Convolutional Neural Network Architectures

Presenter: Wei Li

Advisor: I-Chen Wu

Outline

- Reference
- LeNet-5
- AlexNet
- ZFNet
- Network in Network
- VGG Network
- GoogLeNet
- Residual Network
- Wide Residual Network
- ResNeXt
- DenseNet
- Dual Path Network
- Squeeze-and-Excitation Networks
- Summary of CNN architectures

Reference

- LeNet-5
  - Gradient-Based Learning Applied to Document Recognition
    - The first successful applications of Convolutional Networks
    - Developed by Yann LeCun in **1998**.
- AlexNet & ZFNet
  - ImageNet Classification with Deep Convolutional Neural Networks
    - The first work that popularized Convolutional Networks in Computer Vision.
    - ImageNet ILSVRC challenge **2012** champion
  - Visualizing and Understanding Convolutional Networks
    - An improvement on AlexNet by tweaking the architecture hyperparameters
    - ImageNet ILSVRC challenge **2013** champion

Reference(cont.)

- Network in Network [arXiv:1312.4400v3]
  - Network In Network
    - 1x1 convolution
    - Global average pooling
- VGG Network [arXiv:1409.1556v6]
  - Very Deep Convolutional Networks for Large-Scale Image Recognition
    - University of Oxford -- Visual Geometry Group
    - The **first** and the **second** places in ImageNet ILSVRC challenge **2014** localization and classification tasks

Reference(cont.)

- GoogLeNet
  - [v1] Going Deeper with Convolutions [arXiv:1409.4842]
    - The **first** places in ImageNet ILSVRC challenge **2014** classification tasks
    - Efficient “Inception” module
  - There are also several follow-up versions to the GoogLeNet:
    - [v2] Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift [arXiv:1502.03167v3]
    - [v3] Rethinking the Inception Architecture for Computer Vision [arXiv:1512.00567]
    - [v4] Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning [arXiv:1602.07261v2]

Reference(cont.)

- Residual Network
  - Deep Residual Learning for Image Recognition [arXiv:1512.03385]
    - CVPR 2016 Best Paper Award
    - MSRA: Kaiming He 何恺明(FAIR now)
      - Faster R-CNN
      - Mask R-CNN
    - **1st places** in **all five** main tracks:
      - ImageNet Classification: "Ultra-deep"152-layer nets
      - ImageNet Detection: 16% better than 2nd
      - ImageNet Localization: 27% better than 2nd
      - COCO Detection: 11% better than 2nd
      - COCO Segmentation: 12% better than 2nd
  - Identity Mappings in Deep Residual Networks [arXiv:1603.05027v3]

Reference(cont.)

- Wide Residual Network
  - Wide Residual Networks [arXiv:1605.07146v4]
    - Sergey Zagoruyko
- ResNeXt
  - Aggregated Residual Transformations for Deep Neural Networks [arXiv:1611.05431]
    - Also from creators of ResNet
    - Increases width of residual block through multiple parallel pathways ("cardinality")
    - Parallel pathways similar in spirit to Inception module
- DenseNet
  - Densely Connected Convolutional Networks [arXiv:1608.06993v4]
    - CVPR **2017** Best Paper Award
    - Dense blocks where each layer is connected to every other layer in feedforward fashion

Reference(cont.)

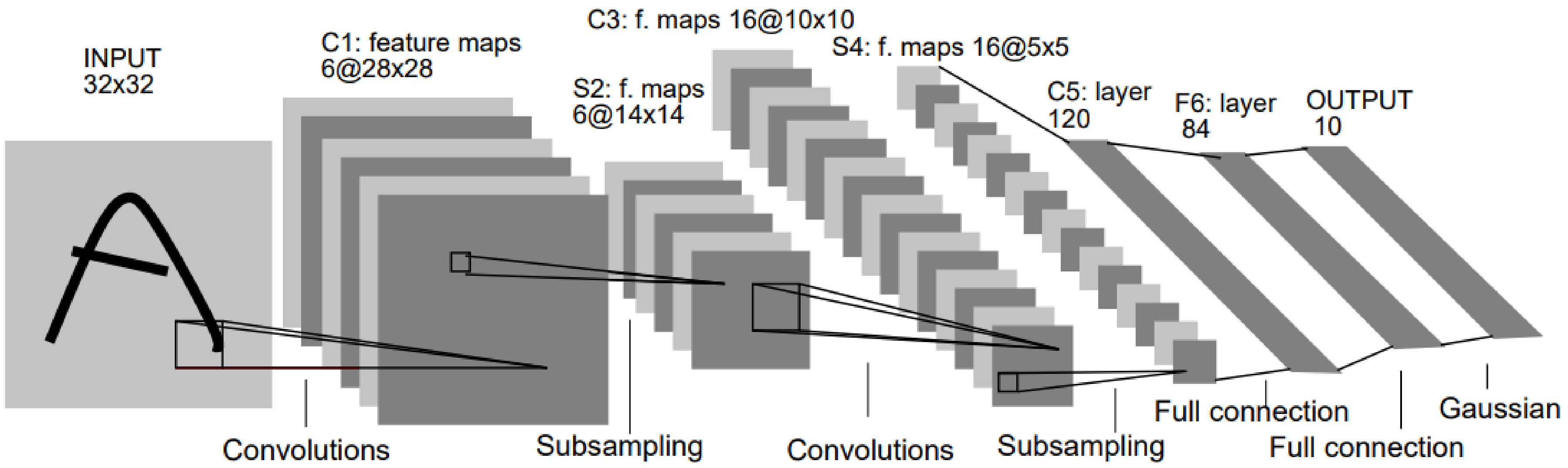
- Dual Path Network
  - Dual Path Networks [arXiv:1707.01629]
    - The **first** places in ImageNet ILSVRC challenge **2017** object localization tasks
    - Qihoo 360 and National University of Singapore
- SENet
  - Squeeze-and-Excitation Networks [arXiv:1709.01507]
    - The **first** places in ImageNet ILSVRC challenge **2017** classification tasks
    - Momenta and University of Oxford.

Basic Networks

- LeNet-5
- AlexNet & ZFNet
- Network in Network

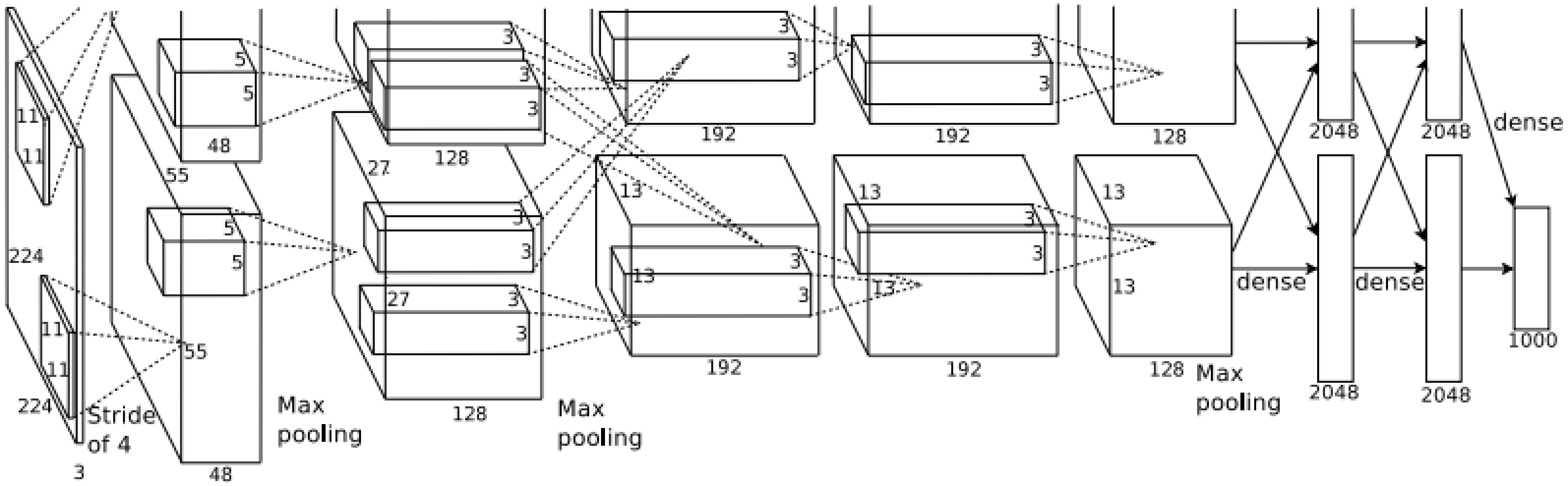
LeNet-5

- **The first successful applications of Convolutional Networks (by LeCun in 1998)**
- Handwritten and machine-printed character recognition.
- **No GPU**
    - NVIDIA STG-2000X(1994) 1MB memory
    - RIVA 128ZX (1998) 8MB memory



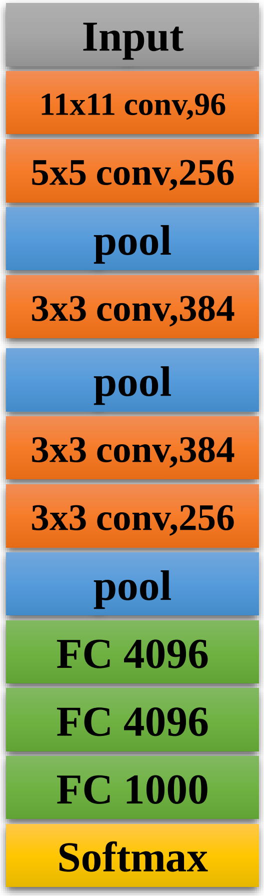
AlexNet

- **First use of ReLU**
- Used Norm layers(not common anymore)
- Data augmentation / Dropout / Weight Decay
- **Used GPU to train**
  - Trained on GTX 580(3 GB memory)
  - Networks spread across 2 GPUs, half the neurons(feature maps) on each GPU
- **ImageNet ILSVRC challenge 2012 champion**



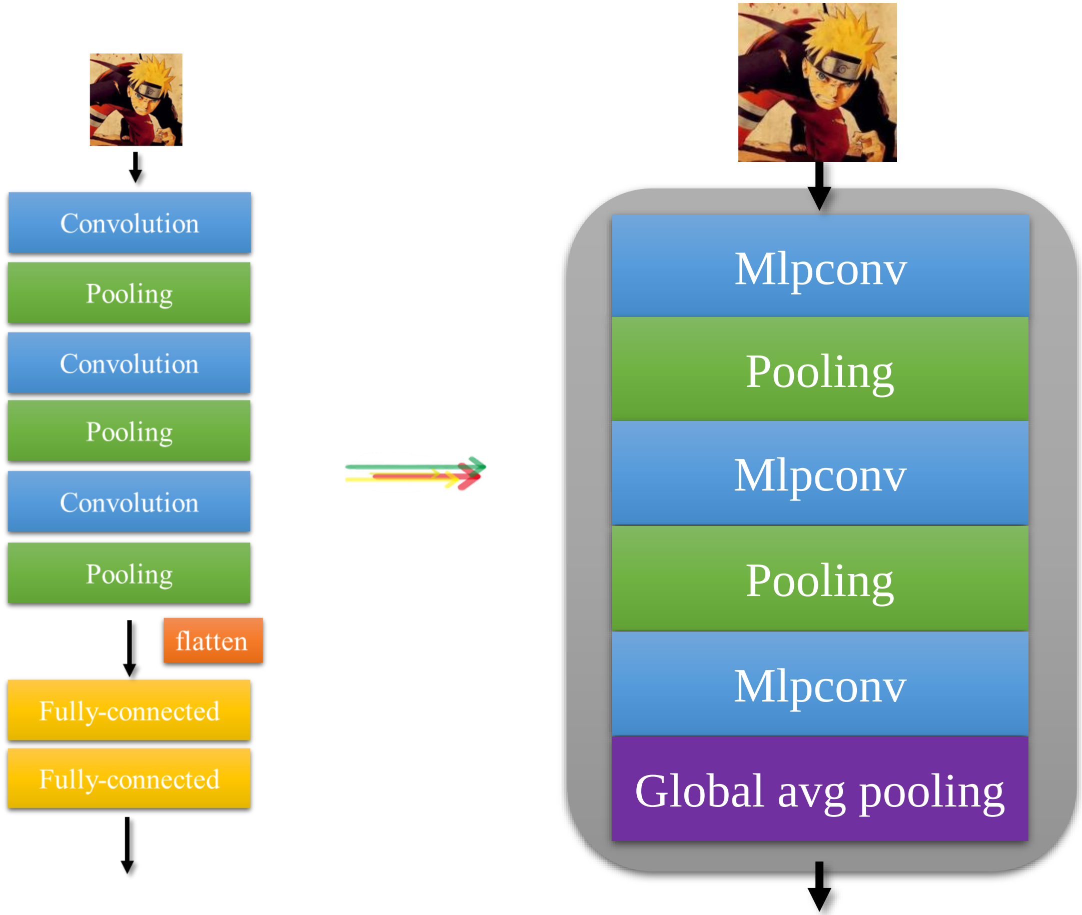
AlexNet(cont.)

Full (simplified) AlexNet architecture:
[227x227x3] INPUT
[55x55x96] CONV1: 96 11x11 filters at stride 4, pad 0
[27x27x96] MAX POOL1: 3x3 filters at stride 2
[27x27x96] NORM1: Normalization layer
[27x27x256] CONV2: 256 5x5 filters at stride 1, pad 2
[13x13x256] MAX POOL2: 3x3 filters at stride 2
[13x13x256] NORM2: Normalization layer
[13x13x384] CONV3: 384 3x3 filters at stride 1, pad 1
[13x13x384] CONV4: 384 3x3 filters at stride 1, pad 1
[13x13x256] CONV5: 256 3x3 filters at stride 1, pad 1
[6x6x256] MAX POOL3: 3x3 filters at stride 2
[4096] FC6: 4096 neurons
[4096] FC7: 4096 neurons
[1000] FC8: 1000 neurons (class scores)



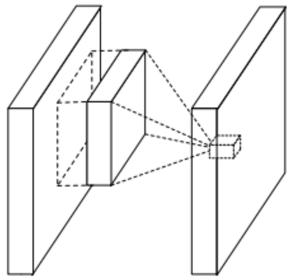
Network in Network

- **Used Mlpconv** layers instead of traditional conv layers
- Removed all of the fully-connected layer, used **global average pooling**



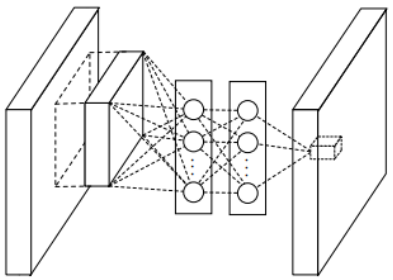
Network in Network(cont.)

