名企 Project-I/II

AI for CV Group
2021

Week 7-8
Acceleration

Contents:

I. Introduction

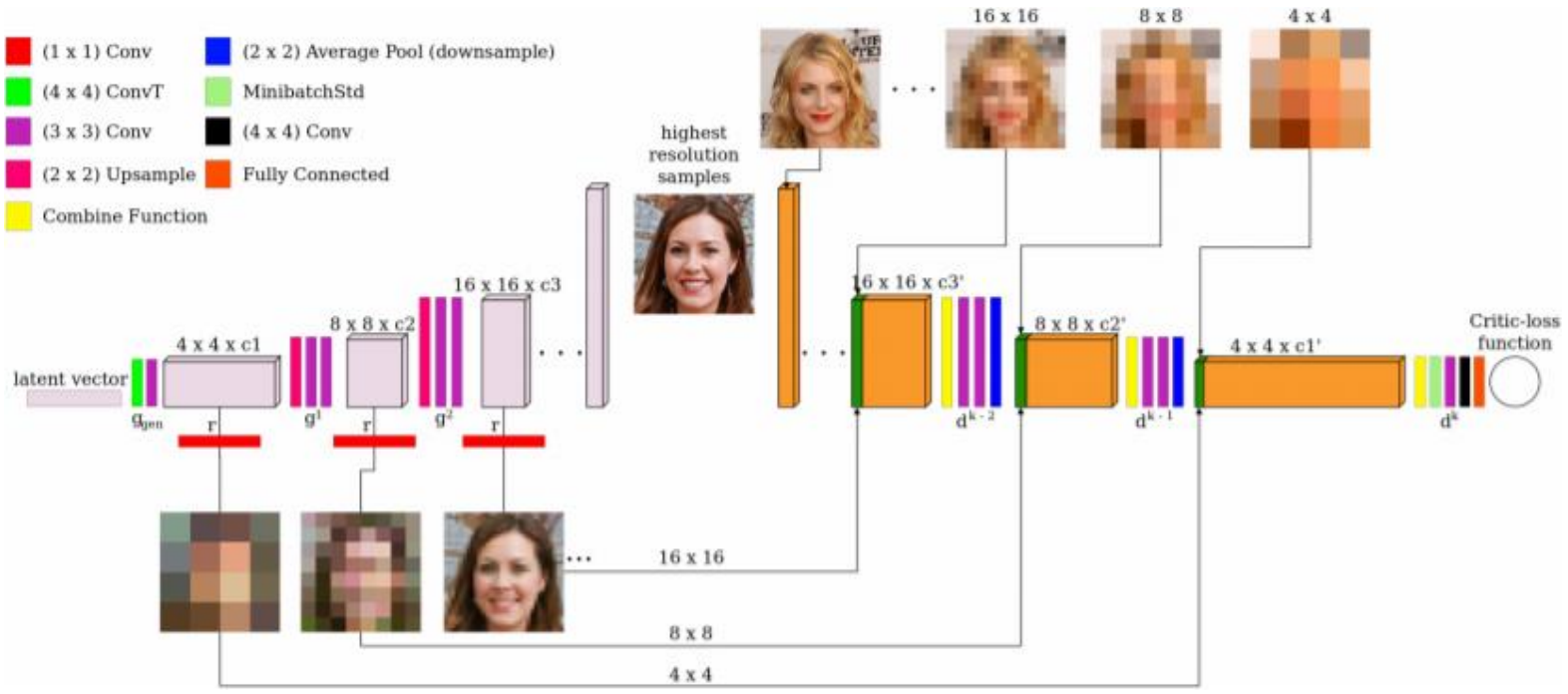
I. Introduction

A. Target

a. **Increase inference speed**

b. **Reduce model size**

c. **A good survey [2020, Cheng]**



I. Introduction

B. Methods

B1. Mathematical Operation

B1.1: Winograd [2015, Lavin]

- Firstly promoted in 1980, 1-dim
- 2015, 2d-conv, small size kernel: 2x2, 3x3, 4x4
- Convert "*" operations to "+" operations

$$\begin{bmatrix} e \\ f \end{bmatrix} = \begin{bmatrix} c & -d \\ d & c \end{bmatrix} \cdot \begin{bmatrix} a \\ b \end{bmatrix}$$

$$originally: e = a \cdot c - b \cdot d \quad (4\ \text{“*”}, 2\ \text{“+”})$$
$$f = a \cdot d + b \cdot c$$

$$converted: e = a \cdot c - b \cdot d = a \cdot (c - d) + d \cdot (a - b)$$
$$f = a \cdot d + b \cdot c = b \cdot (c + d) + d \cdot (a - b)$$
$$(3\ \text{“*”}, 5\ \text{“+”})$$

I. Introduction

B. Methods

B1. Mathematical Operation

B1.2: Low-rank factorization

- Matrix decomposition (SVD)
- Apply to FC / Conv layers
- Good examples:

  Fully-adaptive Feature Sharing in Multi-Task Networks with Applications in Person Attribute Classification
  [2016, Lu]

I. Introduction

B. Methods

B2. Network Design:
MobileNet / ShuffleNet / Efficient Net / GhostNet

B2.1: MobileNet Series

B2.2: ShuffleNet Series

B2.3: Efficient Net Series

B2.4: GhostNet

I. Introduction

B. Methods

B2. Network Design: MobileNets / ShuffleNet / Efficient Net

B2.1: MobileNets Series

- v1 [2017, Howard], v2 [2018, Sandler], v3 [2019, Howard]

- Depthwise Conv
  Inverted Residuals + Linear Bottleneck + R-ASPP (Reduced-Atrous Spatial Pyramid Pooling) [2017, Chen]
  MNasNet [2019, Tan] + NetAdapt [2018, Yang] + v2 + SENet [2017-2019, v1-v4, Hu] + h-swish + LR-ASPP (Lite Reduced-…)

- Depthwise Conv (v1):

I. Introduction

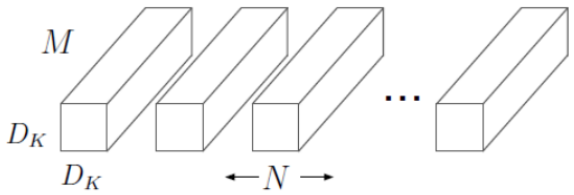
B. Methods

B2. Network Design

B2.1: MobileNets Series

- **Depthwise Conv (v1):**

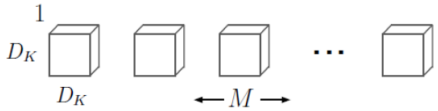
Original Convolution:



(a) Standard Convolution Filters

Image size: $D_F \cdot D_F \cdot M$
Kernel size: $D_k \cdot D_k \cdot M \cdot N$
Operations: $D_k \cdot D_k \cdot M \cdot N \cdot D_F \cdot D_F \equiv Ori$

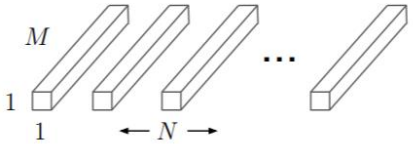
Depthwise Convolution:



(b) Depthwise Convolutional Filters

Step1: Image size: $D_F \cdot D_F \cdot M$
Depthwise Kernel size: $D_k \cdot D_k \cdot 1 \cdot M$
Operations: $D_k \cdot D_k \cdot M \cdot D_F \cdot D_F$

This step "only filters input channels, it does not combine them to create new features." So a 1x1 conv is needed to create new features by computing a linear combination.



(c) 1 × 1 Convolutional Filters called Pointwise Convolution in the context of Depthwise Separable Convolution

Step2: Input size: $D_F \cdot D_F \cdot M$
Pointwise Kernel size: $1 \cdot 1 \cdot M \cdot N$
Operations: $M \cdot N \cdot D_F \cdot D_F$

Total: Operations: $D_k \cdot D_k \cdot M \cdot D_F \cdot D_F + M \cdot N \cdot D_F \cdot D_F \equiv Dep$
How Small:
$$\frac{Dep}{Ori} = \frac{1}{N} + \frac{1}{D_k^2}$$

I. Introduction

B. Methods

B2. Network Design

B2.1: MobileNets Series

- **Depthwise Conv (v1):**

Original Convolution:

D_F=112, M=64, N=128, $D_k = 3$
Image size: $D_F \cdot D_F \cdot M$
Kernel size: $D_k \cdot D_k \cdot M \cdot N$
Operations:
$D_k \cdot D_k \cdot M \cdot N \cdot D_F \cdot D_F = 924844032$

Depthwise Convolution:

Operations:
$D_k \cdot D_k \cdot M \cdot D_F \cdot D_F + M \cdot N \cdot D_F \cdot D_F$=109985792

$$\frac{Dep}{Ori} = \frac{1}{N} + \frac{1}{D_k^2}$$

$$=\frac{109985792}{924844032}$$

=0.1189

Usually, it can be 8~9 times less computation than standard convolutions with only about 1% loss of accuracy.

I. Introduction

B. Methods

B2. Network Design

B2.1: MobileNets Series

- **Inverted Residuals + Linear Bottleneck(v2):**

First impression: very little parameters, very slim blobs

心灵拷问: Then how could it be better?

人话版:
肯定在每层中间增加channel数来增加参数数量，然后在下一层前再缩回去。

高逼格版:
Inverted Residuals
+
Linear Bottlenecks

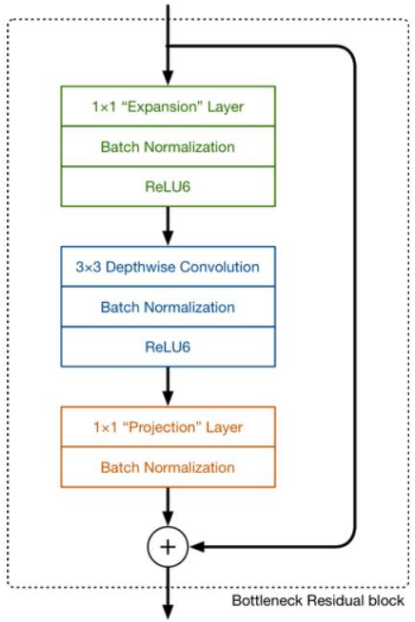
I. Introduction

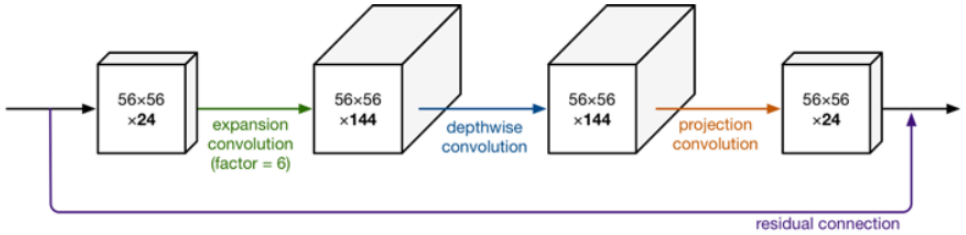
B. Methods

B2. Network Design

B2.1: MobileNets Series

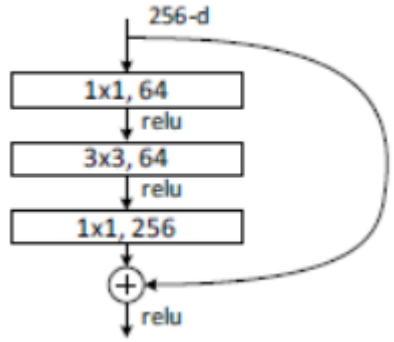
- Inverted Residuals + Linear Bottleneck(v2):





Inverted residual: expand - transfer – reduce

Linear Bottlenecks: Eltwise+ with NO ReLU at the end of a bottleneck



Original residual:
reduce - transfer - expand

Original bottlenecks:
Eltwise+ with ReLU at the end of a bottleneck

I. Introduction

B. Methods

B2. Network Design

B2.1: MobileNets Series

- **Inverted Residuals + Linear Bottleneck(v2):**

人话解释:

Inverted Residuals:
Skip connection 这种bottleneck的结构被证明很有效，所以想用;
但是如果像以前那样先压缩channel，channel数本来自就少，再压没了，所以不如先增大再减少。

Linear Bottlenecks:
ReLU让负半轴为0。本来我参数就不多，学习能力就有限，这一下再让一些参数为0了，就更学不着什么东西了。干脆在eltwise+那里不要ReLU了

Manifold：流形
局部具有欧式空间性质的空间

品质解释:

Inverted Residuals:
作用是uncompress数据，使得我们感兴趣的低维流形(manifold of interest)能够包含于我们的高维空间中。

Linear Bottlenecks:
神经网络被认为可以使MOI嵌入低维空间。如果当前激活空间MOI完整度较高，ReLU会让空间坍塌，丢失信息；并且，其非0部分是做线性变换，实为一个线性分类器。因而采用线性bottleneck

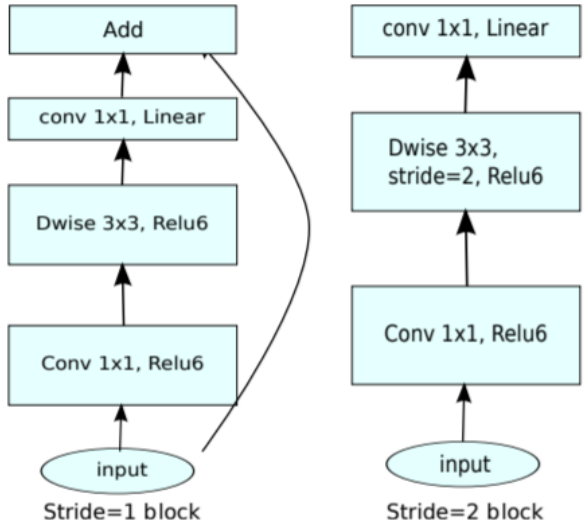
I. Introduction

B. Methods

B2. Network Design

B2.1: MobileNets Series

- Inverted Residuals + Linear Bottleneck(v2):



Use "stride 2 block" to reduce dimension. So no skip connection.

Input	Operator	t	c	n	s
$224^2 \times 3$	conv2d	-	32	1	2
$112^2 \times 32$	bottleneck	1	16	1	1
$112^2 \times 16$	bottleneck	6	24	2	2
$56^2 \times 24$	bottleneck	6	32	3	2
$28^2 \times 32$	bottleneck	6	64	4	2
$14^2 \times 64$	bottleneck	6	96	3	1
$14^2 \times 96$	bottleneck	6	160	3	2
$7^2 \times 160$	bottleneck	6	320	1	1
$7^2 \times 320$	conv2d 1x1	-	1280	1	1
$7^2 \times 1280$	avgpool 7x7	-	-	1	-
$1 \times 1 \times 1280$	conv2d 1x1	-	k	-	

