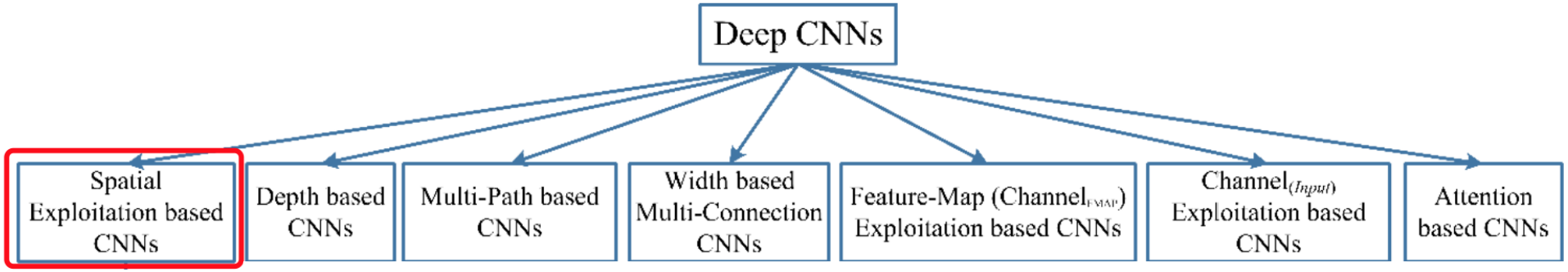
SUSTech Southern University of Science and Technology

ML4CV GROUP

Group Discussion

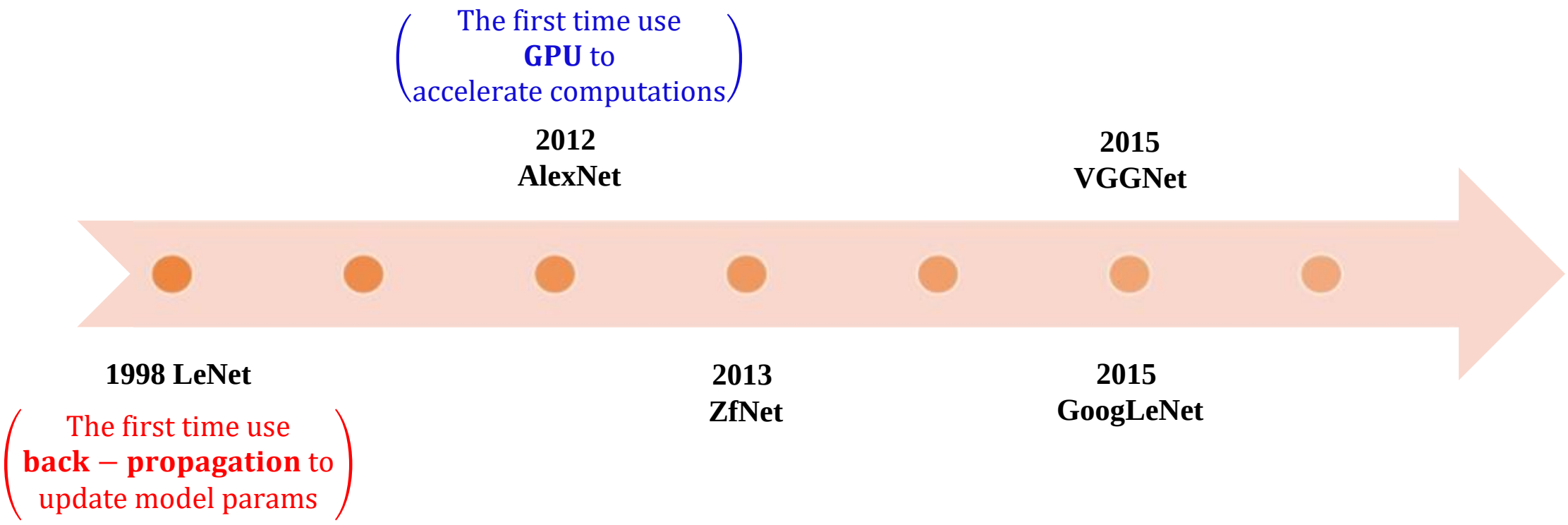
November 2, 2020

Spatial Exploitation based CNNs

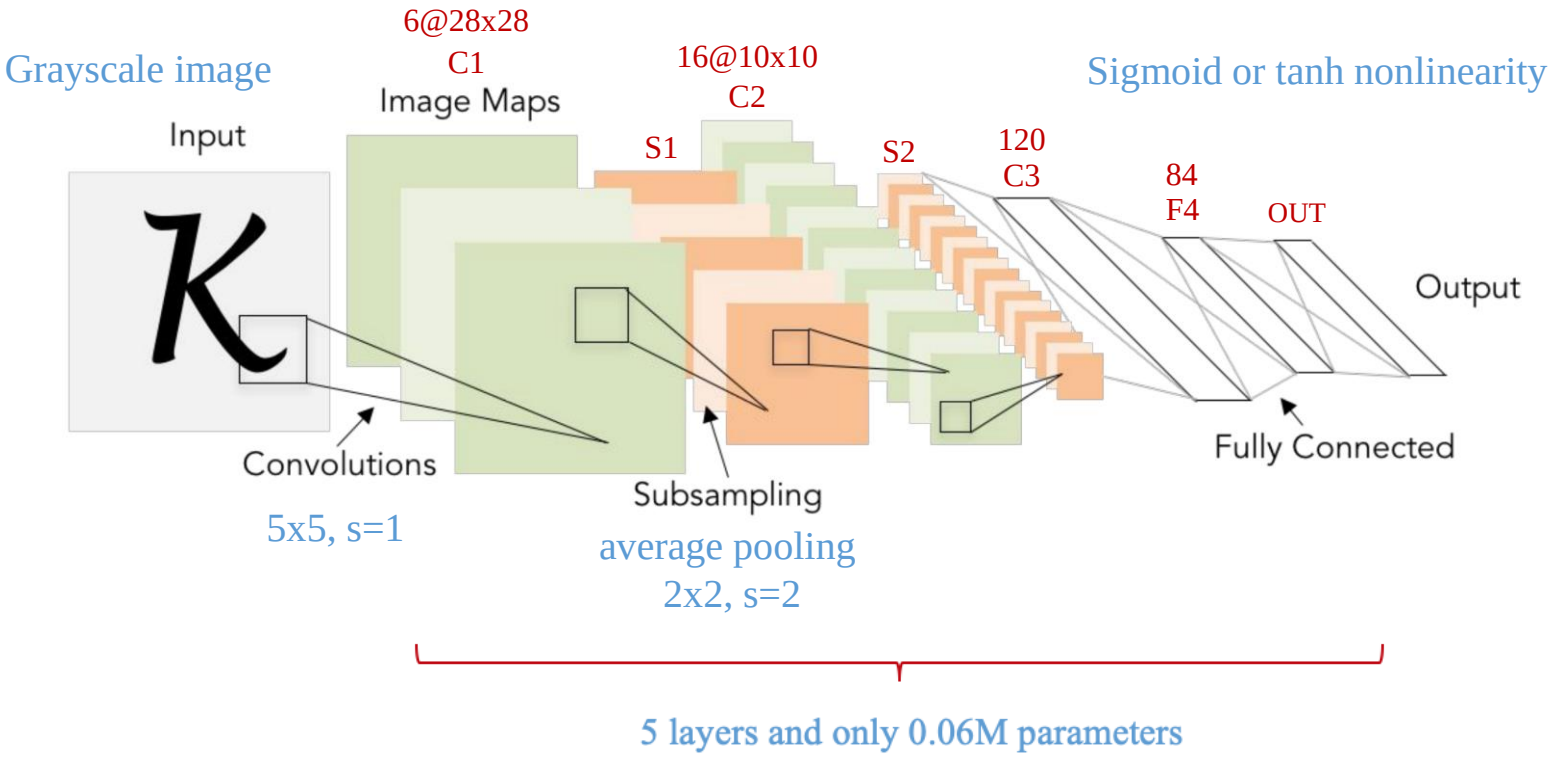


2020.10.12

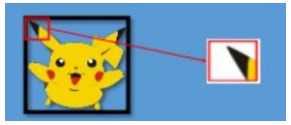
Spatial Exploitation:



LeNet: First popular CNN architecture

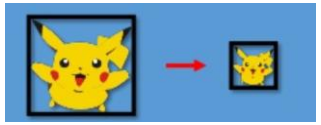


3 key ideas:



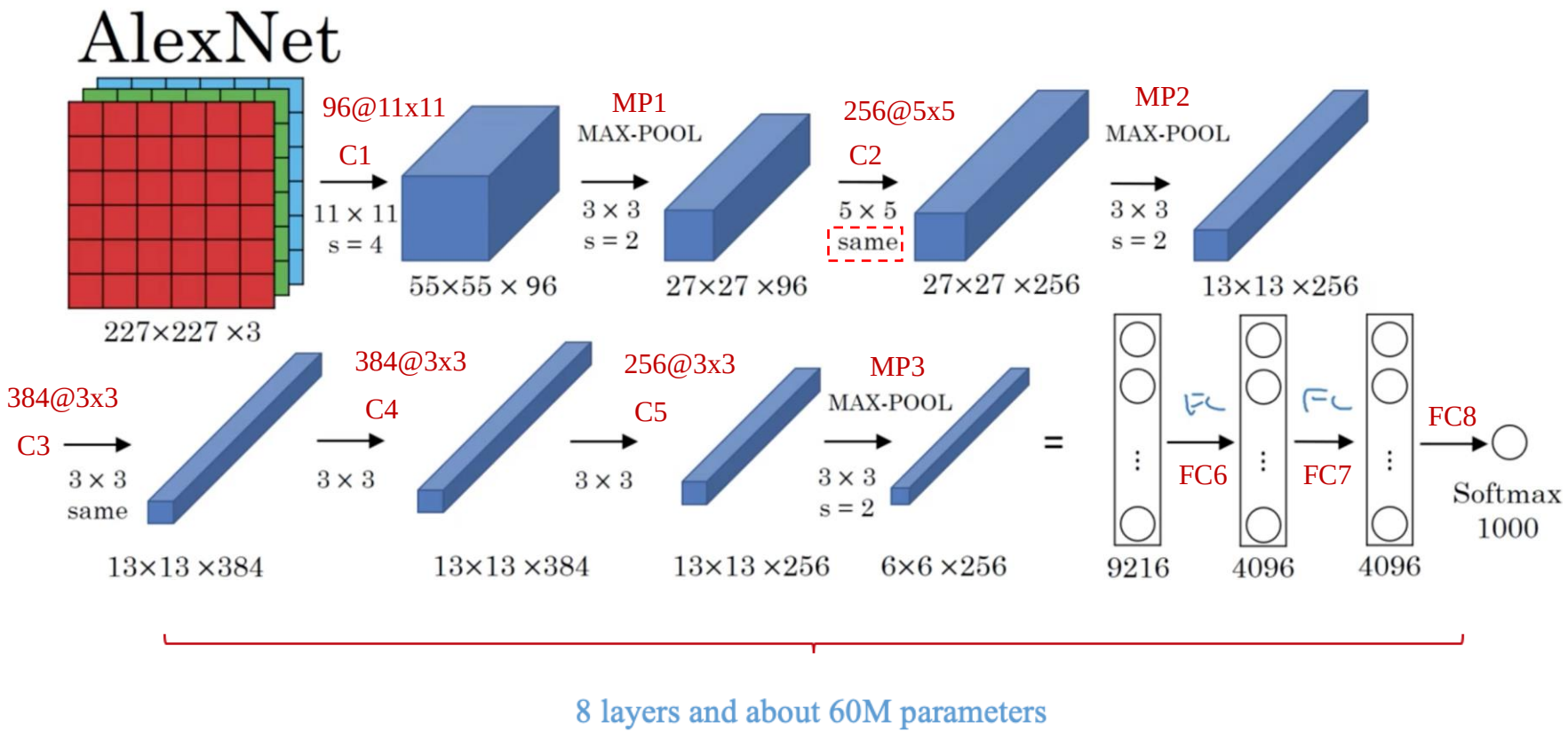
- Local Receptive Fields
- Shared Weights

- Sub-sampling



LeCun et al. Gradient Based Learning Applied to Document Recognition. 1998

AlexNet: First deep CNN architecture

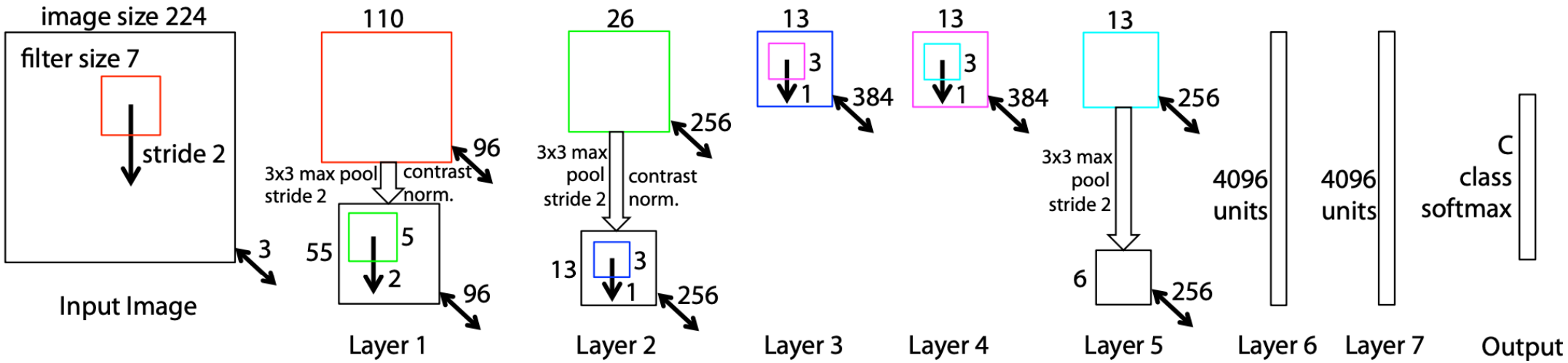


key contributions:

- GPU and training in parallel
- **ReLu** Activation
- Dropout **regularization**
- Heavy data Augmentation

Krizhevsky et al. ImageNet Classification with Deep Convolutional Neural Networks. 2012

ZfNet: Refinement of AlexNet



➢ **AlexNet but:**

CONV1:

- change from (11x11 stride 4) to (7x7 stride 2)

CONV3, 4, 5:

- instead of 384, 384, 256 filters use 512, 1024, 512

key contributions:

- Introduced the idea of parameter tuning

by visualizing the output of intermediate layers

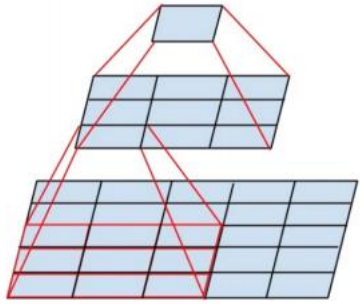
Zeiler and Fergus. Visualizing and Understanding Convolutional Networks. 2013

VGGNet: Small filters, deeper networks

Why small filters:

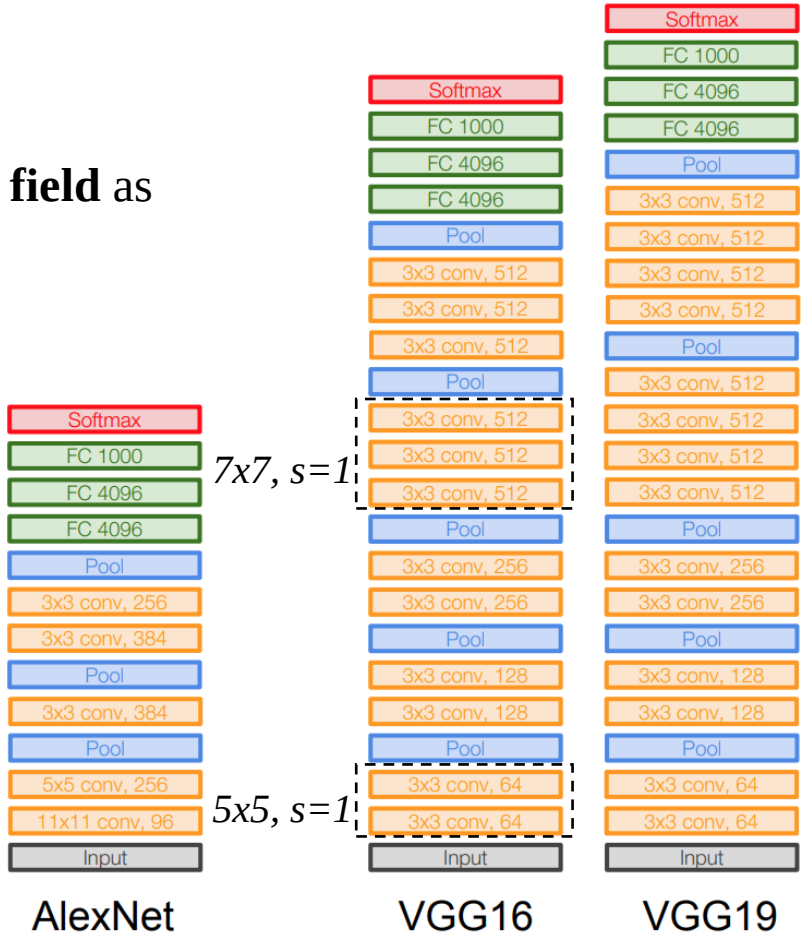
- Stack of three 3x3 conv(stride 1) layers has same **effective receptive field** as one 7x7 conv layer
- But deeper, more non-linearities, and fewer parameters:

$3 * (3^2)$ and (7^2)



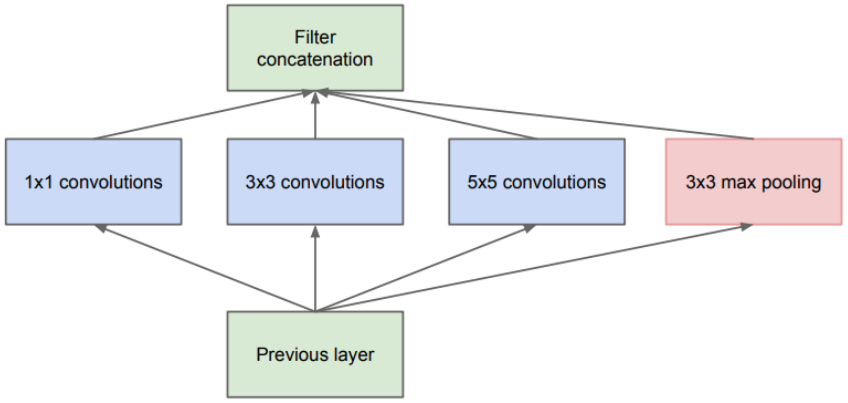
key points:

- Depth is important
- Simplify the network to go deep
- 138M parameters(mostly due to the FC layers)