- Mlpconv layer



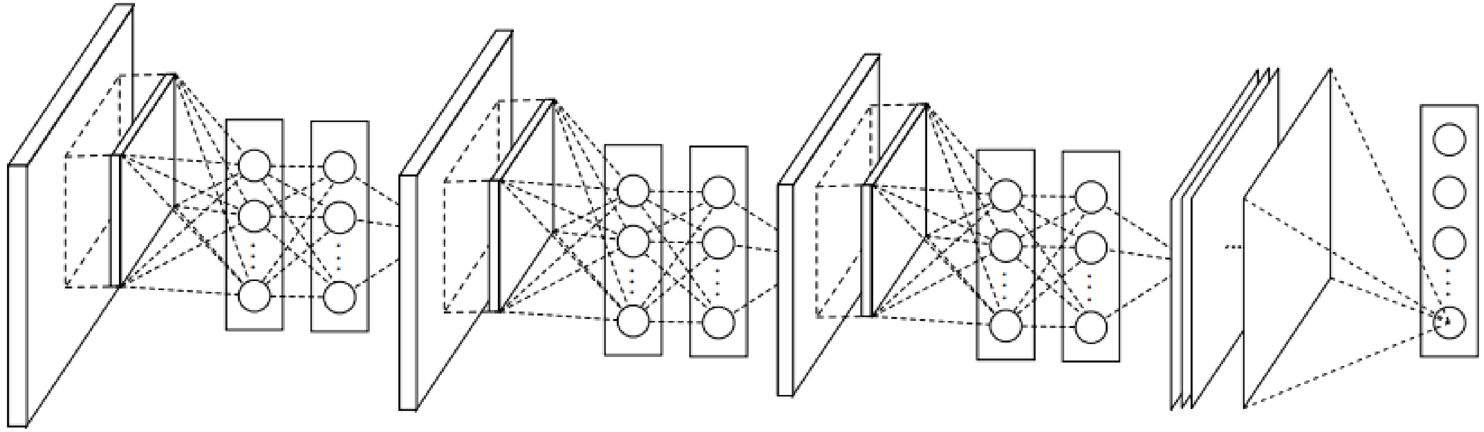
$$f_{i,j,k} = \max(w_k^T x_{i,j}, 0).$$

(a) Linear convolution layer



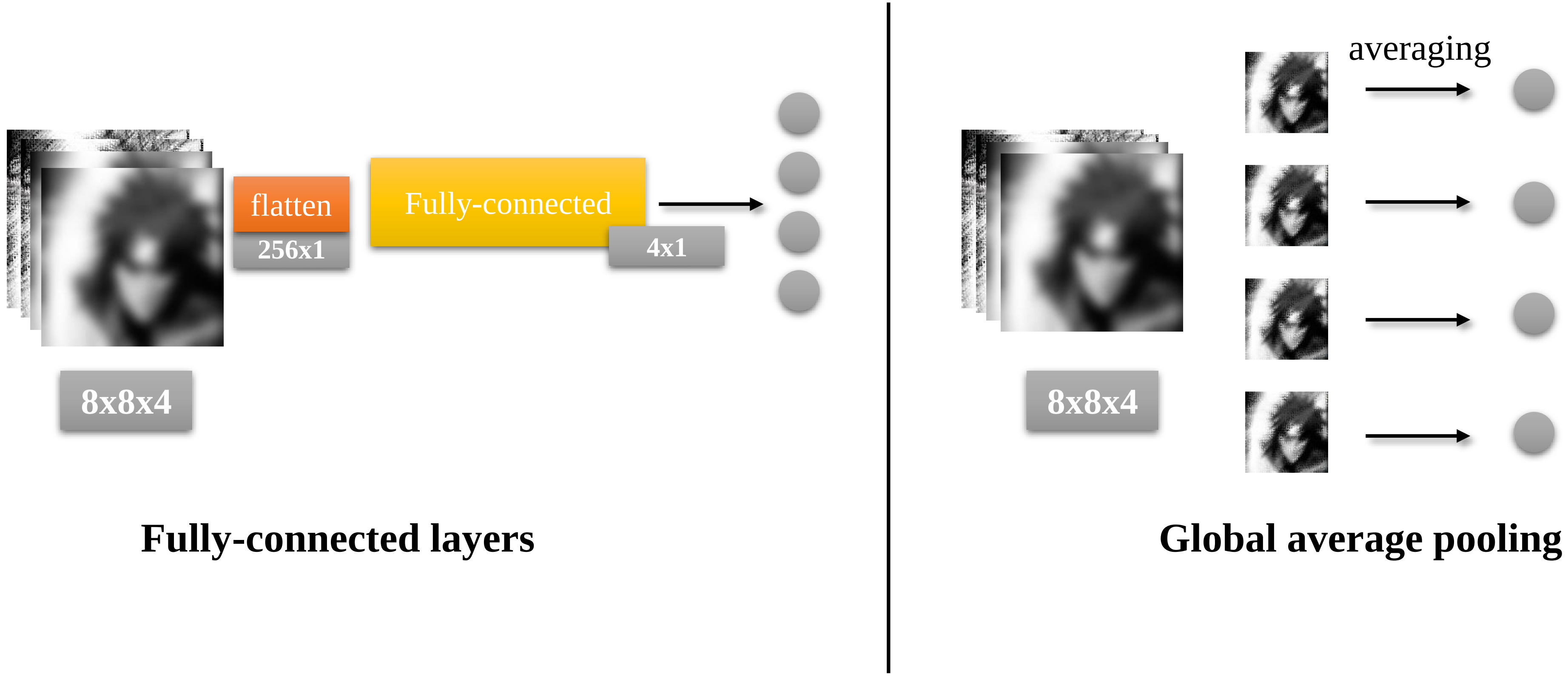
$$\begin{aligned} f^1_{i,j,k_1} &= \max({w^1_{k_1}}^T x_{i,j} + b_{k_1}, 0). \\ &\vdots \\ f^n_{i,j,k_n} &= \max({w^n_{k_n}}^T f^{n-1}_{i,j} + b_{k_n}, 0). \end{aligned}$$

(b) Mlpconv layer



Network in Network(cont.)

- Global average pooling

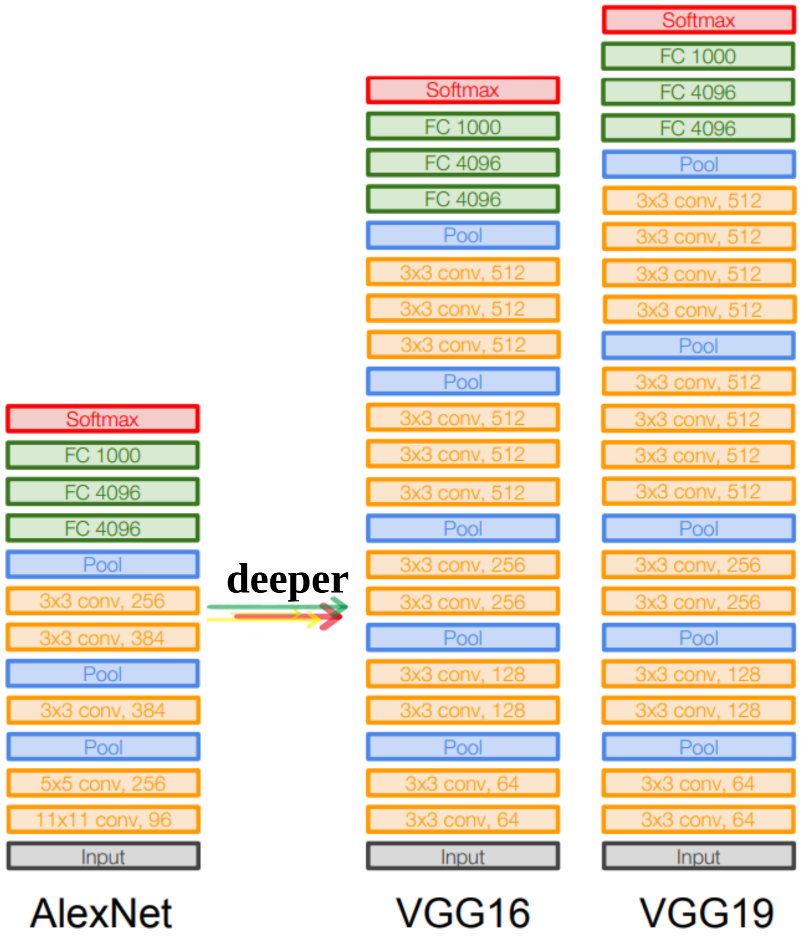


Fully-connected layers **Global average pooling**

VGG and GoogLeNet

- VGG Networks
- GoogLeNet

VGG Network



8 layers (AlexNet) → 19 layers (VGGNet)

11x11(5x5,3x3) conv → 3x3 conv

11.7% top 5 error in ILSVRC'13(ZFNet)
→ 7.3% top 5 error in ILSVRC'14

Why use smaller filters? (3x3 conv)

1. **Stack of three 3x3 conv (stride 1) layers has same effective receptive field as one 7x7 conv layer**
2. **But the network is deeper and more non-linearity**
3. **And fewer parameter:**

$$3 * (C * (3 * 3 * C)) \text{ vs. } C * (7 * 7 * C)$$
$$27C^2 \text{ vs. } 49C^2$$

VGG Network(cont.)

(not counting biases)

INPUT: [224x224x3] memory: 224*224*3=150K params: 0
CONV3-64: [224x224x64] memory: **224*224*64=3.2M** params: (3*3*3)*64 = 1,728
CONV3-64: [224x224x64] memory: **224*224*64=3.2M** params: (3*3*64)*64 = 36,864
POOL2: [112x112x64] memory: 112*112*64=800K params: 0
CONV3-128: [112x112x128] memory: 112*112*128=1.6M params: (3*3*64)*128 = 73,728
CONV3-128: [112x112x128] memory: 112*112*128=1.6M params: (3*3*128)*128 = 147,456
POOL2: [56x56x128] memory: 56*56*128=400K params: 0
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*128)*256 = 294,912
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*256)*256 = 589,824
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*256)*256 = 589,824
POOL2: [28x28x256] memory: 28*28*256=200K params: 0
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*256)*512 = 1,179,648
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*512)*512 = 2,359,296
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*512)*512 = 2,359,296
POOL2: [14x14x512] memory: 14*14*512=100K params: 0
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
POOL2: [7x7x512] memory: 7*7*512=25K params: 0
FC: [1x1x4096] memory: 4096 params: 7*7*512*4096 = **102,760,448**
FC: [1x1x4096] memory: 4096 params: 4096*4096 = 16,777,216
FC: [1x1x1000] memory: 1000 params: 4096*1000 = 4,096,000

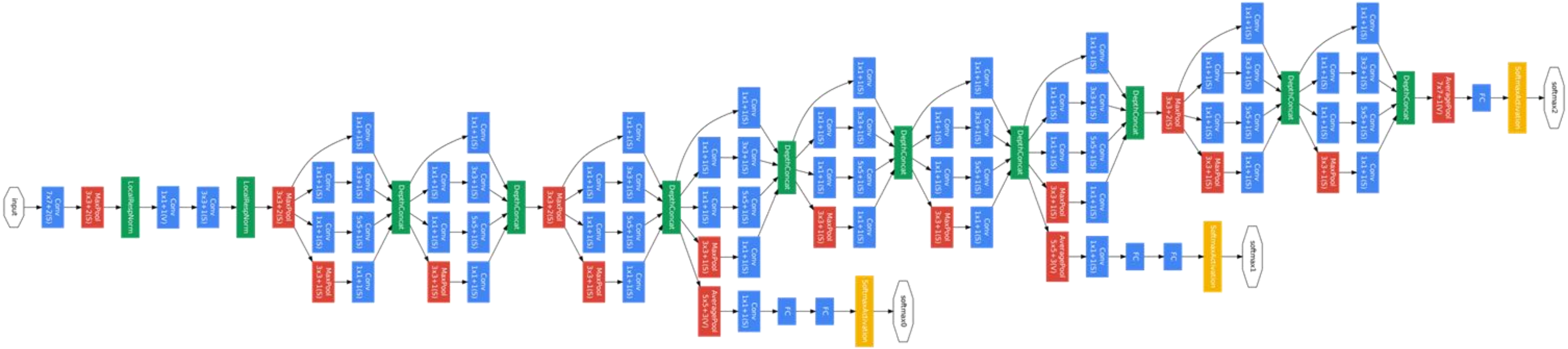
Most memory is in early CONV

Most params are in late FC

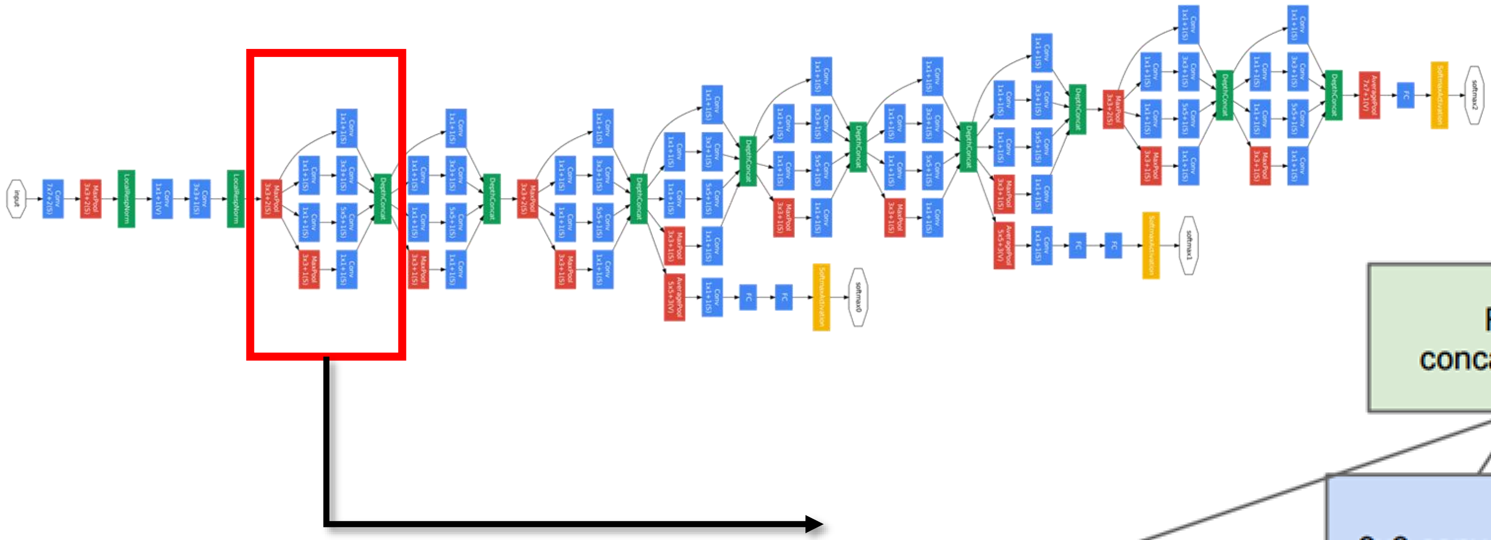
TOTAL memory: 24M * 4 bytes ~= 96MB / image (only forward! ~*2 for bwd)
TOTAL params: 138M parameters