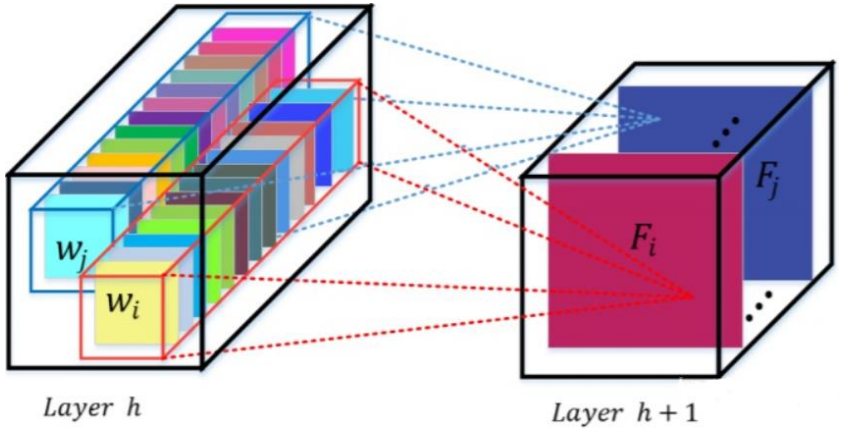
I. Introduction

B. Methods

B2. Network Design

B2.1: MobileNets Series

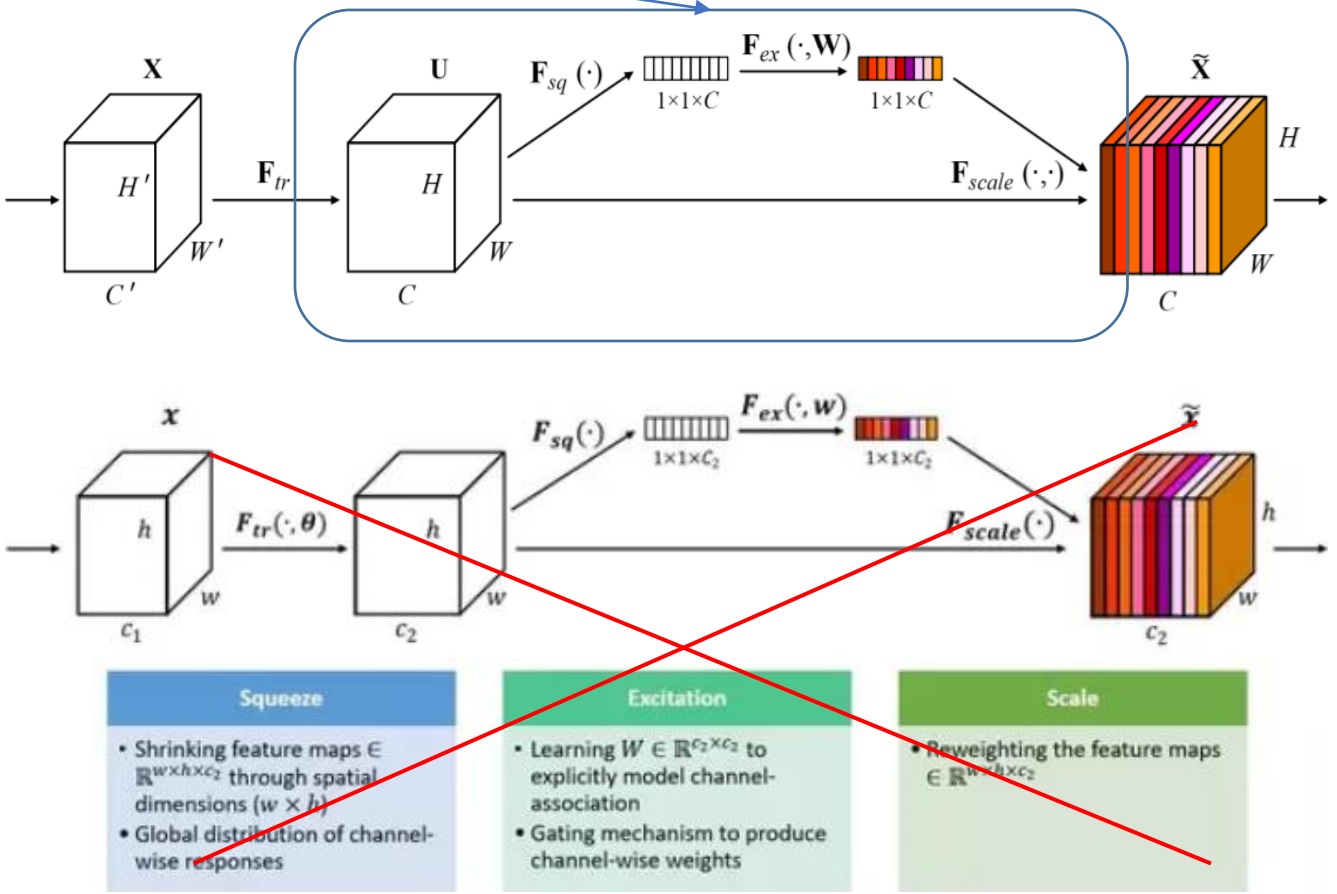
- SENet (v3): pdf, code



➢ Original Conv: Spatial Info Condense

➢ SENet: Get importance among channels

➢ Squeeze-Excitation Block



I. Introduction

B. Methods

B2. Network Design

B2.1: MobileNets Series

- SENet (v3): pdf, code

➢ Squeeze-Excitation Block

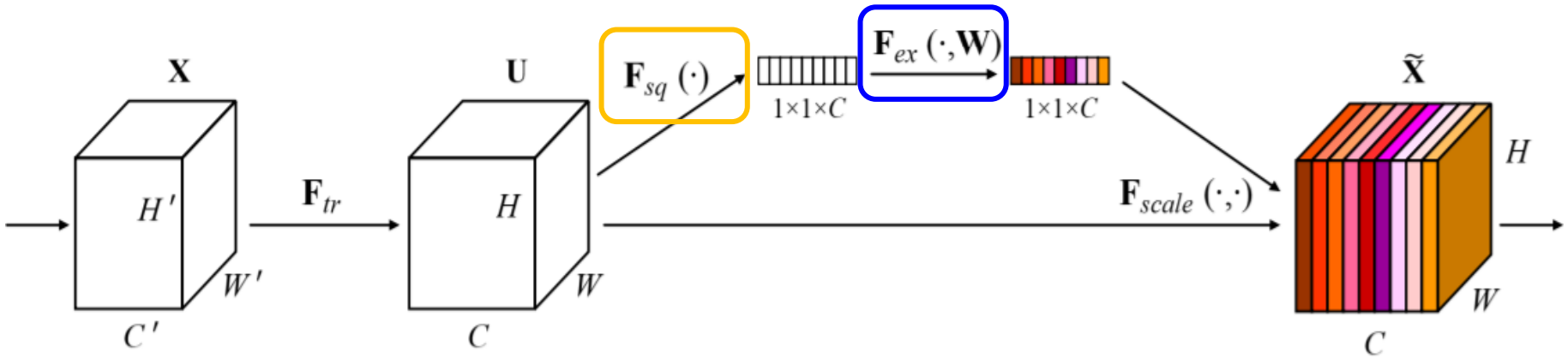
Sigmoid – *FC* – *ReLU* – *FC*:

$$\mathbf{s} = \mathbf{F}_{ex}(\mathbf{z}, \mathbf{W}) = \sigma(g(\mathbf{z}, \mathbf{W})) = \sigma(\mathbf{W}_2\delta(\mathbf{W}_1\mathbf{z}))$$

$$\mathbf{W}_1 \in \mathbb{R}^{\frac{C}{r}\times C} \quad \mathbf{W}_2 \in \mathbb{R}^{C\times\frac{C}{r}}$$

Ratio r	top-1 err.	top-5 err.	Params
2	22.29	6.00	45.7M
4	22.25	6.09	35.7M
8	22.26	5.99	30.7M
16	22.28	6.03	28.1M
32	22.72	6.20	26.9M
original	23.30	6.55	25.6M

Global Ave. Pooling: $z_c = \mathbf{F}_{sq}(\mathbf{u}_c) = \frac{1}{H \times W}\sum_{i=1}^{H}\sum_{j=1}^{W} u_c(i,j)$



I. Introduction

B. Methods

B2. Network Design

B2.1: MobileNets Series

- SENet (v3): pdf, code

➢ Squeeze-Excitation Block

Global Ave. Pooling: $z_c = \mathbf{F}_{sq}(\mathbf{u}_c) = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} u_c(i,j)$

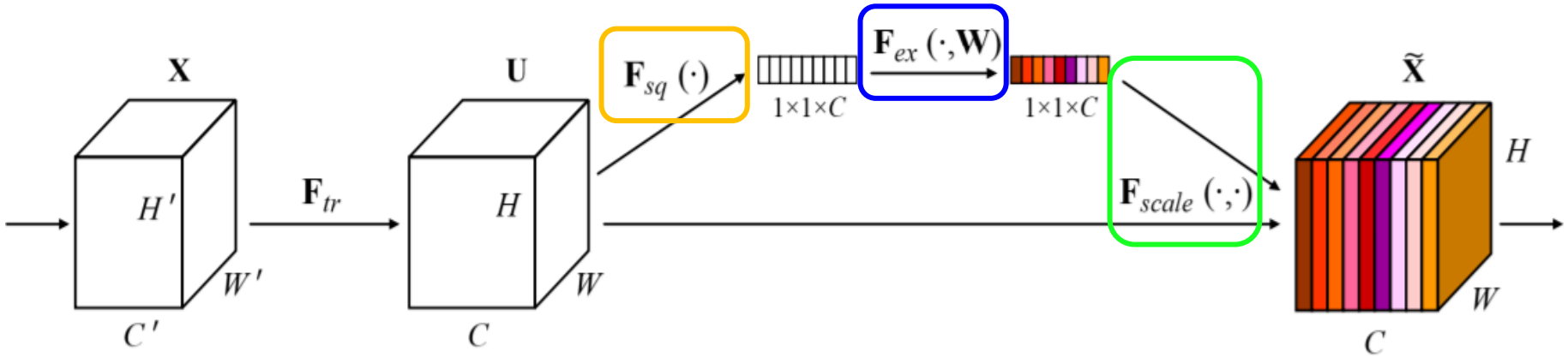
Sigmoid – *FC* – *ReLU* – *FC*:

$$\mathbf{s} = \mathbf{F}_{ex}(\mathbf{z}, \mathbf{W}) = \sigma(g(\mathbf{z}, \mathbf{W})) = \sigma(\mathbf{W}_2 \delta(\mathbf{W}_1 \mathbf{z}))$$

$$\mathbf{W}_1 \in \mathbb{R}^{\frac{C}{r} \times C} \quad \mathbf{W}_2 \in \mathbb{R}^{C \times \frac{C}{r}}$$

Channel – wise multiplication:

$$\tilde{\mathbf{x}}_c = \mathbf{F}_{scale}(\mathbf{u}_c, s_c) = s_c \, \mathbf{u}_c$$



