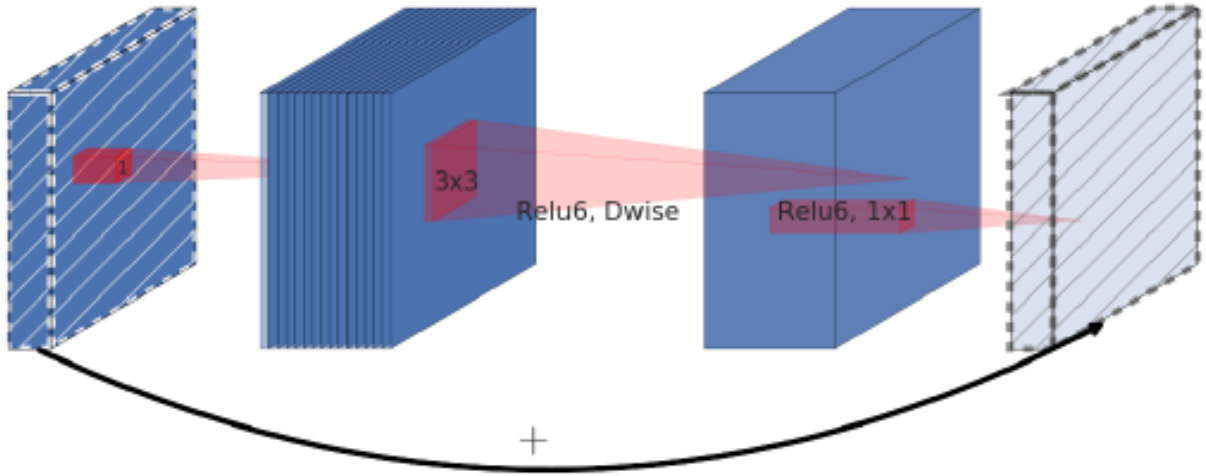
Regular Conv



MobileNet V1



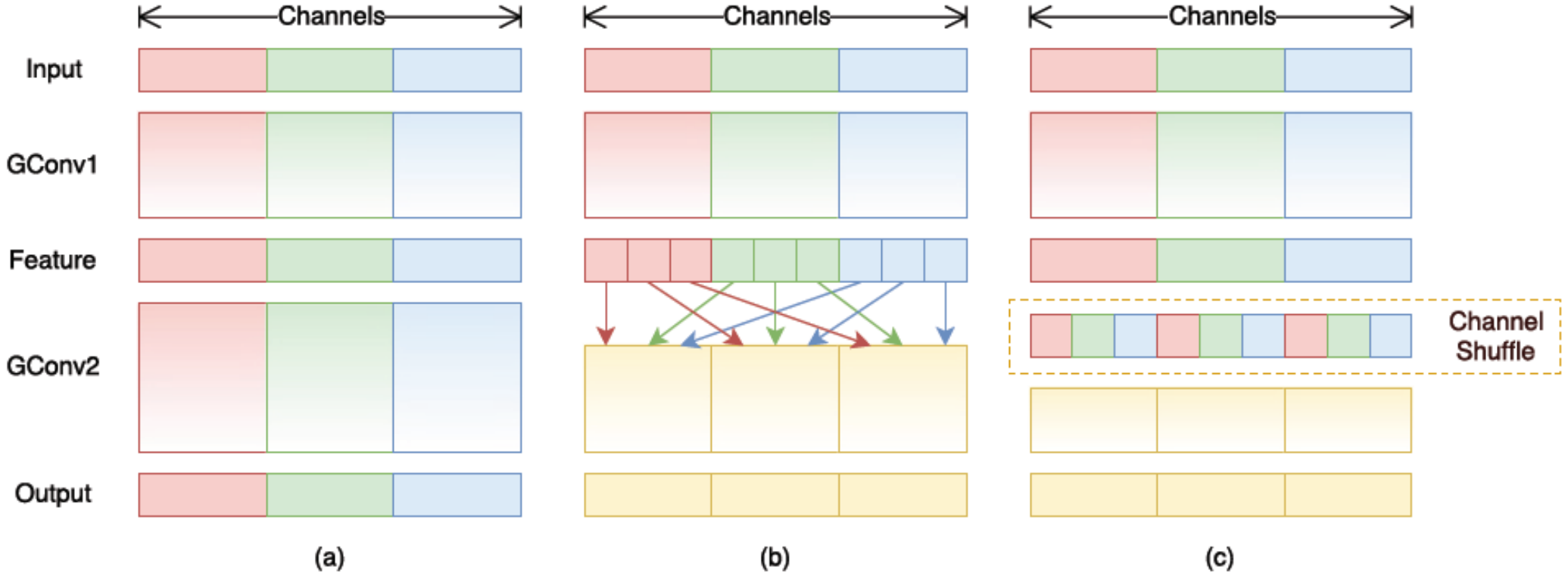
MobileNet V2



Howard et al. "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications". CVPR2017
Sandler et al. "Inverted Residuals and Linear Bottlenecks: Mobile Networks for Classification, Detection and Segmentation". CVPR2018

ShuffleNet

- *Grouped 1x1 convolution*
- *Channel shuffle to enhance Information flow across channels*



Zhang et al. ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices. CVPR2018

Comparison Result

Network	Top 1	Params	MAdds	CPU
MobileNetV1	70.6	4.2M	575M	123ms
ShuffleNet (1.5)	69.0	**2.9M**	292M	-
ShuffleNet (x2)	70.9	4.4M	524M	-
NasNet-A	74.0	5.3M	564M	192ms
MobileNetV2	**71.7**	**3.4M**	**300M**	**80ms**
MobileNetV2 (1.4)	**74.7**	6.9M	585M	**149ms**

Future Trends

- Non-fine-tuning or Unsupervised Compression
- Self-adaptive Compression
- Network Acceleration for other tasks
- Hardware-Software Co-design
- Binarized Neural Networks

Recent Advances in Efficient Computation of Deep Convolutional Neural Networks. Frontiers of Information Technology & Electronic Engineering (FITEE), Vol.19, No.1, pp.64-77, 2018

谢谢大家！

Q&A

Email: jcheng@nlpr.ia.ac.cn
主页： www.nlpr.ia.ac.cn/jcheng
www.airia.cn