GoogLeNet

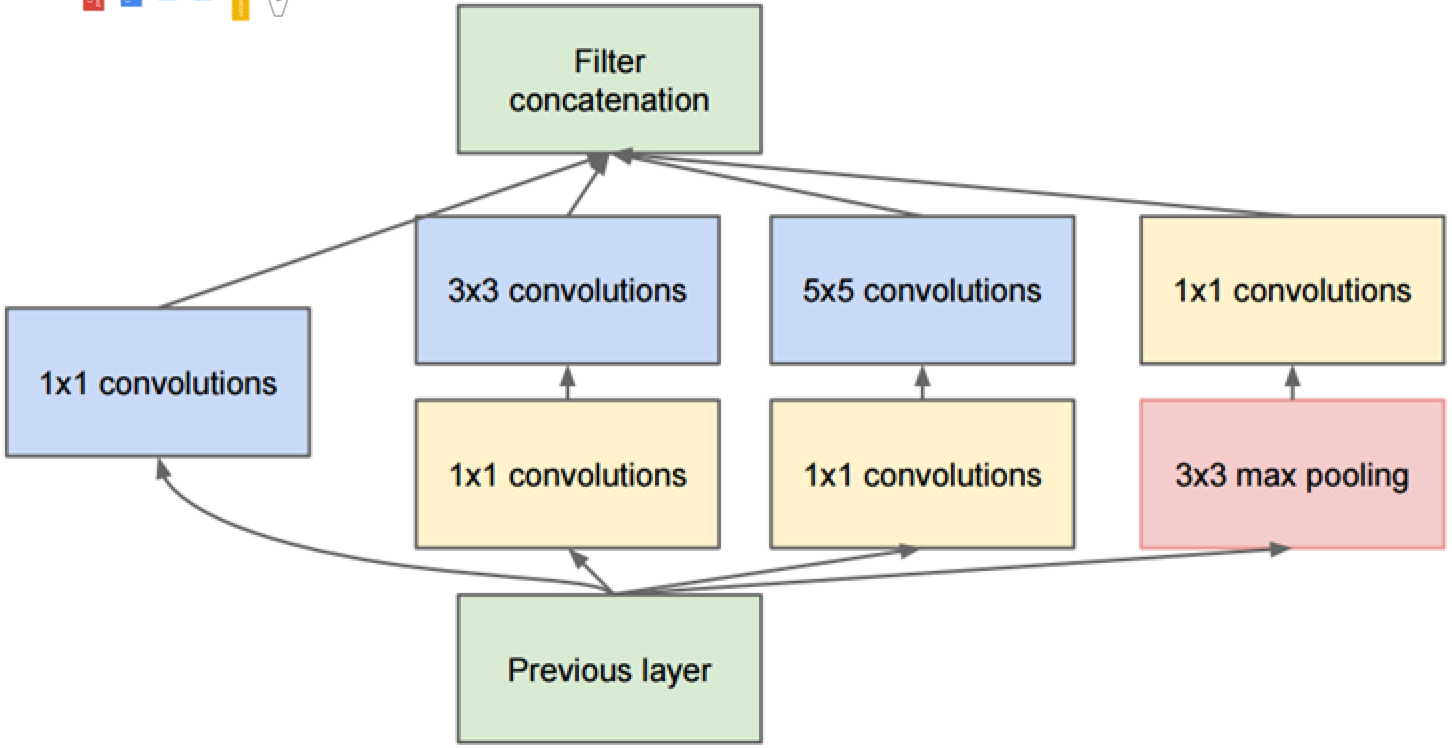
- **Efficient "Inception" module**
- No FC layers
- 12x less params than AlexNet
- **ILSVRC'14 classification winner (6.7% top 5 error)**



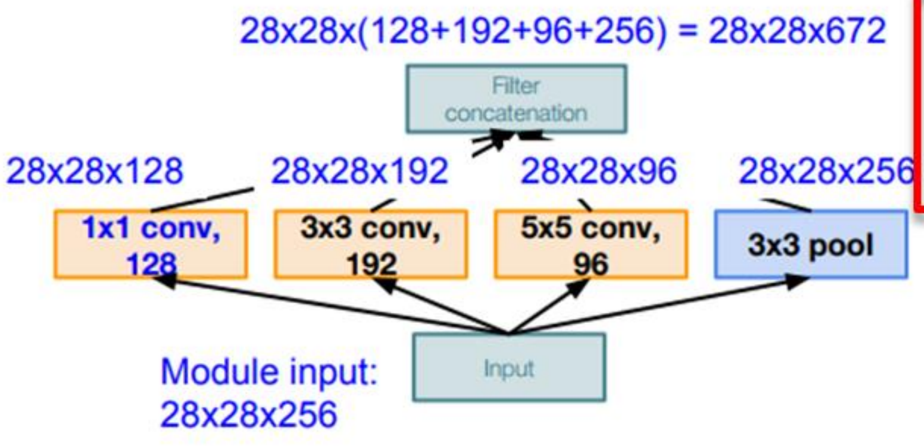
GoogLeNet(cont.)



- **Inception module**



Full Inception module



Naive Inception module

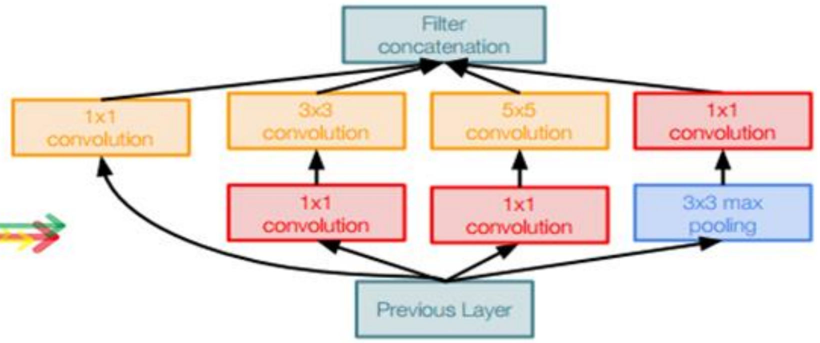
Conv Ops:
[1x1 conv, 128] 28x28x128x1x1x256
[3x3 conv, 192] 28x28x192x3x3x256
[5x5 conv, 96] 28x28x96x5x5x256
Total: 854M ops

Very expensive compute

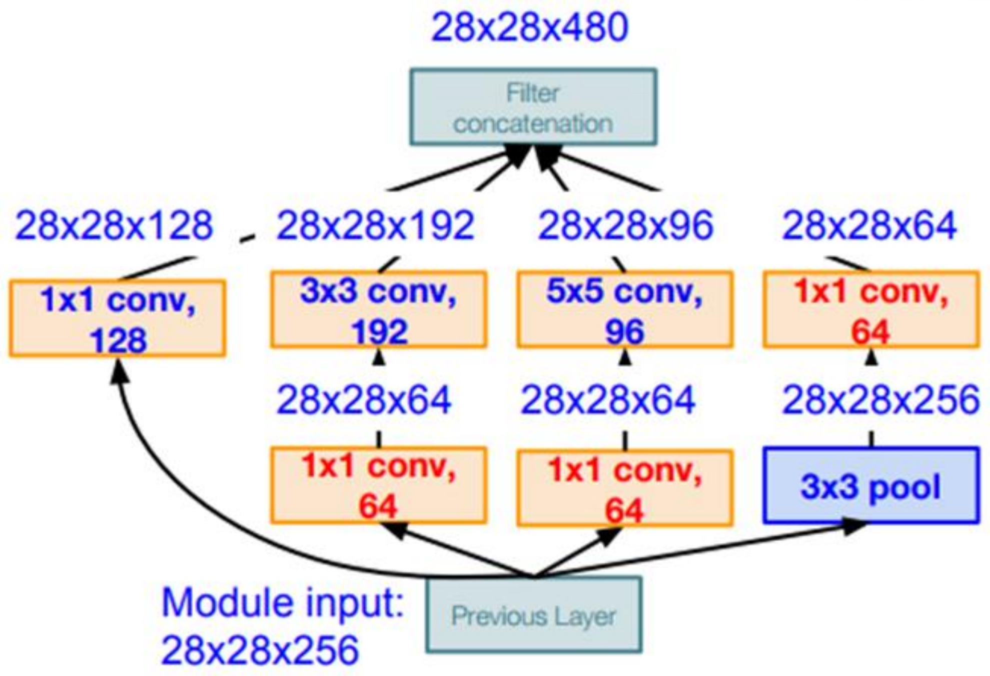
Pooling layer also preserves feature depth, which means total depth after concatenation can only grow at every layer!

Solution: "bottleneck" layers that use 1x1 convolutions to reduce feature depth

1x1 conv "bottleneck" layers



Inception module with dimension reduction



Inception module with dimension reduction

Conv Ops:
[1x1 conv, 64] 28x28x64x1x1x256
[1x1 conv, 64] 28x28x64x1x1x256
[1x1 conv, 128] 28x28x128x1x1x256
[3x3 conv, 192] 28x28x192x3x3x64
[5x5 conv, 96] 28x28x96x5x5x64
[1x1 conv, 64] 28x28x64x1x1x256
Total: 358M ops

Compared to 854M ops for naive version
Bottleneck can also reduce depth after pooling layer

Residual Network and Variants

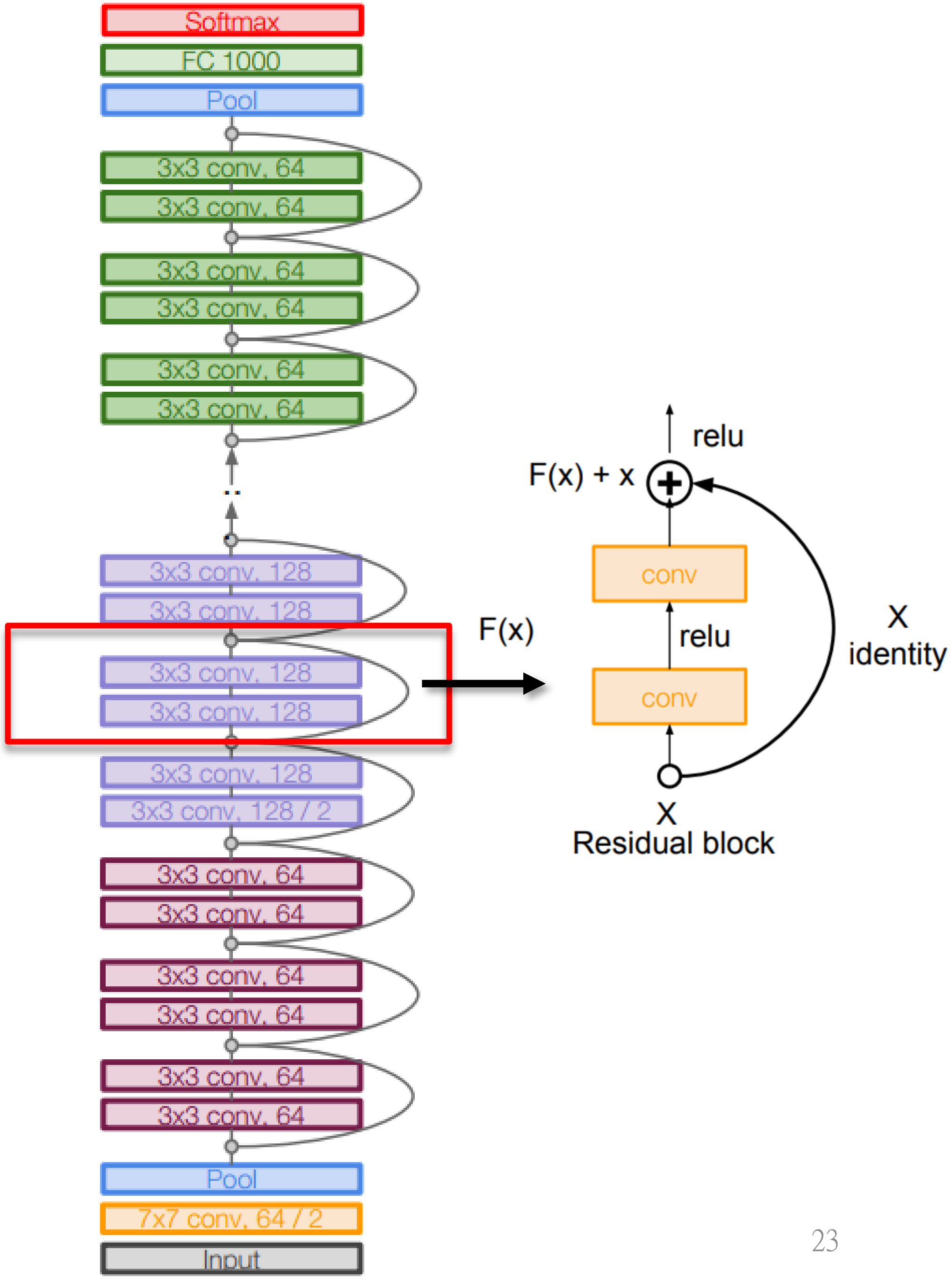
- Residual Network (Identity Mapping)
- Wide Residual Network
- ResNeXt

Residual Network

- MSRA - Kaiming He
- 152-layer model for ImageNet
- ILSVRC’15 classification winner (3.57% top 5 error)
- Swept all classification and detection competitions in ILSVRC’15 and COCO’15