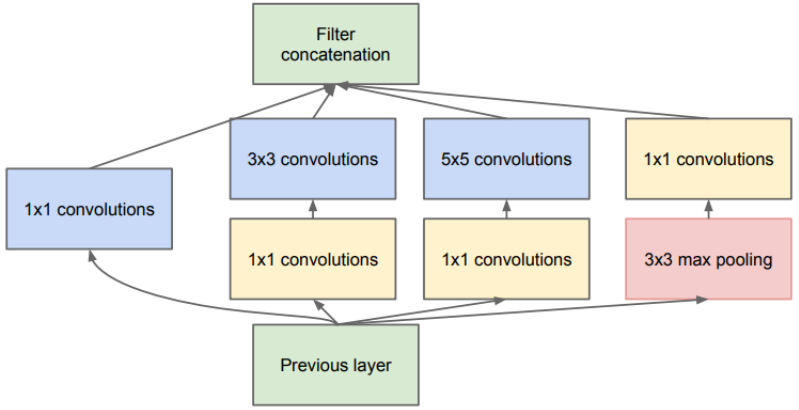


Simonyan and Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. 2015

GoogLeNet: The Inception Module

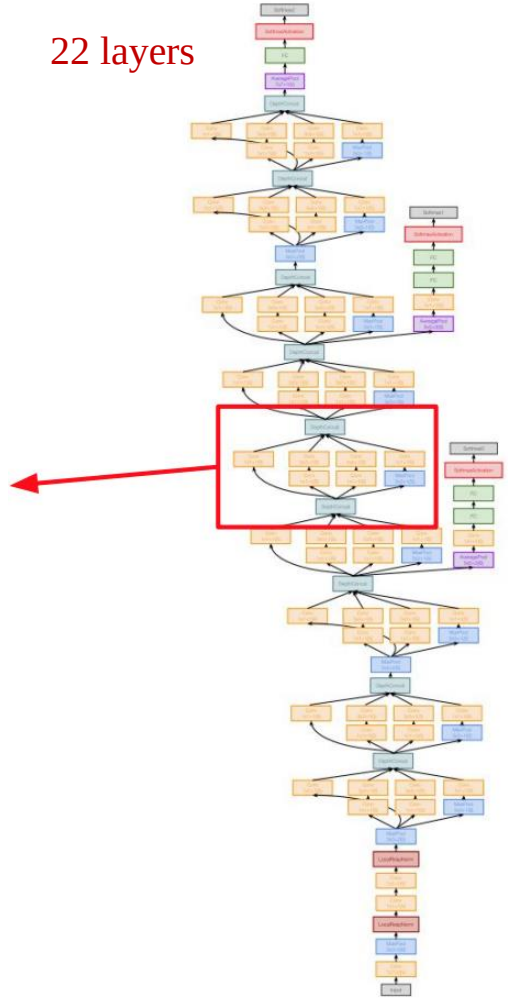


Naïve Inception Module



Inception Module

22 layers

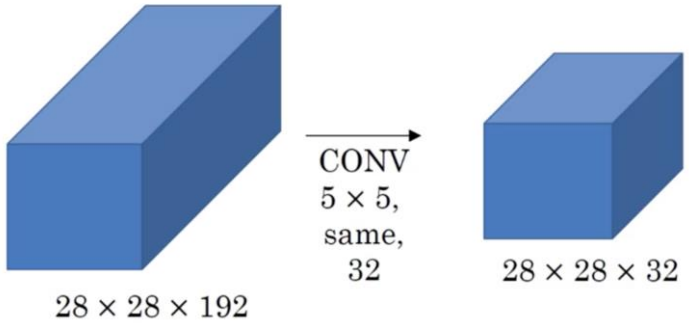


Parallel paths with different receptive field sizes and operations

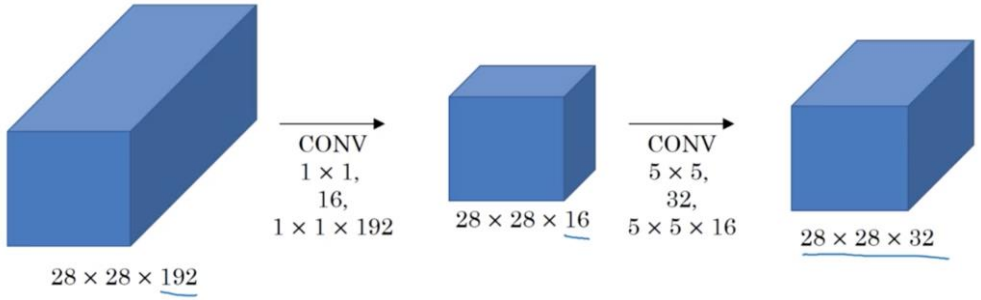
- Capture sparse patterns of correlations in the stack of feature maps

Szegedy et al. Going deeper with convolutions. 2015

GoogLeNet: With computational efficiency



Conv operations: $28 * 28 * 32 * 5 * 5 * 192 = 120M$



Conv operations: $28 * 28 * 16 * 1 * 1 * 192 + 28 * 28 * 32 * 5 * 5 * 16 = 12.4M$

2.4M 10M

1x1 conv "bottleneck" layers:

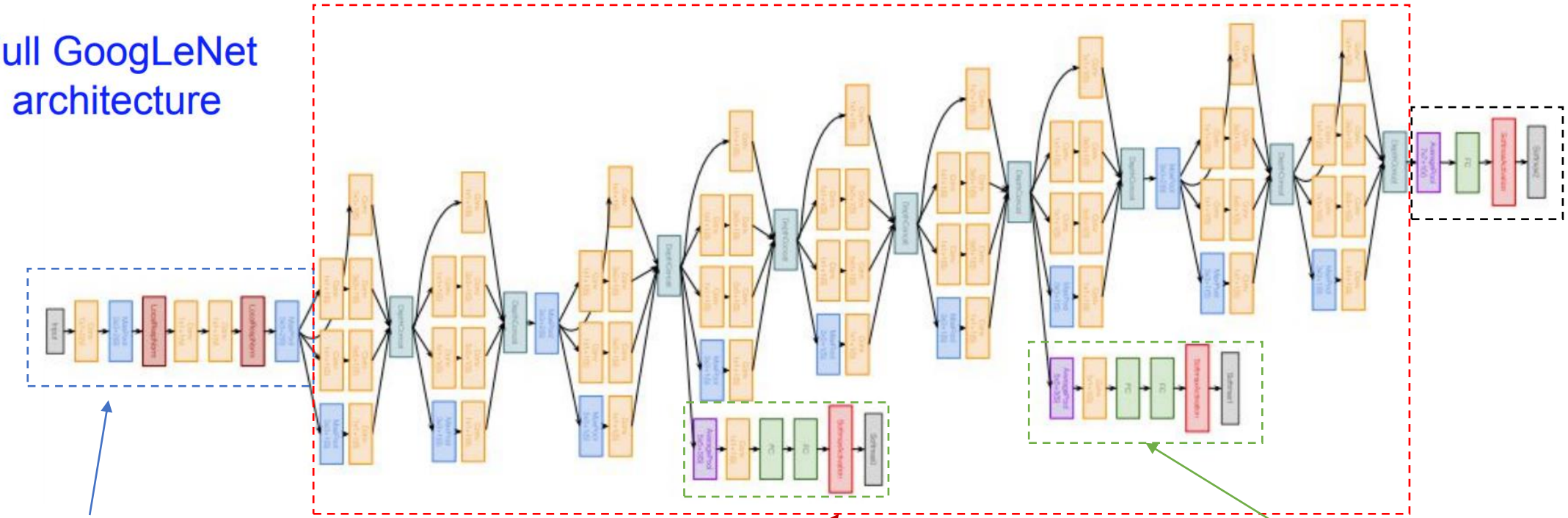
- Preserves spatial dimensions, reduce feature depth

key ideas:

- 22 layers
- Efficient "inception" module
- Use "bottleneck" layer to reduce feature depth
- No FC layers
- Only 4M parameters

Szegedy et al. Going deeper with convolutions. 2015

GoogLeNet: Architecture



Szegedy et al. Going deeper with convolutions. 2015

Depth Based CNNs

Zhiqiang Wang
October 12, 20220
Southern University of Science and Technology

Highway Networks

如何训练深度网络

Question:

Theoretical and empirical evidence show that depth of neural networks is a crucial ingredient for their success. However, network training becomes more difficult with increasing depth.

layers of a plain feedforward neural network

$$\mathbf{y} = H(\mathbf{x}, \mathbf{W_H}).$$

H is usually an afffine transform followed by a non-linear activation function.

architecture of highway networks

$$\mathbf{y} = H(\mathbf{x}, \mathbf{W_H}) \cdot T(\mathbf{x}, \mathbf{W_T}) + \mathbf{x} \cdot C(\mathbf{x}, \mathbf{W_C}).$$

T denotes transform gate and ***C*** denotes carry gate.

For simplicity, we can set ***C = 1 - T***

$$\mathbf{y} = H(\mathbf{x}, \mathbf{W_H}) \cdot T(\mathbf{x}, \mathbf{W_T}) + \mathbf{x} \cdot (1 - T(\mathbf{x}, \mathbf{W_T})).$$

$$\mathbf{y} = \begin{cases} \mathbf{x}, & \text{if } T(\mathbf{x}, \mathbf{W_T}) = 0, \\ H(\mathbf{x}, \mathbf{W_H}), & \text{if } T(\mathbf{x}, \mathbf{W_T}) = 1. \end{cases}$$

A block with two non-linear transform.

$$T(\mathbf{x}) = \sigma(\mathbf{W_T}^T \mathbf{x} + \mathbf{b_T})$$

$\boldsymbol{b_T}$ can be initialized with a negative value such that the network is initially biased towards carry behavior.

Highway Networks

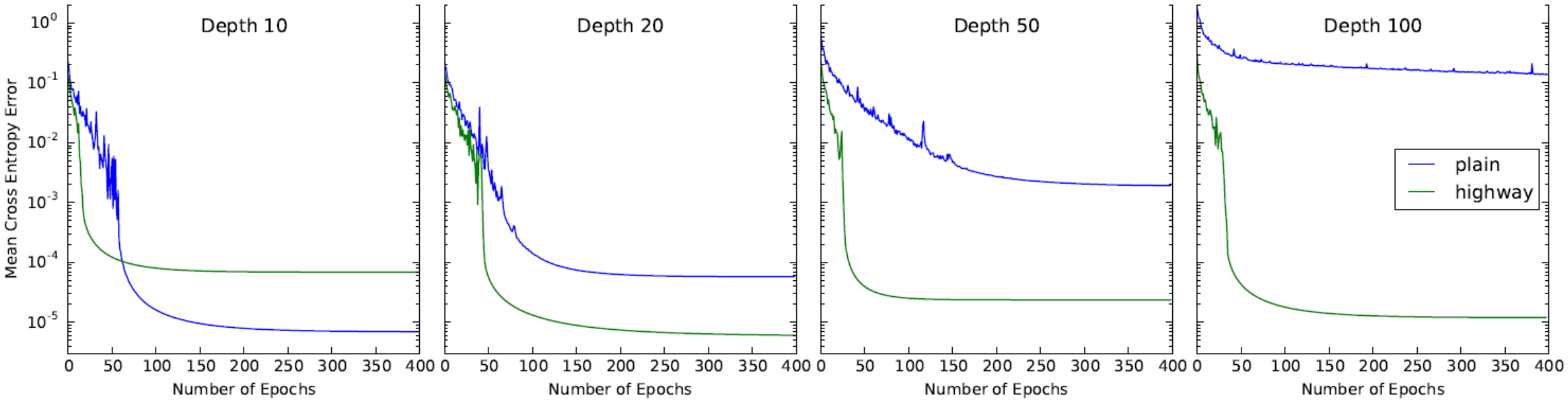


Figure 1. Comparison of optimization of plain networks and highway networks of various depths. All networks were optimized using SGD with momentum. The curves shown are for the best hyperparameter settings obtained for each configuration using a random search. Plain networks become much harder to optimize with increasing depth, while highway networks with up to 100 layers can still be optimized well.

ResNet

Question:

Deeper neural networks are more difficult to train.

Instead of hoping each few stacked layers directly fit a desired underlying mapping, we explicity let these layers fit a residual mapping.

$$\mathbf{y} = \mathcal{F}(\mathbf{x}, \{W_i\}) + \mathbf{x}.$$

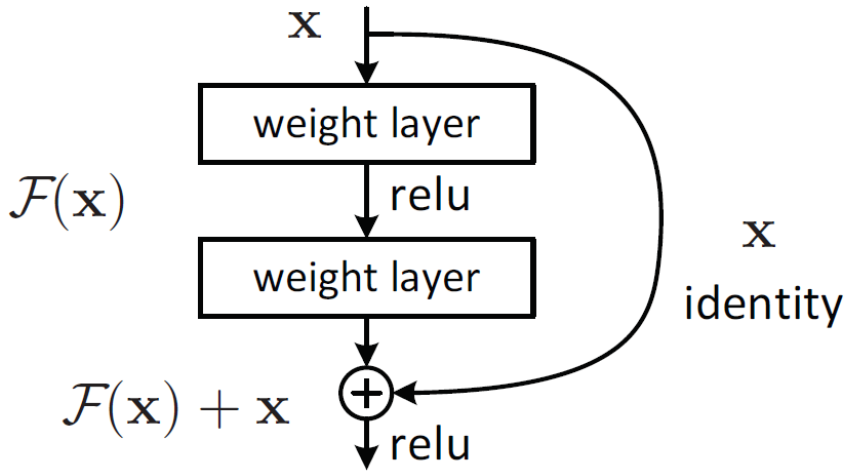


Figure 2. Residual learning: a building block.

Matching dimensions. (Size & Depth)

$$\mathbf{y} = \mathcal{F}(\mathbf{x}, \{W_i\}) + W_s\mathbf{x}.$$

The form of the residual function ***F*** is flexible.
(2 or 3 layers, while more layers are possible)

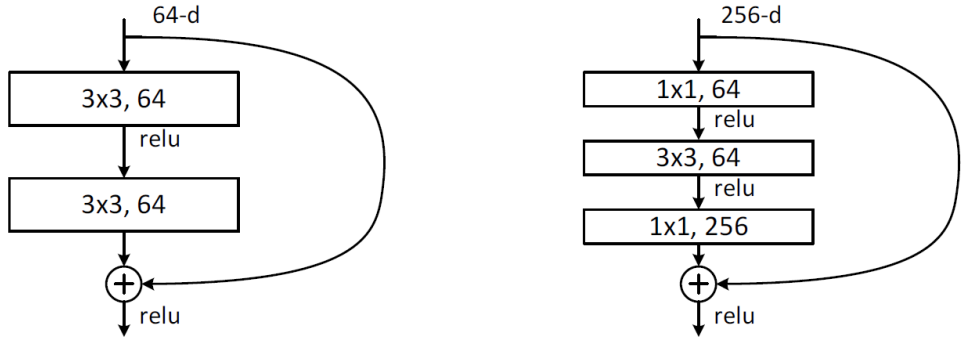


Figure 5. A deeper residual function $\mathcal{F}$ for ImageNet. Left: a building block (on 56×56 feature maps) as in Fig. 3 for ResNet-34. Right: a "bottleneck" building block for ResNet-50/101/152.

ResNet

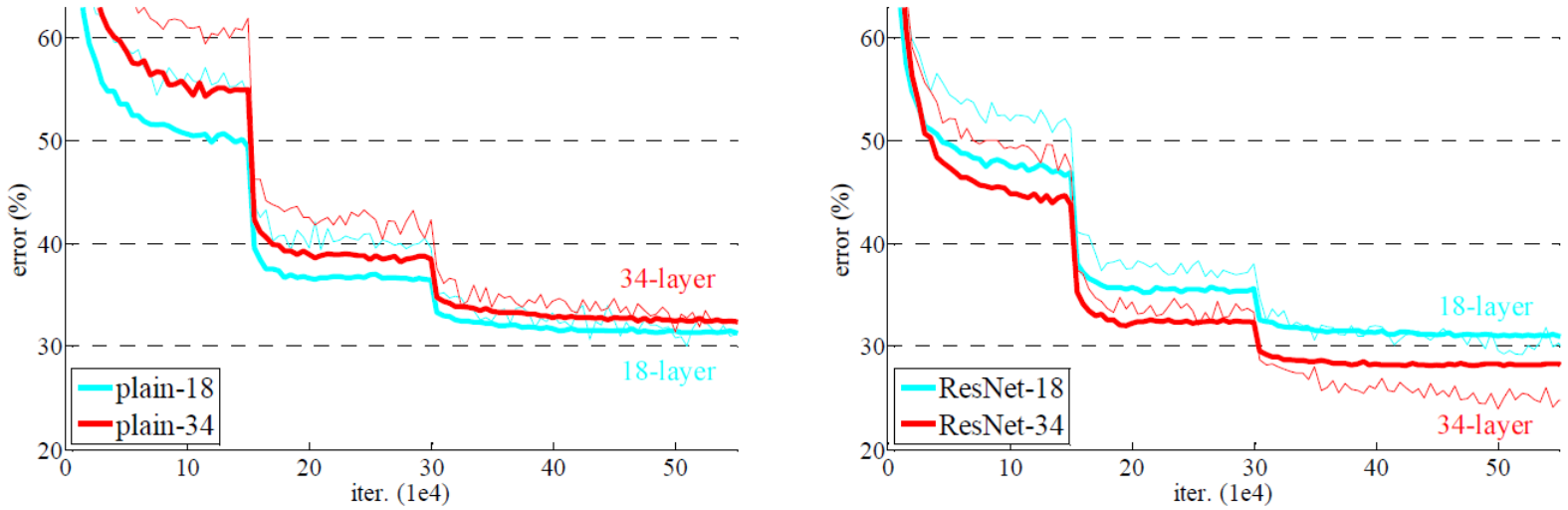


Figure 4. Training on **ImageNet**. Thin curves denote training error, and bold curves denote validation error of the center crops. Left: plain networks of 18 and 34 layers. Right: ResNets of 18 and 34 layers. In this plot, the residual networks have no extra parameter compared to their plain counterparts.

	plain	ResNet
18 layers	27.94	27.88
34 layers	28.54	**25.03**

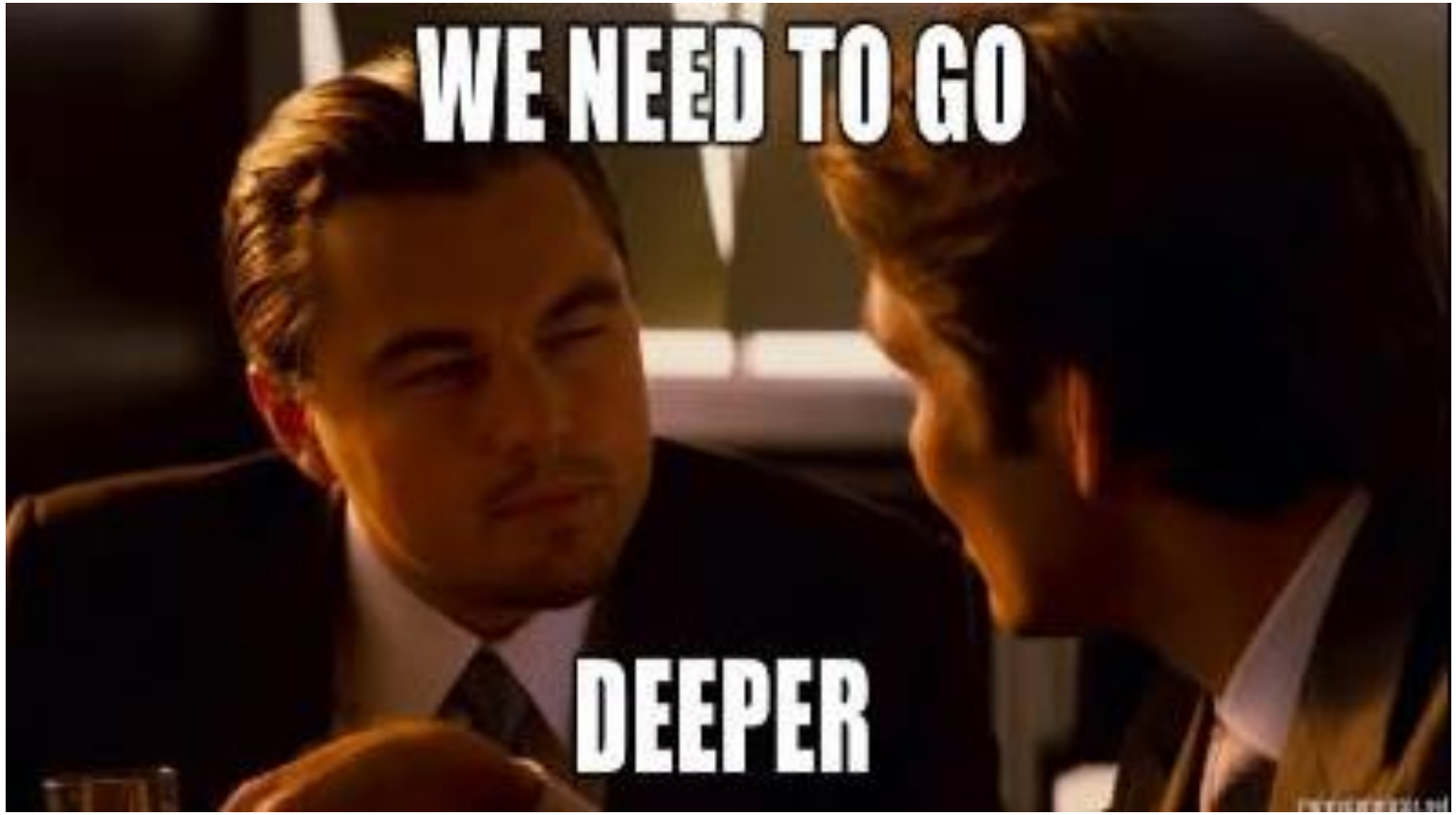
Table 2. Top-1 error (%, 10-crop testing) on ImageNet validation. Here the ResNets have no extra parameter compared to their plain counterparts. Fig. 4 shows the training procedures.

深层网络的退化问题在ResNet中得到了解决，并且预测的准确率随着网络的加深而提高了。

相比于plain networks，同等的ResNet在提升准确率上表现得更好。

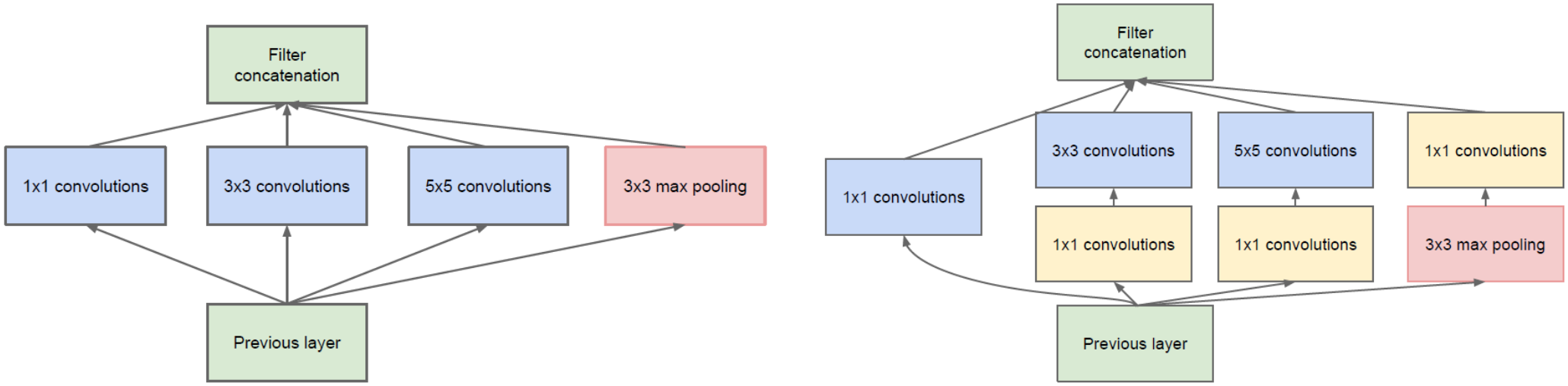
ResNet在训练的前期收敛更快。

Inception



Inception v1

The main hallmark of this architecture is the improved utilization of the computing resources inside the network. This was achieved by a carefully crafted design that allows for increasing the depth and width of the network while keeping the computational budget constant.