I. Introduction

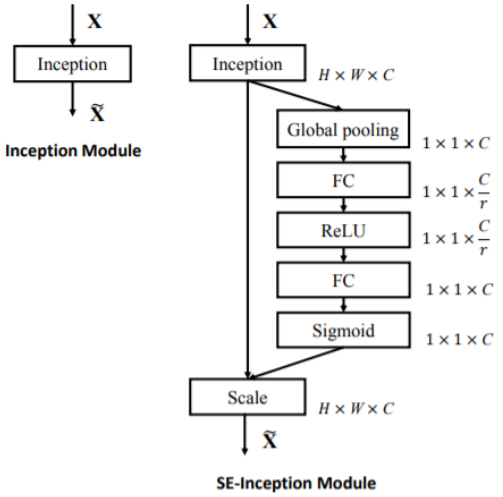
B. Methods

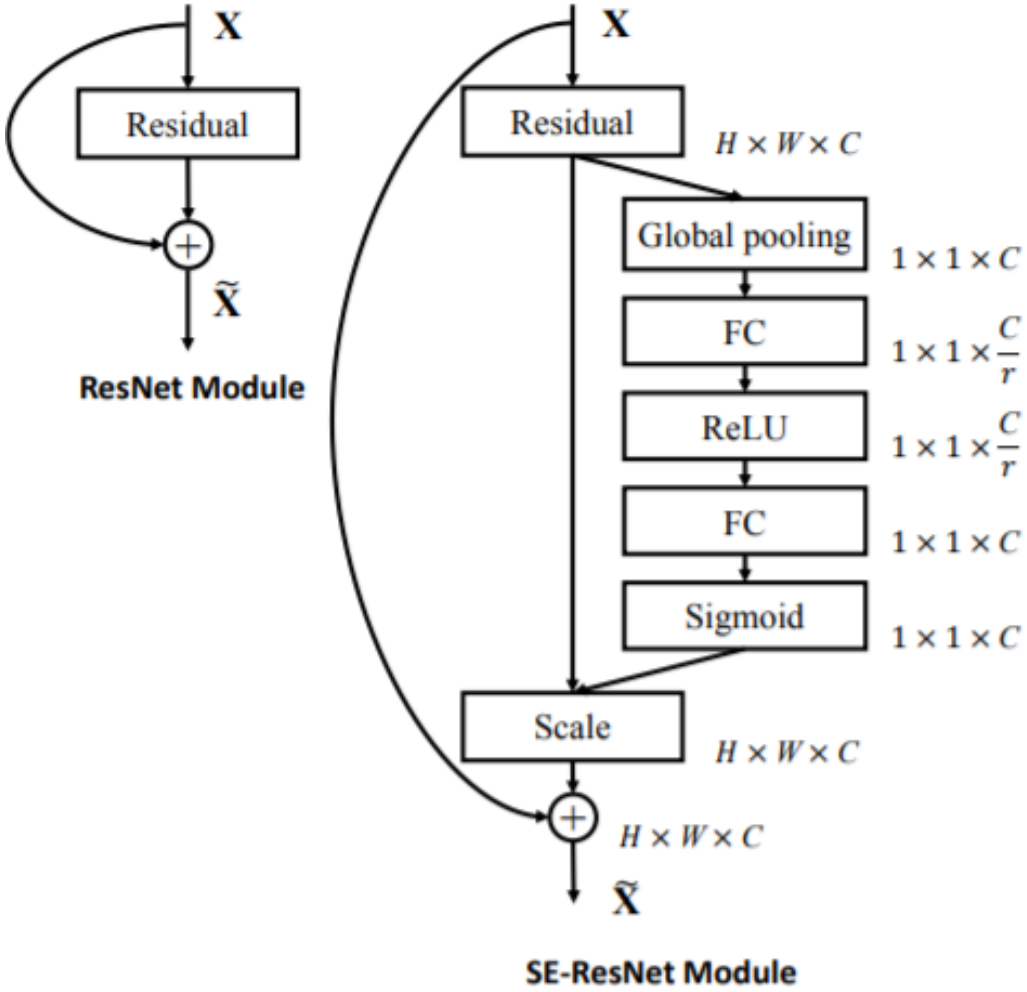
B2. Network Design

B2.1: MobileNets Series

- SENet (v3): pdf, code

➢ Classical Structure





I. Introduction

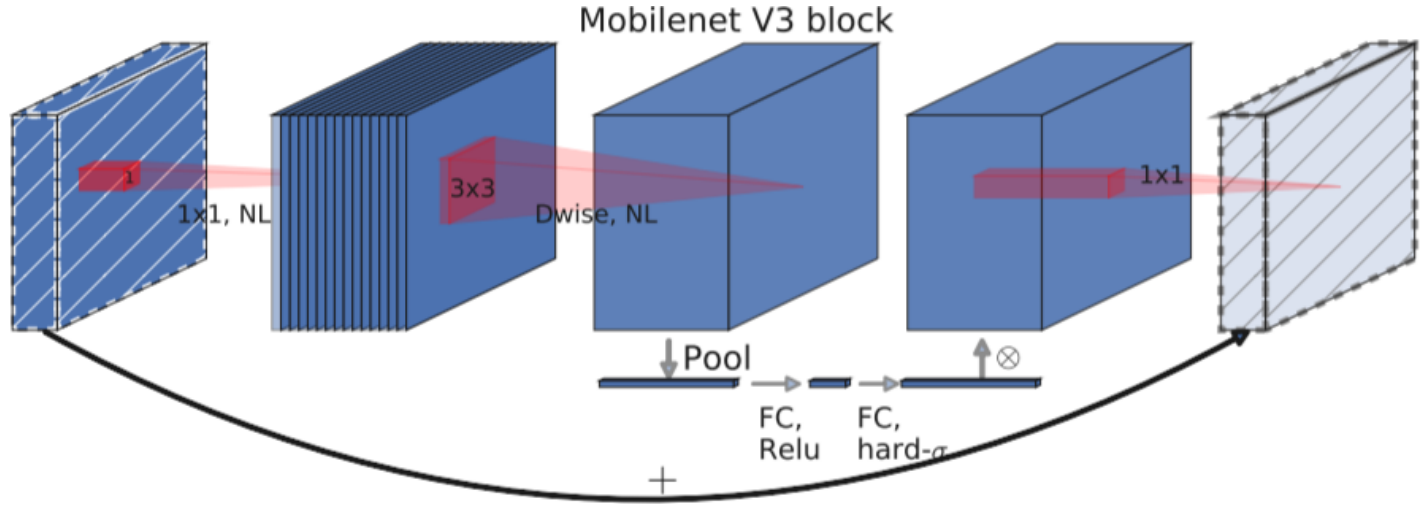
B. Methods

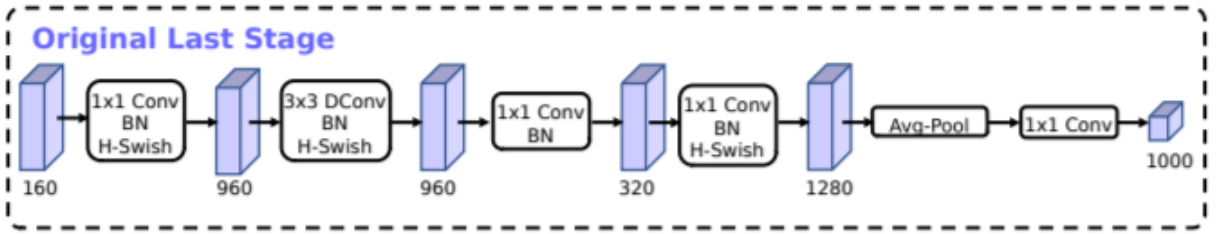
B2. Network Design

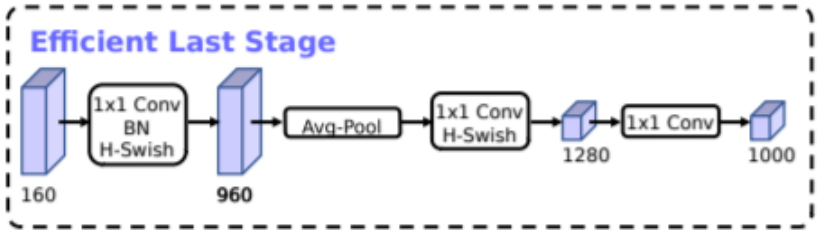
B2.1: MobileNets Series

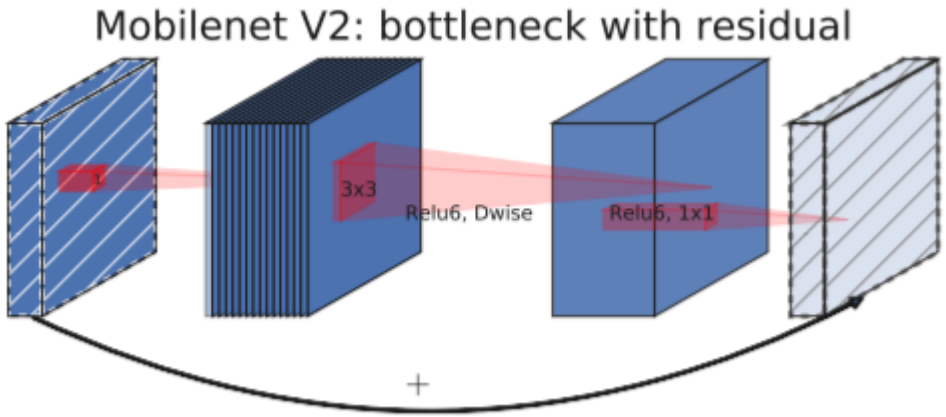
- SENet (v3): pdf, code

➢ MobileNet v3:
MobileNet v2 (shrink last layers) + SENet









I. Introduction

B. Methods

B2. Network Design

B2.1: MobileNets Series

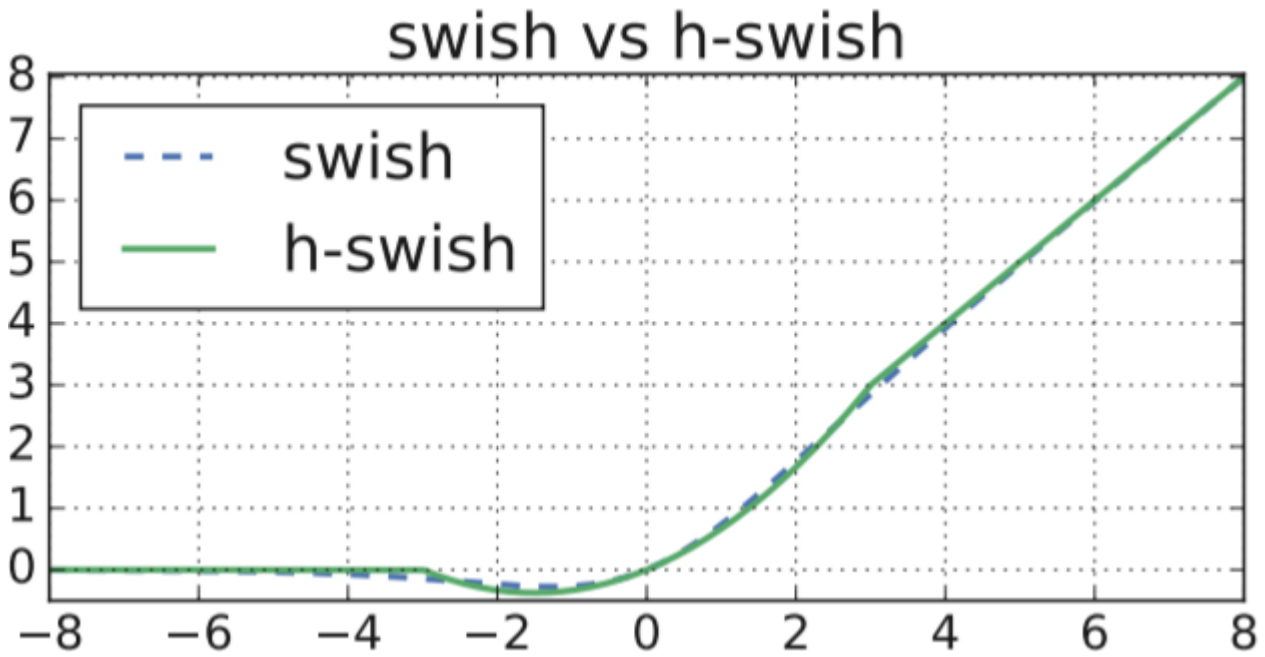
- **h-swish (v3):**

➢ Form:

$$\text{h-swish}[x] = x\frac{\text{ReLU6}(x+3)}{6}$$

➢ Reason:

- Better to optimize on both software & hardware;
- Better to quantize on hardware



I. Introduction

B. Methods

B2. Network Design

B2.1: MobileNets Series

- Summary (v3):

Input	Operator	exp size	#out	SE	NL	s
$224^2 \times 3$	conv2d, 3x3	-	16	-	HS	2
$112^2 \times 16$	bneck, 3x3	16	16	✓	RE	2
$56^2 \times 16$	bneck, 3x3	72	24	-	RE	2
$28^2 \times 24$	bneck, 3x3	88	24	-	RE	1
$28^2 \times 24$	bneck, 5x5	96	40	✓	HS	2
$14^2 \times 40$	bneck, 5x5	240	40	✓	HS	1
$14^2 \times 40$	bneck, 5x5	240	40	✓	HS	1
$14^2 \times 40$	bneck, 5x5	120	48	✓	HS	1
$14^2 \times 48$	bneck, 5x5	144	48	✓	HS	1
$14^2 \times 48$	bneck, 5x5	288	96	✓	HS	2
$7^2 \times 96$	bneck, 5x5	576	96	✓	HS	1
$7^2 \times 96$	bneck, 5x5	576	96	✓	HS	1
$7^2 \times 96$	conv2d, 1x1	-	576	✓	HS	1
$7^2 \times 576$	pool, 7x7	-	-	-	-	1
$1^2 \times 576$	conv2d 1x1, NBN	-	1024	-	HS	1
$1^2 \times 1024$	conv2d 1x1, NBN	-	k	-	-	1

Input	Operator	exp size	#out	SE	NL	s
$224^2 \times 3$	conv2d	-	16	-	HS	2
$112^2 \times 16$	bneck, 3x3	16	16	-	RE	1
$112^2 \times 16$	bneck, 3x3	64	24	-	RE	2
$56^2 \times 24$	bneck, 3x3	72	24	-	RE	1
$56^2 \times 24$	bneck, 5x5	72	40	✓	RE	2
$28^2 \times 40$	bneck, 5x5	120	40	✓	RE	1
$28^2 \times 40$	bneck, 5x5	120	40	✓	RE	1
$28^2 \times 40$	bneck, 3x3	240	80	-	HS	2
$14^2 \times 80$	bneck, 3x3	200	80	-	HS	1
$14^2 \times 80$	bneck, 3x3	184	80	-	HS	1
$14^2 \times 80$	bneck, 3x3	184	80	-	HS	1
$14^2 \times 80$	bneck, 3x3	480	112	✓	HS	1
$14^2 \times 112$	bneck, 3x3	672	112	✓	HS	1
$14^2 \times 112$	bneck, 5x5	672	160	✓	HS	2
$7^2 \times 160$	bneck, 5x5	960	160	✓	HS	1
$7^2 \times 160$	bneck, 5x5	960	160	✓	HS	1
$7^2 \times 160$	conv2d, 1x1	-	960	-	HS	1
$7^2 \times 960$	pool, 7x7	-	-	-	-	1
$1^2 \times 960$	conv2d 1x1, NBN	-	1280	-	HS	1
$1^2 \times 1280$	conv2d 1x1, NBN	-	k	-	-	1

Table 1. Specification for MobileNetV3-Large. SE denotes whether there is a Squeeze-And-Excite in that block. NL denotes the type of nonlinearity used. Here, HS denotes h-swish and RE denotes ReLU. NBN denotes no batch normalization. s denotes stride.

I. Introduction

B. Methods

B2. Network Design

B2.2: ShuffleNet Series

- v1 [2017, Zhang], v2 [2018, Sandler]
- Group Conv + Channel Shuffle
  4 Guidelines
- Group Conv + Channel Shuffle (v1):

I. Introduction

B. Methods

B2. Network Design

B2.2: ShuffleNet Series

- Group Conv + Channel Shuffle (v1):

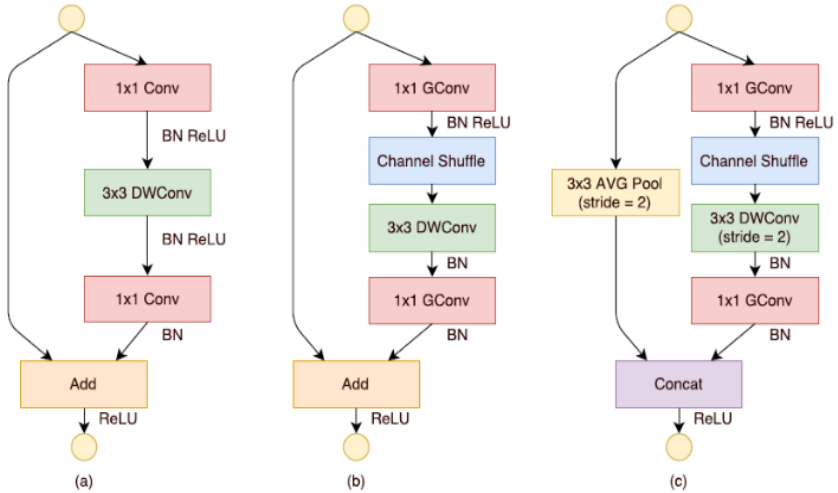
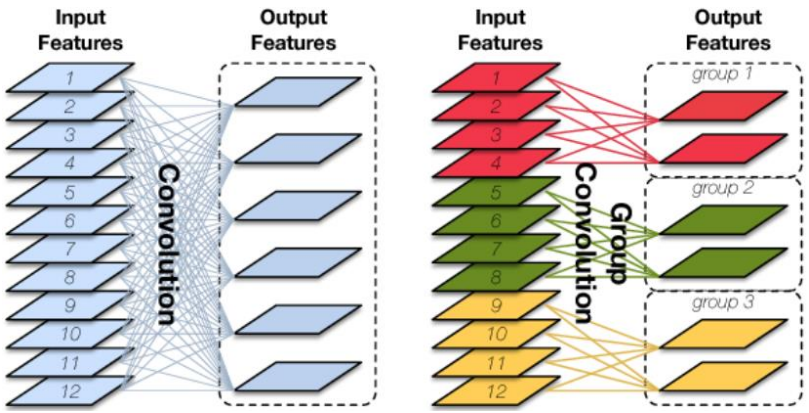


Figure 2: ShuffleNet Units. a) bottleneck unit [9] with depthwise convolution (DWConv) [3, 12]; b) ShuffleNet unit with pointwise group convolution (GConv) and channel shuffle; c) ShuffleNet unit with stride = 2.



Why Group Conv?

A. Because of less computation

Standard Conv:

Image size: $D_F \cdot D_F \cdot M$

Kernel size: $D_k \cdot D_k \cdot M \cdot N$

Operations: $D_k \cdot D_k \cdot M \cdot N \cdot D_F \cdot D_F \equiv Ori$

Group Conv:

Image size: $D_F \cdot D_F \cdot M$

Kernel size: $D_k \cdot D_k \cdot \frac{M}{g} \cdot \frac{N}{g}$

Operations: $D_k \cdot D_k \cdot \frac{M}{g} \cdot \frac{N}{g} \cdot D_F \cdot D_F \cdot g = \frac{Ori}{g} \equiv GConv$

Why 1x1 Conv?

A. For channel combining

Why 1x1 GConv?

A. Faster combining channels

Anything need to be improved?

A. Yes. Because each group of feature is only a fractional response of the input. So we need to fix it.

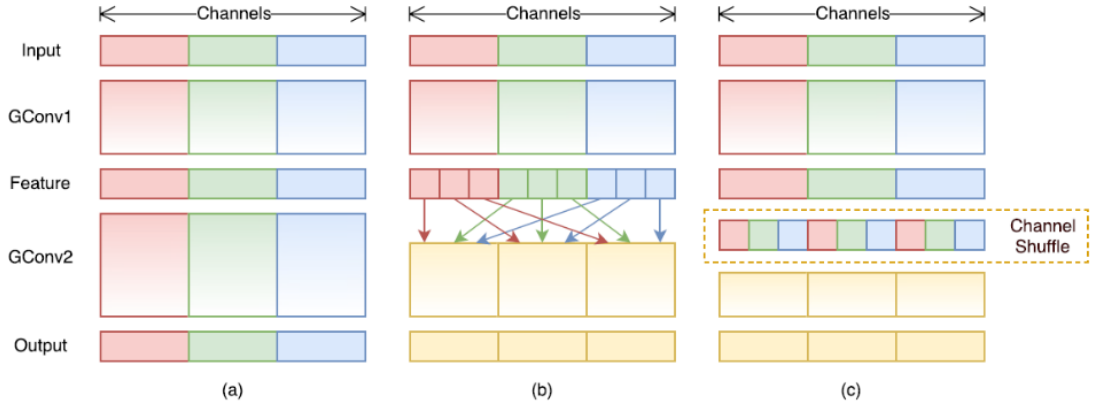
I. Introduction

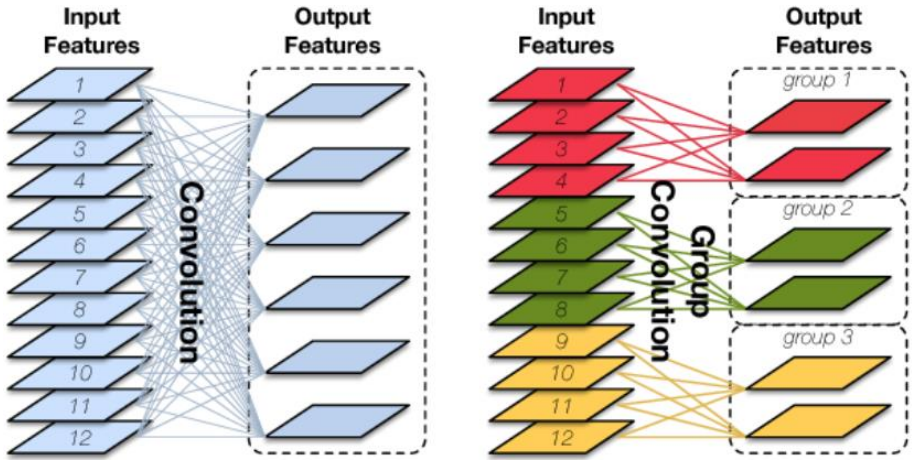
B. Methods

B2. Network Design

B2.2: ShuffleNet Series

- Group Conv + Channel Shuffle (v1):





Anything need to be improved?

A. Yes. Because each group of feature is only a fractional response of the input. So we need to fix it.

How to improve? / to shuffle channel?

A. Use channel shuffle tech. After doing GConv, we shuffle the feature maps

I. Introduction

B. Methods

B2. Network Design

B2.2: ShuffleNet Series

- Group Conv + Channel Shuffle (v1):

Layer	Output size	KSize	Stride	Repeat	Output channels (g groups)				
					$g = 1$	$g = 2$	$g = 3$	$g = 4$	$g = 8$
Image	224×224				3	3	3	3	3
Conv1	112×112	3×3	2	1	24	24	24	24	24
MaxPool	56×56	3×3	2						
Stage2	28×28		2	1	144	200	240	272	384
	28×28		1	3	144	200	240	272	384
Stage3	14×14		2	1	288	400	480	544	768
	14×14		1	7	288	400	480	544	768
Stage4	7×7		2	1	576	800	960	1088	1536
	7×7		1	3	576	800	960	1088	1536
GlobalPool	1×1	7×7							
FC					1000	1000	1000	1000	1000
Complexity					143M	140M	137M	133M	137M

Table 1. ShuffleNet architecture. The complexity is evaluated with FLOPs, i.e. the number of floating-point multiplication-adds. Note that for Stage 2, we do not apply group convolution on the first pointwise layer because the number of input channels is relatively small.

I. Introduction

B. Methods

B2. Network Design

B2.2: ShuffleNet Series

- 4 Guidelines (v2):

G1. MAC (Memory access cost) becomes minimal when input/output has same the size.

G2. Excessive GConv increases MAC.

G3. Network fragmentation reduces degree of parallelism

G4. Eltwise operations are non-negligible

I. Introduction

B. Methods

B2. Network Design

B2.2: ShuffleNet Series

- 4 Guidelines (v2):

G1. MAC (Memory access cost) becomes minimal when input/output has same the size

G2. Excessive GConv increases MAC

G3. Network fragmentation reduces degree of parallelism

G4. Eltwise operations are non-negligible

Explain: (assume 1x1 kernel)

Feature map size (In):	$w \times h \times c_1$
Feature map size (Out):	$w \times h \times c_2$
Conv FLOPs (B):	whc_1c_2
(In) memory:	whc_1
(Out) memory:	whc_2
Kernels (In+Out) memory:	c_1c_2

MAC: Memory Access Cost

$$
\begin{aligned}
MAC &= hw(c_1 + c_2) + c_1c_2 \\
&= \sqrt{(hw(c_1 + c_2))^2} + \frac{B}{hw} \\
&= \sqrt{(hw)^2 \cdot (c_1 + c_2)^2} + \frac{B}{hw} \\
&\geq \sqrt{(hw)^2 \cdot 4c_1c_2} + \frac{B}{hw} \\
&= 2\sqrt{hw \cdot (hwc_1c_2)} + \frac{B}{hw} \\
&= 2\sqrt{hwB} + \frac{B}{hw}
\end{aligned}
$$

$\boldsymbol{MAC_{min}\ can\ be\ gotten\ when\ c_1 = c_2}$

I. Introduction

B. Methods

B2. Network Design

B2.2: ShuffleNet Series

- **4 Guidelines (v2):**

G1. MAC (Memory access cost) becomes minimal when input/output has same the size.

G2. Excessive GConv increases MAC.