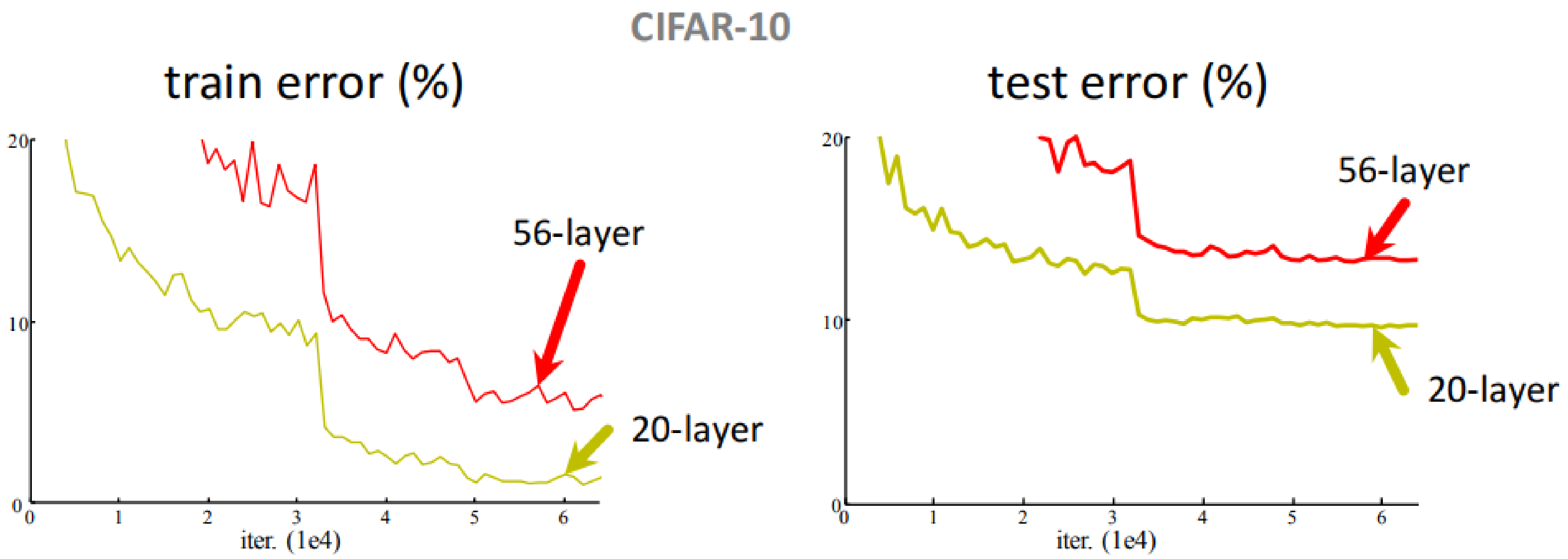
Full ResNet Architecture:

- Stack residual blocks
- Every residual block has two 3x3 conv layers
- Periodically, double # of filters and downsample spatially using stride 2 (/2 in each dimension)
- Additional conv layer at the beginning
- No FC layers at the end (only FC 1000 to outputclasses)



Residual Network(cont.)

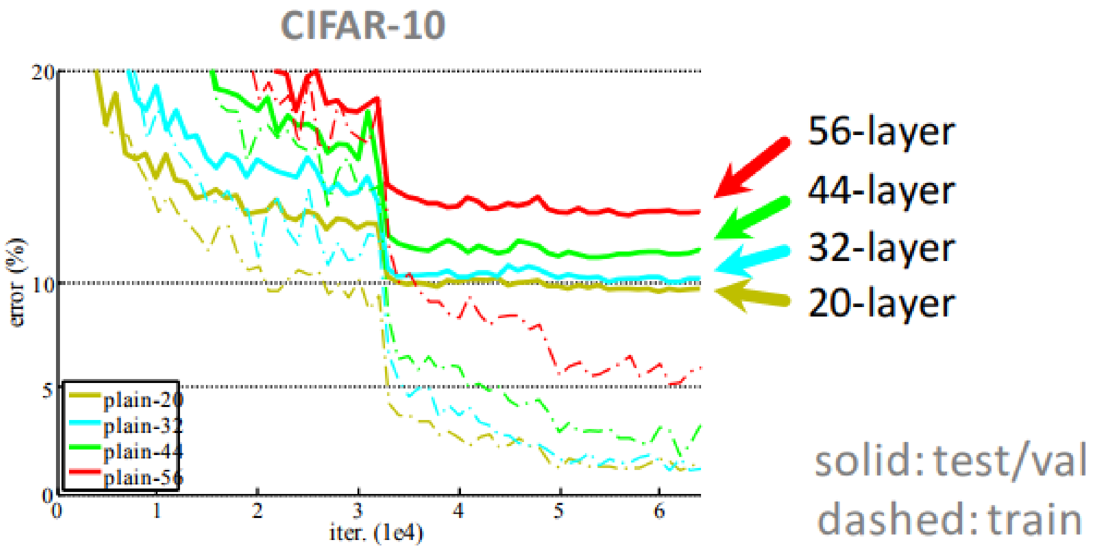
- Simply stacking layers

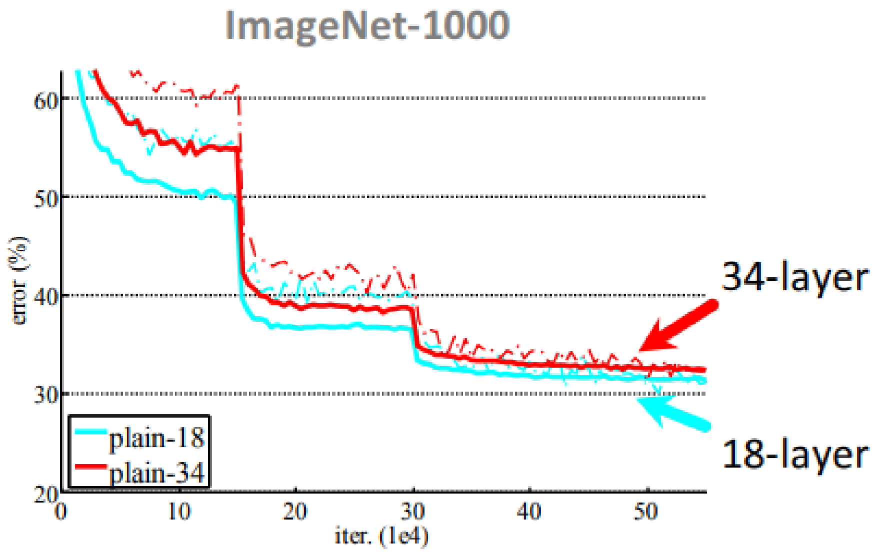


- *Plain* nets: stacking 3x3 conv layers...
- 56-layer net has **higher training error** and test error than 20-layer net

Residual Network(cont.)

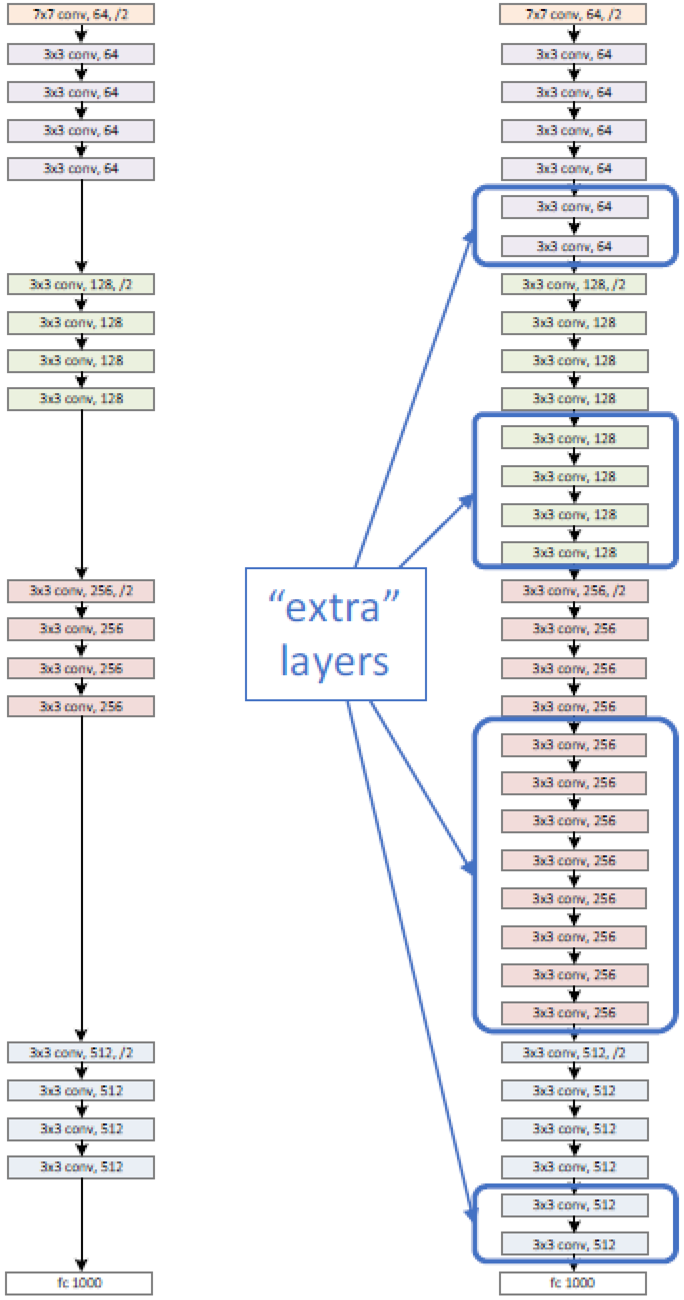
- Simply stacking layers





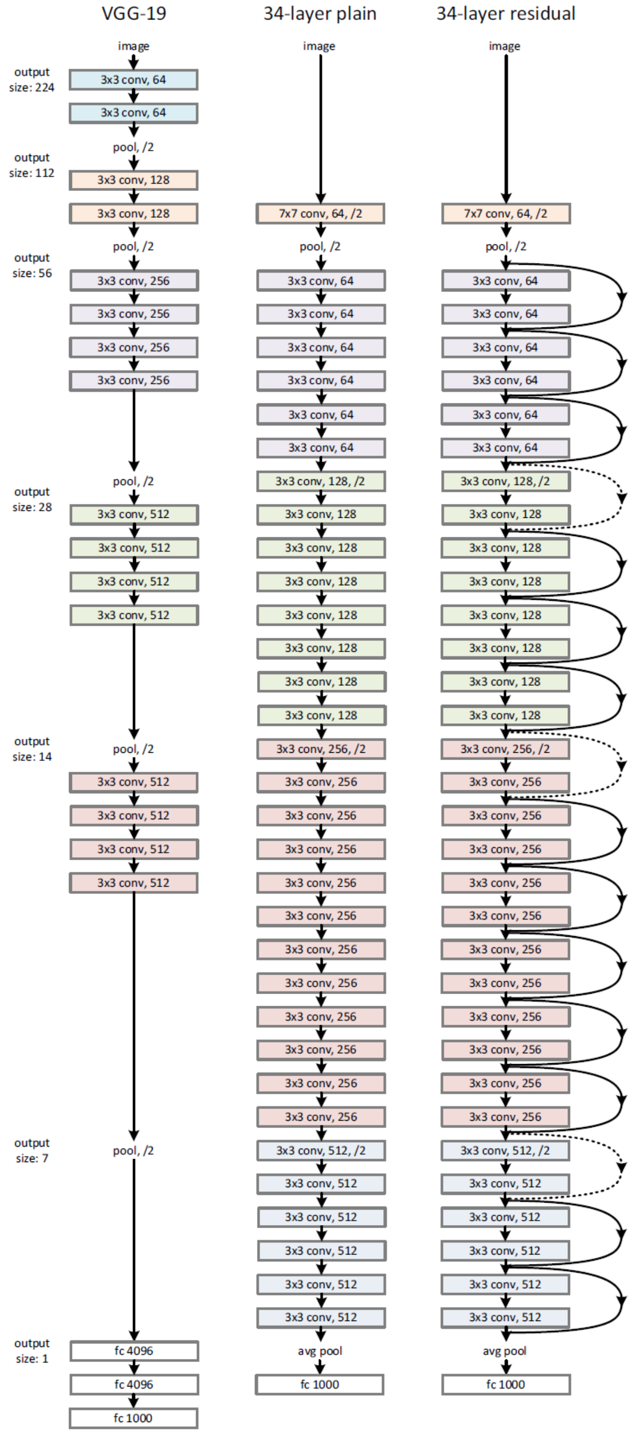
- "Overly deep" plain nets have **higher training error**
- A general phenomenon, observed in many datasets

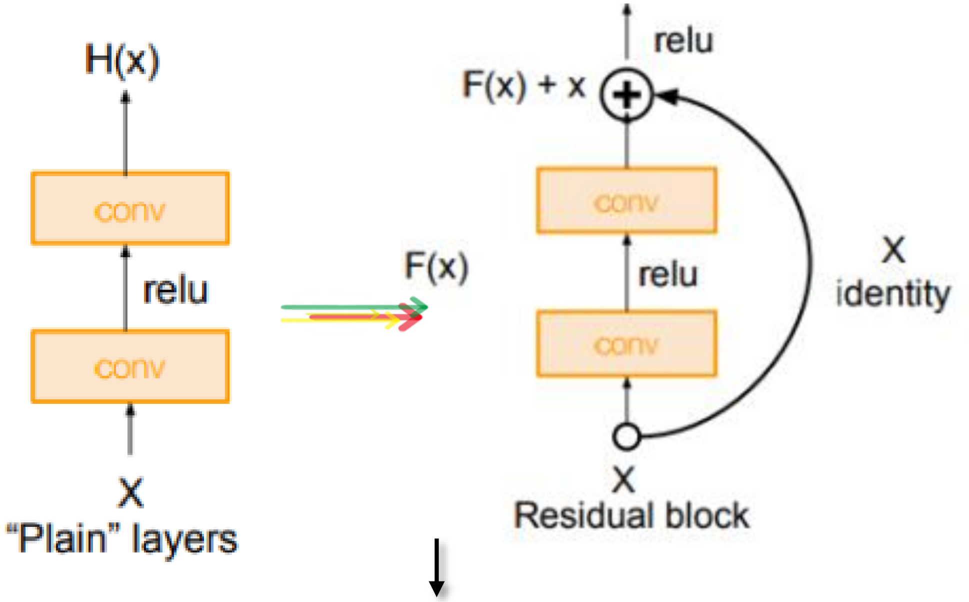
a shallower model
(18 layers)



a deeper counterpart
(34 layers)