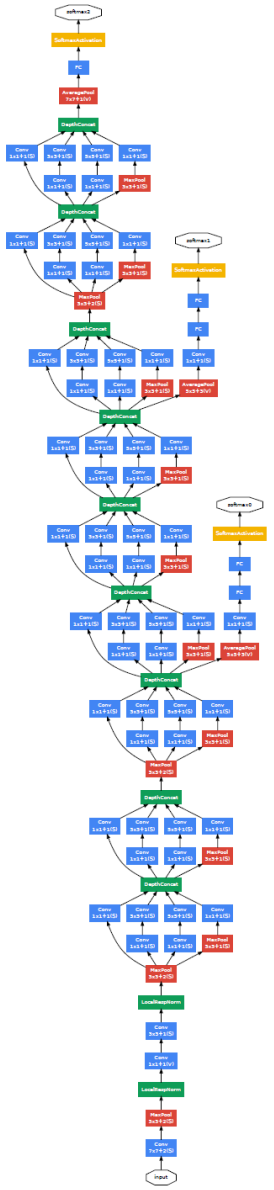


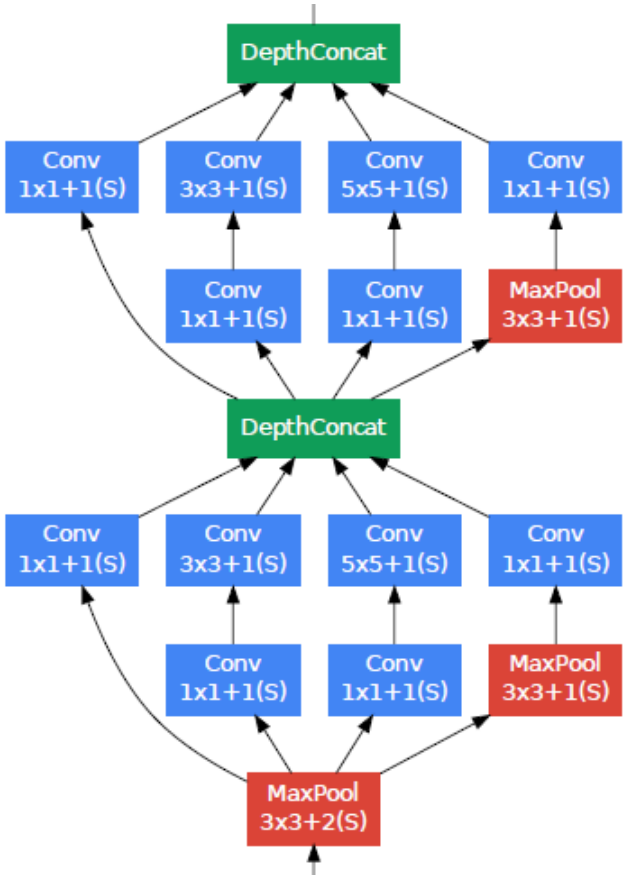
(a) Inception module, naïve version

(b) Inception module with dimension reductions

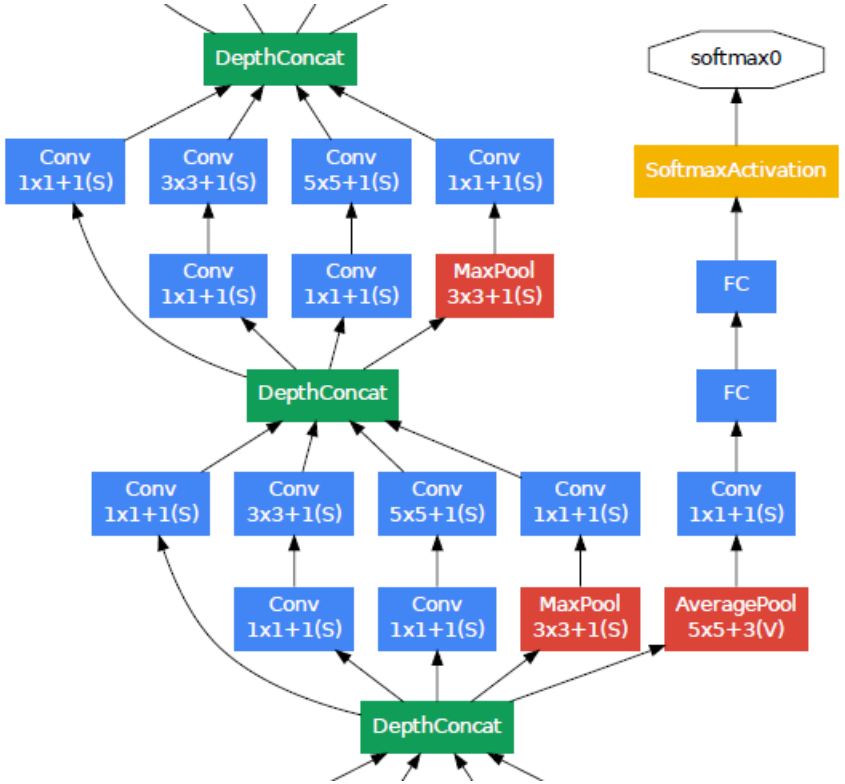
Figure 2: Inception module

GoogLeNet





Inception v1



Auxiliary classifier

Inception v2 & Inception v3

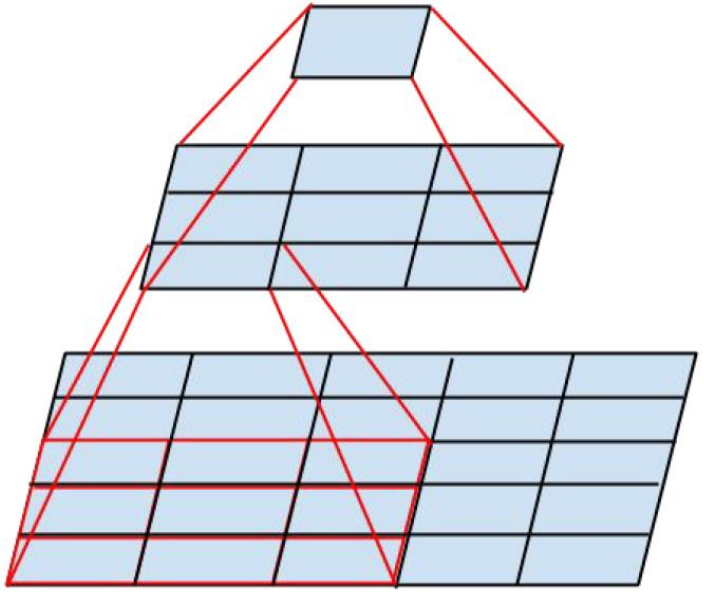


Figure 1. Mini-network replacing the 5×5 convolutions.

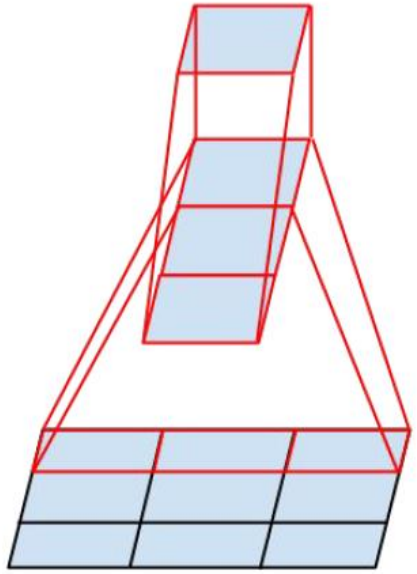


Figure 3. Mini-network replacing the 3×3 convolutions. The lower layer of this network consists of a 3×1 convolution with 3 output units.

Inception v2 & Inception v3

The idea of Inception-v3 was to reduce the computational cost of deep networks without affecting the generalization by replacing large size filters with small and asymmetric filters and using 1*1 convolution as a bottleneck before the large filters.

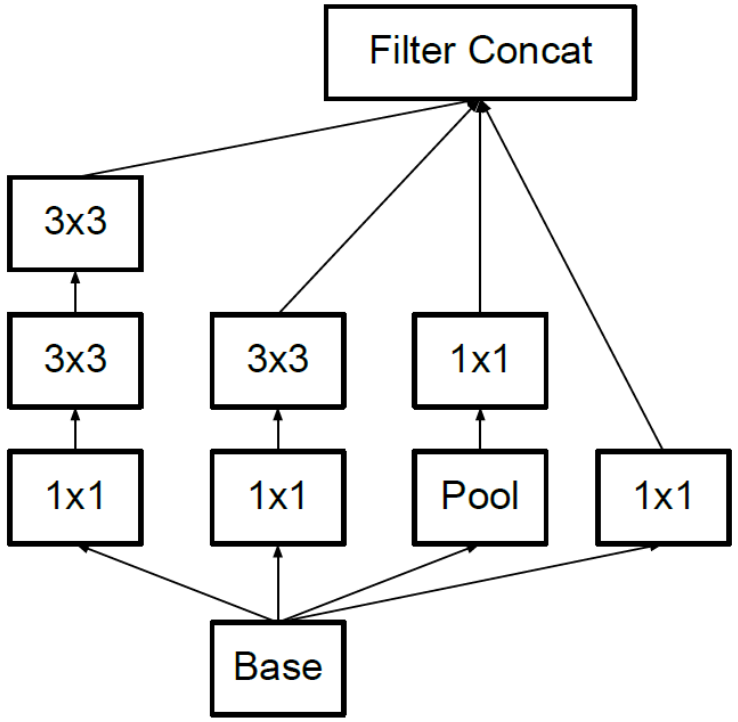


Figure 5. Inception modules where each 5×5 convolution is replaced by two 3×3 convolution, as suggested by principle 3 of Section 2.

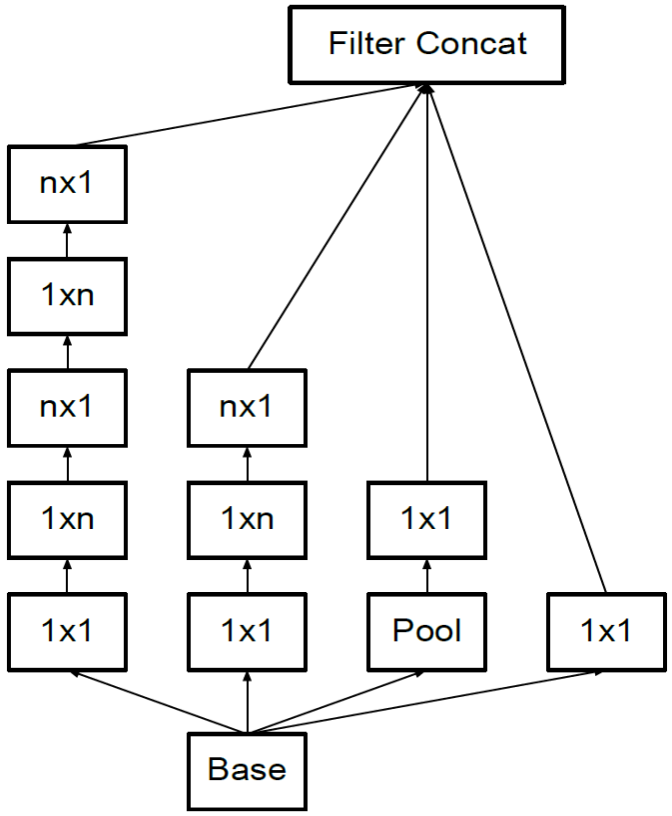


Figure 6. Inception modules after the factorization of the $n \times n$ convolutions. In our proposed architecture, we chose $n = 7$ for the 17×17 grid. (The filter sizes are picked using principle 3)

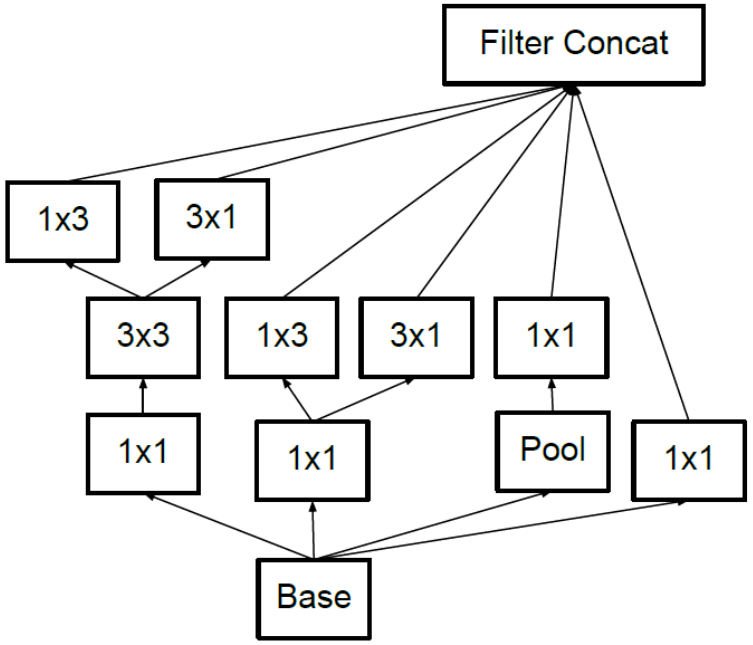


Figure 7. Inception modules with expanded the filter bank outputs. This architecture is used on the coarsest (8×8) grids to promote high dimensional representations, as suggested by principle 2 of Section 2. We are using this solution only on the coarsest grid, since that is the place where producing high dimensional sparse representation is the most critical as the ratio of local processing (by 1×1 convolutions) is increased compared to the spatial aggregation.

Inception v2 & Inception v3

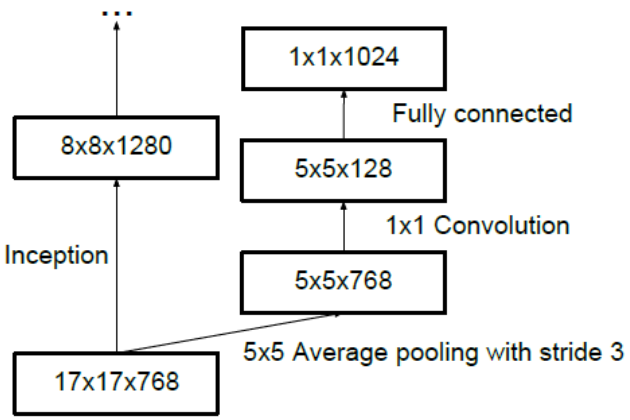


Figure 8. Auxiliary classifier on top of the last 17×17 layer. Batch normalization[7] of the layers in the side head results in a 0.4% absolute gain in top-1 accuracy. The lower axis shows the number of itertions performed, each with batch size 32.

在训练的早期，辅助分类器的存在并没促进收敛；
而训练快结束时，辅助分类器的存在会提升网络的准确率。
结论：辅助分类器起到的是正则化的作用。

Grid Size Reduction

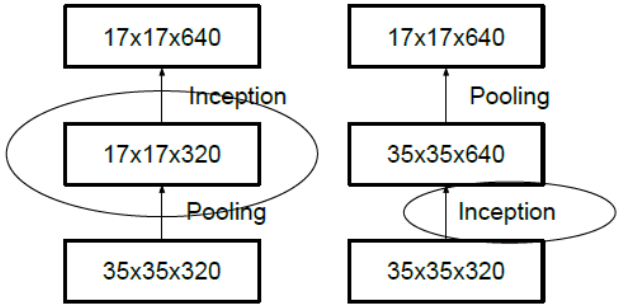


Figure 9. Two alternative ways of reducing the grid size. The solution on the left violates the principle 1 of not introducing an representational bottleneck from Section 2. The version on the right is 3 times more expensive computationally.

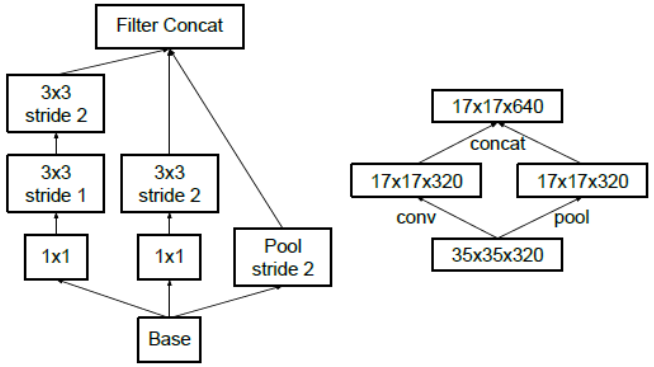


Figure 10. Inception module that reduces the grid-size while expands the filter banks. It is both cheap and avoids the representational bottleneck as is suggested by principle 1. The diagram on the right represents the same solution but from the perspective of grid sizes rather than the operations.

左图：
先池化，再1*1卷积升维
信息丢失太大
右图：
先升维，再池化
计算开销非常大

池化和卷积并行
在降低计算量的同时，避免表征瓶颈

Inception v2 & Inception v3

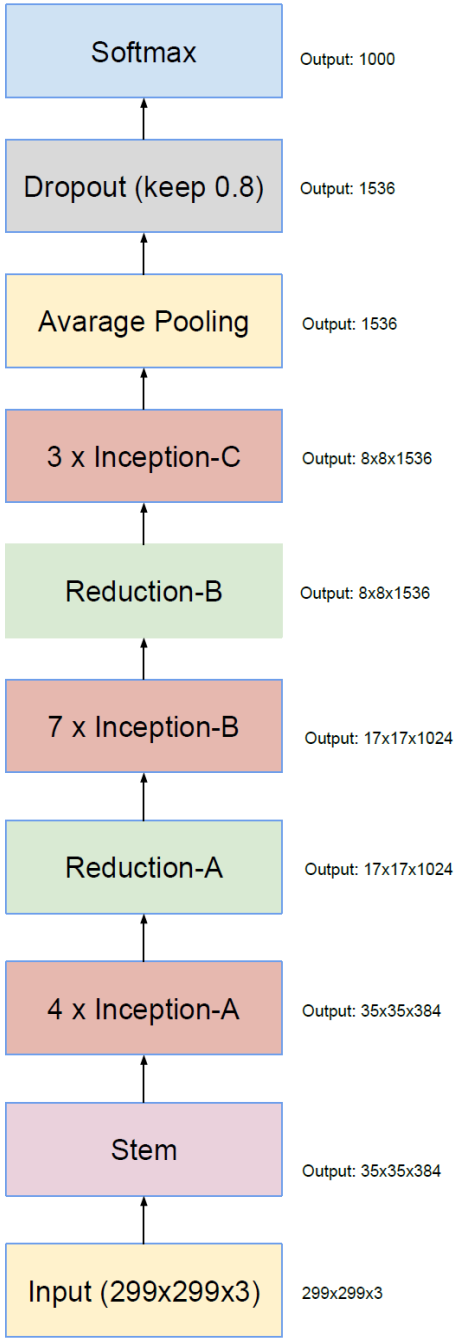
Architecture of Inception v2

type	patch size/stride or remarks	input size
conv	3×3/2	299×299×3
conv	3×3/1	149×149×32
conv padded	3×3/1	147×147×32
pool	3×3/2	147×147×64
conv	3×3/1	73×73×64
conv	3×3/2	71×71×80
conv	3×3/1	35×35×192
3×Inception	As in figure 5	35×35×288
5×Inception	As in figure 6	17×17×768
2×Inception	As in figure 7	8×8×1280
pool	8 × 8	8 × 8 × 2048
linear	logits	1 × 1 × 2048
softmax	classifier	1 × 1 × 1000