G3. Network fragmentation reduces degree of parallelism

G4. Eltwise operations are non-negligible

Explain:

Feature map size (In):	$w \times h \times c_1$
Feature map size (Out):	$w \times h \times c_2$
Conv FLOPs (B):	$wh\frac{c_1}{g}\frac{c_2}{g}g$
(In) memory:	whc_1
(Out) memory:	whc_2
Kernels (In+Out) memory:	$\frac{c_1}{g}\frac{c_2}{g}g$
Group number:	g

$$MAC = hw(c_1 + c_2) + \frac{c_1}{g}\frac{c_2}{g}g$$

$$= hw(c_1 + c_2) + \frac{c_1 c_2}{g}$$

$$= Bg(\frac{1}{c_1} + \frac{1}{c_2}) + \frac{B}{hw}$$

MAC is getting bigger with the increase of group number.

So g shouldn't be too big.

I. Introduction

B. Methods

B2. Network Design

B2.2: ShuffleNet Series

- 4 Guidelines (v2):

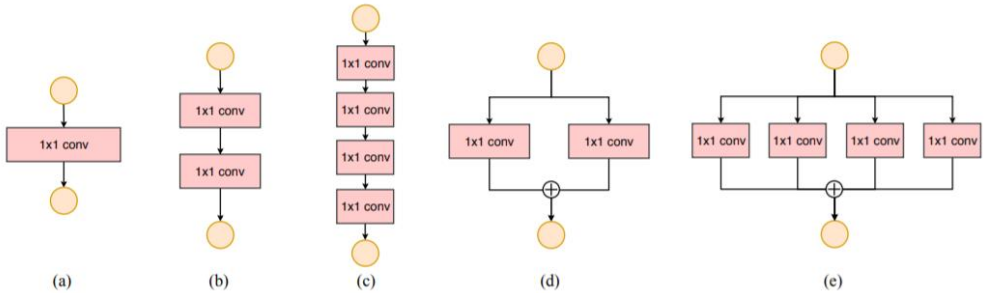
G1. MAC (Memory access cost) becomes minimal when input/output has same the size

G2. Excessive GConv increases MAC

G3. Network fragmentation reduces degree of parallelism

G4. Eltwise operations are non-negligible

Explain:



	GPU (Batches/sec.)			CPU (Images/sec.)		
	c=128	c=256	c=512	c=64	c=128	c=256
1-fragment	2446	1274	434	40.2	10.1	2.3
2-fragment-series	1790	909	336	38.6	10.1	2.2
4-fragment-series	752	745	349	38.4	10.1	2.3
2-fragment-parallel	1537	803	320	33.4	9.1	2.2
4-fragment-parallel	691	572	292	35.0	8.4	2.1

$Speed$: $a > b > d > c > e$

The more branches, the slower the system.

The more fragments, the slower the system.

I. Introduction

B. Methods

B2. Network Design

B2.2: ShuffleNet Series

- 4 Guidelines (v2):

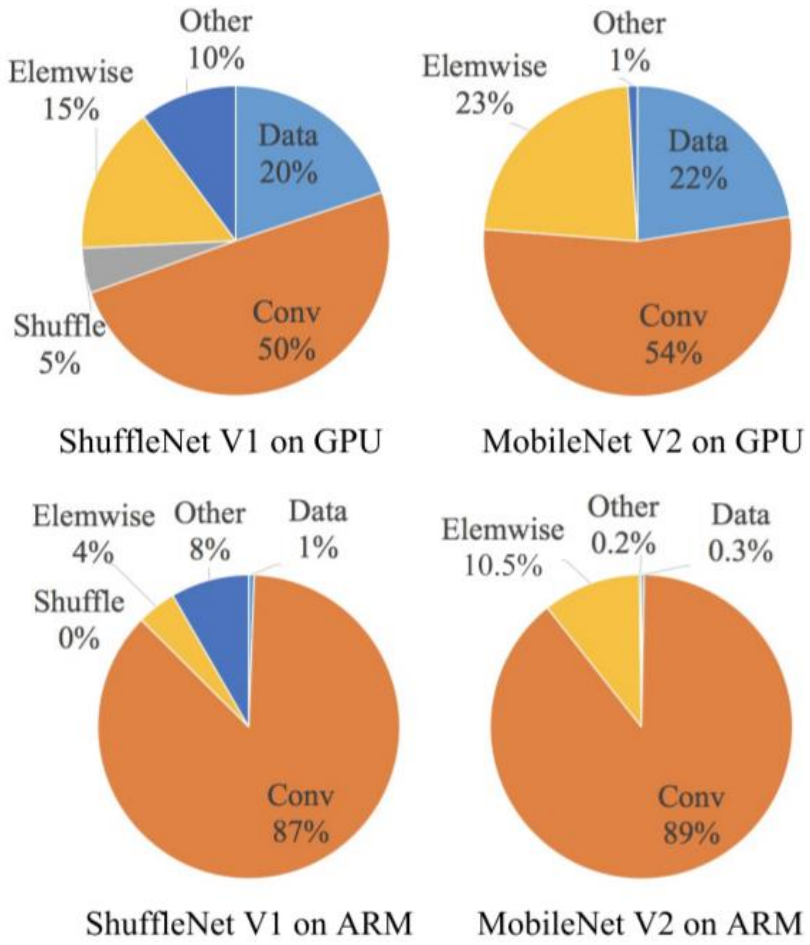
G1. MAC (Memory access cost) becomes minimal when input/output has same the size

G2. Excessive GConv increases MAC

G3. Network fragmentation reduces degree of parallelism

G4. Eltwise operations are non-negligible

Explain:



Reduce the usage of Eltwise

I. Introduction

B. Methods

B2. Network Design

B2.2: ShuffleNet Series

- **4 Guidelines (v2):**

G1. MAC (Memory access cost) becomes minimal when input/output has same the size

G2. Excessive GConv increases MAC

G3. Network fragmentation reduces degree of parallelism

G4. Eltwise operations are non-negligible

E.G.
Q. How other networks break the rules?

ShuffleNet – V1:
X G2: rely on GConv heavily
X G1: it's bottleneck-like building blocks

MobileNet – V2:
X G1: inverted bottleneck structure
X G4: Eltwise+ at thick feature

Auto-generated structures:
X G3: highly fragmented

I. Introduction

B. Methods

B2. Network Design

B2.2: ShuffleNet Series

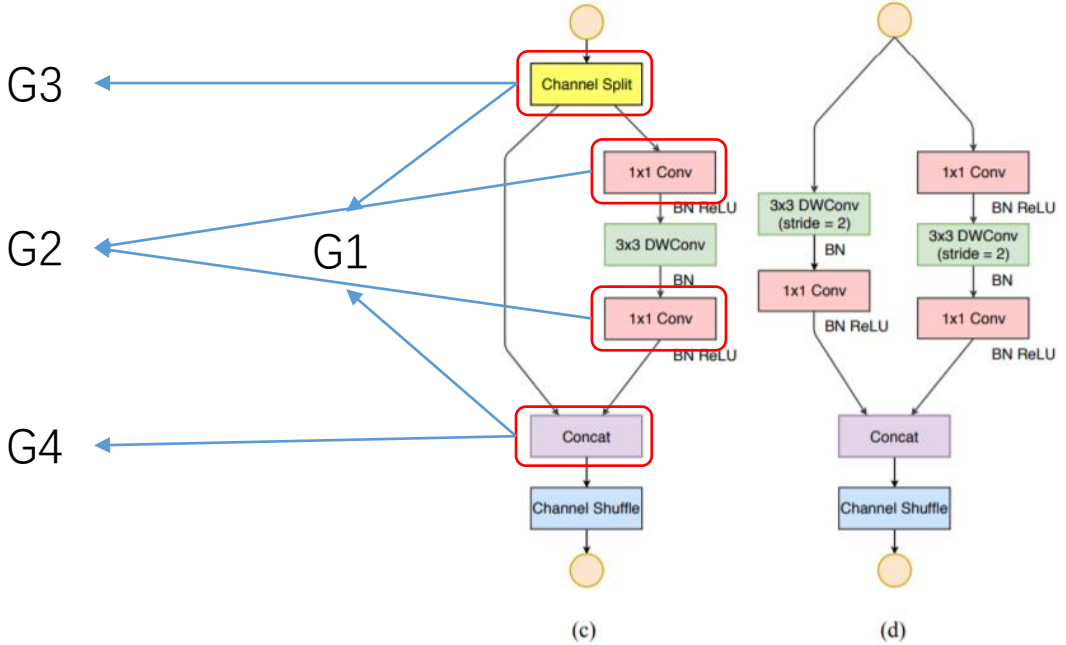
- **Summary (v2):**

G1. MAC (Memory access cost) becomes minimal when input/output has same the size

G2. Excessive GConv increases MAC

G3. Network fragmentation reduces degree of parallelism

G4. Eltwise operations are non-negligible



Layer	Output size	KSize	Stride	Repeat	Output channels			
					0.5×	1×	1.5×	2×
Image	224×224				3	3	3	3
Conv1	112×112	3×3	2	1	24	24	24	24
MaxPool	56×56	3×3	2					
Stage2	28×28		2	1	48	116	176	244
	28×28		1	3				
Stage3	14×14		2	1	96	232	352	488
	14×14		1	7				
Stage4	7×7		2	1	192	464	704	976
	7×7		1	3				
Conv5	7×7	1×1	1	1	1024	1024	1024	2048
GlobalPool	1×1	7×7						
FC					1000	1000	1000	1000
FLOPs					41M	146M	299M	591M
# of Weights					1.4M	2.3M	3.5M	7.4M

Table 5: Overall architecture of ShuffleNet v2, for four different levels of complexities.

I. Introduction

B. Methods

B2. Network Design

B2.3: EfficientNet

- v1 [2019, Tan]
- NAS + B0 Network + Compound Model Scaling

- Compound Model Scaling (v1):

I. Introduction

B. Methods

B2. Network Design

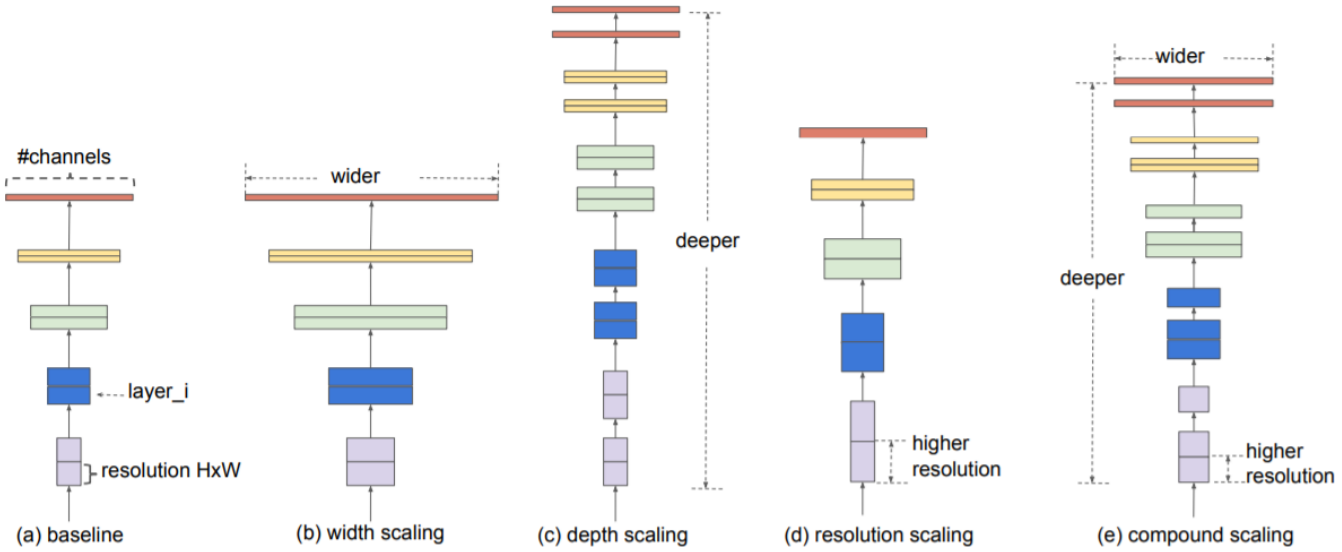
B2.3: EfficientNet

- **Compound Model Scaling (v1):**

➢ **NAS Mathematical Form**

$$\mathcal{N} = \bigodot_{i=1\ldots s} \mathcal{F}_i^{L_i}\left(X_{\langle H_i, W_i, C_i\rangle}\right)$$

i: *stage id*
L_i: *# of Conv in ith stage*
F_i: *Convolution operation*
$\bigodot$:
sign of concat: $\left(F_5(\ldots F_1(X < \cdots >))\right)$
N: *The network*



➢ **Constraint**

$$\text{depth: } d = \alpha^\phi$$
$$\text{width: } w = \beta^\phi$$
$$\text{resolution: } r = \gamma^\phi$$
$$\text{s.t. } \alpha \cdot \beta^2 \cdot \gamma^2 \approx 2$$
$$\alpha \geq 1, \beta \geq 1, \gamma \geq 1$$

α, β, γ: *scale, searched by small range grid*
$\alpha|\beta^2|\gamma^2$: *double α will double FLOPs, but double β|γ will increase 4 times, so α|β²|γ² not α|β|γ*
ϕ: *hyperparameter: how many more resources are available for model scaling.*

➢ **Target**

$$\max_{d,w,r} \quad Accuracy(\mathcal{N}(d,w,r))$$
$$s.t. \quad \mathcal{N}(d,w,r) = \bigodot_{i=1\ldots s} \hat{\mathcal{F}}_i^{d\cdot\hat{L}_i}\left(X_{\langle r\cdot\hat{H}_i, r\cdot\hat{W}_i, w\cdot\hat{C}_i\rangle}\right)$$
$$\text{Memory}(\mathcal{N}) \leq \text{target\_memory}$$
$$\text{FLOPS}(\mathcal{N}) \leq \text{target\_flops}$$

d: *scaling factor of depth*
r: *scaling factor of resolution*
w: *scaling factor of width (channel)*

I. Introduction

B. Methods

B2. Network Design

B2.3: EfficientNet

- B0 Network (v1):

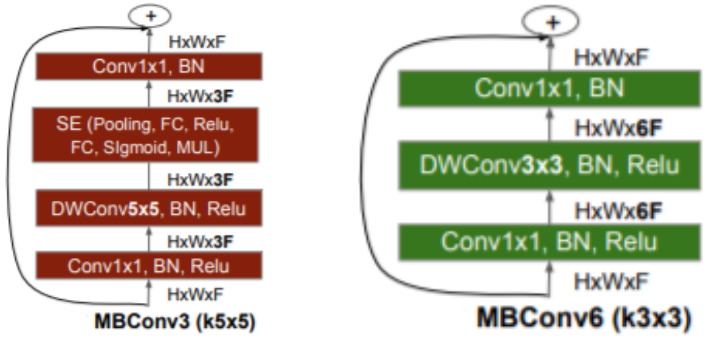
➢ B1-7 network — How to search from B0

- STEP 1: we first fix $\phi = 1$, assuming twice more resources available, and do a small grid search of α, β, γ based on Equation 2 and 3. In particular, we find the best values for EfficientNet-B0 are $\alpha = 1.2, \beta = 1.1, \gamma = 1.15$, under constraint of $\alpha \cdot \beta^2 \cdot \gamma^2 \approx 2$
- STEP 2: we then fix α, β, γ as constants and scale up baseline network with different ϕ using Equation 3, to obtain EfficientNet-B1 to B7

➢ B0 network — Starting point to search

Stage i	Operator $\hat{\mathcal{F}}_i$	Resolution $\hat{H}_i \times \hat{W}_i$	#Channels $\hat{C}_i$	#Layers $\hat{L}_i$
1	Conv3x3	224×224	32	1
2	MBConv1, k3x3	112×112	16	1
3	MBConv6, k3x3	112×112	24	2
4	MBConv6, k5x5	56×56	40	2
5	MBConv6, k3x3	28×28	80	3
6	MBConv6, k5x5	14×14	112	3
7	MBConv6, k5x5	14×14	192	4
8	MBConv6, k3x3	7×7	320	1
9	Conv1x1 & Pooling & FC	7×7	1280	1

MBConv: *Mobilenet v2 Inverted Residuals + Linear Bottleneck + SE Block*



How to get B0: [MNasNet](#) $(2019, Tan)$
Initial Searching Target: $Acc(m) \times [FLOPs(m)/T]^w$
$Acc(m)$: $models(m)'s\ accuracy$
T: $Target\ FLOPs$
w: $-0.07, hyperparameter$: $tradeoff$ $between\ accuracy\ \&\ FLOPs$