- Richer solution space
- A deeper model should not have **higher training error**
- A solution *by construction*:
  - original layers: copied from a learned shallower model
  - extra layers: set as identity
  - at least the same training error
- Optimization difficulties: solvers cannot find the solution when going deeper…





$H(x) = g(x)$

Need to learn $\boldsymbol{g(.)}$

$H(x) = F(x) + x$

Need to learn $\boldsymbol{F(.)}$, but it's easier to learn

assume $\boldsymbol{x = 2.9}$,

after one "plain" layers(or Residual block). $\boldsymbol{H_1(x) = 3.0, F_1(x) = 0.1}$

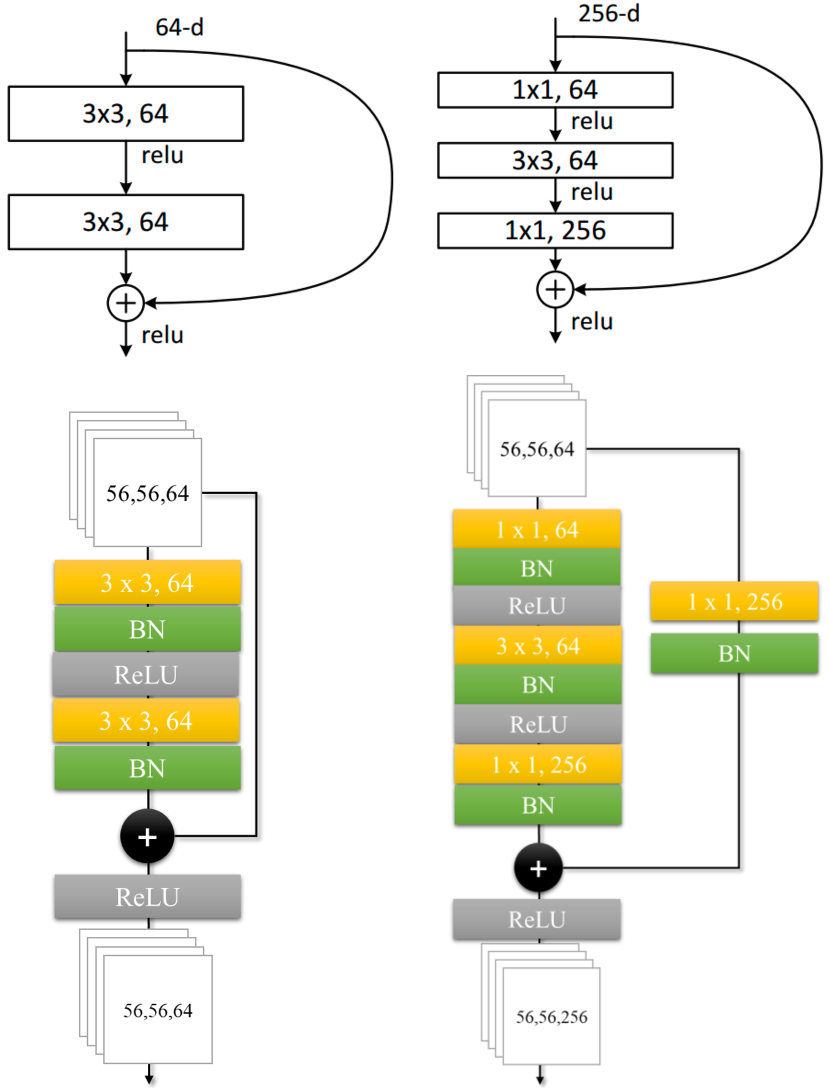
after another "plain" layers(or Residual block). $\boldsymbol{H_2(x) = 3.1, F_2(x) = 0.2}$

Plain layer: $\boldsymbol{\Delta = \frac{3.1 - 3.0}{3.0} = 3.3\%}$

Residual block: $\boldsymbol{\Delta = \frac{0.2 - 0.1}{0.1} = 100\%}$

"We hypothesize that it is easier to optimize the residual mapping than to optimize the original, unreferenced mapping"——authors

Residual Network(bottleneck)



Residual block: **2 layers** **(3-3)**

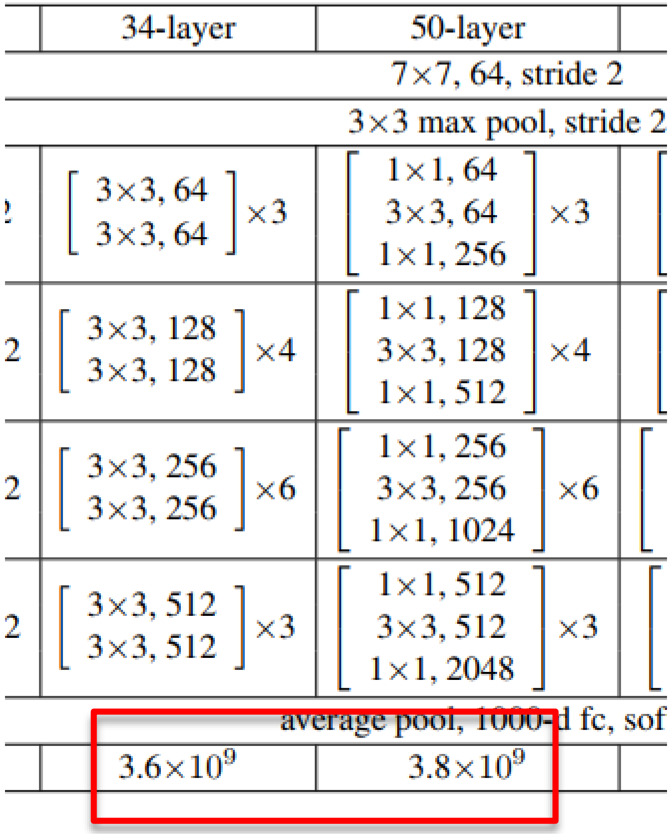
Params:

$$(3 \times 3 \times 64) * 64 + (3 \times 3 \times 64) * 64 = 73K$$

Bottleneck: **3 layers** **(1-3-1)**

Params:

$$(1 \times 1 \times 64) * 64 + (3 \times 3 \times 64) * 64 + (1 \times 1 \times 64) * 256 + (1 \times 1 \times 64) * 256 = 73K$$



	34-layer	50-layer
	7×7, 64, stride 2	
	3×3 max pool, stride 2	
	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
2	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$
2	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$
2	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	average pool, 1000-d fc, sof	
	3.6×10^9	3.8×10^9

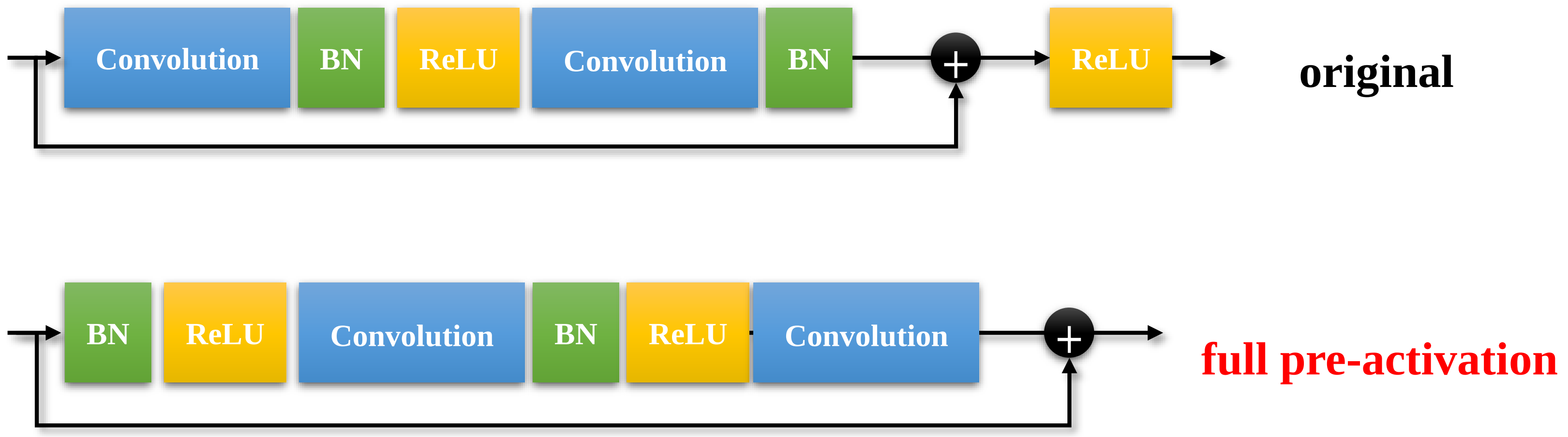
both designs have similar time complexity

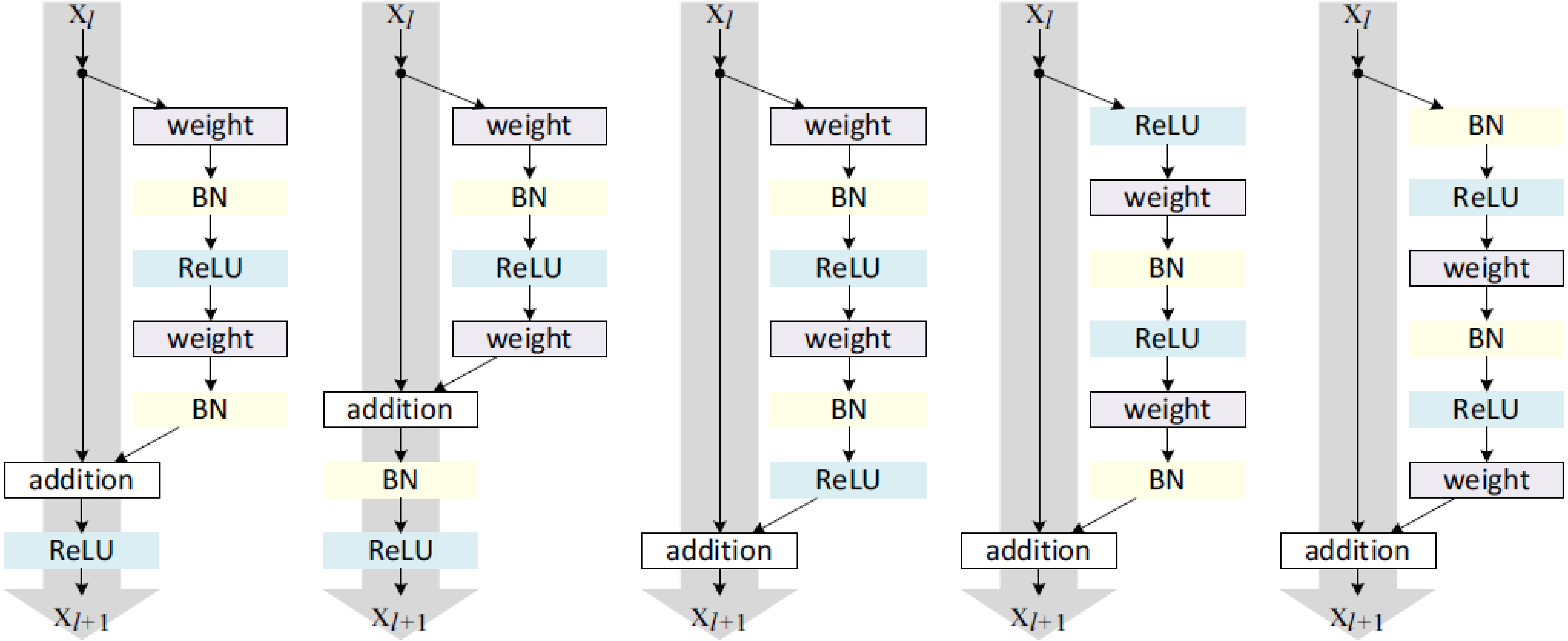
Residual Network(Architectures for ImageNet)

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Residual Network(Identity Mappings)

- Residual Network
  - Deep Residual Learning for Image Recognition [arXiv:1512.03385]
  - **Identity Mappings in Deep Residual Networks [arXiv:1603.05027v3]**





(a) original

(b) BN after addition

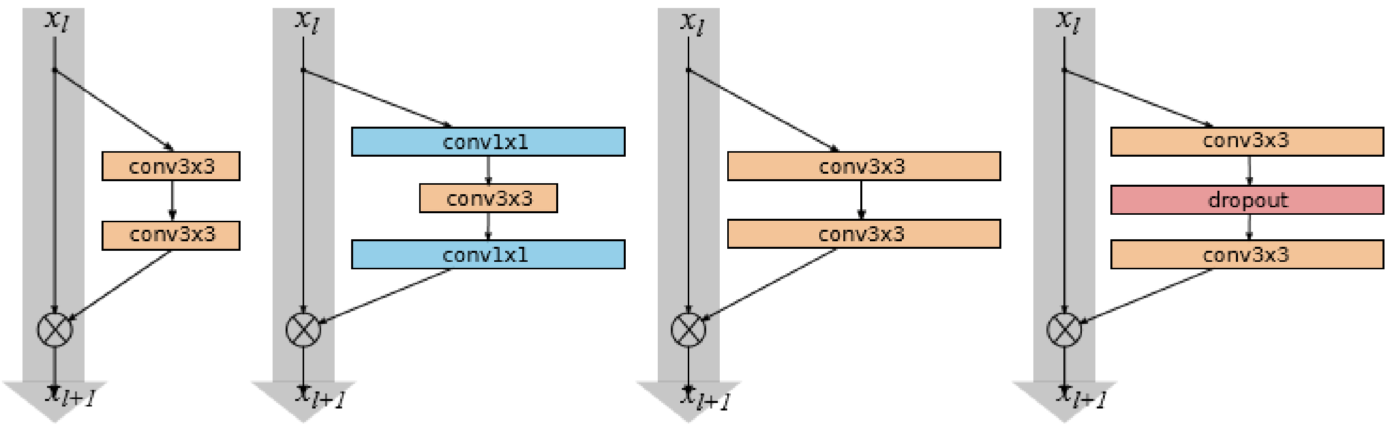
(c) ReLU before addition

(d) ReLU-only pre-activation

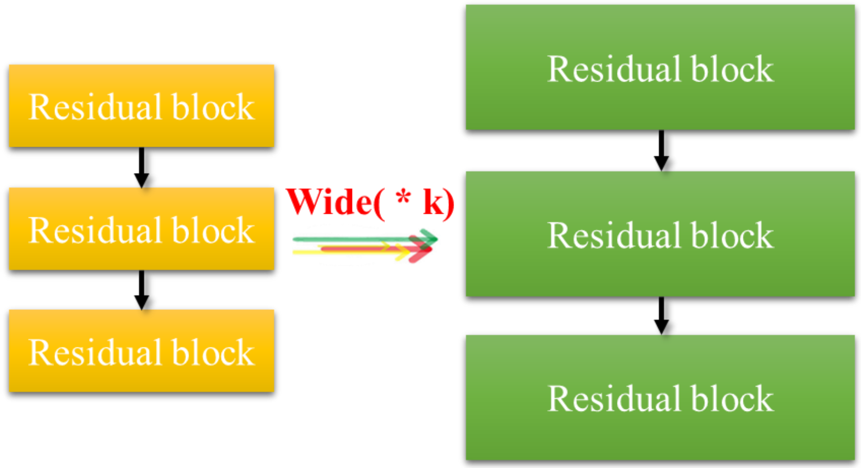
(e) **full pre-activation**

case	Fig.	ResNet-110	ResNet-164
original Residual Unit [1]	Fig. 4(a)	6.61	5.93
BN after addition	Fig. 4(b)	8.17	6.50
ReLU before addition	Fig. 4(c)	7.84	6.14
ReLU-only pre-activation	Fig. 4(d)	6.71	5.91
full pre-activation	Fig. 4(e)	**6.37**	**5.46**