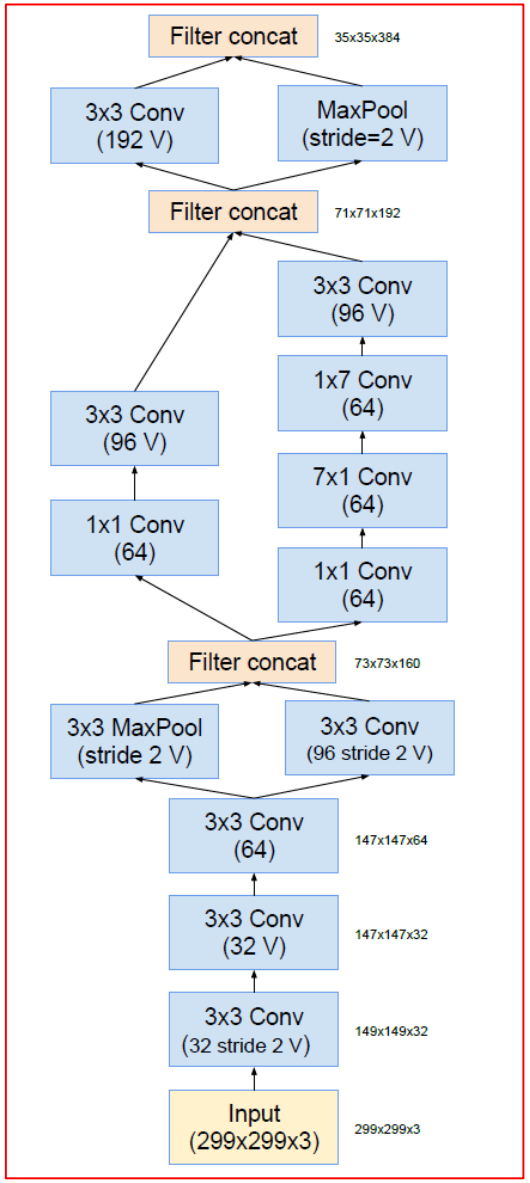
Network	Top-1 Error	Top-5 Error	Cost Bn Ops
GoogLeNet [20]	29%	9.2%	1.5
BN-GoogLeNet	26.8%	-	**1.5**
BN-Inception [7]	25.2%	7.8	2.0
Inception-v2	23.4%	-	3.8
Inception-v2 RMSProp	23.1%	6.3	3.8
Inception-v2 Label Smoothing	22.8%	6.1	3.8
Inception-v2 Factorized 7 × 7	21.6%	5.8	4.8
Inception-v2 BN-auxiliary	**21.2**%	**5.6**%	4.8

Inception v3

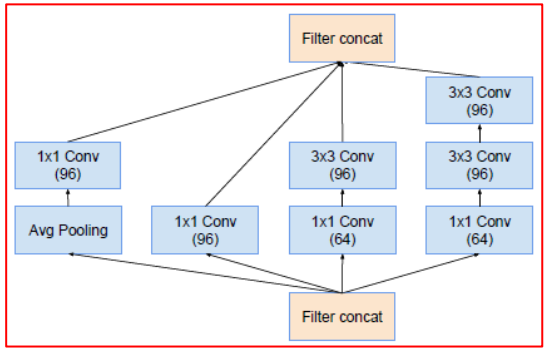
Inception v4



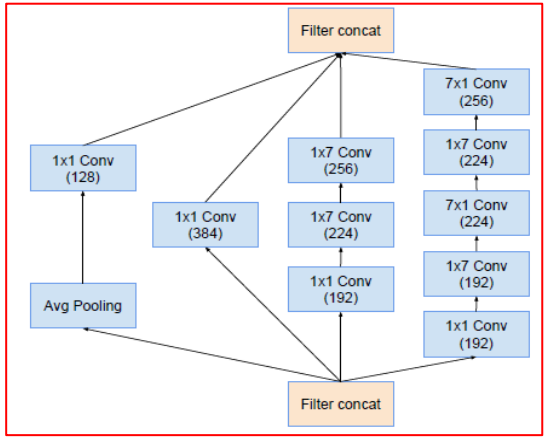
Stem



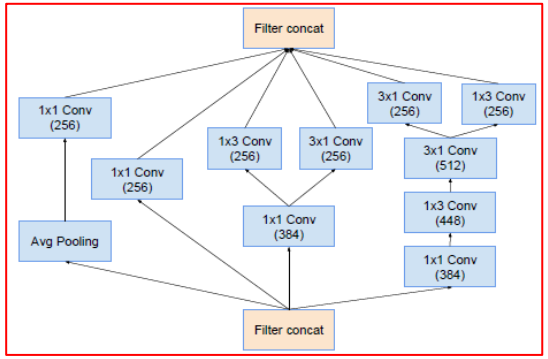
Inception-A



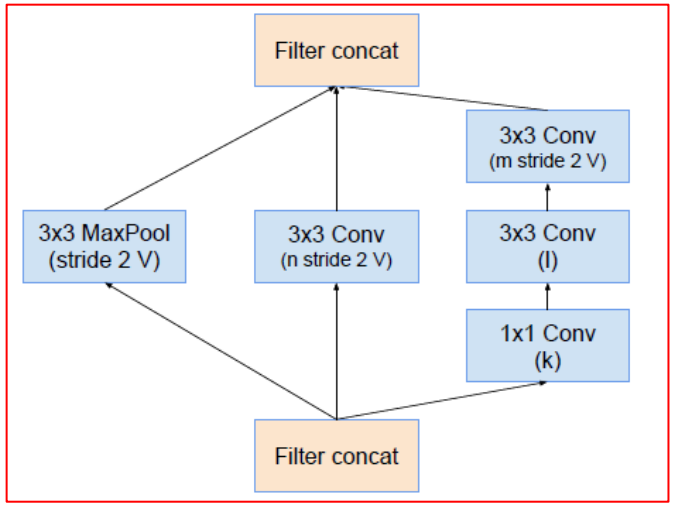
Inception-B



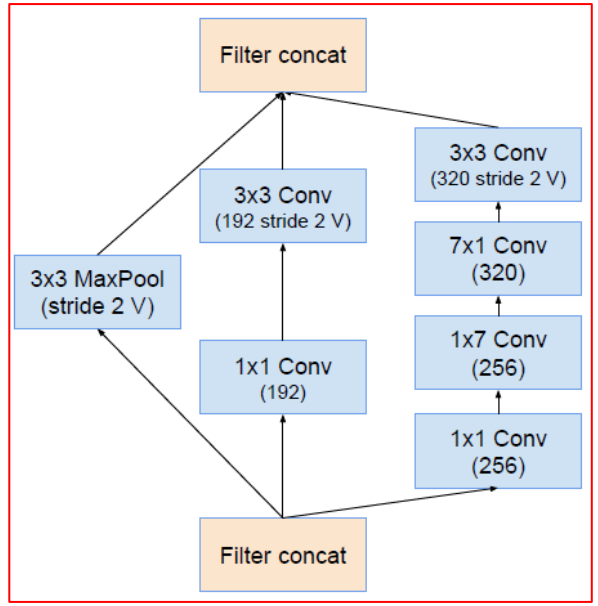
Inception-C



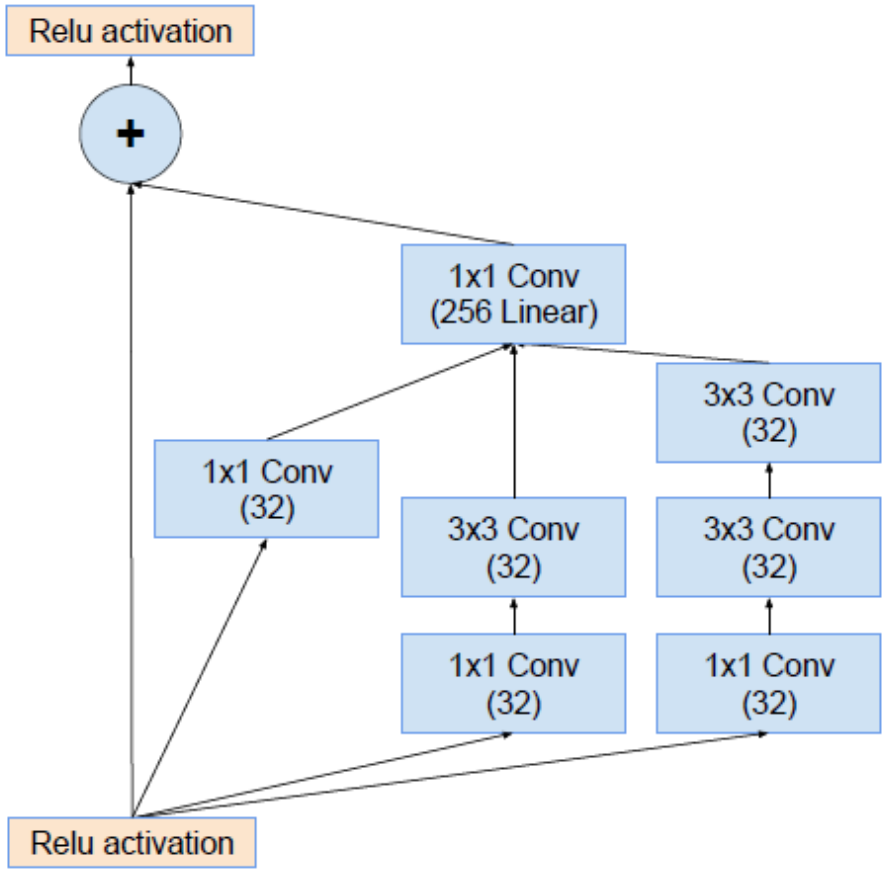
Reduction-A

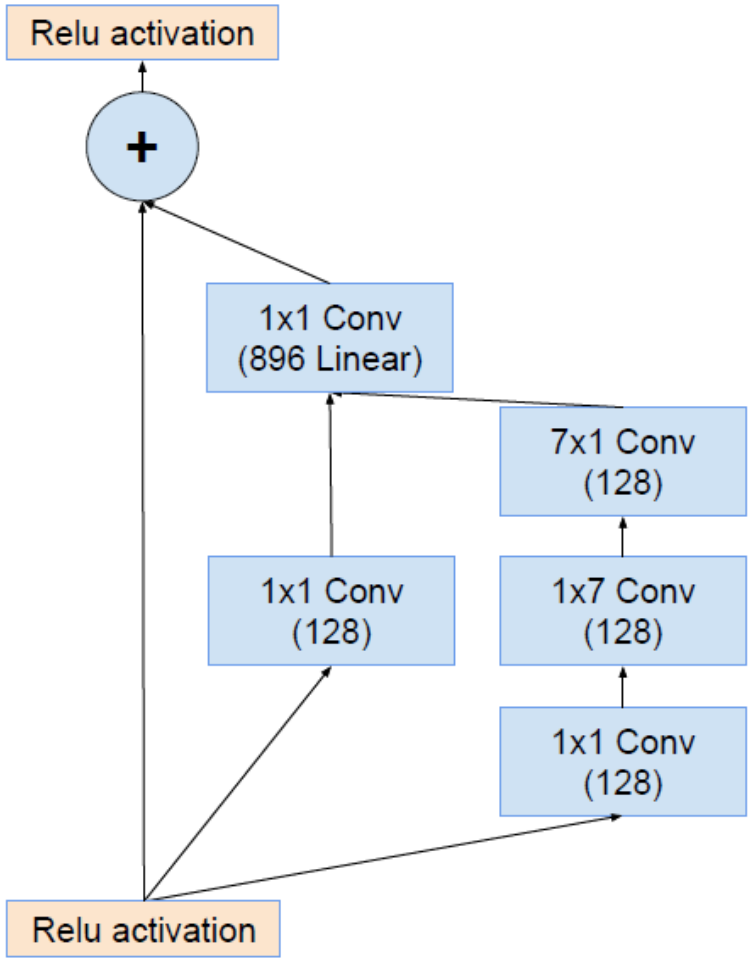


Reduction-B



Inception-ResNet





Inception-ResNet

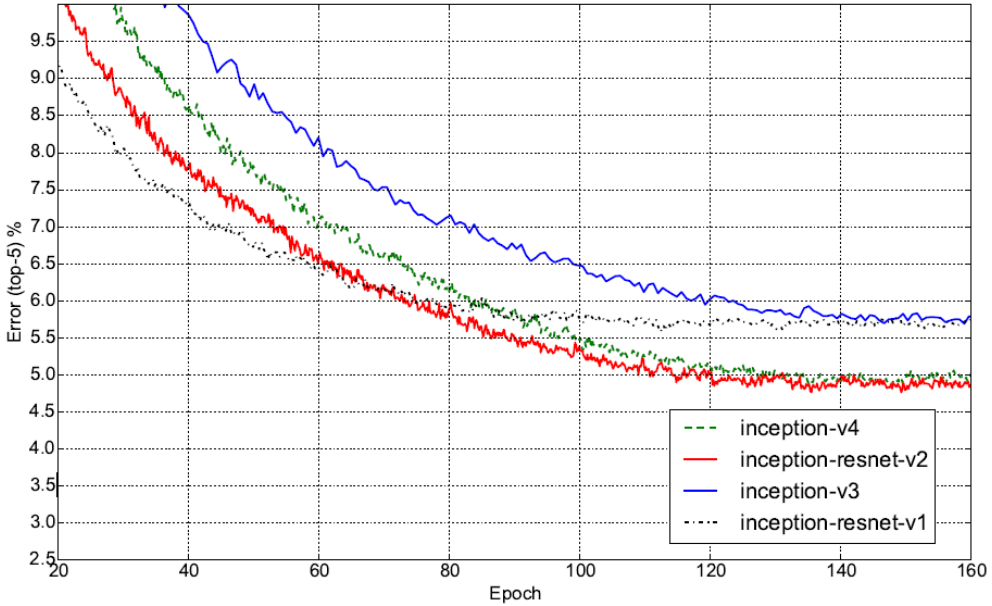


Figure 25. Top-5 error evolution of all four models (single model, single crop). Showing the improvement due to larger model size. Although the residual version converges faster, the final accuracy seems to mainly depend on the model size.

Network	Top-1 Error	Top-5 Error
BN-Inception [6]	25.2%	7.8%
Inception-v3 [15]	21.2%	5.6%
Inception-ResNet-v1	21.3%	5.5%
Inception-v4	20.0%	5.0%
Inception-ResNet-v2	19.9%	4.9%

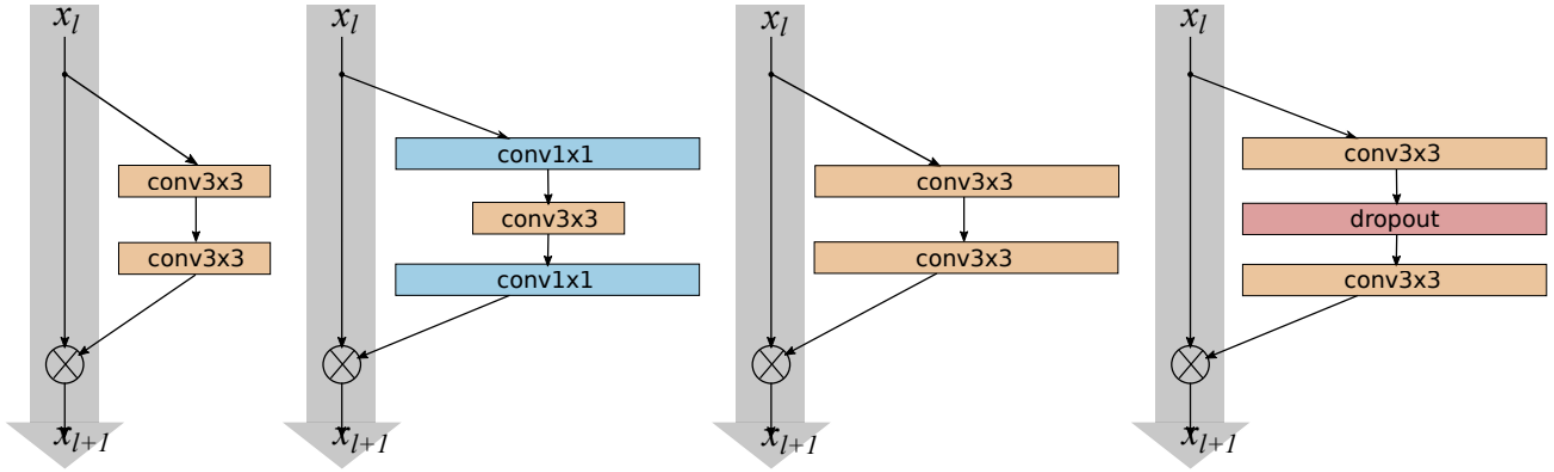
Table 2. Single crop - single model experimental results. Reported on the non-blacklisted subset of the validation set of ILSVRC 2012.

Inception-ResNet v1 ： 计算量和Inception v3相当
Inception-ResNet v2 ： 计算量和Inception v4相当

Although the residual version converges faster, the final accuracy seems to mainly depend on the model size.

Width-based CNNs

Wide ResNet (WRN)



(a) basic (b) bottleneck (c) basic-wide (d) wide-dropout

Figure 1: Various residual blocks used in the paper. Batch normalization and ReLU precede each convolution (omitted for clarity)

block type	depth	# params	time,s	CIFAR-10
$B(1,3,1)$	40	1.4M	85.8	6.06
$B(3,1)$	40	1.2M	67.5	5.78
$B(1,3)$	40	1.3M	72.2	6.42
$B(3,1,1)$	40	1.3M	82.2	5.86
$B(3,3)$	28	1.5M	67.5	5.73
$B(3,1,3)$	22	1.1M	59.9	5.78

Table 2: Test error (%, median over 5 runs) on CIFAR-10 of residual networks with $k = 2$ and different block types. Time column measures one training epoch.

l	CIFAR-10
1	6.69
2	5.43
3	5.65
4	5.93

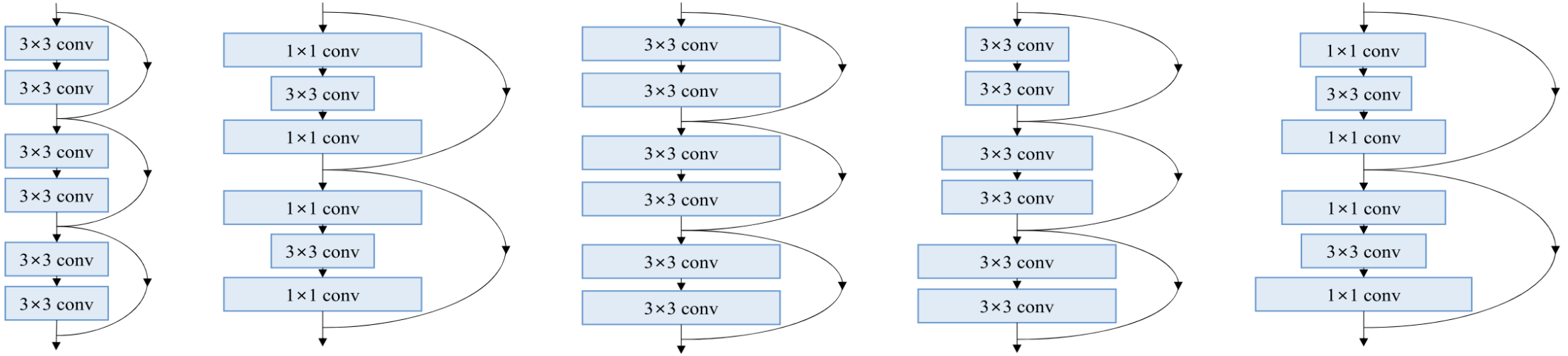
Table 3: Test error (%, median over 5 runs) on CIFAR-10 of WRN-40-2 (2.2M) with various l.

depth	k	# params	CIFAR-10	CIFAR-100
40	1	0.6M	6.85	30.89
40	2	2.2M	5.33	26.04
40	4	8.9M	4.97	22.89
40	8	35.7M	4.66	-
28	10	36.5M	**4.17**	20.50
28	12	52.5M	4.33	**20.43**
22	8	17.2M	4.38	21.22
22	10	26.8M	4.44	20.75
16	8	11.0M	4.81	22.07
16	10	17.1M	4.56	21.59

Table 4: Test error (%) of various wide networks on CIFAR-10 and CIFAR-100 (ZCA pre-processing).

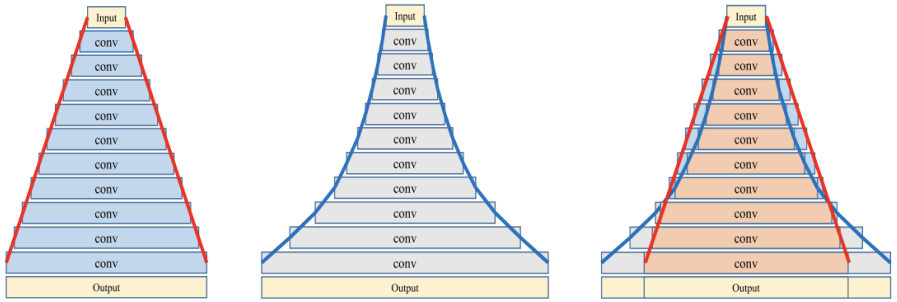
	depth-k	# params	CIFAR-10	CIFAR-100
NIN [20]			8.81	35.67
DSN [19]			8.22	34.57
FitNet [24]			8.39	35.04
Highway [28]			7.72	32.39
ELU [5]			6.55	24.28
original-ResNet[11]	110	1.7M	6.43	25.16
	1202	10.2M	7.93	27.82
stoc-depth[14]	110	1.7M	5.23	24.58
	1202	10.2M	4.91	-
pre-act-ResNet[13]	110	1.7M	6.37	-
	164	1.7M	5.46	24.33
	1001	10.2M	4.92(4.64)	22.71
WRN (ours)	40-4	8.9M	4.53	21.18
	16-8	11.0M	4.27	20.43
	28-10	36.5M	**4.00**	**19.25**

Deep Pyramidal Residual Networks



(a) basic (b) bottleneck (c) wide (d) pyramidal (e) pyramidal bottleneck

Figure 1. Schematic illustration of (a) basic residual units [7], (b) bottleneck residual units [7], (c) wide residual units [34], (d) our pyramidal residual units, and (e) our pyramidal bottleneck residual units.