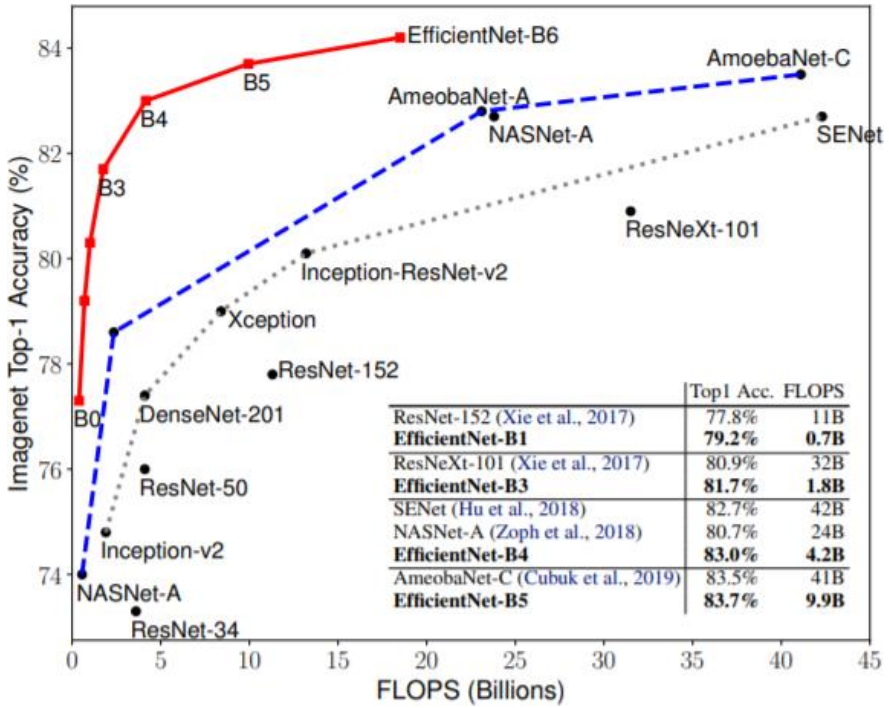
I. Introduction

B. Methods

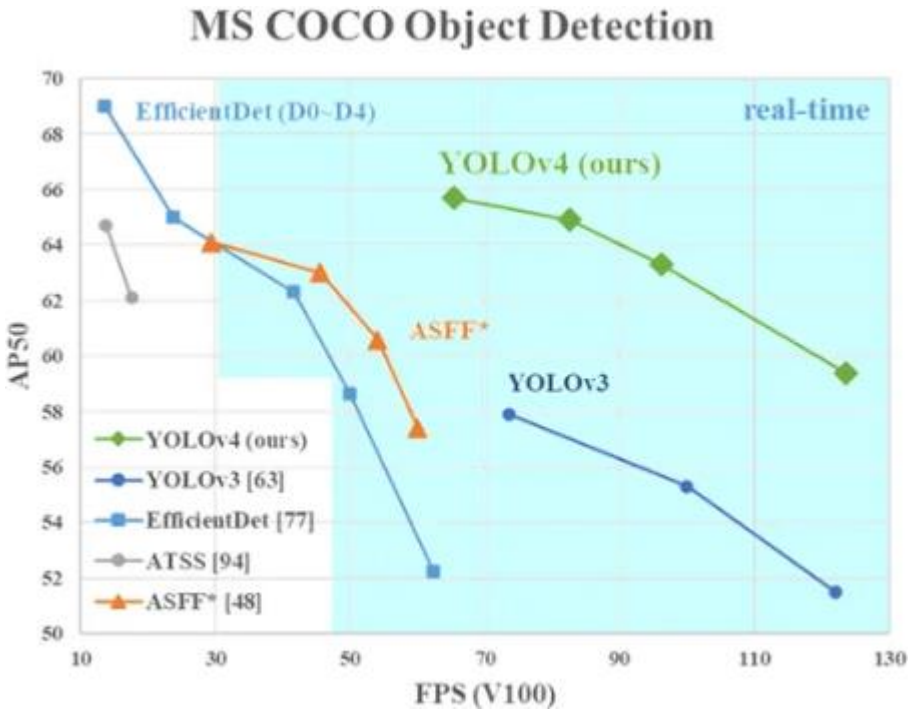
B2. Network Design

B2.3: EfficientNet

- Summary (v1):



	Top1 Acc.	FLOPS
ResNet-152 (Xie et al., 2017)	77.8%	11B
EfficientNet-B1	**79.2%**	**0.7B**
ResNeXt-101 (Xie et al., 2017)	80.9%	32B
EfficientNet-B3	**81.7%**	**1.8B**
SENet (Hu et al., 2018)	82.7%	42B
NASNet-A (Zoph et al., 2018)	80.7%	24B
EfficientNet-B4	**83.0%**	**4.2B**
AmeobaNet-C (Cubuk et al., 2019)	83.5%	41B
EfficientNet-B5	**83.7%**	**9.9B**



[*Yolo's better*]

I. Introduction

B. Methods

B2. Network Design

B2.4: GhostNet

- v1 [2020, Han]
- Ghost Module
- Model Rubik's Cube/TinyNets [2020, Han], (could be combined with any tiny nets)
- Compound Model Scaling (v1):

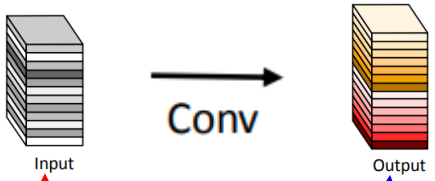
I. Introduction

B. Methods

B2. Network Design

B2.4: GhostNet

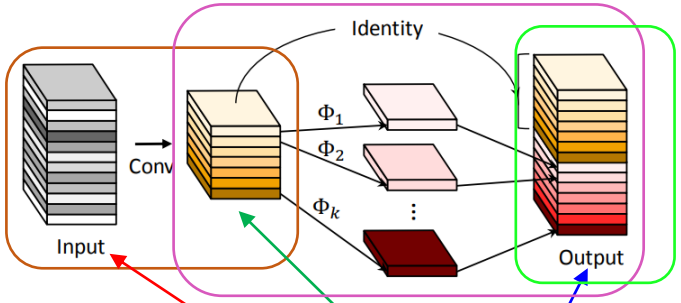
- **Ghost Module (v1):**



(a) The convolutional layer

$X \in \mathbb{R}^{c \times h \times w} \quad * \quad f + b = Y \in \mathbb{R}^{n \times h' \times w'}$

$FLOPs = k \cdot k \cdot c \cdot h' \cdot w' \cdot n$



(b) The Ghost module

Step1: $Y' = X * f' \in \mathbb{R}^{c \times k \times k \times m}, \quad m \leq n$

intrinsic feature: by using Identity Mapping

Step2: $y_{ij} = \Phi_{i,j}(y'_i),$

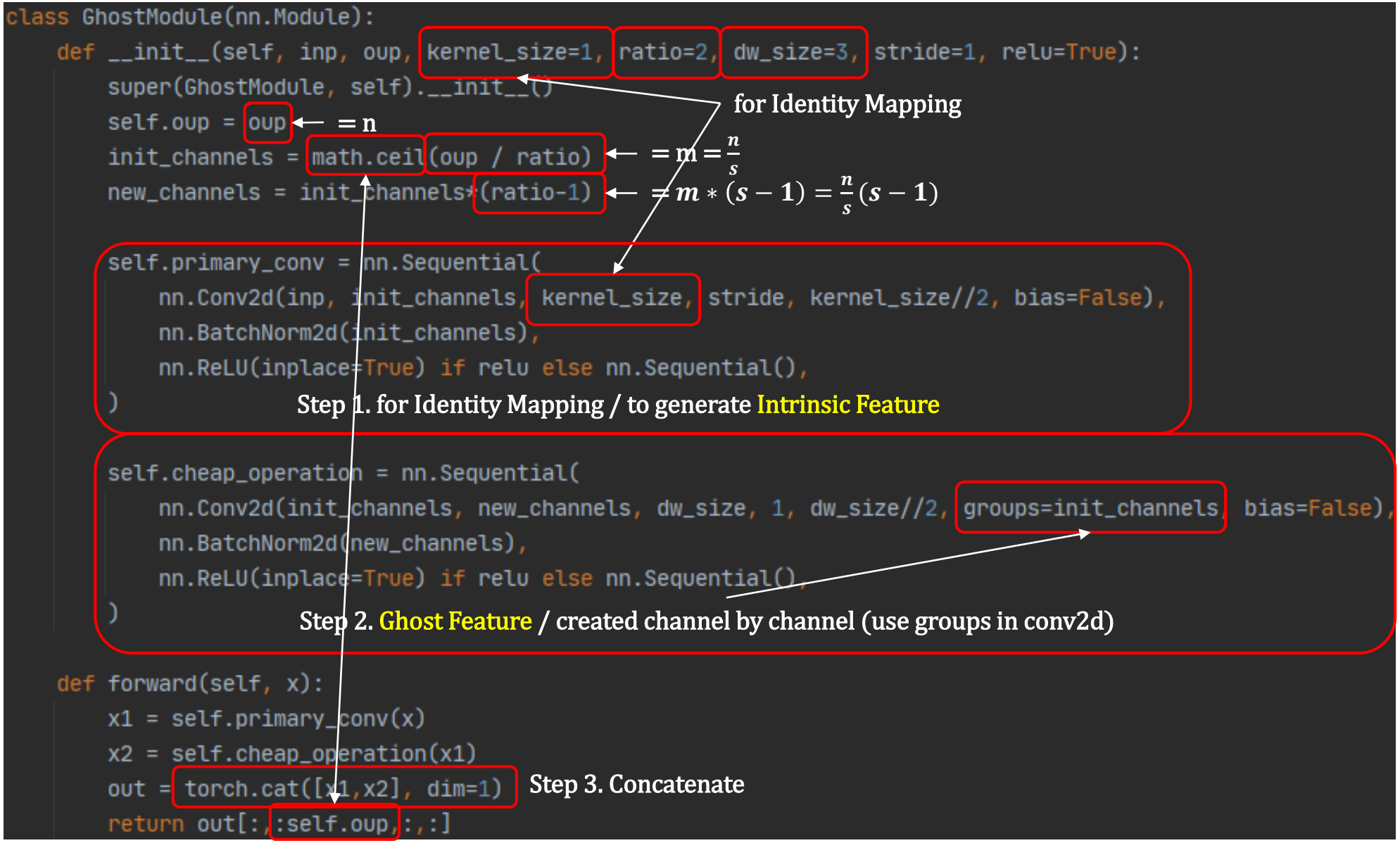
$\forall\, i = 1, \ldots, m, \quad j = 1, \ldots, s$

use groups in conv

- 1, intrinsic feature
- $s - 1$, ghost feature

Step3: concatenate intrinsic & ghost feature

Let's see the code.


class GhostModule(nn.Module):
def __init__(self, inp, oup, kernel_size=1, ratio=2, dw_size=3, stride=1, relu=True):
super(GhostModule, self).__init__()
self.oup = oup
init_channels = math.ceil(oup / ratio)
new_channels = init_channels*(ratio-1)
self.primary_conv = nn.Sequential(
nn.Conv2d(inp, init_channels, kernel_size, stride, kernel_size//2, bias=False),
nn.BatchNorm2d(init_channels),
nn.ReLU(inplace=True) if relu else nn.Sequential(),
)
self.cheap_operation = nn.Sequential(
nn.Conv2d(init_channels, new_channels, dw_size, 1, dw_size//2, groups=init_channels, bias=False),
nn.BatchNorm2d(new_channels),
nn.ReLU(inplace=True) if relu else nn.Sequential(),
)
def forward(self, x):
x1 = self.primary_conv(x)
x2 = self.cheap_operation(x1)
out = torch.cat([x1,x2], dim=1)
return out[:,:self.oup,:,:]
for Identity Mapping
= n
= m = n/s
= m * (s − 1) = n/s (s − 1)
Step 1. for Identity Mapping / to generate Intrinsic Feature
Step 2. Ghost Feature / created channel by channel (use groups in conv2d)
Step 3. Concatenate

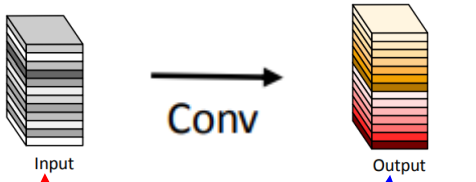
I. Introduction

B. Methods

B2. Network Design

B2.4: GhostNet

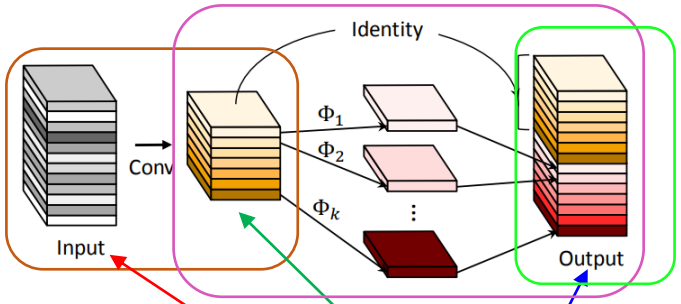
- **Ghost Module (v1):**



(a) The convolutional layer

$X \in \mathbb{R}^{c\times h\times w} \quad * \; f + b = Y \in \mathbb{R}^{n\times h'\times w'}$

$FLOPs = k \cdot k \cdot c \cdot h' \cdot w' \cdot n$



(b) The Ghost module

Step1: $Y' = X * f' \in \mathbb{R}^{c\times k\times k\times m}, \quad m \le n$

intrinsic feature: by using Identity Mapping

Step2: $y_{ij} = \Phi_{i,j}(y'_i),$

$\forall\, i = 1, \ldots, m, \quad j = 1, \ldots, s$

use groups in conv

- 1, intrinsic feature
- $s-1$, ghost feature

Step3: concatenate intrinsic & ghost feature

Speed Up Ratio

$$r_s = \frac{n \cdot h' \cdot w' \cdot c \cdot k \cdot k}{\frac{n}{s} \cdot h' \cdot w' \cdot c \cdot k \cdot k + (s-1) \cdot \frac{n}{s} \cdot h' \cdot w' \cdot d \cdot d} = \frac{c \cdot k \cdot k}{\frac{1}{s} \cdot c \cdot k \cdot k + \frac{s-1}{s} \cdot d \cdot d} \approx \frac{s \cdot c}{s + c - 1} \approx s, \tag{4}$$

where $d \times d$ has the similar magnitude as that of $k \times k$, and $s \ll c$. Similarly, the compression ratio can be calculated as

$$r_c = \frac{n \cdot c \cdot k \cdot k}{\frac{n}{s} \cdot c \cdot k \cdot k + (s-1) \cdot \frac{n}{s} \cdot d \cdot d} \approx \frac{s \cdot c}{s + c - 1} \approx s, \tag{5}$$

which is equal to that of the speed-up ratio by utilizing the proposed Ghost module.

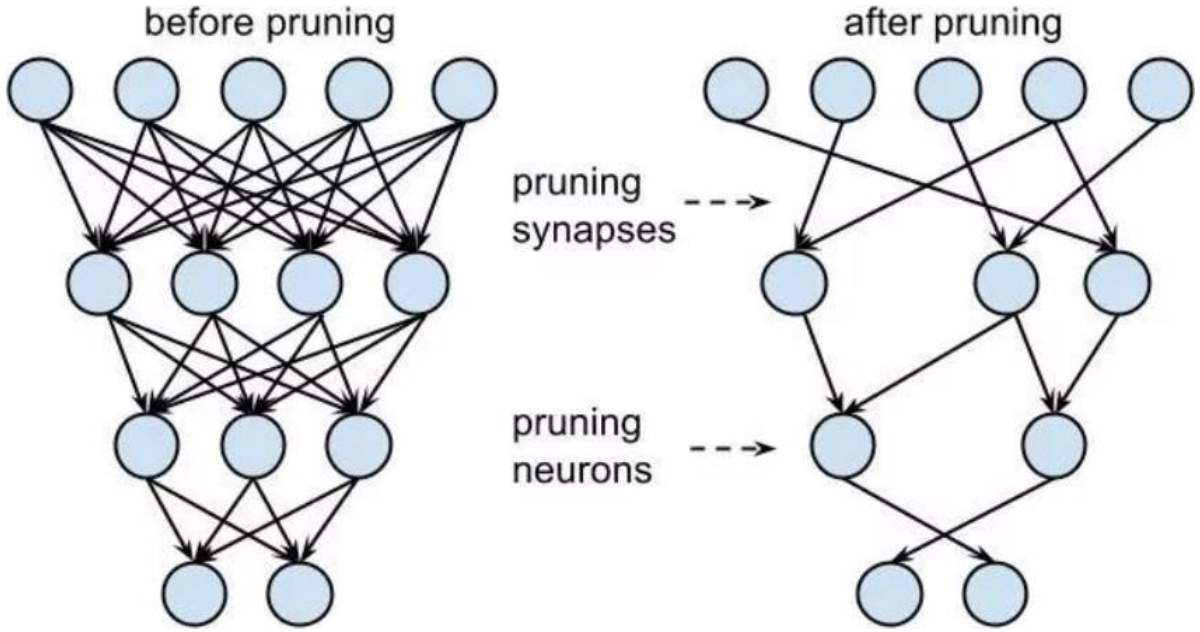
I. Introduction

B. Methods

B3. Model

B3.1: Model/Parameter prune

- Remove connections
- Remove kernels
- Different rules to remove
- Great Resources



I. Introduction

B. Methods

B3. Model

B3.2: Model Quantization

- Common in engineering end
- A different area comparing with what we discussed before
- Highly relying on hardware
- Different companies have different solutions
- A good introduction: part I, part II / corresponding document
  A great paper: 2017, Jacob, Google
  A great summary