Wide Residual Network



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

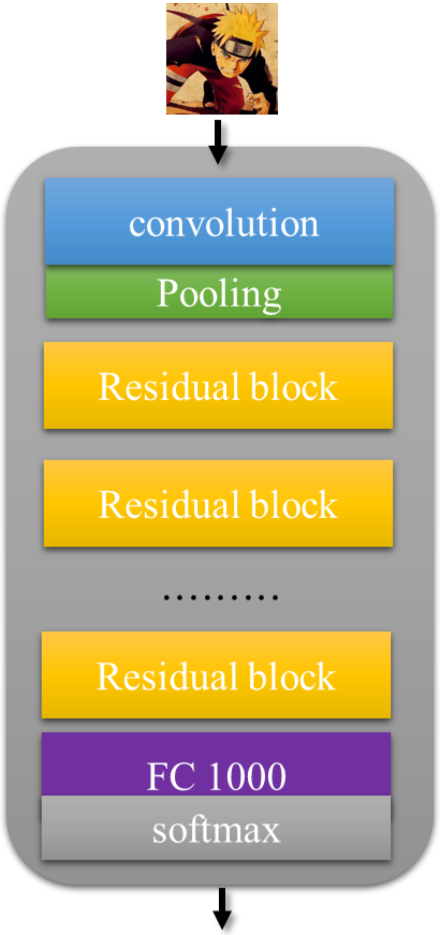


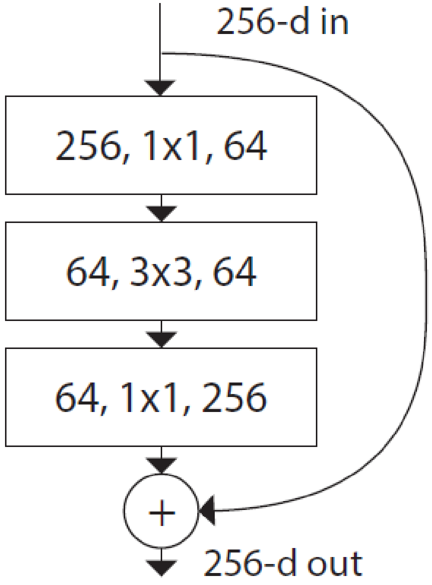
n_filters = [16, 32, 64] **n_filters = [16*k, 32*k, 64*k]**

	depth-k	# params	CIFAR-10	CIFAR-100
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**

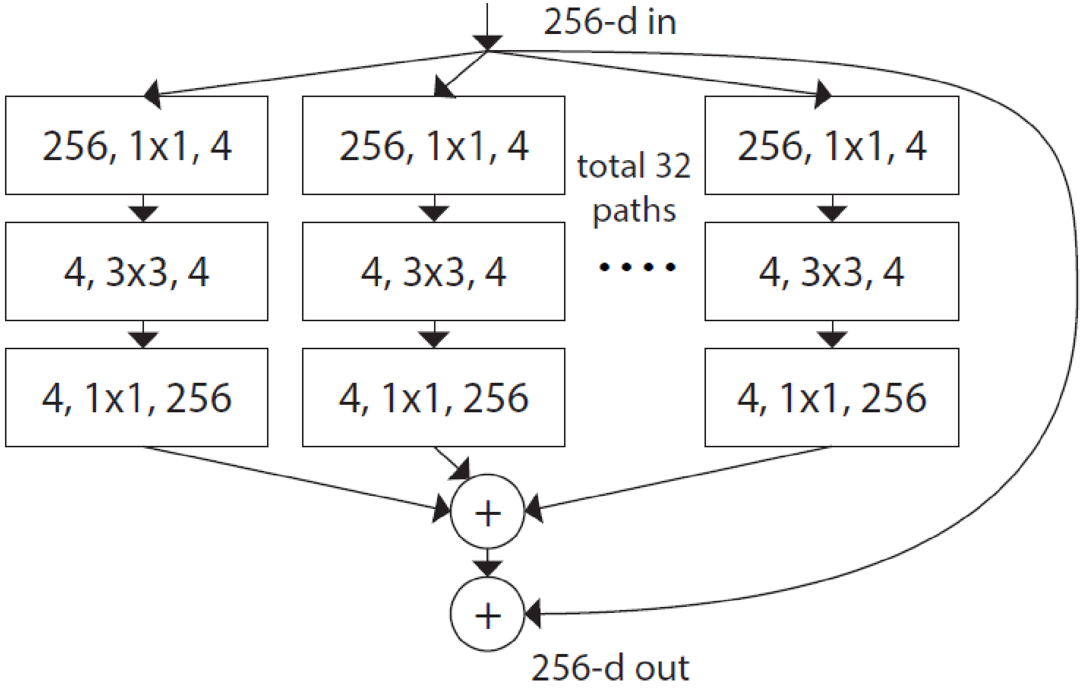
depth	k	dropout	CIFAR-10	CIFAR-100	SVHN
16	4		5.02	24.03	1.85
16	4	✓	5.24	23.91	1.64
28	10		4.00	19.25	-
28	10	✓	**3.89**	**18.85**	-
52	1		6.43	29.89	2.08
52	1	✓	6.28	29.78	1.70

ResNeXt





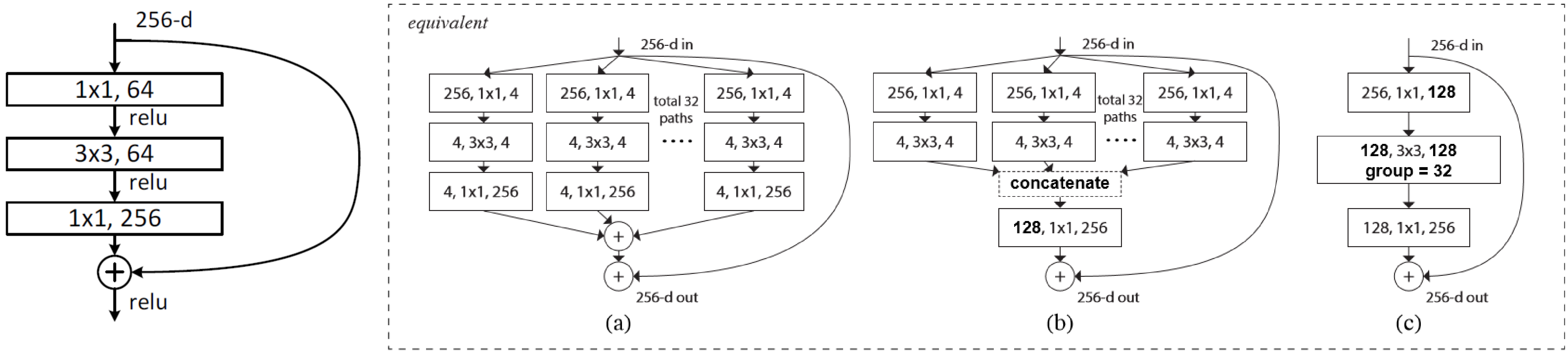
ResNet



ResNeXt

ResNeXt(cont.)

- Cardinality
- Parallel pathways similar in spirit to Inception module



ResNeXt(cont.)

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.		**25.5×10^6**	**25.0×10^6**
FLOPs		**4.1×10^9**	**4.2×10^9**

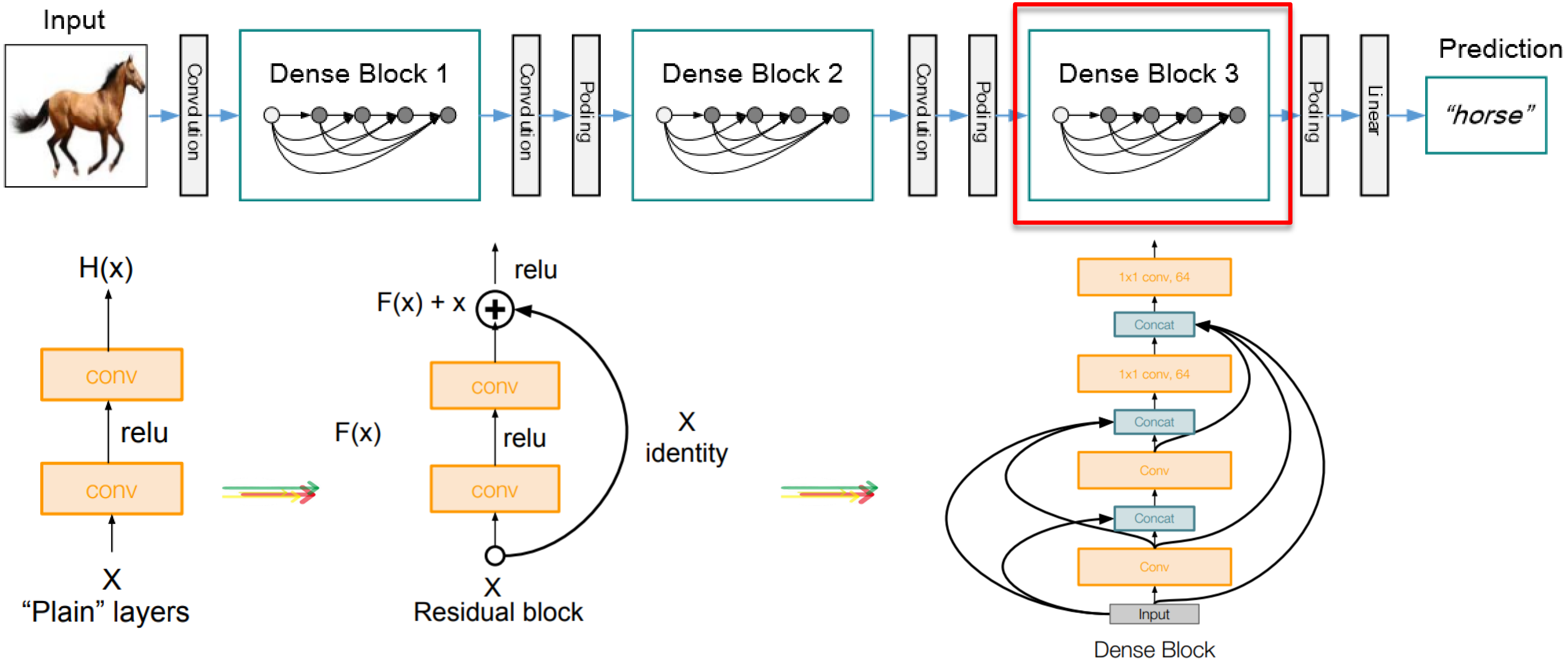
	224×224		320×320 / 299×299	
	top-1 err	top-5 err	top-1 err	top-5 err
ResNet-101 [14]	22.0	6.0	-	-
ResNet-200 [15]	21.7	5.8	20.1	4.8
Inception-v3 [39]	-	-	21.2	5.6
Inception-v4 [37]	-	-	20.0	5.0
Inception-ResNet-v2 [37]	-	-	19.9	4.9
ResNeXt-101 (**64 × 4d**)	20.4	5.3	**19.1**	**4.4**

	# params	CIFAR-10	CIFAR-100
Wide ResNet [43]	36.5M	4.17	20.50
ResNeXt-29, 8×64d	34.4M	3.65	17.77
ResNeXt-29, 16×64d	68.1M	**3.58**	**17.31**

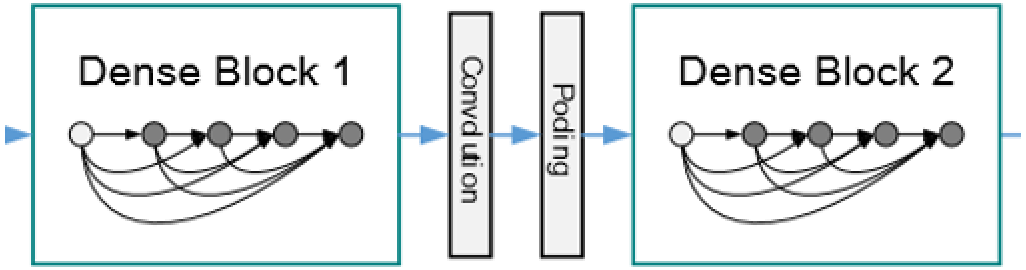
Other advanced Networks

- Densely Connected Convolutional Networks(DenseNet)
- Dual Path Networks(DPN)
- Squeeze-and-Excitation Networks(SENet)

DenseNet


Input
Convolution
Dense Block 1
Convolution
Pooling
Dense Block 2
Convolution
Pooling
Dense Block 3
Pooling
Linear
Prediction
"horse"
H(x)
conv
relu
conv
X
"Plain" layers
relu
F(x) + x
conv
F(x)
relu
conv
X
identity
X
Residual block
1x1 conv, 64
Concat
1x1 conv, 64
Concat
Conv
Concat
Conv
Input
Dense Block

DenseNet(cont.)