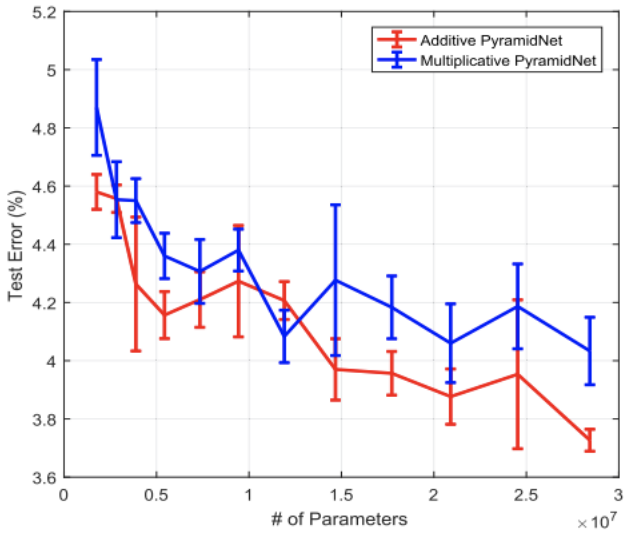


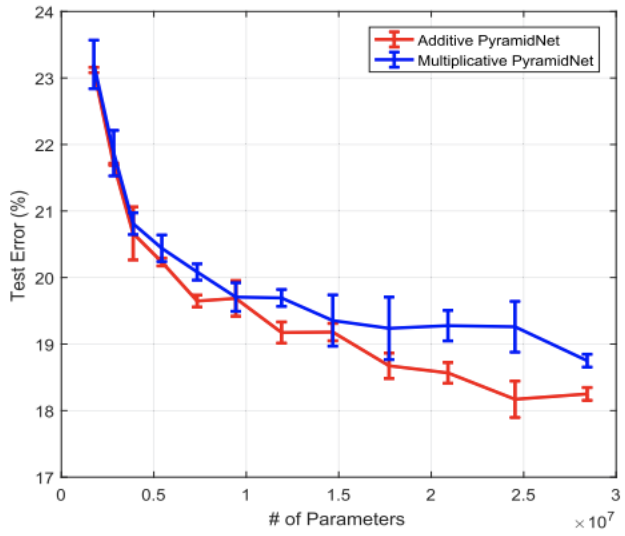
(a) (b) (c)

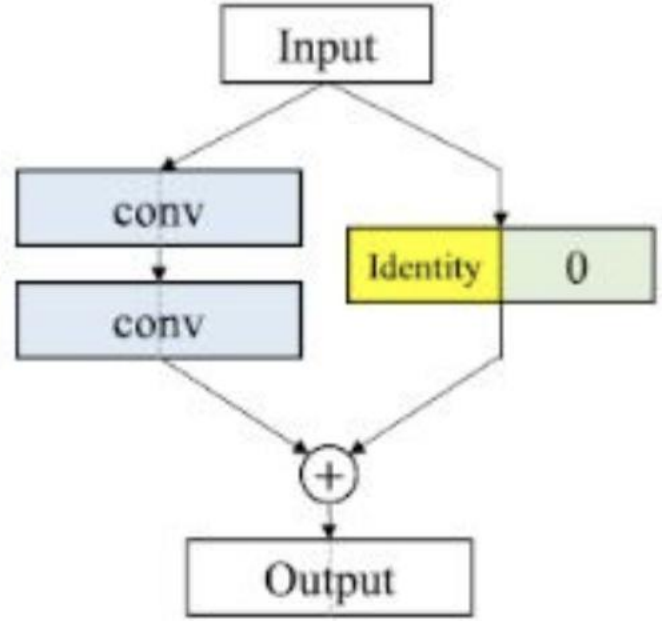
Figure 2. Visual illustrations of (a) additive PyramidNet, (b) multiplicative PyramidNet, and (c) a comparison of (a) and (b).

$$D_k = \begin{cases} 16, & \text{if } k = 1, \\ \lfloor D_{k-1} + \alpha/N \rfloor, & \text{if } 2 \leq k \leq N+1, \end{cases} \tag{2}$$

$$D_k = \begin{cases} 16, & \text{if } k = 1, \\ \lfloor D_{k-1} \cdot \alpha^{\frac{1}{N}} \rfloor, & \text{if } 2 \leq k \leq N+1. \end{cases} \tag{3}$$







Network	# of Params	Output Feat. Dim.	Depth	CIFAR-10	CIFAR-100
NiN [18]	-	-	-	8.81	35.68
All-CNN [27]	-	-	-	7.25	33.71
DSN [17]	-	-	-	7.97	34.57
FitNet [21]	-	-	-	8.39	35.04
Highway [29]	-	-	-	7.72	32.39
Fractional Max-pooling [4]	-	-	-	4.50	27.62
ELU [29]	-	-	-	6.55	24.28
ResNet [7]	1.7M	64	110	6.43	25.16
ResNet [7]	10.2M	64	1001	-	27.82
ResNet [7]	19.4M	64	1202	7.93	-
Pre-activation ResNet [8]	1.7M	64	164	5.46	24.33
Pre-activation ResNet [8]	10.2M	64	1001	4.62	22.71
Stochastic Depth [10]	1.7M	64	110	5.23	24.58
Stochastic Depth [10]	10.2M	64	1202	4.91	-
FractalNet [14]	38.6M	1,024	21	4.60	23.73
SwapOut v2 (width×4) [26]	7.4M	256	32	4.76	22.72
Wide ResNet (width×4) [34]	8.7M	256	40	4.97	22.89
Wide ResNet (width×10) [34]	36.5M	640	28	4.17	20.50
Weighted ResNet [24]	19.1M	64	1192	5.10	-
DenseNet [9]	27.2M	2,320	100	3.74	19.25
PyramidNet ($\alpha = 48$)	1.7M	64	110	4.58±0.06	23.12±0.04
PyramidNet ($\alpha = 84$)	3.8M	100	110	4.26±0.23	20.66±0.40
PyramidNet ($\alpha = 270$)	28.3M	286	110	**3.73**±0.04	**18.25**±0.10
PyramidNet (bottleneck, $\alpha = 270$)	27.0M	1,144	164	**3.48**±0.20	**17.01**±0.39

Table 4. Top-1 error rates (%) on CIFAR datasets. All the results of PyramidNets are produced with additive PyramidNets, and α denotes the widening factor. "Output Feat. Dim." denotes the feature dimension of just before the last softmax classifier.

Xception

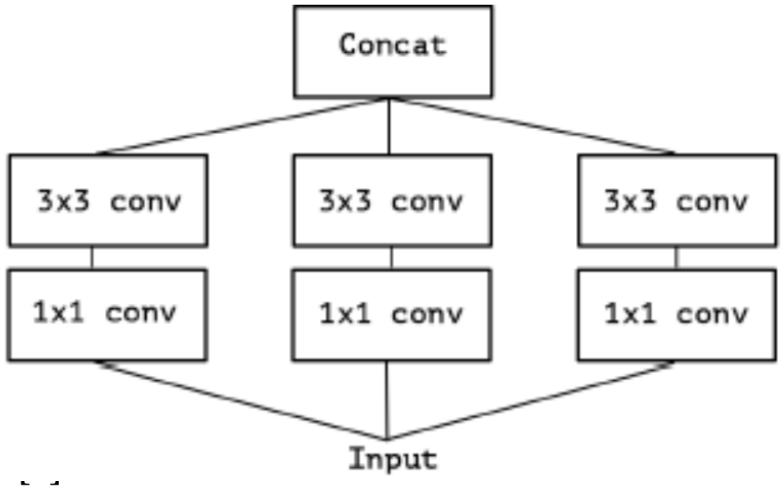
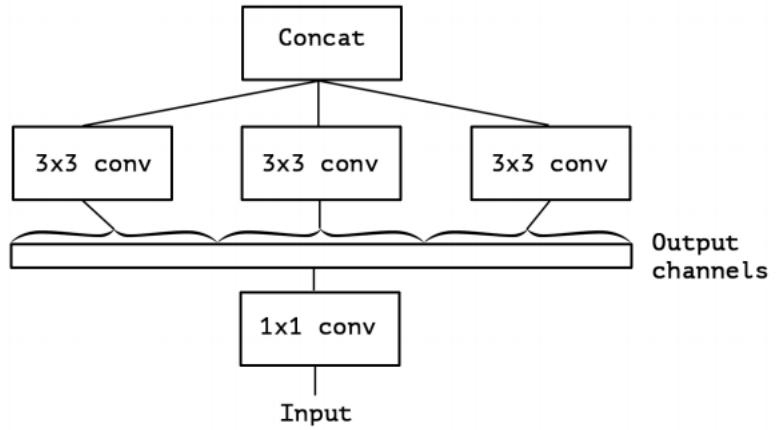


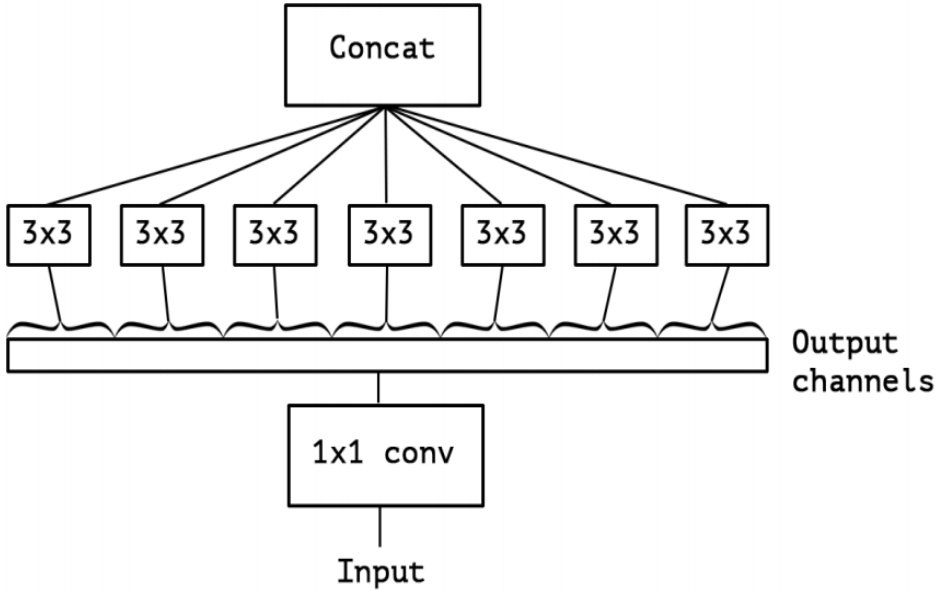
Table 1. Classification performance comparison on ImageNet (single crop, single model). VGG-16 and ResNet-152 numbers are only included as a reminder. The version of Inception V3 being benchmarked does not include the auxiliary tower.

	Top-1 accuracy	Top-5 accuracy
VGG-16	0.715	0.901
ResNet-152	0.770	0.933
Inception V3	0.782	0.941
Xception	**0.790**	**0.945**

Table 2. Classification performance comparison on JFT (single crop, single model).

	FastEval14k MAP@100
Inception V3 - no FC layers	6.36
Xception - no FC layers	6.70
Inception V3 with FC layers	6.50
Xception with FC layers	**6.78**





ResneXt (Aggregated Residual Transform Network)

1. split-transform-merge (between inception and resnet)

$$\sum_{i=1}^{D} w_i x_i, \tag{1}$$

$$\mathcal{F}(\mathbf{x}) = \sum_{i=1}^{C} \mathcal{T}_i(\mathbf{x}), \tag{2}$$

$$\mathbf{y} = \mathbf{x} + \sum_{i=1}^{C} \mathcal{T}_i(\mathbf{x}), \tag{3}$$

C is the size of the set of transformations to be aggregated. We refer to C as **cardinality**

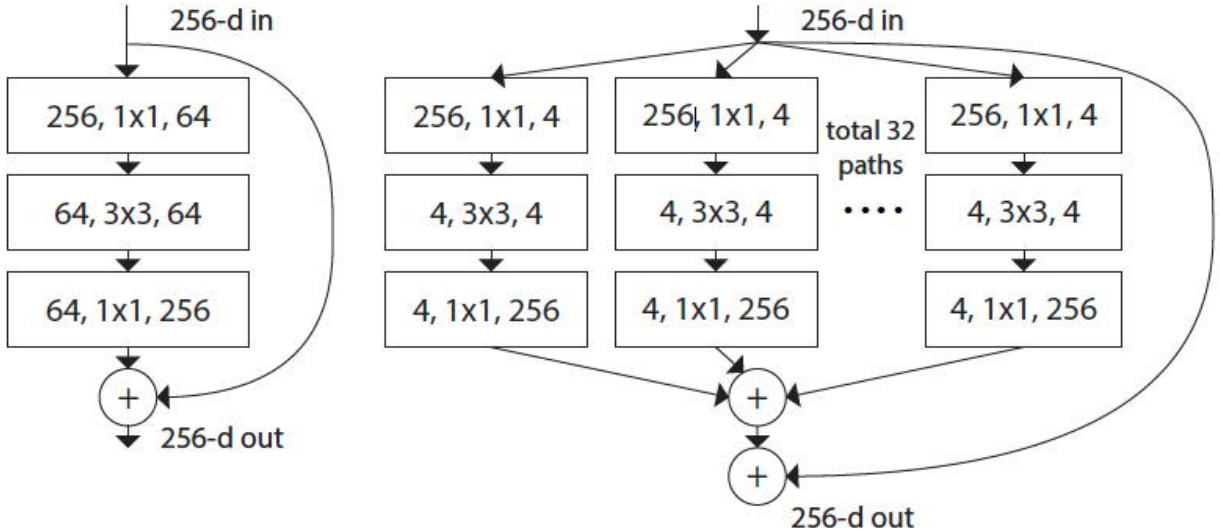


Figure 1. **Left**: A block of ResNet [14]. **Right**: A block of ResNeXt with cardinality = 32, with roughly the same complexity. A layer is shown as (# in channels, filter size, # out channels).

2. Grouped convolutions

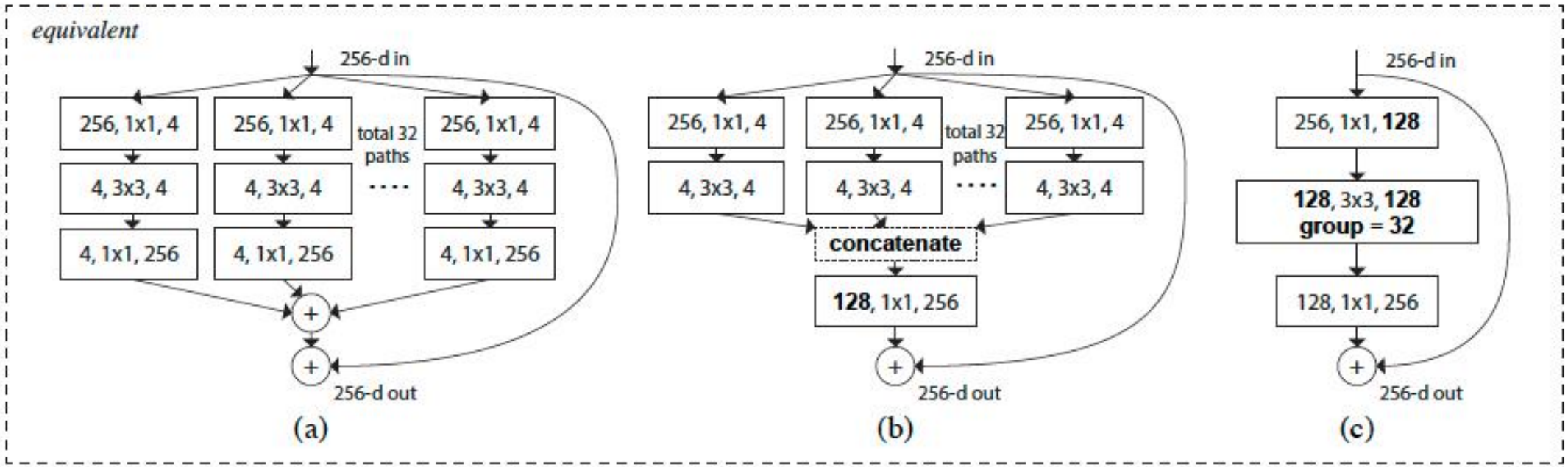


Figure 3. Equivalent building blocks of ResNeXt. (**a**): Aggregated residual transformations, the same as Fig. 1 right. (**b**): A block equivalent to (a), implemented as early concatenation. (**c**): A block equivalent to (a,b), implemented as grouped convolutions [24]. Notations in **bold** text highlight the reformulation changes. A layer is denoted as (# input channels, filter size, # output channels).

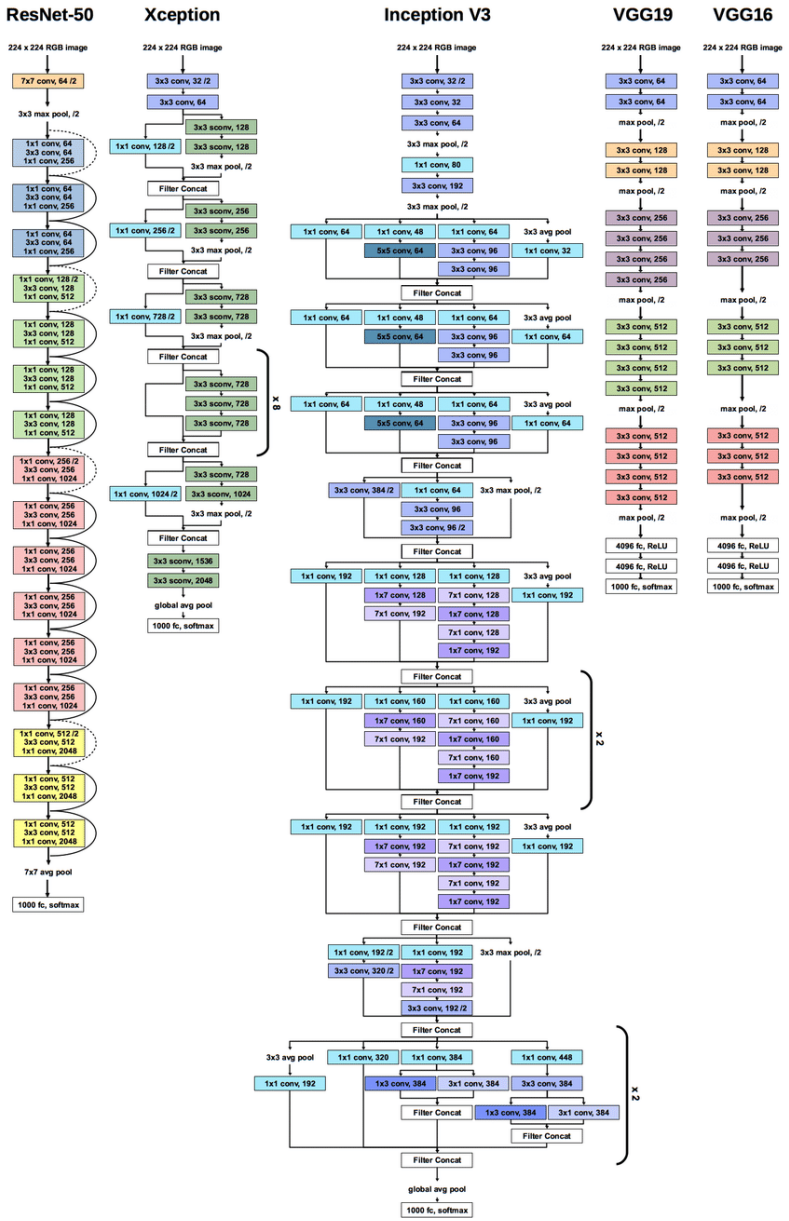
stage	output	ResNet-50	**ResNeXt-50 (32×4d)**
conv1	112×112	7×7, 64, stride 2	7×7, 64, stride 2
conv2	56×56	3×3 max pool, stride 2	3×3 max pool, stride 2
		$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128, C{=}32 \\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

Table 1. (**Left**) ResNet-50. (**Right**) ResNeXt-50 with a 32×4d template (using the reformulation in Fig. 3(c)). Inside the brackets are the shape of a residual block, and outside the brackets is the number of stacked blocks on a stage. "C=32" suggests grouped convolutions [24] with 32 groups. *The numbers of parameters and FLOPs are similar between these two models.*

Inception family

Input→Stem→A→ReducitonA→B→ReductionB→C→Avg Pooling(+Linear)→feature

Inception-V1（GoogLeNet）、
BN-Inception、
Inception-V2、
Inception-V3、
Inception-ResNet-V1、
Inception-V4、
Inception-ResNet-V2



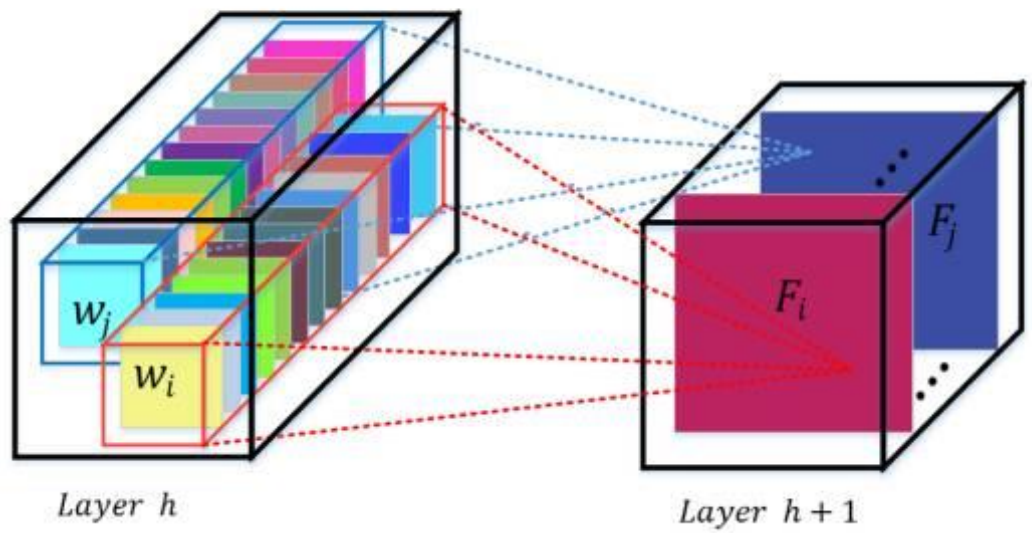
Feature-Map (ChannelFMap) Exploitation based CNNs

Some of the feature-maps impart little or no role in object discrimination.