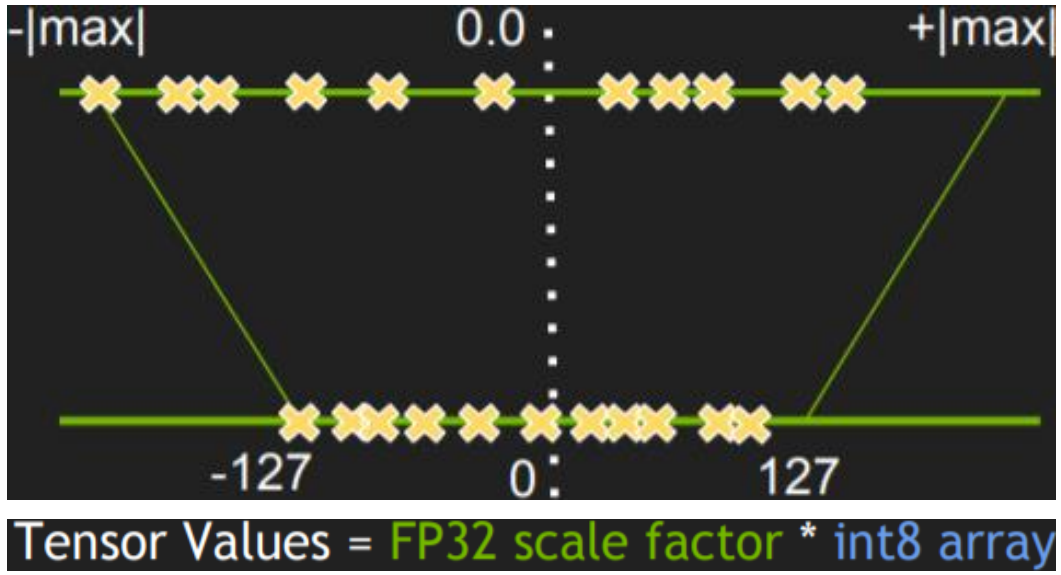
I. Introduction

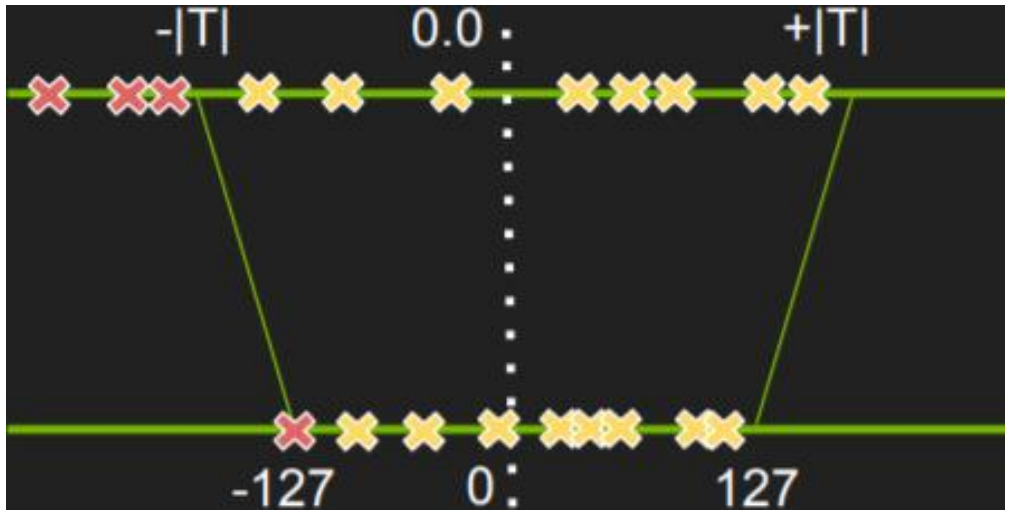
B. Methods

B3. Model

B3.2: Model Quantization

- Basic idea (E.g. Float 32 -> Int 8)





- Practical way
  - Target: PyTorch Model → TensorRT
  - **Method1**: PyTorch → Onnx → TensorRT
  - Method2: PyTorch → TensorRT
  - Some useful prior works:
    Link1 / **Link2**

- Optional Practice:
  - Link2 → Then Method1 → Then Method2
  - Not compulsory.

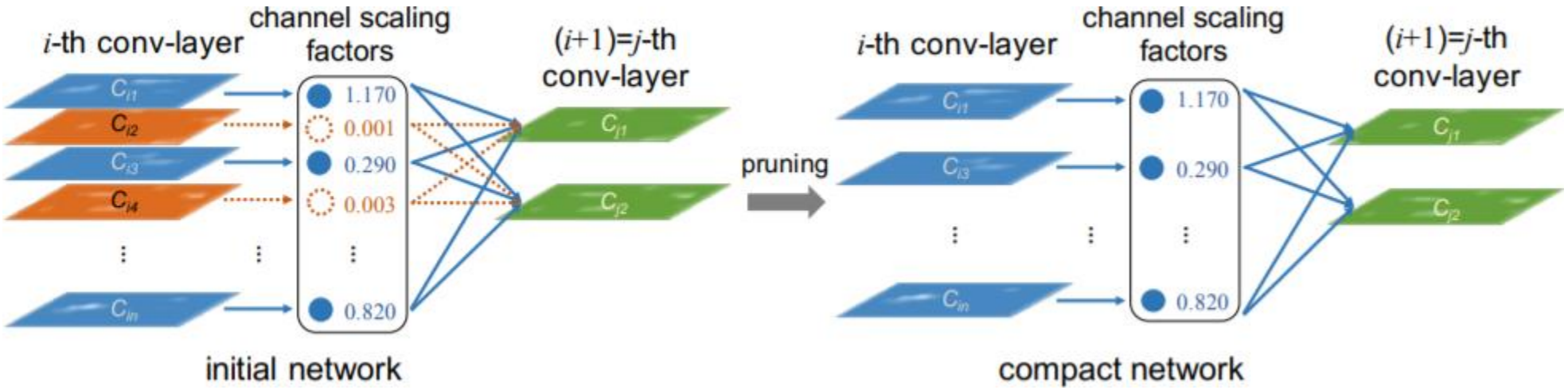
II. Network Slimming

II. Network Slimming

C. Theory [Paper, 2017, Liu]

- **Illustration:**
  a. Layers-prune method
  b. Remove layers with light kernel weights

- **Questions:**
  a. How to get weights?
  b. How to get layers with light weights?
  c. How to remove light-layers and corresponding kernel?
  d. How to remain the rest layers ?
  e. How to maintain accuracy?



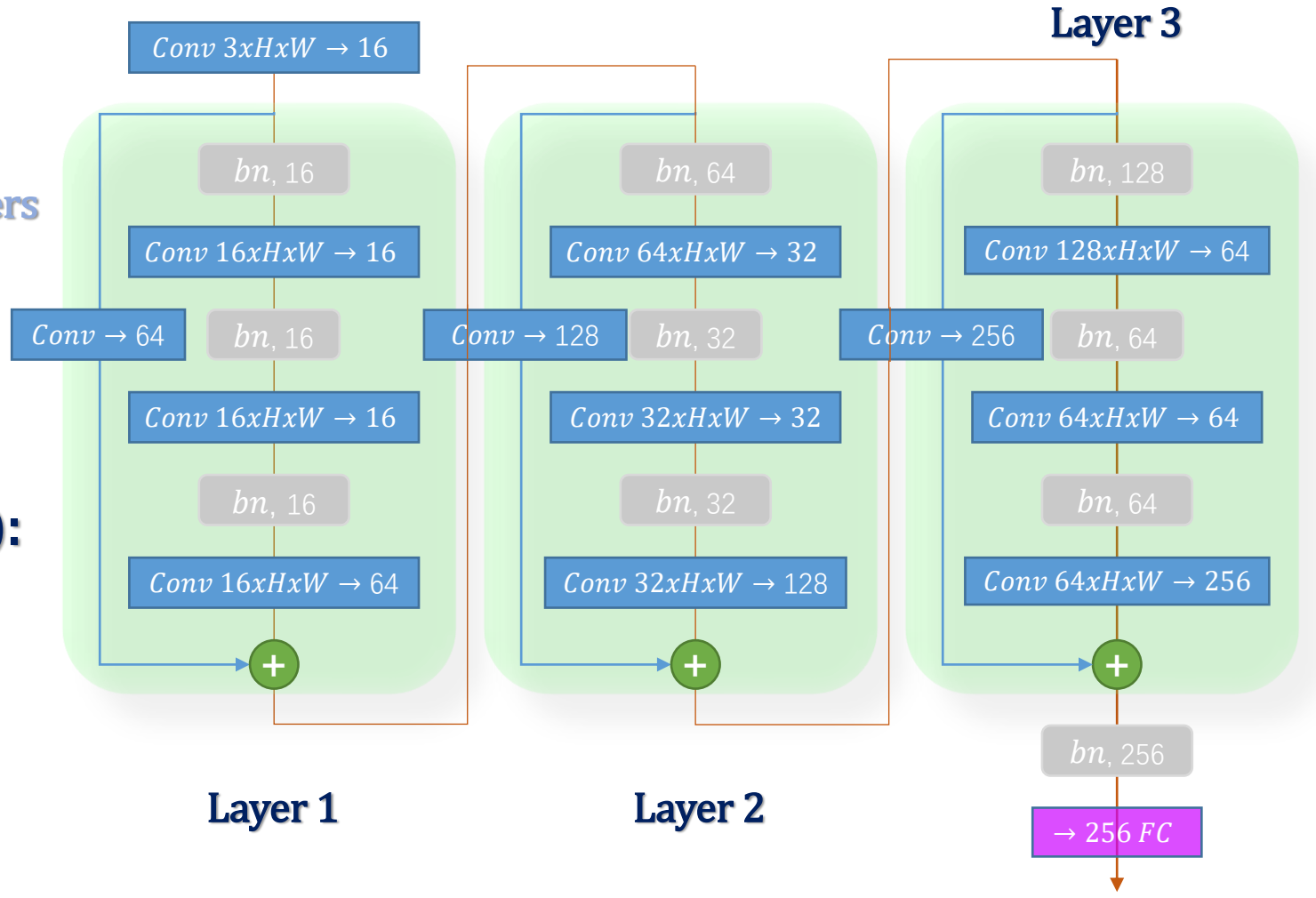
II. Network Slimming

C. Theory [Paper, 2017, Liu]

- **Pipeline:**
  a. Train original model
  b. Load trained model + prune layers
  c. Finetune pruned model

- **E.g.: Resnet (Identity Mapping, 2016):**

➢ Original structure
(simplified inner blocks)
(channel # just for illustration)



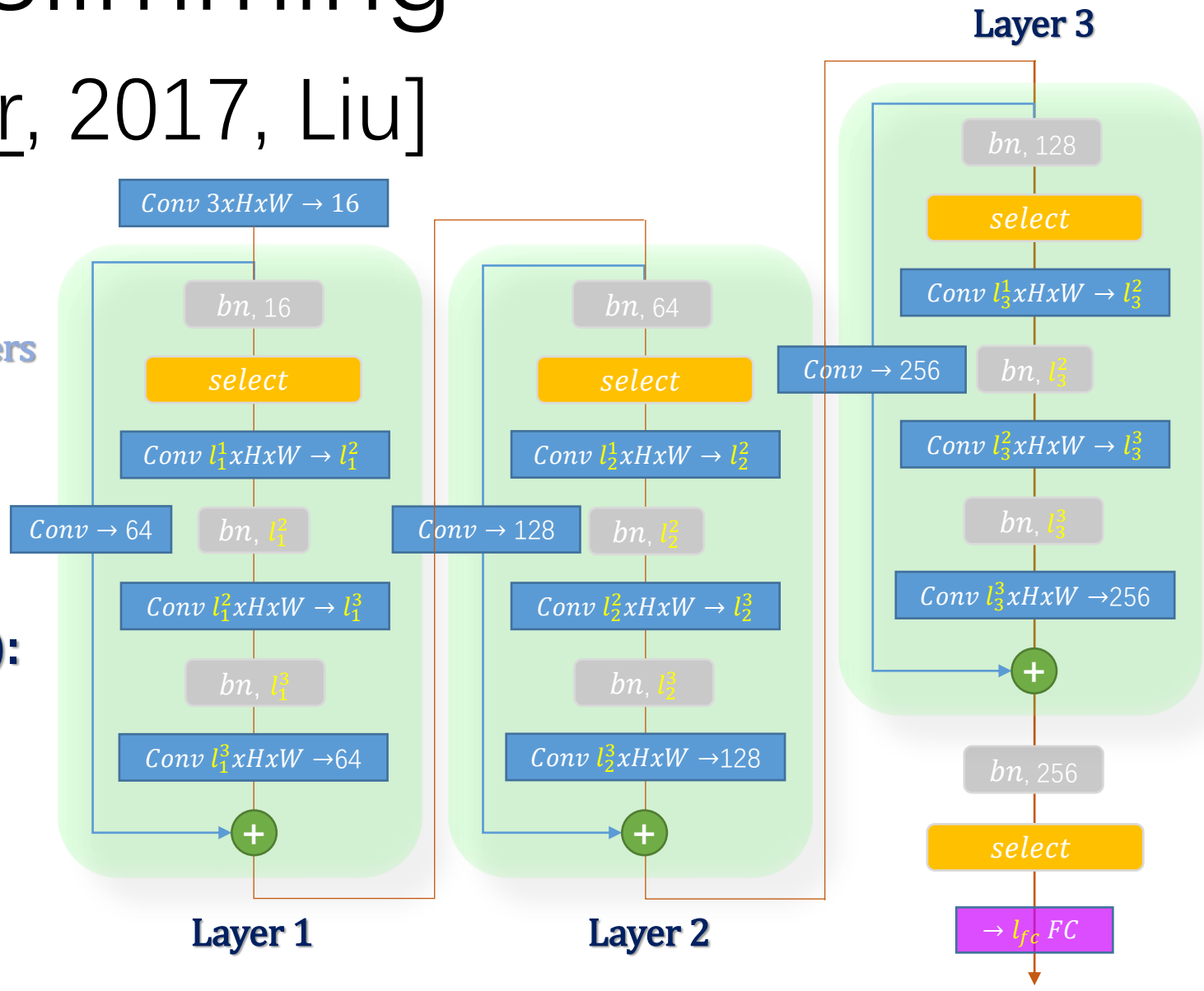
II. Network Slimming

C. Theory [Paper, 2017, Liu]

- **Pipeline:**
  a. Train original model
  b. Load trained model + prune layers
  c. Finetune pruned model

- **E.g.: Resnet (Identity Mapping, 2016):**

➢ Pruning structure
(simplified inner blocks)
(channel # just for illustration)



II. Network Slimming

C. Theory [Paper, 2017, Liu]

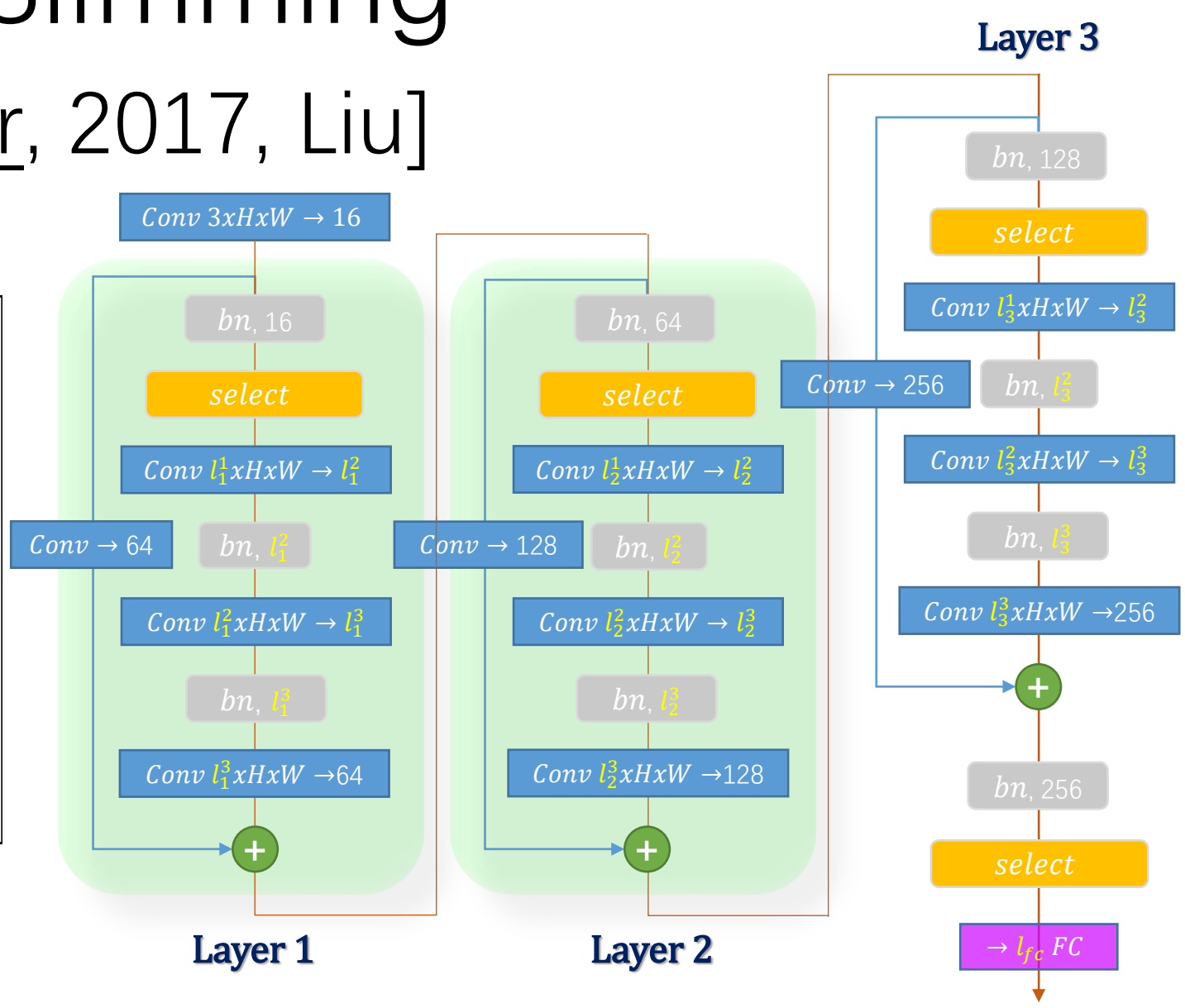
Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