- Denseblock
- Transition layer
  - BN-ReLU-1x1conv-ave pooling
- Bottleneck layers
  - BN-ReLU-Conv(1x1)-BN-ReLU-Conv(3x3)
  - Called **DenseNet-B**
- Compression
  - If a dense block contains $\boldsymbol{m}$ feature-maps, we let the following transition layer generate $\boldsymbol{\theta} * \boldsymbol{m}$ output feature maps. $(0 < \theta \leq 1)$
  - Called **DenseNet-C**
- **DenseNet-BC**
  - Used bottleneck layer and compression

- DenseNet architectures for ImageNet.

Layers	Output Size	DenseNet-121 $(k = 32)$	DenseNet-169 $(k = 32)$	DenseNet-201 $(k = 32)$	DenseNet-161 $(k = 48)$
Convolution	112×112	7×7 conv, stride 2			
Pooling	56×56	3×3 max pool, stride 2			
Dense Block (1)	56×56	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$
Transition Layer (1)	56×56	1×1 conv			
	28×28	2×2 average pool, stride 2			
Dense Block (2)	28×28	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$
Transition Layer (2)	28×28	1×1 conv			
	14×14	2×2 average pool, stride 2			
Dense Block (3)	14×14	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 24$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 48$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 36$
Transition Layer (3)	14×14	1×1 conv			
	7×7	2×2 average pool, stride 2			
Dense Block (4)	7×7	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 16$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 24$
Classification Layer	1×1	7×7 global average pool			
		1000D fully-connected, softmax			

Table 1. DenseNet architectures for ImageNet. The growth rate for the first 3 networks is $k = 32$, and $k = 48$ for DenseNet-161. Note that each "conv" layer shown in the table corresponds the sequence BN-ReLU-Conv.

DenseNet(cont.)

- **Depth=40, Growthrate=12, Bottleneck=True, Compression =0.5**
- **CONV = BN-ReLU-conv**

Input：**(32,32,3)**

conv(24,3,3) **(32,32,24)**

(The first dense block)

CONV(48,1,1)-CONV(12,3,3)-merge(36) % after merge, nb_filter=24+12=36

CONV(48,1,1)-CONV(12,3,3)-merge(48) % after merge, nb_filter=36+12=48

CONV(48,1,1)-CONV(12,3,3)-merge(60)

CONV(48,1,1)-CONV(12,3,3)-merge(72)

CONV(48,1,1)-CONV(12,3,3)-merge(84)

CONV(48,1,1)-CONV(12,3,3)-merge(96) % we have 6 layers in this block, so the output nb_filter=24+72=96 **(32,32,96)**

(The first Transition layer)

CONV(48,1,1) % nb_filter=nb_filter*compression=96*0.5=48

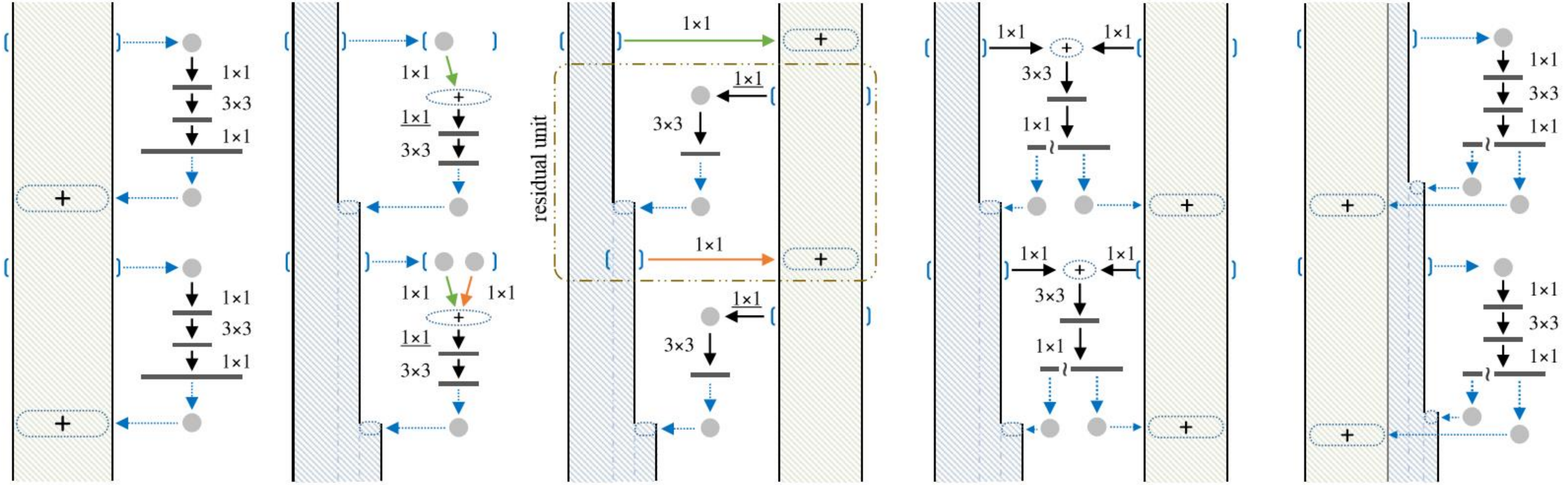
AveragePool(2,2,(2,2)) % pool_size=2,2 strides=(2,2) **(16,16,48)**

- Error rates (%) on CIFAR and SVHN datasets.

Method	Depth	Params	C10	C10+	C100	C100+	SVHN
Network in Network [22]	-	-	10.41	8.81	35.68	-	2.35
All-CNN [31]	-	-	9.08	7.25	-	33.71	-
Deeply Supervised Net [20]	-	-	9.69	7.97	-	34.57	1.92
Highway Network [33]	-	-	-	7.72	-	32.39	-
FractalNet [17]	21	38.6M	10.18	5.22	35.34	23.30	2.01
with Dropout/Drop-path	21	38.6M	7.33	4.60	28.20	23.73	1.87
ResNet [11]	110	1.7M	-	6.61	-	-	-
ResNet (reported by [13])	110	1.7M	13.63	6.41	44.74	27.22	2.01
ResNet with Stochastic Depth [13]	110	1.7M	11.66	5.23	37.80	24.58	1.75
	1202	10.2M	-	4.91	-	-	-
Wide ResNet [41]	16	11.0M	-	4.81	-	22.07	-
	28	36.5M	-	4.17	-	20.50	-
with Dropout	16	2.7M	-	-	-	-	1.64
ResNet (pre-activation) [12]	164	1.7M	11.26*	5.46	35.58*	24.33	-
	1001	10.2M	10.56*	4.62	33.47*	22.71	-
DenseNet ($k = 12$)	40	1.0M	**7.00**	5.24	**27.55**	24.42	1.79
DenseNet ($k = 12$)	100	7.0M	**5.77**	**4.10**	**23.79**	**20.20**	1.67
DenseNet ($k = 24$)	100	27.2M	**5.83**	**3.74**	**23.42**	**19.25**	**1.59**
DenseNet-BC ($k = 12$)	100	0.8M	**5.92**	4.51	**24.15**	22.27	1.76
DenseNet-BC ($k = 24$)	250	15.3M	**5.19**	**3.62**	**19.64**	**17.60**	1.74
DenseNet-BC ($k = 40$)	190	25.6M	-	**3.46**	-	**17.18**	-

Dual Path Network

- The **first** places in ImageNet ILSVRC challenge **2017** object localization tasks
- ResNet + DenseNet



(a) Residual Network (b) Densely Connected Network (c) Densely Connected Network (with shared connections) (d) Dual Path Architecture (e) DPN

- DPN architectures for ImageNet.

stage	output	DenseNet-161 (k=48)	ResNeXt-101 (32×4d)	ResNeXt-101 (64×4d)	DPN-92 (32×3d)	DPN-98 (40×4d)
conv1	112x112	7 × 7, 96, stride 2	7 × 7, 64, stride 2	7 × 7, 64, stride 2	7 × 7, 64, stride 2	7 × 7, 96, stride 2
conv2	56x56	3 × 3 max pool, stride 2	3 × 3 max pool, stride 2	3 × 3 max pool, stride 2	3 × 3 max pool, stride 2	3 × 3 max pool, stride 2
		$\begin{bmatrix} 1\times1, 192 \\ 3\times3, 48 \end{bmatrix} \times 6$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128, G=32 \\ 1\times1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256, G=64 \\ 1\times1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 96 \\ 3\times3, 96, G=32 \\ 1\times1, 256\ (+16) \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 160 \\ 3\times3, 160, G=40 \\ 1\times1, 256\ (+16) \end{bmatrix} \times 3$
conv3	28×28	$\begin{bmatrix} 1\times1, 192 \\ 3\times3, 48 \end{bmatrix} \times 12$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256, G=32 \\ 1\times1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512, G=64 \\ 1\times1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1\times1, 192 \\ 3\times3, 192, G=32 \\ 1\times1, 512\ (+32) \end{bmatrix} \times 4$	$\begin{bmatrix} 1\times1, 320 \\ 3\times3, 320, G=40 \\ 1\times1, 512\ (+32) \end{bmatrix} \times 6$
conv4	14×14	$\begin{bmatrix} 1\times1, 192 \\ 3\times3, 48 \end{bmatrix} \times 36$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512, G=32 \\ 1\times1, 1024 \end{bmatrix} \times 23$	$\begin{bmatrix} 1\times1, 1024 \\ 3\times3, 1024, G=64 \\ 1\times1, 1024 \end{bmatrix} \times 23$	$\begin{bmatrix} 1\times1, 384 \\ 3\times3, 384, G=32 \\ 1\times1, 1024\ (+24) \end{bmatrix} \times 20$	$\begin{bmatrix} 1\times1, 640 \\ 3\times3, 640, G=40 \\ 1\times1, 1024\ (+32) \end{bmatrix} \times 20$
conv5	7×7	$\begin{bmatrix} 1\times1, 192 \\ 3\times3, 48 \end{bmatrix} \times 24$	$\begin{bmatrix} 1\times1, 1024 \\ 3\times3, 1024, G=32 \\ 1\times1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 2048 \\ 3\times3, 2048, G=64 \\ 1\times1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 768 \\ 3\times3, 768, G=32 \\ 1\times1, 2048\ (+128) \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 1280 \\ 3\times3, 1280, G=40 \\ 1\times1, 2048\ (+128) \end{bmatrix} \times 3$
	1×1	global average pool 1000-d fc, softmax	global average pool 1000-d fc, softmax	global average pool 1000-d fc, softmax	global average pool 1000-d fc, softmax	global average pool 1000-d fc, softmax
# params		$\mathbf{28.9} \times 10^6$	$\mathbf{44.3} \times 10^6$	$\mathbf{83.7} \times 10^6$	$\mathbf{37.8} \times 10^6$	$\mathbf{61.7} \times 10^6$
FLOPs		$\mathbf{7.7} \times 10^9$	$\mathbf{8.0} \times 10^9$	$\mathbf{15.5} \times 10^9$	$\mathbf{6.5} \times 10^9$	$\mathbf{11.7} \times 10^9$

Table 2: Comparison with state-of-the-art CNNs on ImageNet-1k dataset. Single crop validation error rate (%) on validation set. *: Performance reported by [21], †: With Mean-Max Pooling (see Appendix A).

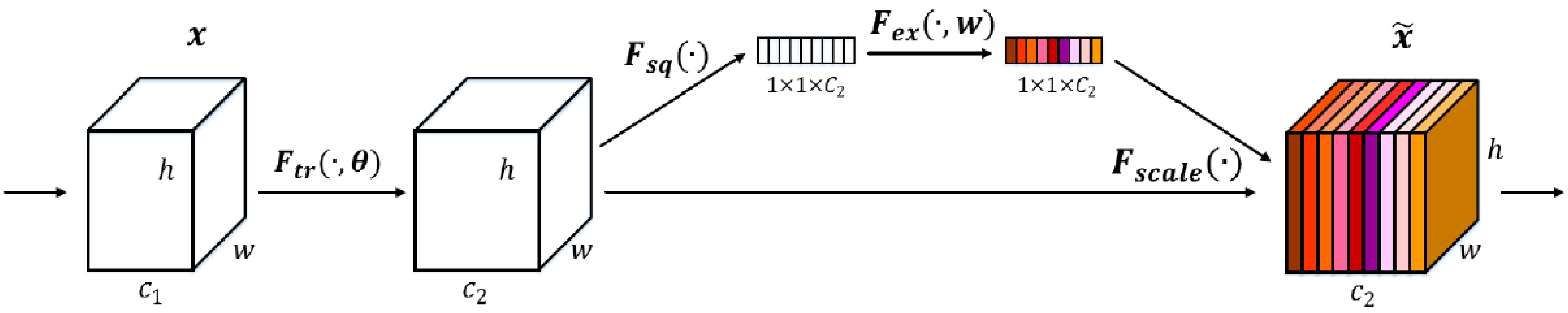
Method	Model Size	GFLOPs	x224 top-1	x224 top-5	x320 / x299 top-1	x320 / x299 top-5
DenseNet-161(k=48) [8]	111 MB	7.7	22.2	–	–	–
ResNet-101* [5]	170 MB	7.8	22.0	6.0	–	–
ResNeXt-101 (32 × 4d) [21]	170 MB	8.0	21.2	5.6	–	–
DPN-92 (32 × 3d)	**145 MB**	**6.5**	**20.7**	**5.4**	**19.3**	**4.7**
ResNet-200 [6]	247 MB	15.0	21.7	5.8	20.1	4.8
Inception-resnet-v2 [20]	227 MB	–	–	–	19.9	4.9
ResNeXt-101 (64 × 4d) [21]	320 MB	15.5	20.4	5.3	19.1	4.4
DPN-98 (40 × 4d)	**236 MB**	**11.7**	**20.2**	**5.2**	**18.9**	**4.4**
Very deep Inception-resnet-v2 [23]	531 MB	–	–	–	19.10	4.48
Very Deep PolyNet [23]	365 MB	–	–	–	18.71	4.25
DPN-131 (40 × 4d)	304 MB	16.0	19.93	5.12	18.62	4.23
DPN-131 (40 × 4d) †	**304 MB**	**16.0**	**19.93**	**5.12**	**18.55**	**4.16**

Table 3: Comparison with state-of-the-art CNNs on Places365-Standard dataset. 10 crops validation accuracy rate (%) on validation set.