Convolution

A convolutional filer is expected to be an informative combination

- Fusing **channel-wise** and **spatial** information
- Within local receptive fields

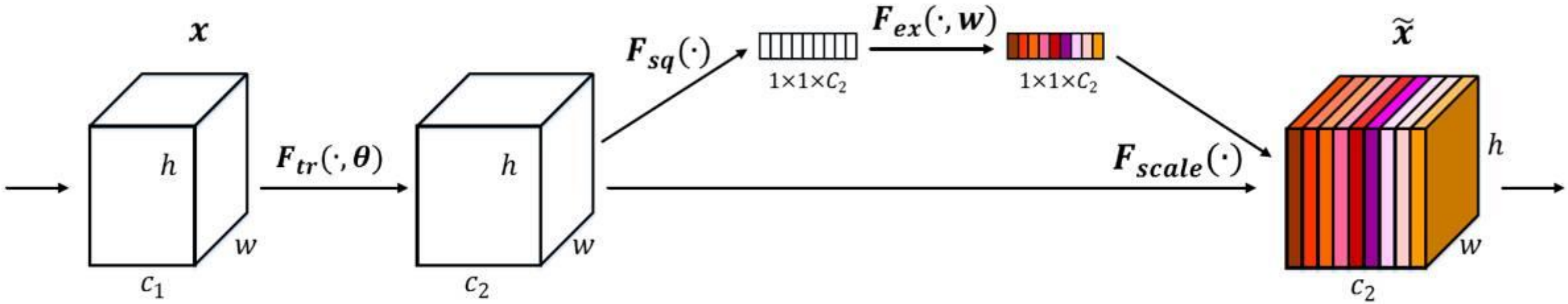


Squeeze-and-Excitation (SE) Networks

- If a network can be enhanced from the aspect of **channel relationship**?

- **Motivation:**
  - Explicitly model channel-interdependencies within modules
  - Feature recalibration
    - Selectively enhance useful features and suppress less useful ones

Squeeze-and-Excitation Module



Squeeze

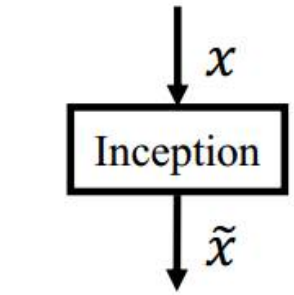
- Shrinking feature maps $\in \mathbb{R}^{w\times h\times c_2}$ through spatial dimensions $(w \times h)$
- Global distribution of channel-wise responses

Excitation

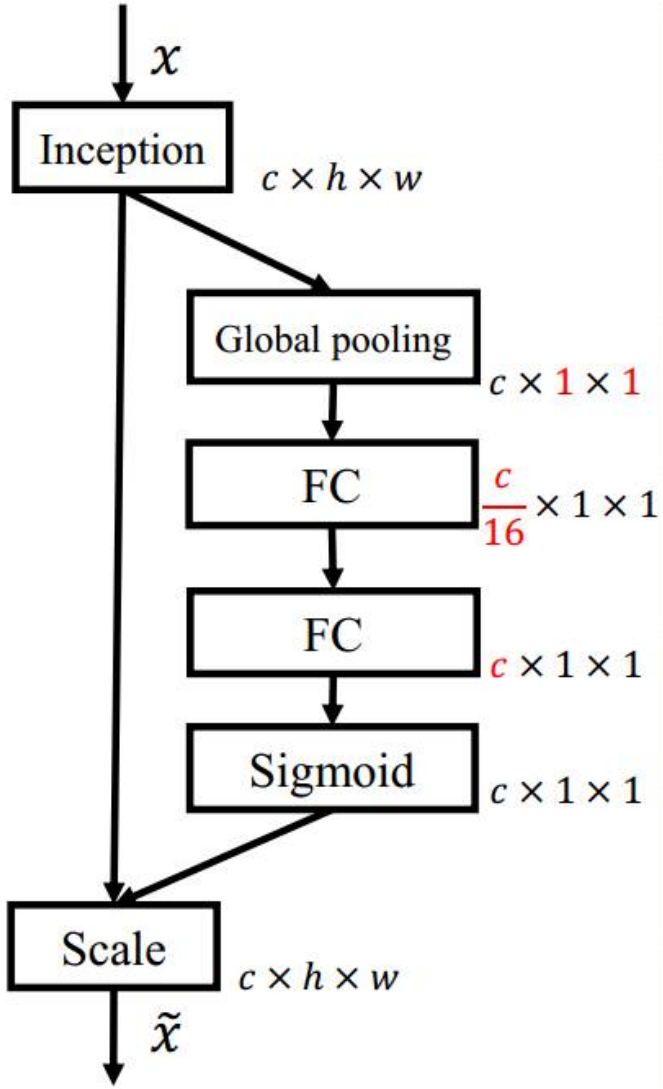
- Learning $W \in \mathbb{R}^{c_2\times c_2}$ to explicitly model channel-association
- Gating mechanism to produce channel-wise weights

Scale

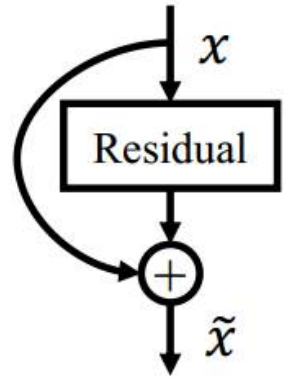
- Reweighting the feature maps $\in \mathbb{R}^{w\times h\times c_2}$



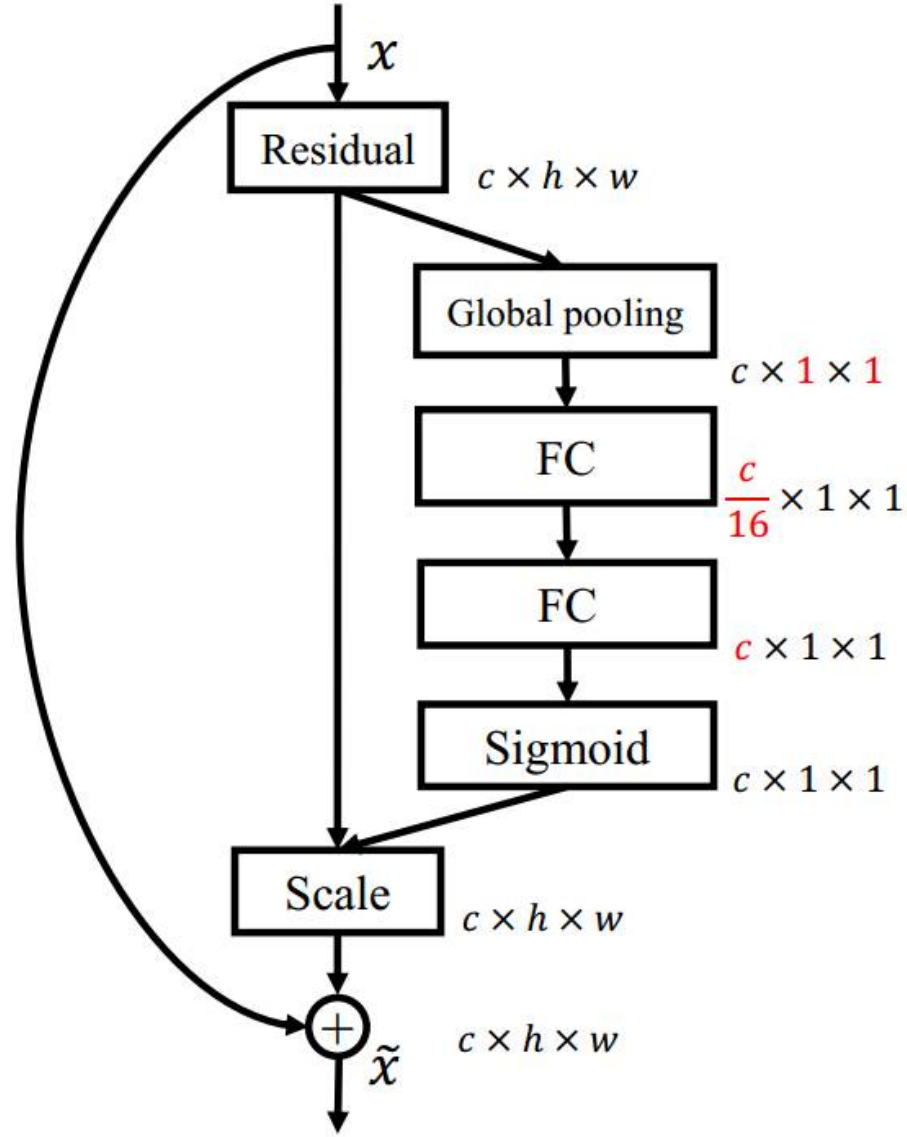
Inception Module



SE-Inception Module



ResNet Module



SE-ResNet Module

Model and Computational Complexity

SE-ResNet-50 vs. ResNet-50

- Parameters: 2%~10% additional parameters
- Computation cost: <1% additional computation (theoretical)
- GPU inference time: 10% additional time
- CPU inference time: <2% additional time

Benefits against Network Depth

	Original		Our re-implementation		SE-module	
	top-1 err.	top-5 err.	top-1 err.	top-5 err.	top-1 err.	top-5 err.
ResNet-50 [1]	24.7	7.8	24.80	7.48	**23.29**$_{(1.51)}$	**6.62**$_{(0.86)}$
ResNet-101 [1]	23.6	7.1	23.17	6.52	**22.38**$_{(0.79)}$	**6.07**$_{(0.45)}$
ResNet-152 [1]	23.0	6.7	22.42	6.34	**21.57**$_{(0.85)}$	**5.73**$_{(0.61)}$

Table 1. Error rates (%) of single-crop results on the ImageNet-1k validation set.

Competitive Inner-Imaging Squeeze and Excitation for Residual Network (CMPESE Network)

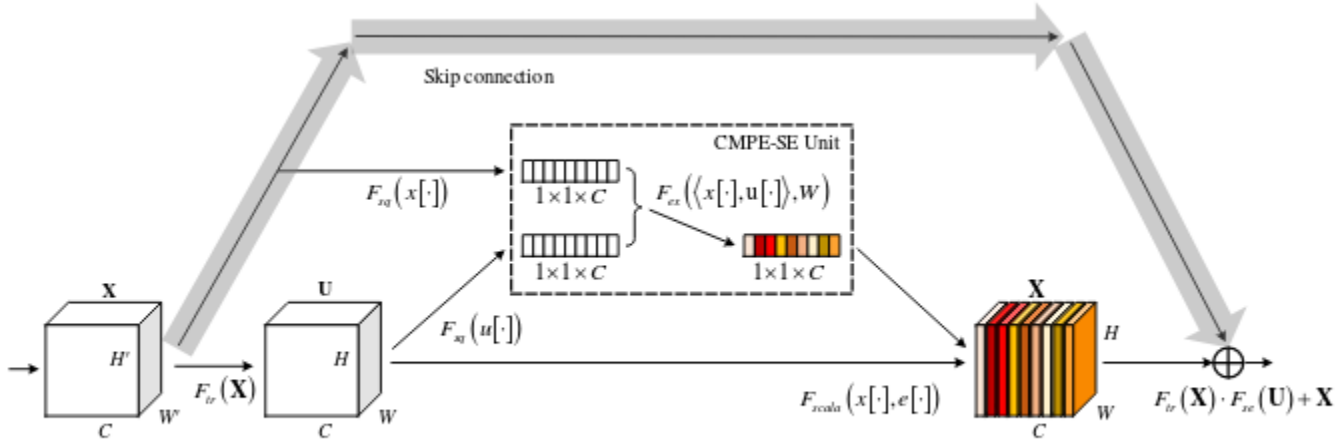
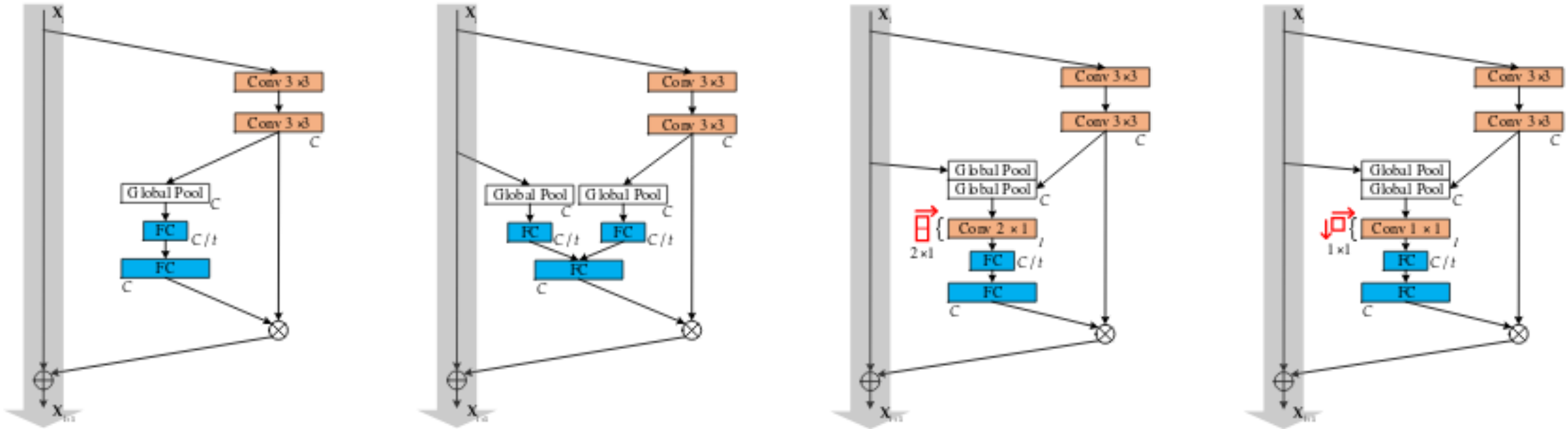


Figure 1: Competitive Squeeze-Excitation Architecture for Residual block.



(a) SE-ResNet module

(b) CMPE-SE-ResNet module (with double FC squeezes)

(c) CMPE-SE-ResNet module (with conv 2×1 pair-view)

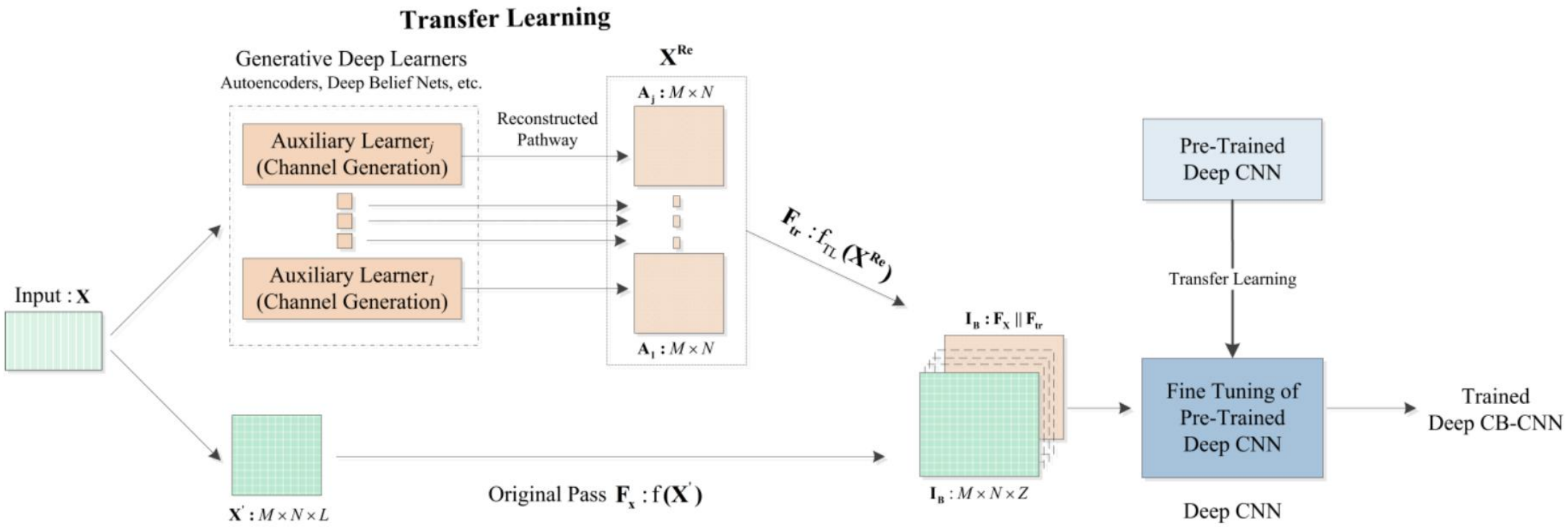
(d) CMPE-SE-ResNet module (with conv 1×1 pair-view)

Figure 2: Difference of SE-ResNet module and CMPE-SE-ResNet modules. (a) Typical SE Residual block, the orange rectangles represent the convolution and blue indicates fully connected layer, the white one is global average pooling. (b) CMPE-SE residual block of the version with double fully connected embedding for squeezed signals, which are merged in the excitation layer. (c) CMPE-SE residual block with 2×1 convolutional pair-view, after stacking the squeezed signals, the red pane and arrow indicate the size and the scanning direction of concolution. (d) CMPE-SE residual block with 1×1 convolutional pair-view.