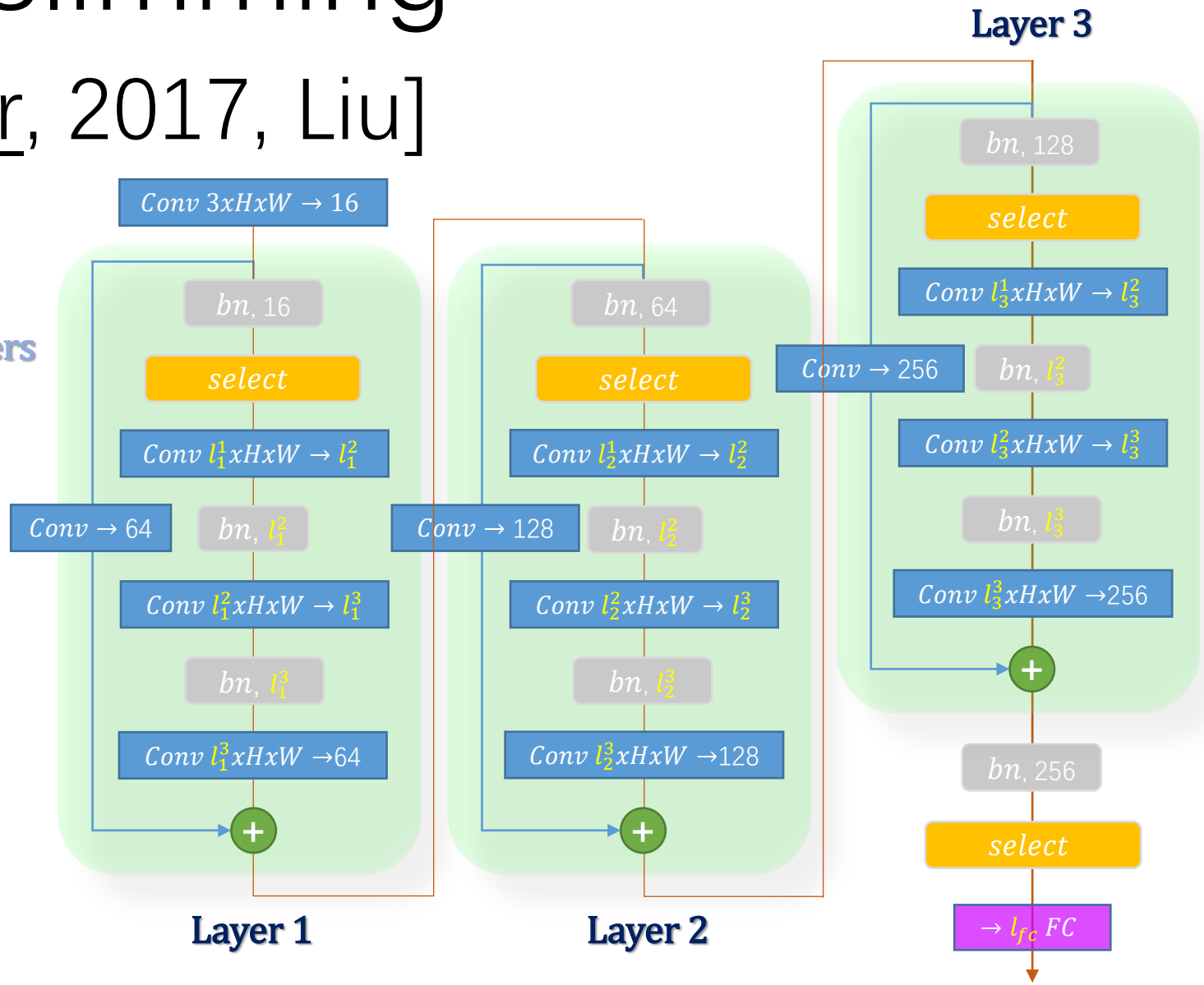


II. Network Slimming

C. Theory [Paper, 2017, Liu]

- **Pipeline:**
  a. Train original model
  b. Load trained model + prune layers
  c. Finetune pruned model

- **Prune Pipeline:**
  a. Sort bn's γ(weight)
  b. Set prune ratio
  c. Create a mask where
     $m_i = 0\ (\gamma_i \leq ratio)$,
     $m_i = 1\ (\gamma_i > ratio)$
  d. Remain channels where $m_i = 1$



II. Network Slimming

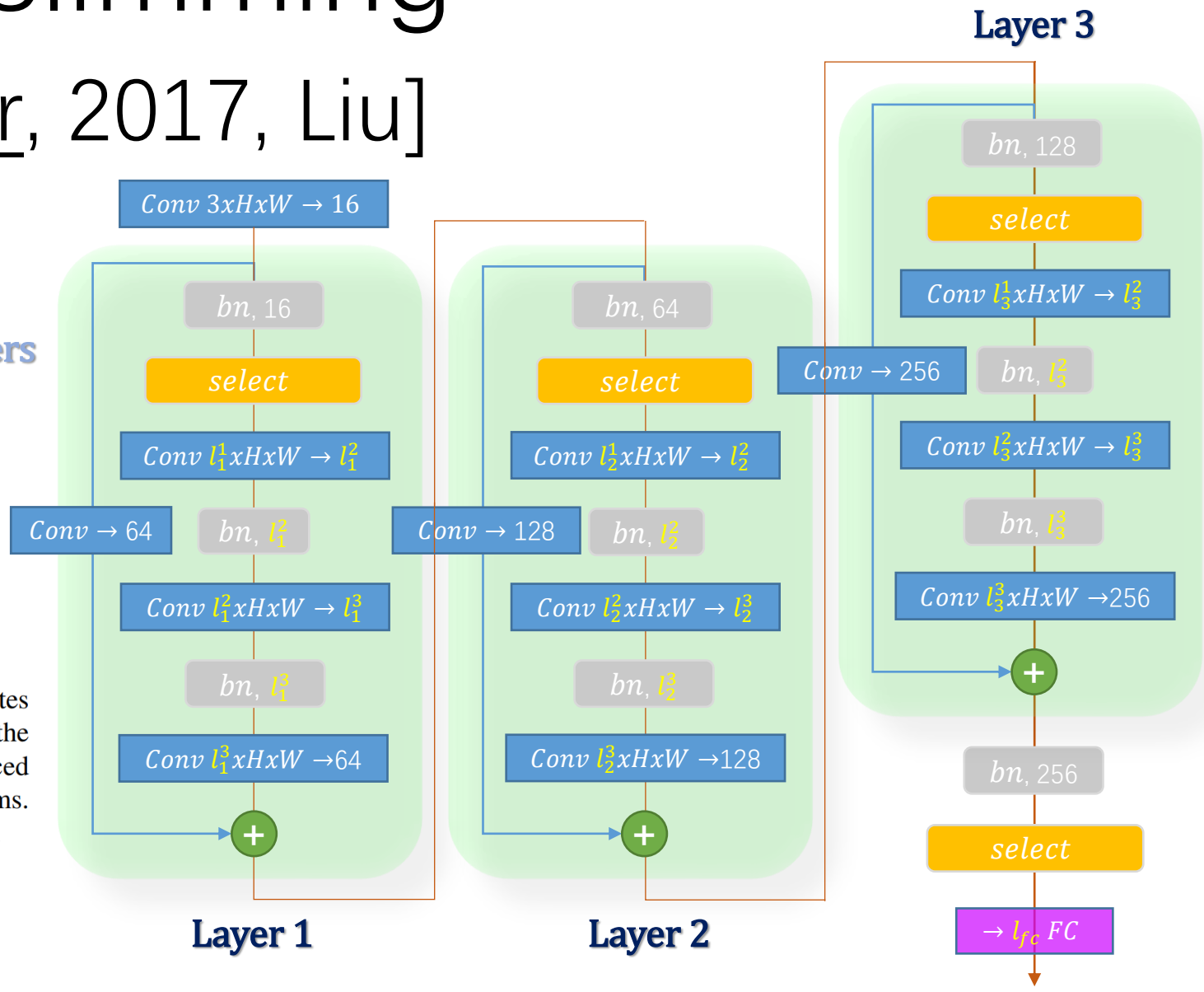
C. Theory [Paper, 2017, Liu]

- **Pipeline:**
  a. Train original model
  b. Load trained model + prune layers
  c. Finetune pruned model

- **Math Theory:**

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

where (x, y) denote the train input and target, W denotes the trainable weights, the first sum-term corresponds to the normal training loss of a CNN, $g(\cdot)$ is a sparsity-induced penalty on the scaling factors, and λ balances the two terms. In our experiment, we choose $g(s) = |s|$



II. Network Slimming

D. Code

- **Resnet e.g.:**

➢ Potential problems:

- Python main.py –dataset cifar10 –arch resnet –depth 20
  Update Anaconda

- IndexError: index 0 is out of bounds for dimension 0 with size 0
  main.py → .data[0] → .data

- print(…, correct)
  float(correct)

II. Network Slimming

D. Code

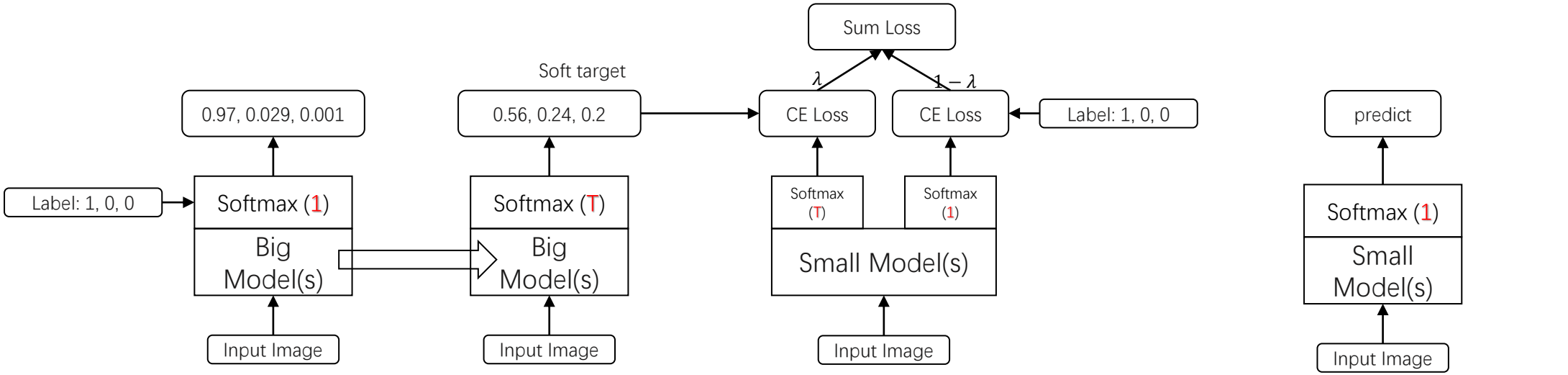
- **Yolo v3 - Prune:**
  - Do it:

III. Knowledge Distillation

III. Knowledge Distillation

E. Theory

- **Original - Paper [2015, Hinton]:**



a. Train big model(s)

b. Generate soft target by using trained model

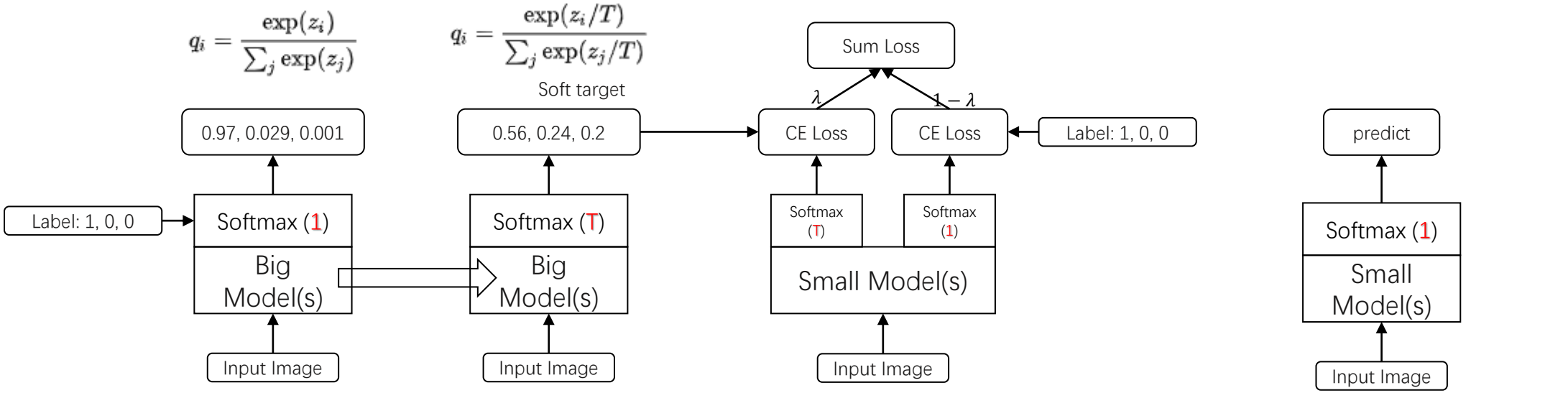
c. Train small model guided by big model

d. Predict by using small model

III. Knowledge Distillation

E. Theory

- **Original - Paper [2015, Hinton]:**



a. Train big model(s)

b. Generate soft target by using trained model

c. Train small model guided by big model

d. Predict by using small model

III. Knowledge Distillation

E. Theory

- **Original - Paper [2015, Hinton]:**

III. Knowledge Distillation

E. Theory

- **Problems:**
  - Can only be used in classification tasks
  - Knowledge can only be distilled at last layers
  - Distilled layer type can only be FC (fully connected) layers

III. Knowledge Distillation

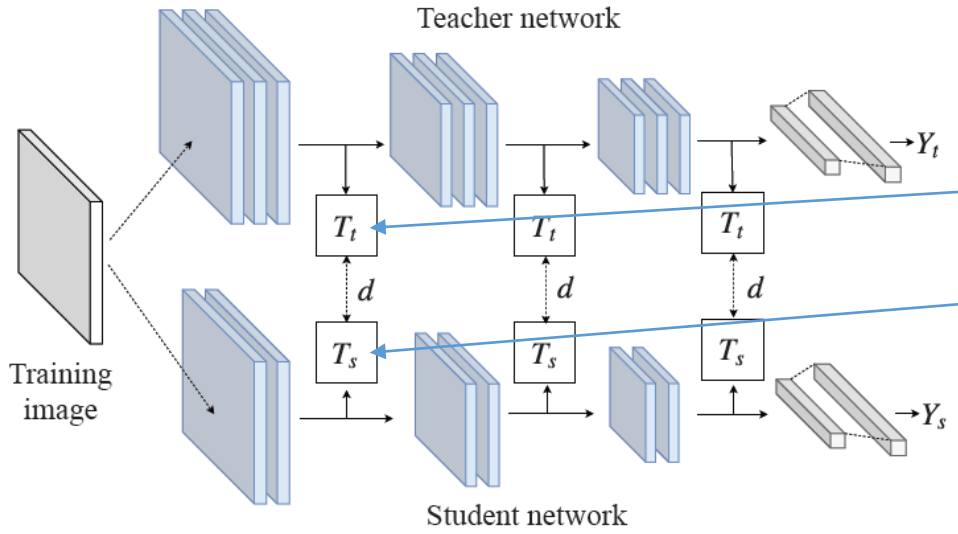
E. Theory

- **Problems:**
  - Can only be used in classification tasks
  - Knowledge can only be distilled at last layers
  - Distilled layer type can only be FC (fully connected) layers
- **What we need / want:**
  - Used in tasks other than classification (detection / segmentation / image transform / …)
  - Knowledge could be distilled at other layers besides last ones
  - Distilled layer type could be feature layers other than FC layers

III. Knowledge Distillation

F. Feature Distillation

- **paper: A Comprehensive Overhaul of Feature Distillation [2019, Heo]**



4 factors:

- Teacher transform, T_t
- Student transform, T_s
- Distillation feature position
- Distance function