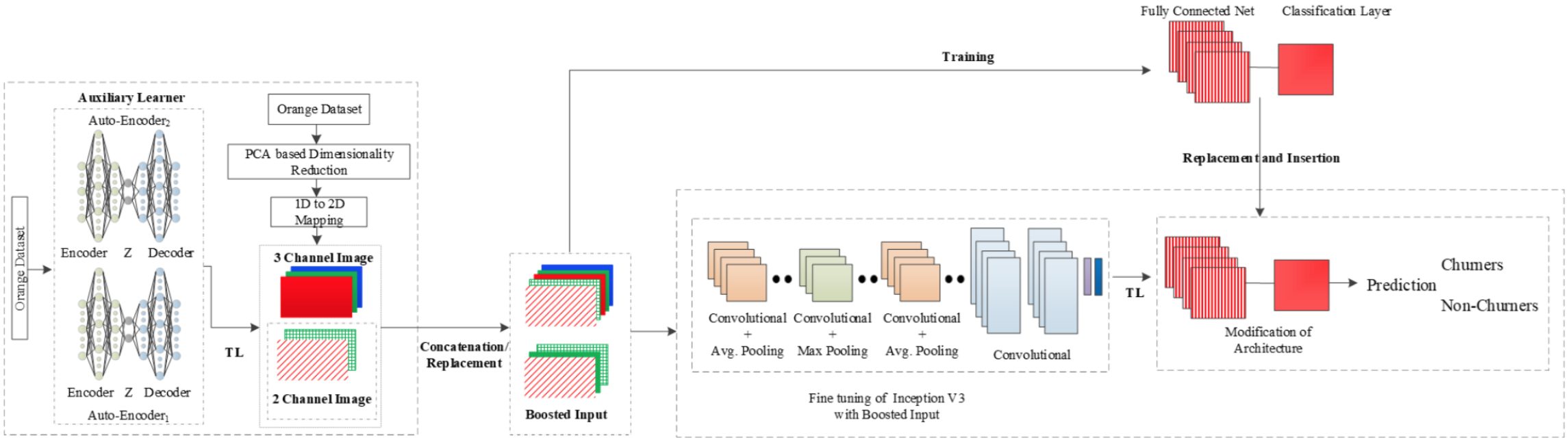
Channel Boosted Convolutional Neural Network using Transfer Learning

Qiushi Lin

Channel Boosted deep CNN (CB-CNN)



Details of the working of CB-CNN



Training of AE

- Sparsity constraint for AE is defined as

$$\gamma_n = \gamma$$

$$\gamma_n = \frac{1}{k}\sum_{m=1}^{k}\left[\alpha_n(\mathbf{x}(m))\right]$$

- Unsupervised learning loss

$$\Omega_{Sparsity} = \sum_{n=1}^{z} KL(\gamma \,\|\, \gamma_n)$$

Fine-tuning using boosted input

- fine-tuning of the pre-trained Inception-V3 is performed
- the final layer (fully connected net) of the network is trained from scratch
- the whole network is fine-tuned using the boosted input

Advantage of deploying TL

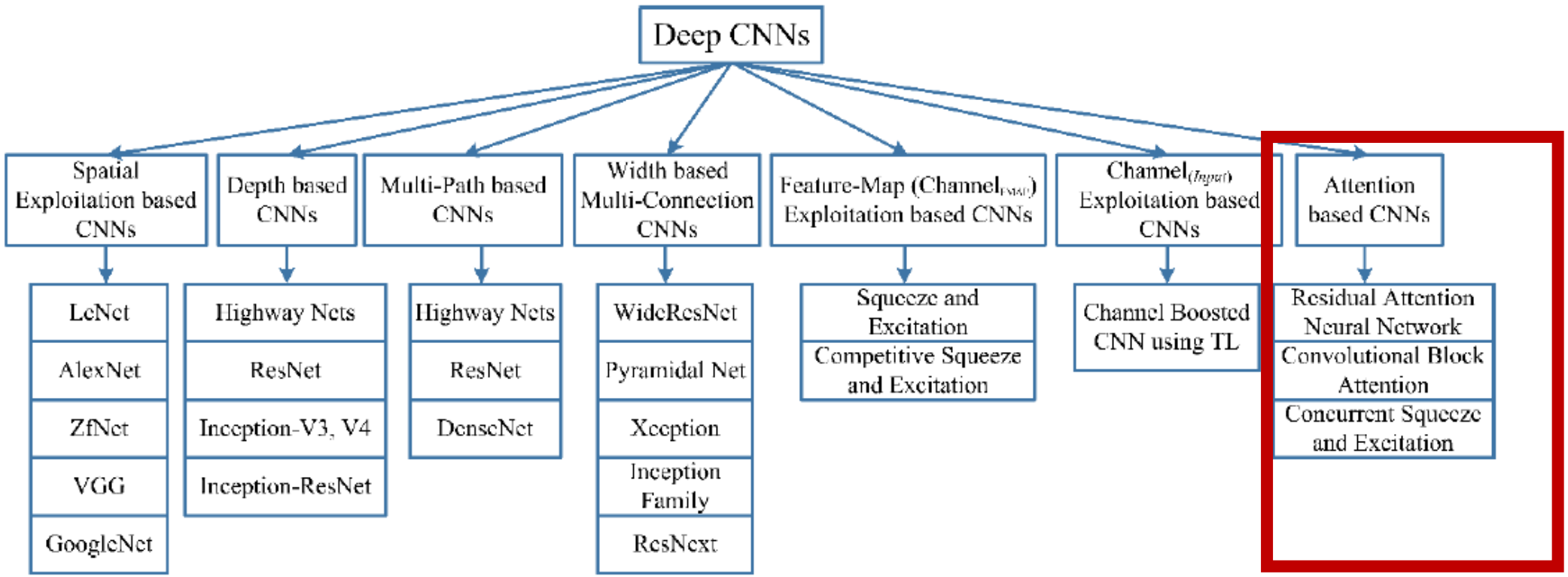
- the transferred feature maps from the pre-trained CNN reduce the training cost and help in improving the generalization

- augmentation of the auxiliary channels available through TL from the already trained Deep NN (generative model) with original feature maps improves the classifier's representation capacity for complex classification problems.

Attention based CNN

Speaker: Zhu Liu

Outline


Deep CNNs
Spatial Exploitation based CNNs
Depth based CNNs
Multi-Path based CNNs
Width based Multi-Connection CNNs
Feature-Map (Channel$_{(Map)}$) Exploitation based CNNs
Channel$_{(Input)}$ Exploitation based CNNs
Attention based CNNs
LeNet
AlexNet
ZfNet
VGG
GoogleNet
Highway Nets
ResNet
Inception-V3, V4
Inception-ResNet
Highway Nets
ResNet
DenseNet
WideResNet
Pyramidal Net
Xception
Inception Family
ResNext
Squeeze and Excitation
Competitive Squeeze and Excitation
Channel Boosted CNN using TL
Residual Attention Neural Network
Convolutional Block Attention
Concurrent Squeeze and Excitation

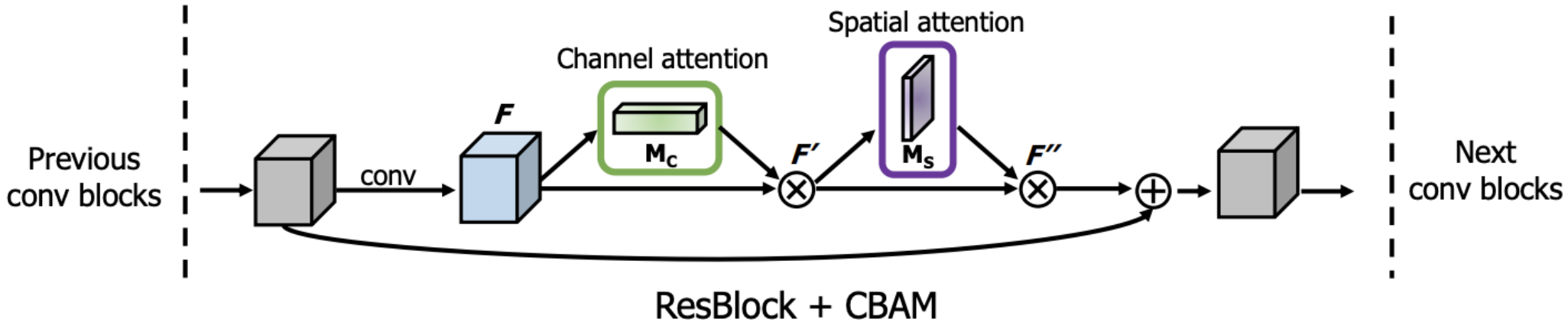
Outline

- CBAM: Convolutional Block Attention Module (ECCV 2018; 992)
- Concurrent Spatial and Channel ‘Squeeze & Excitation’ in Fully Convolutional Networks (MICCAI 2018; 134)
- Residual Attention Network for Image Classification (CVPR 2018; 1142)

Attention? Attention!

- WHAT to attend?
- HOW to get the attention weight map (importance distribution)?
- WHERE to put the attention module on the original network?

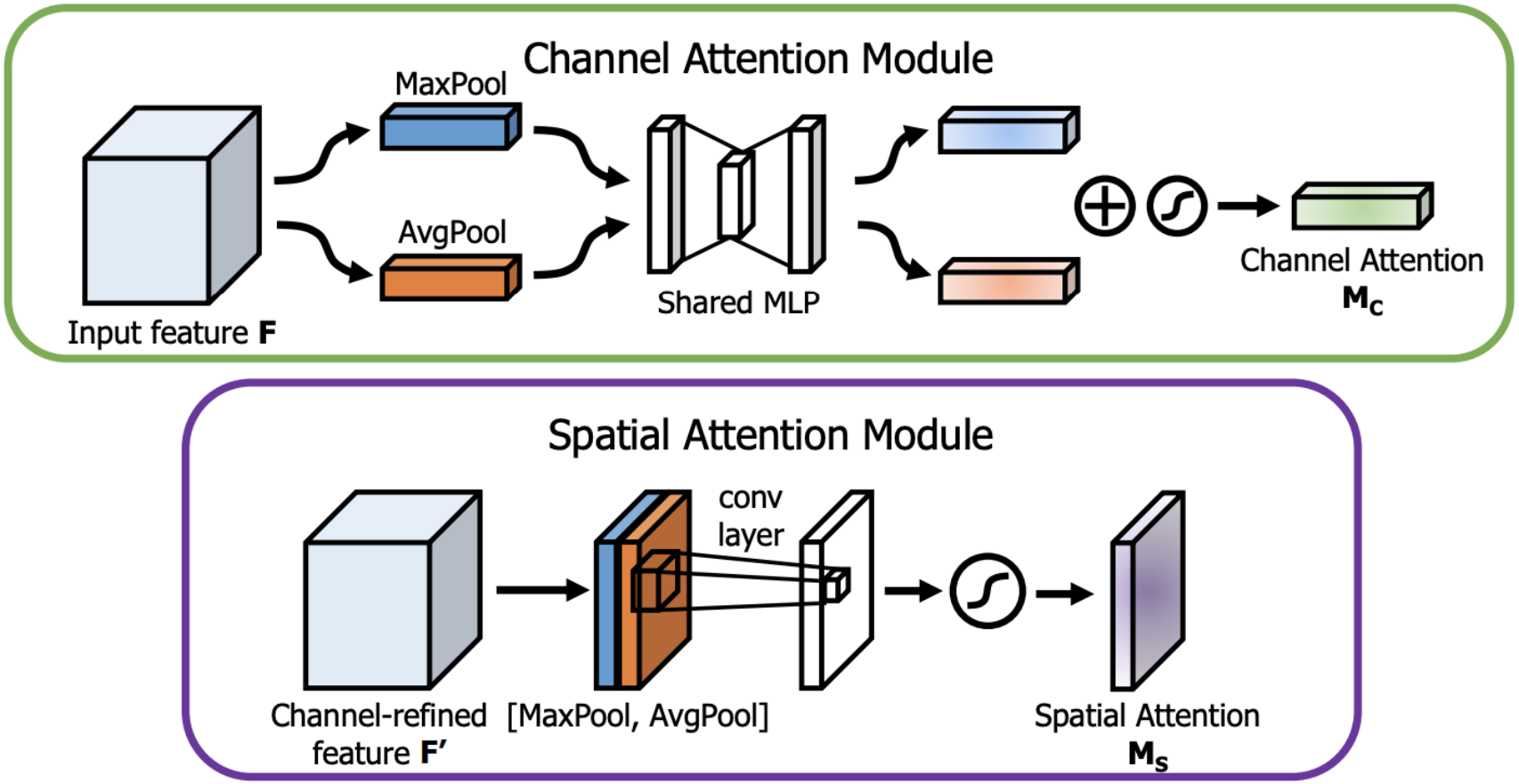
CBAM



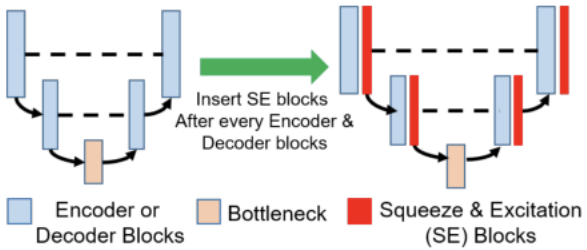
ResBlock + CBAM

$$\mathbf{F}' = \mathbf{M_c}(\mathbf{F}) \otimes \mathbf{F},$$
$$\mathbf{F}'' = \mathbf{M_s}(\mathbf{F}') \otimes \mathbf{F}',$$

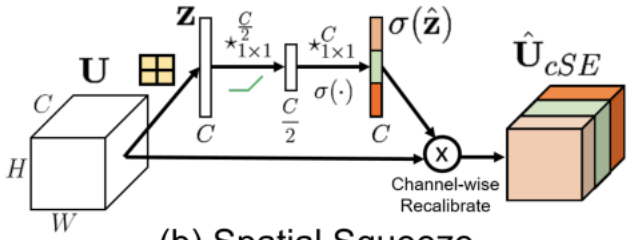
CBAM


Channel Attention Module
MaxPool
AvgPool
Input feature F
Shared MLP
Channel Attention
M_C
Spatial Attention Module
conv
layer
Channel-refined
feature F'
[MaxPool, AvgPool]
Spatial Attention
M_S

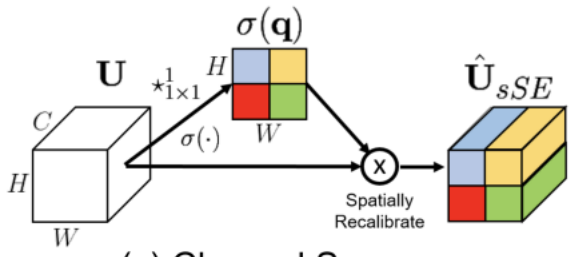
SE-sc



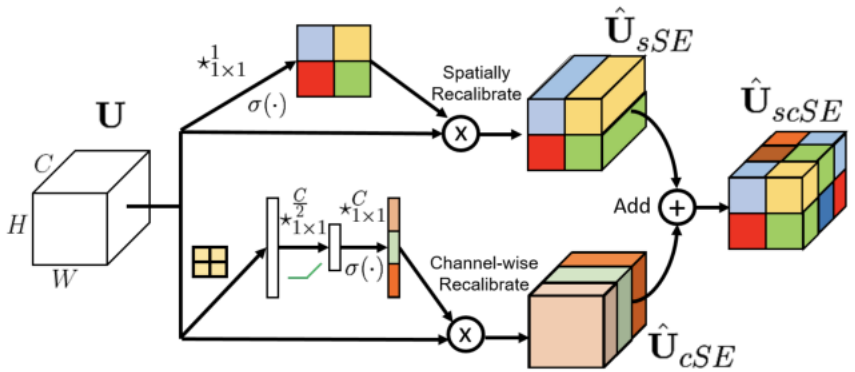
(a) SE block in F-CNN



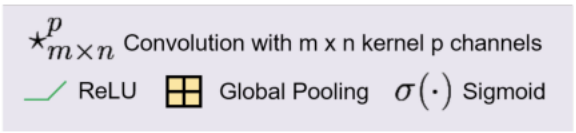
(b) Spatial Squeeze
and Channel Excitation (cSE)



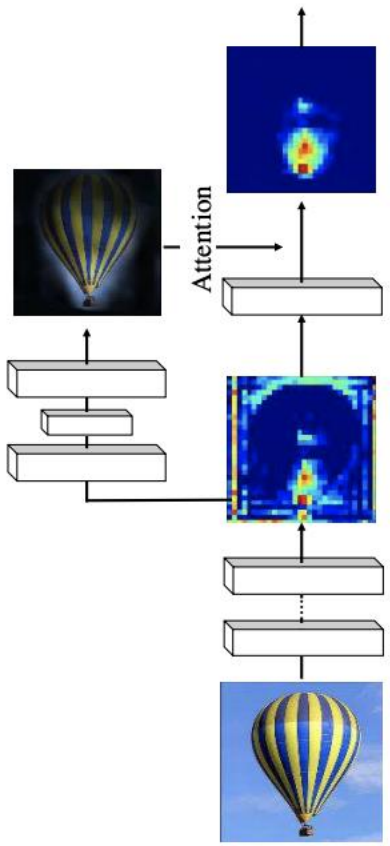
(c) Channel Squeeze
and Spatial Excitation (sSE)



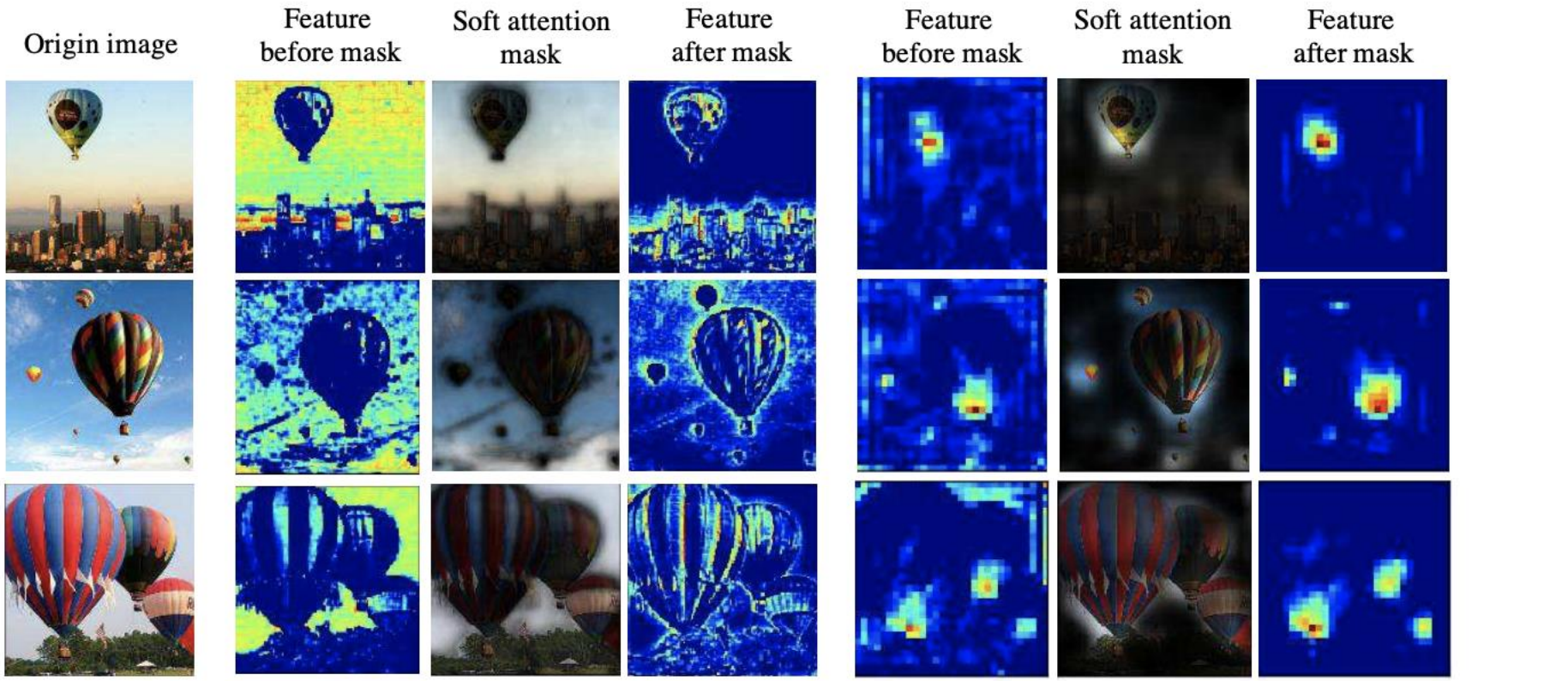
(d) Concurrent Spatial and Channel
Squeeze and Channel Excitation (scSE)



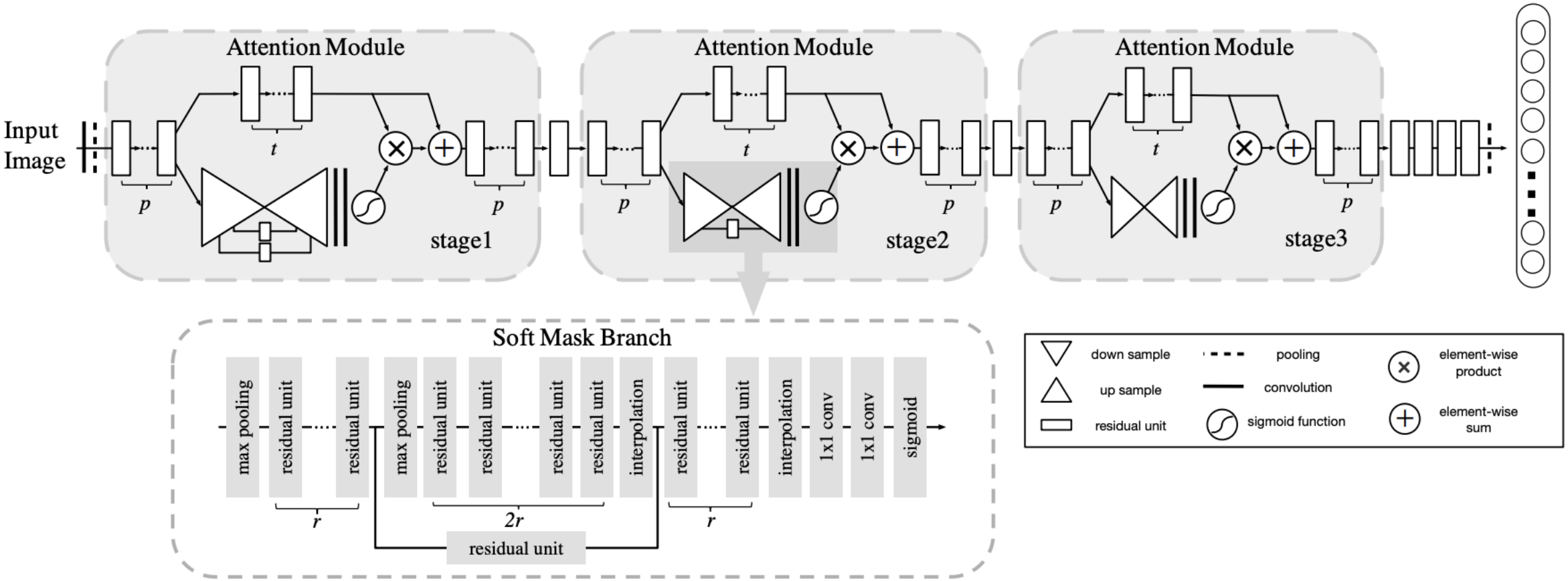
Residual Attention Module



Attention mechanism



Residual Attention Module

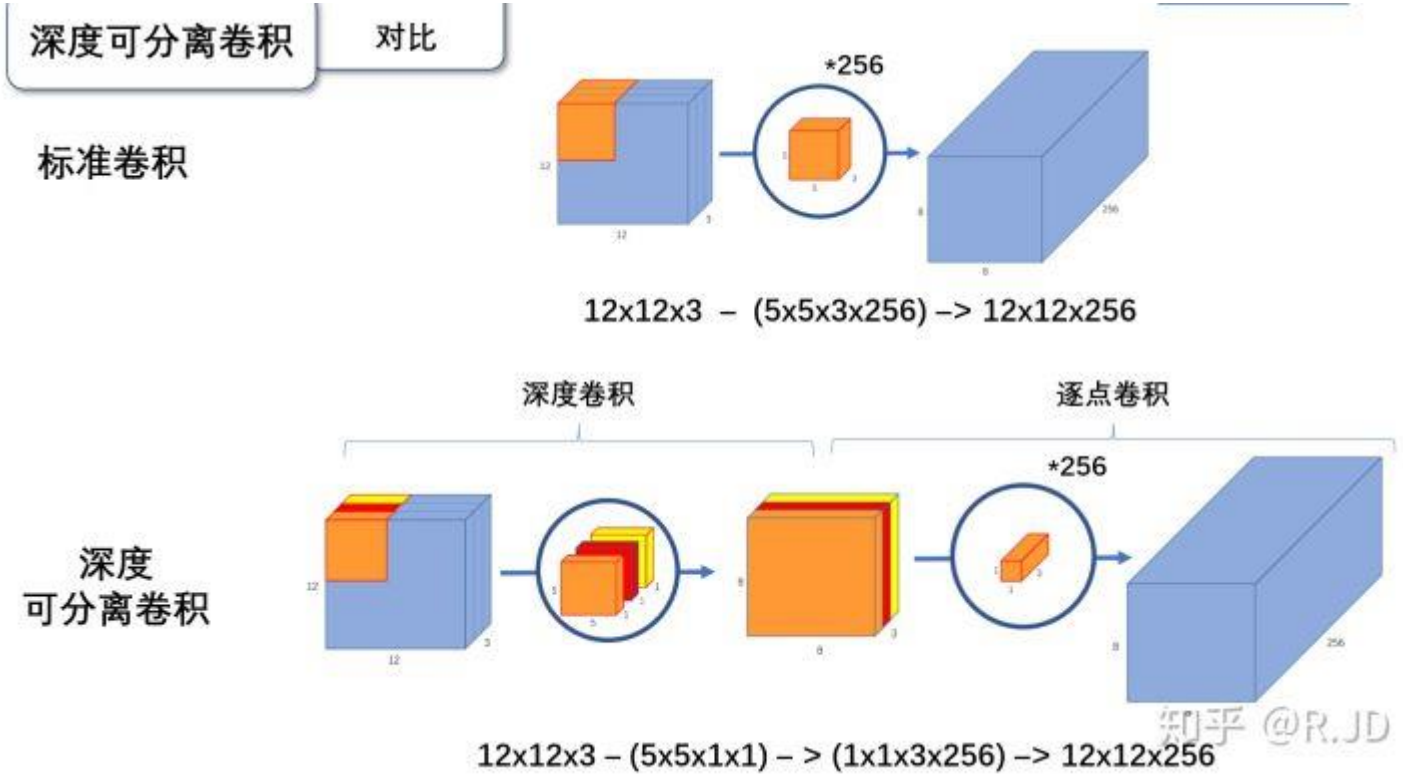


Light CNN

Minghui Chen

MobileNet v1

深度可分离卷积（Depthwise seperable convolution）=深度卷积(depthwise convolution)+逐点卷积(pointwise convolution)。



Ref: https://zhuanlan.zhihu.com/p/70703846

Input channel M
Kernel size $D_K * D_K$

Output channel N
Output feature size $D_F * D_F$

By expressing convolution as a two step process of filtering and combining we get a reduction in computation of:

$$\frac{D_K \cdot D_K \cdot M \cdot D_F \cdot D_F + M \cdot N \cdot D_F \cdot D_F}{D_K \cdot D_K \cdot M \cdot N \cdot D_F \cdot D_F} = \frac{1}{N} + \frac{1}{D_K^2}$$

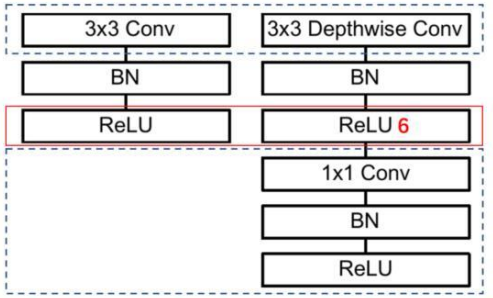


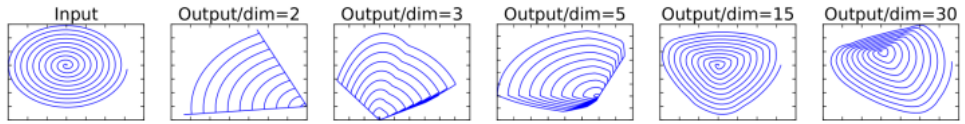
Figure 3. Left: Standard convolutional layer with batchnorm and ReLU. Right: Depthwise Separable convolutions with Depthwise and Pointwise layers followed by batchnorm and ReLU.

ReLU6就是普通的ReLU但是限制最大输出值为6（对输出值做clip），这是为了在移动端设备float16的低精度的时候，也能有很好的数值分辨率，如果对ReLU的激活范围不加限制，输出范围为0到正无穷，如果激活值非常大，分布在一个很大的范围内，则低精度的float16无法很好地精确描述如此大范围的数值，带来精度损失。

MobileNet v2

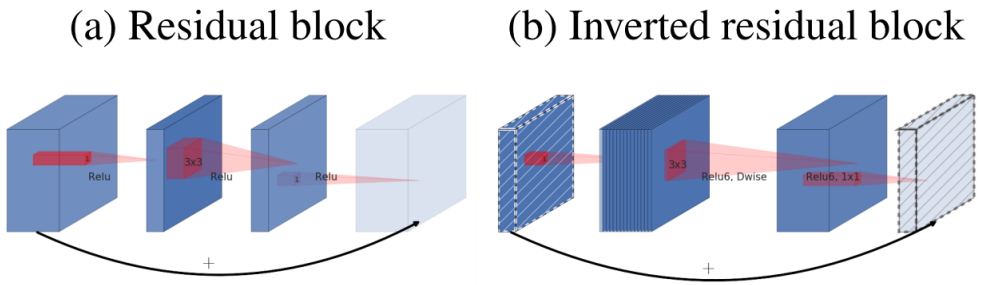
关键改进点：Linear Bottleneck和Inverted Residual。

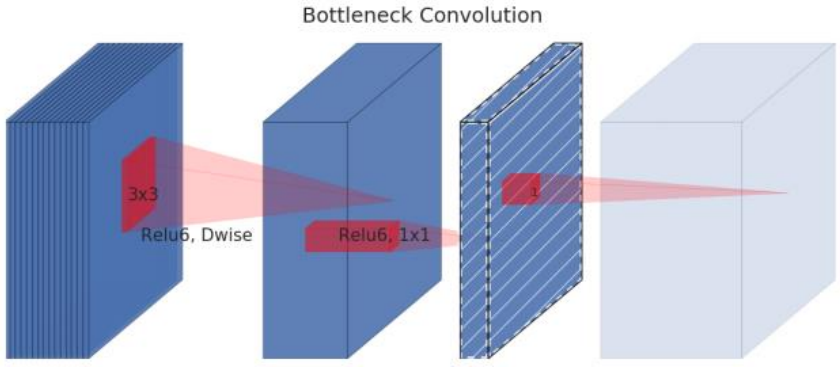
MobileNet v1的问题：深度卷积部分的卷积核比较容易训废掉：训完之后发现深度卷积训出来的卷积核有不少是空的，并将这个锅递给了ReLU。
做了个实验，就是对一个n维空间中的一个“东西”做ReLU运算，然后（利用T的逆矩阵T^{-1}恢复）对比ReLU之后的结果与Input的结果相差有多大。



概括而言，就是对低维度做ReLU运算，很容易造成信息的丢失。而在高维度进行ReLU运算的话，信息的丢失则会很少。
这就解释了为什么深度卷积的卷积核有不少是空。作者解决的方案：将ReLU替换成线性激活函数，但不是所有ReLU都换成线性激活函数，而是将每一个Block的最后一个ReLU换成线性激活函数。

现在还有个问题是，深度卷积本身没有改变通道的能力，来的是多少通道输出就是多少通道。如果来的通道很少的话，DW深度卷积只能在低维度上工作，这样效果并不会很好，所以我们要“扩张”通道。既然我们已经知道PW逐点卷积也就是1×1卷积可以用来升维和降维，那就可以在DW深度卷积之前使用PW卷积进行升维（升维倍数为t，t=6），再在一个更高维的空间中进行卷积操作来提取特征：





- **ResNet 的微结构（Residual Module）**

C_{in} → 1×1 PW → $0.25\times C_{in}$ → 3×3 Regular CONV → $0.25\times C_{in}$ → 1×1 PW → $C_{out}(\approx C_{in})$

- **MobileNet v2 的微结构（Inverted Residual Module）**

C_{in} → 1×1 PW → $6\times C_{in}$ → 3×3 DW → $6\times C_{in}$ → 1×1 PW → $C_{out}(\approx C_{in})$

- **v1 的微结构**

→ 3×3 DW → $ReLU6$ → 1×1 PW → $ReLU6$

- **v2 的微结构**

→ 1×1 PW → $ReLU6$ → 3×3 DW → $ReLU6$ → 1×1 PW → $Linear$

MobileNet v3

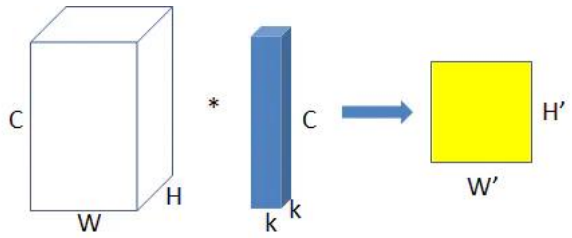
多个小细节的改动（trick的堆叠）：架构基于NAS实现的MnasNet，新的激活函数h-swish(x)，引入SE结构。（不展开了）

ShuffleNet v1

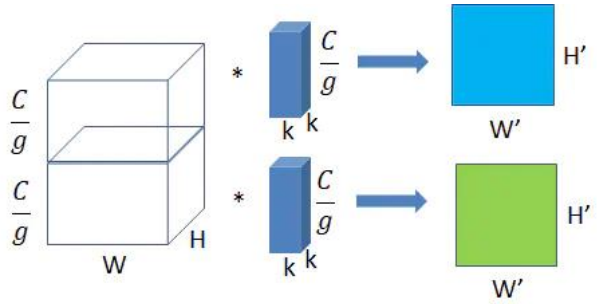
关键创新点：Pointwise Group Convolution, Channel Shuffle。

分组卷积和深度可分离卷积

普通卷积

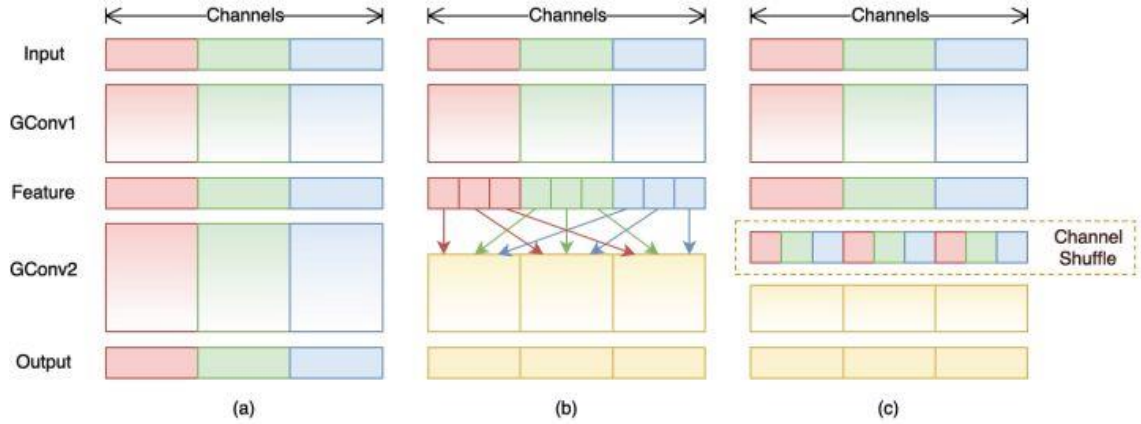


分组卷积：分组卷积最早是出现在AlexNet，当时这么做是为了解决显存不够的问题，单卡无法容纳模型。
当g=1时，分组卷积为普通卷积；当g=C时，分组卷积为深度卷积（depthwise convolution）。

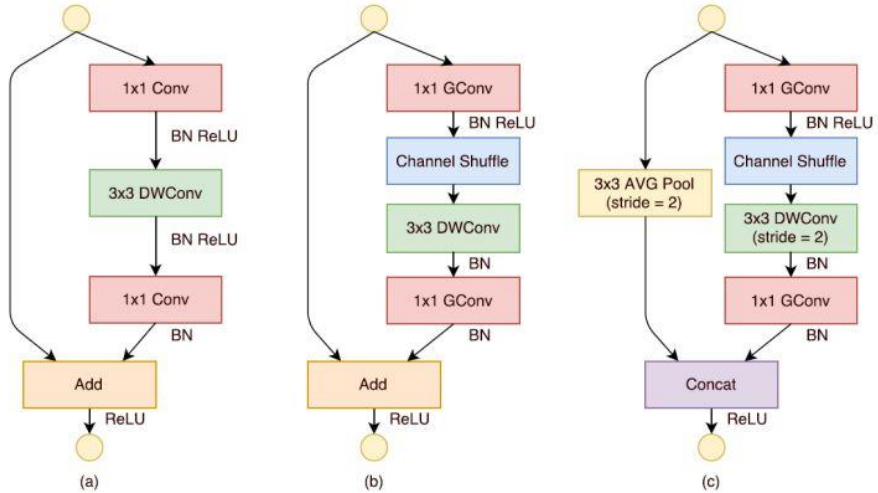


而逐点分组卷积（Pointwise Group Convolution）则是在其后接的1*1卷积的深度（通道）也做同样分组操作，其实就是对Depthwise separable convolution加了个分组操作，使得密集的1*1卷积操作变得相对稀疏，降低计算量。

我们可以看出组内Pointwise卷积可以非常有效的缓解性能瓶颈问题。然而这个策略的一个非常严重的问题是卷积直接的信息沟通不畅，网络趋近于一个由多个结构类似的网络构成的模型集成，精度大打折扣。



可以看到当堆积GConv层后一个问题是不同组之间的特征图是不通信的，这就好像分了三个互不相干的路，大家各走各的，这目测会降低网络的特征提取能力。这样你也可以理解为什么Xception，MobileNet等网络采用密集的x卷积，因为要保证group convolution之后不同组的特征图之间的信息交流。



ShuffleNet v2

关键改进点：Channel Split。

四个高效网络设计指南

G1: 输入输出具有相同channel的时候，内存消耗是最小的。

Formally, following the notations in **G1** and Eq. 1, the relation between MAC and FLOPs for 1×1 group convolution is

$$\begin{aligned} \text{MAC} &= hw(c_1 + c_2) + \frac{c_1 c_2}{g} \\ &= hwc_1 + \frac{Bg}{c_1} + \frac{B}{hw}, \end{aligned} \tag{2}$$

G2: 过多的分组卷积操作会增大MAC，从而使模型速度变慢。
根据G1准则中的MAC计算公式，可以看出在B不变时，g越大，MAC也越大。

		GPU (Batches/sec.)				CPU (Images/sec.)		
g	c for ×1	×1	×2	×4	c for ×1	×1	×2	×4
1	128	2451	1289	437	64	40.0	10.2	2.3
2	180	1725	873	341	90	35.0	9.5	2.2
4	256	1026	644	338	128	32.9	8.7	2.1
8	360	634	445	230	180	27.8	7.5	1.8

Table 2: Validation experiment for **Guideline 2**. Four values of group number g are tested, while the total FLOPs under the four values is fixed by varying the total channel number c. Input image size is 56×56.

G3: 模型中的分支数量越少，模型速度越快。
作者认为，模型中的网络结构太复杂（分支和基本单元过多）会降低网络的并行程度，模型速度越慢。

	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

Table 3: Validation experiment for **Guideline 3**. c denotes the number of channels for *1-fragment*. The channel number in other fragmented structures is adjusted so that the FLOPs is the same as *1-fragment*. Input image size is 56×56.

G4: Element-wise操作不能被忽略。
Element-wise包括Add/ReLU/short-cut/depthwise convolution等操作。元素操作类型操作虽然FLOPs非常低，但是带来的时间消耗还是非常明显的，尤其是在GPU上。元素操作操作虽然基本上不增加FLOPs，但是所带来的时间消耗占比却不可忽视。也即Small FLOPs heavy MAC

通道分离（Channel Split）：

1.在每个单元的开始，通过Channel split将c特征通道的输入被分为两支，分别带有 c−c' 和c'个通道。按照准则**G3**，一个分支的结构仍然保持不变。另一个分支由三个卷积组成， 为满足**G1**，令输入和输出通道相同。与 ShuffleNet V1 不同的是，两个 1×1 卷积不再是组卷积(GConv)，因为Channel Split分割操作已经产生了两个组。
2.卷积之后，把两个分支拼接(Concat)起来，从而通道数量保持不变 (G1)，而且也没有Add操作（element-wise操作）（**G4**）。然后进行与ShuffleNetV1相同的Channel Shuffle操作来保证两个分支间能进行信息交流。
3.depthwise convolution保留

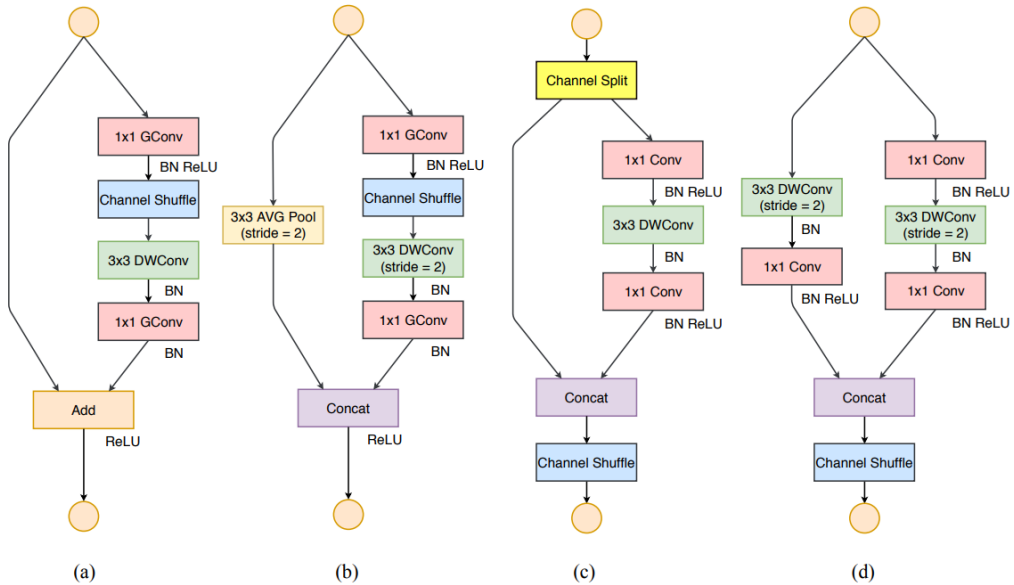


Fig. 3: Building blocks of ShuffleNet v1 [15] and this work. (a): the basic ShuffleNet unit; (b) the ShuffleNet unit for spatial down sampling (2×); (c) our basic unit; (d) our unit for spatial down sampling (2×). **DWConv**: depthwise convolution. **GConv**: group convolution.

Ref: https://zhuanlan.zhihu.com/p/67009992