- **code**

III. Knowledge Distillation

F. Feature Distillation

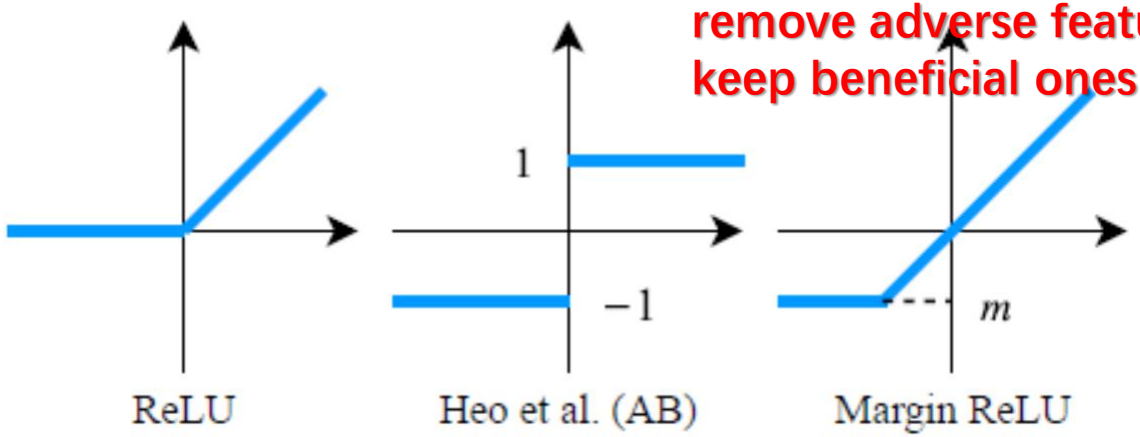
4 factors:

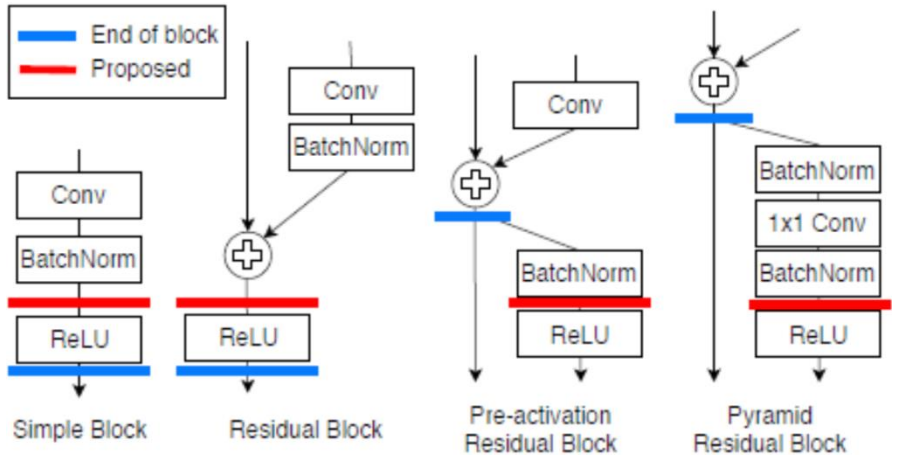
- **Teacher transform, T_t, Margin ReLU**
- **Student transform, T_s, 1x1 conv, no info missing**
- **Distillation feature position, pre-ReLU**
- **Distance function**

Truly suppressed values which are smaller than the margin

$$d_p(\boldsymbol{T},\boldsymbol{S}) = \sum_i^{WHC} \begin{cases} 0 & if\ S_i \le T_i \le 0 \\ (T_i - S_i)^2 & otherwise \end{cases}$$

remove adverse features, keep beneficial ones





III. Knowledge Distillation

F. Feature Distillation

4 factors:

- **Teacher transform, T_t, Margin ReLU**
- **Student transform, T_s, 1x1 conv, no info missing**
- **Distillation feature position, pre-ReLU**
- **Distance function**

	Baseline	+ Position (Sec.3.1)	+ BN (Sec.3.3)	+ loss (Sec.3.2)
Error	26.37	24.81	24.68	24.08
Diff	-	-1.56	-0.13	-0.60

Truly suppressed values which are smaller than the margin

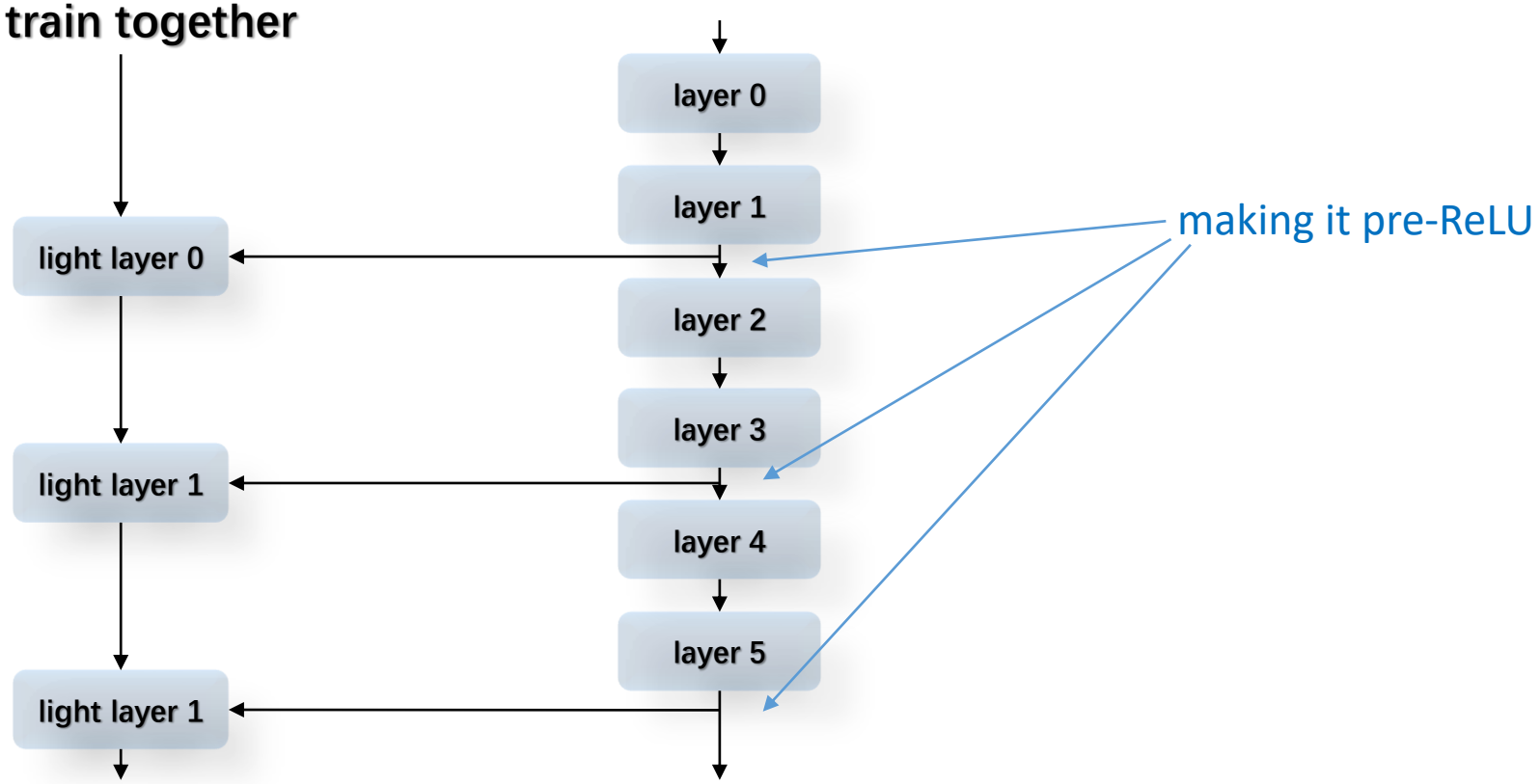
$$d_p(\boldsymbol{T},\boldsymbol{S}) = \sum_i^{WHC} \begin{cases} 0 & if\ S_i \le T_i \le 0 \\ (T_i - S_i)^2 & otherwise \end{cases}$$

III. Knowledge Distillation

F. Feature Distillation

- **A possible training procedure to embed this method**

➢ **Method 1: train together**

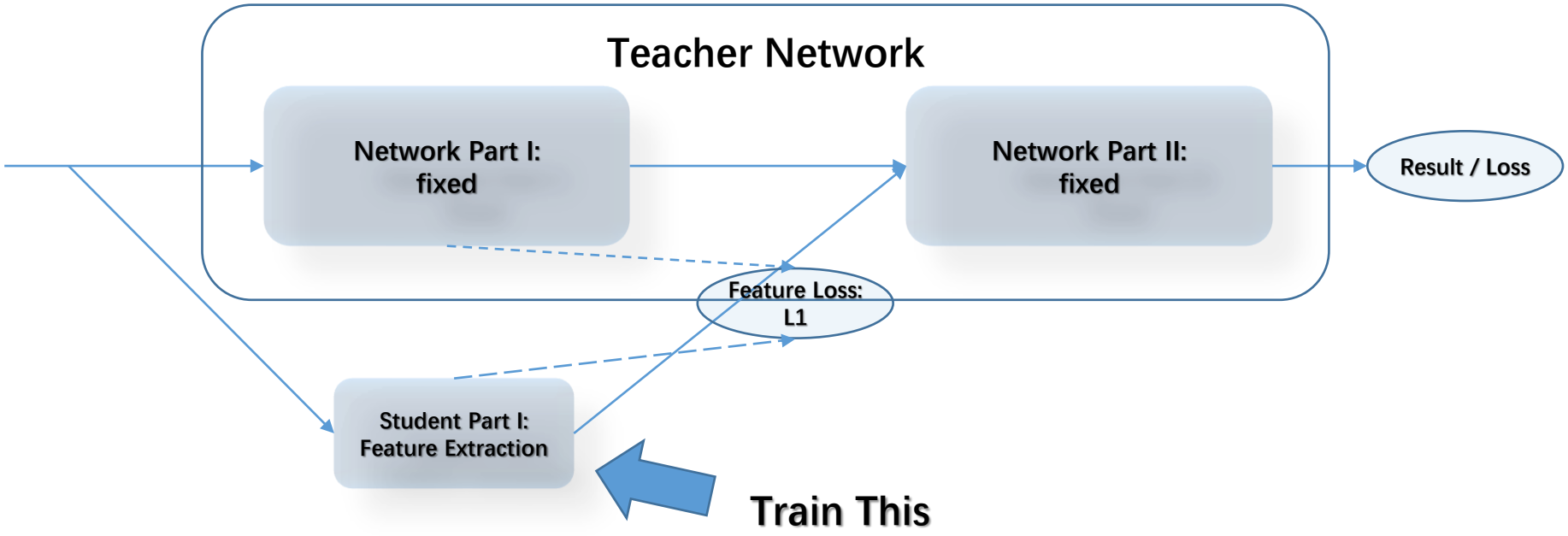


III. Knowledge Distillation

F. Feature Distillation

- **A possible training procedure to embed this method**

➢ **Method 2: train separately**

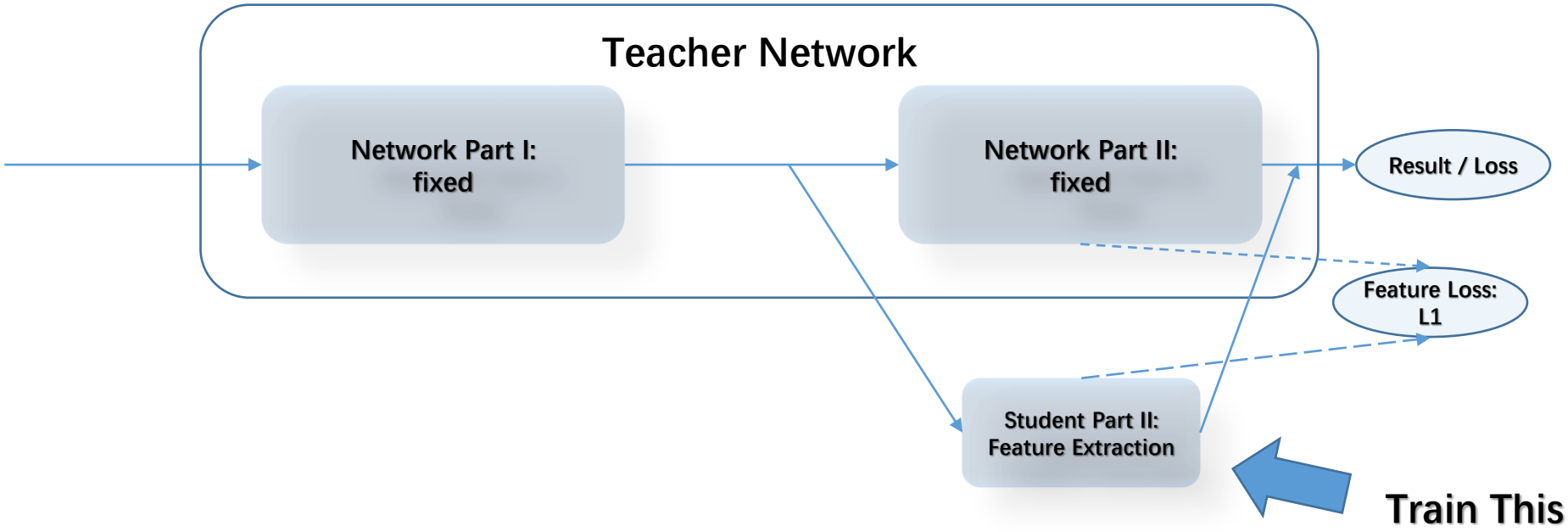


III. Knowledge Distillation

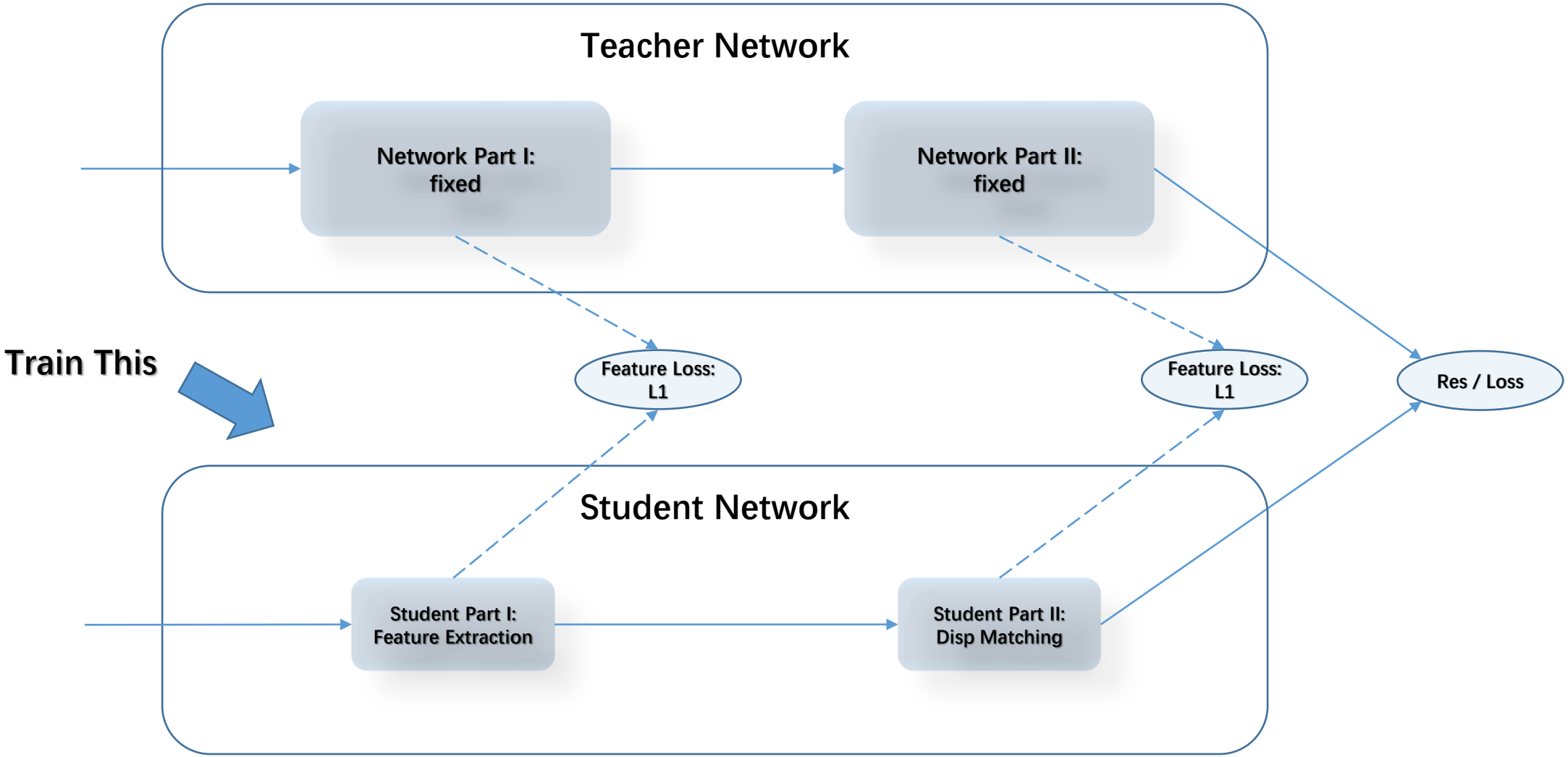
F. Feature Distillation

- **A possible training procedure to embed this method**

➢ **Method 2: train separately**



III. Knowledge Distillation



Summary

- **What we are doing:**

- **It could be done like:**