Method	Model Size	top-1 acc.	top-5 acc.
AlexNet [24]	223 MB	53.17	82.89
GoogleLeNet [24]	44 MB	53.63	83.88
VGG-16 [24]	518 MB	55.24	84.91
ResNet-152 [24]	226 MB	54.74	85.08
ResNeXt-101 [3]	165 MB	56.21	86.25
CRU-Net-116 [3]	163 MB	56.60	86.55
DPN-92 (32 × 3d)	**138 MB**	**56.84**	**86.69**

Squeeze-and-Excitation Networks(SENet)

- The **first** places in ImageNet ILSVRC challenge **2017** classification tasks



Squeeze

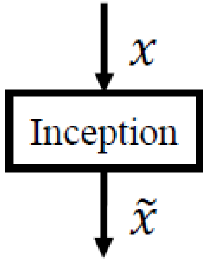
- Shrinking feature maps $\in \mathbb{R}^{w \times h \times c_2}$ through spatial dimensions $(w \times h)$
- Global distribution of channel-wise responses

Excitation

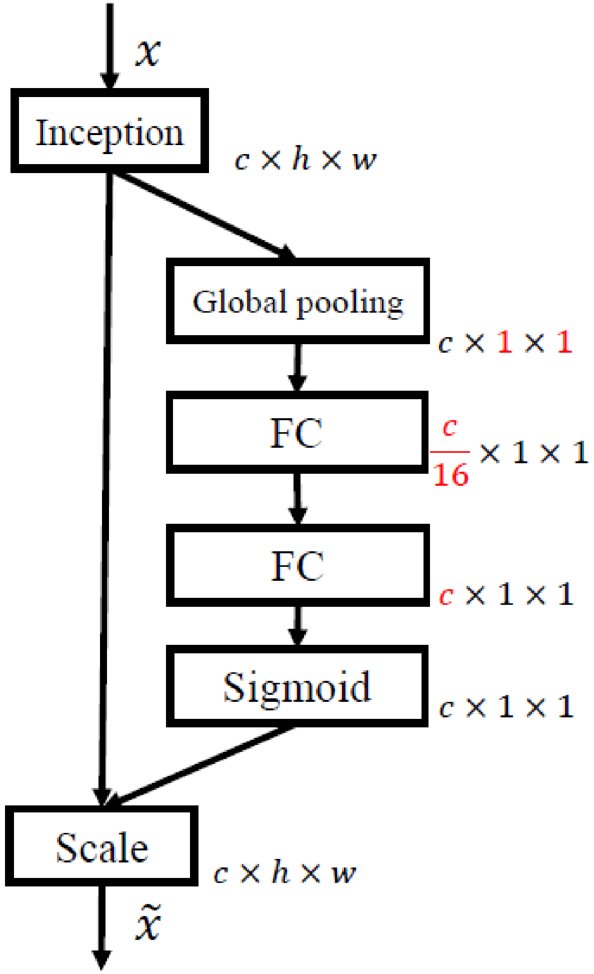
- Learning $W \in \mathbb{R}^{c_2 \times c_2}$ to explicitly model channel-association
- Gating mechanism to produce channel-wise weights

Scale

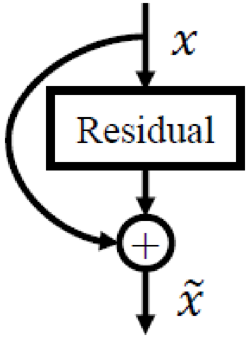
- Reweighting the feature maps $\in \mathbb{R}^{w \times h \times c_2}$



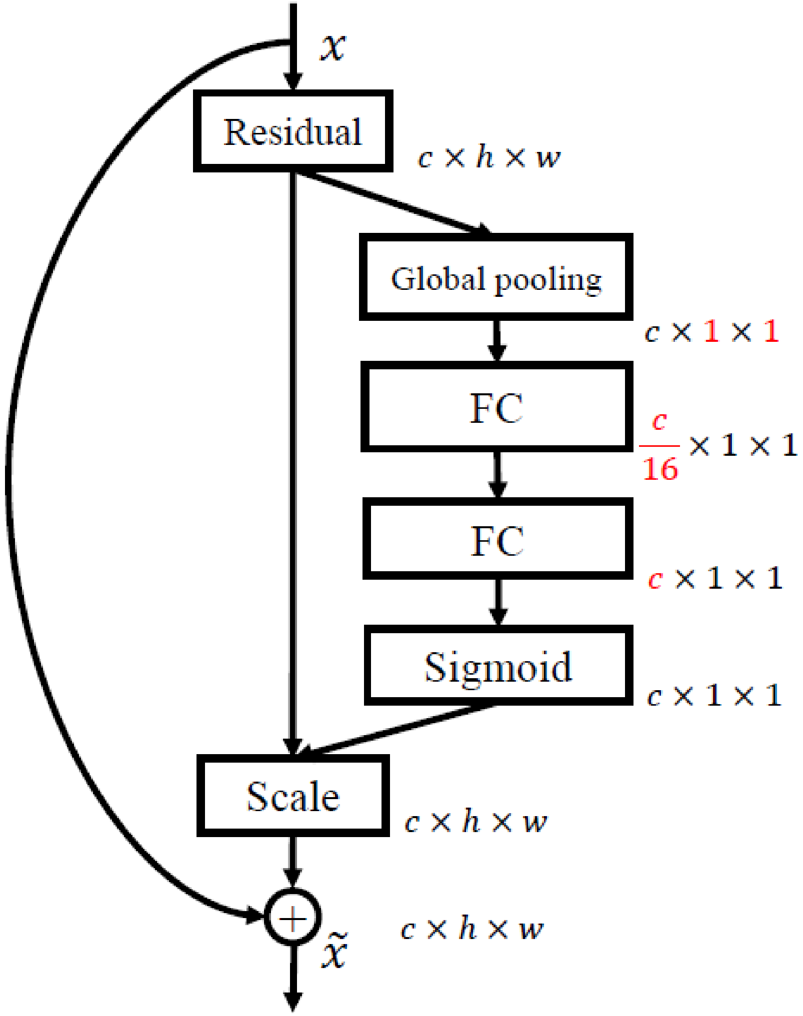
Inception Module



SE-Inception Module



ResNet Module



SE-ResNet Module

- SENet architectures for ImageNet.

Output size	ResNet-50	SE-ResNet-50	SE-ResNeXt-50 (32×4d)
112×112	$conv$, 7×7, 64, stride 2		
56×56	$max\ pool$, 3×3, stride 2		
	$\begin{bmatrix} conv, 1\times1, 64 \\ conv, 3\times3, 64 \\ conv, 1\times1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} conv, 1\times1, 64 \\ conv, 3\times3, 64 \\ conv, 1\times1, 256 \\ fc, [16, 256] \end{bmatrix} \times 3$	$\begin{bmatrix} conv, 1\times1, 128 & \\ conv, 3\times3, 128 & C=32 \\ conv, 1\times1, 256 & \\ fc, [16, 256] & \end{bmatrix} \times 3$
28×28	$\begin{bmatrix} conv, 1\times1, 128 \\ conv, 3\times3, 128 \\ conv, 1\times1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} conv, 1\times1, 128 \\ conv, 3\times3, 128 \\ conv, 1\times1, 512 \\ fc, [32, 512] \end{bmatrix} \times 4$	$\begin{bmatrix} conv, 1\times1, 256 & \\ conv, 3\times3, 256 & C=32 \\ conv, 1\times1, 512 & \\ fc, [32, 512] & \end{bmatrix} \times 4$
14×14	$\begin{bmatrix} conv, 1\times1, 256 \\ conv, 3\times3, 256 \\ conv, 1\times1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} conv, 1\times1, 256 \\ conv, 3\times3, 256 \\ conv, 1\times1, 1024 \\ fc, [64, 1024] \end{bmatrix} \times 6$	$\begin{bmatrix} conv, 1\times1, 512 & \\ conv, 3\times3, 512 & C=32 \\ conv, 1\times1, 1024 & \\ fc, [64, 1024] & \end{bmatrix} \times 6$
7×7	$\begin{bmatrix} conv, 1\times1, 512 \\ conv, 3\times3, 512 \\ conv, 1\times1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} conv, 1\times1, 512 \\ conv, 3\times3, 512 \\ conv, 1\times1, 2048 \\ fc, [128, 2048] \end{bmatrix} \times 3$	$\begin{bmatrix} conv, 1\times1, 1024 & \\ conv, 3\times3, 1024 & C=32 \\ conv, 1\times1, 2048 & \\ fc, [128, 2048] & \end{bmatrix} \times 3$
1×1	$global\ average\ pool$, 1000-d fc, $softmax$		

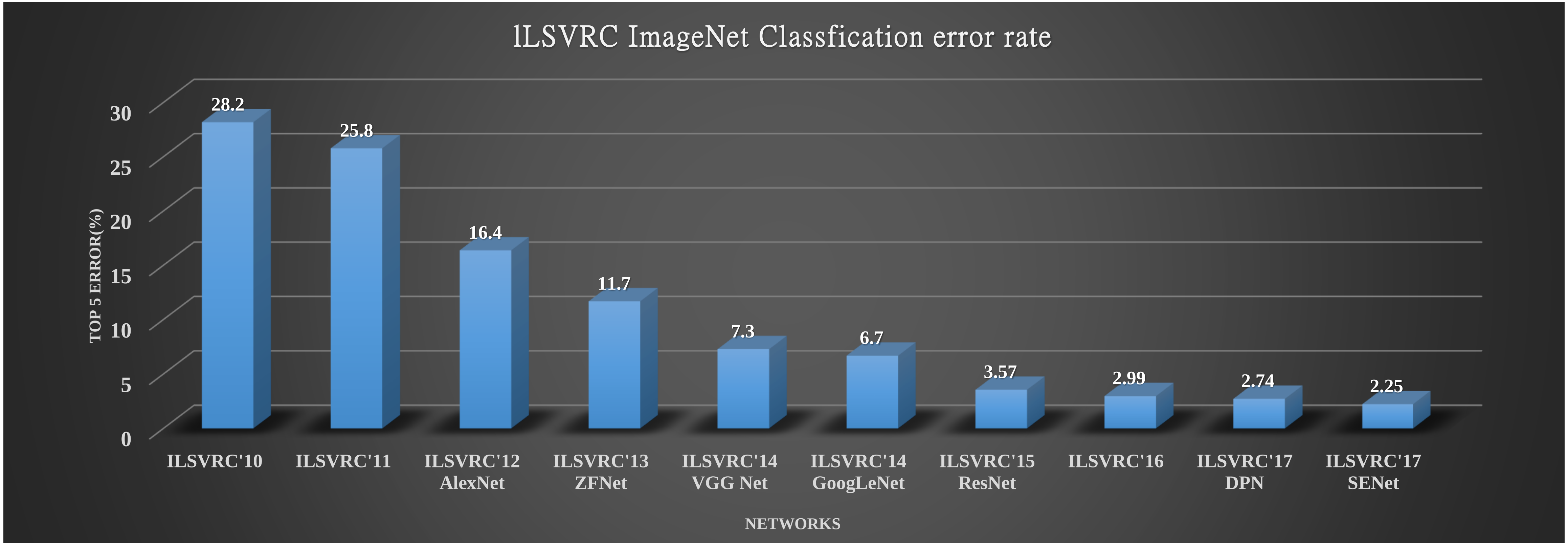
Table 1. (**Left**) ResNet-50. (**Middle**) SE-ResNet-50. (**Right**) SE-ResNeXt-50 with a 32×4d template. The shapes and operations with specific parameter settings of a residual building block are listed inside the brackets and the number of stacked blocks in a stage is presented outside. The inner brackets following by *fc* indicates the output dimension of the two fully connected layers in a SE-module.

- Error rates (%) on ImageNet datasets.

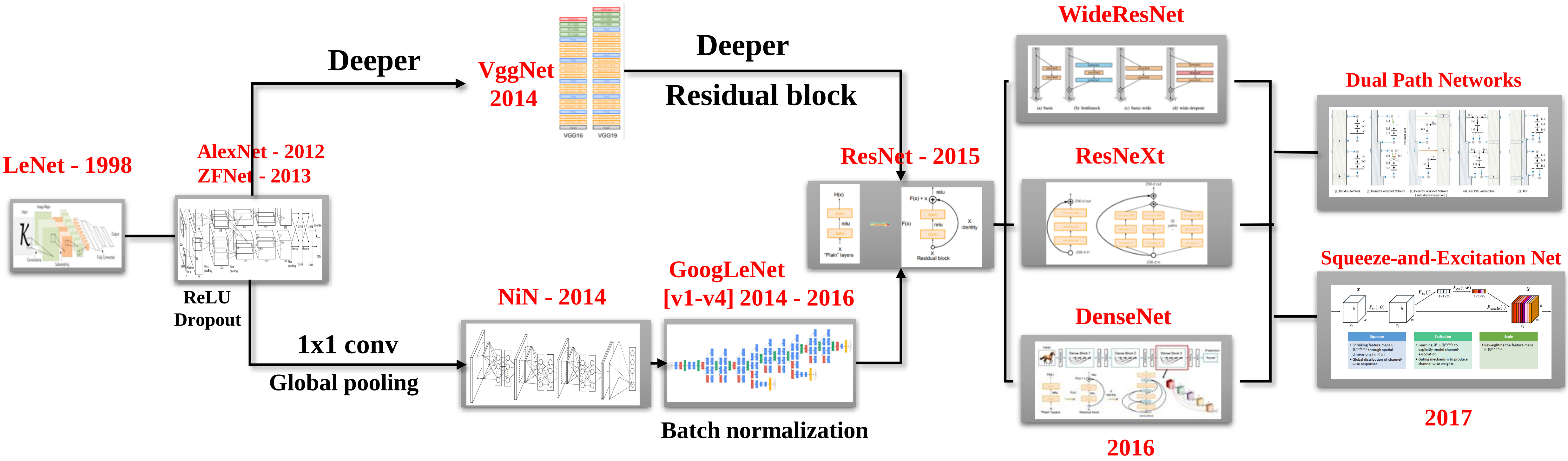
	224 × 224		320 × 320 / 299 × 299	
	top-1 err.	top-5 err.	top-1 err.	top-5 err.
ResNet-152 [9]	23.0	6.7	21.3	5.5
ResNet-200 [10]	21.7	5.8	20.1	4.8
Inception-v3 [40]	-	-	21.2	5.6
Inception-v4 [38]	-	-	20.0	5.0
Inception-ResNet-v2 [38]	-	-	19.9	4.9
ResNeXt-101 (64 × 4d) [43]	20.4	5.3	19.1	4.4
DenseNet-161 (k = 48) [12]	22.2	-	-	-
Very Deep PolyNet [47]	-	-	18.71	4.25
DPN-131 [5]	19.93	5.12	18.55	4.16
SENet	**18.68**	**4.47**	**17.28**	**3.79**

	original		re-implementation			SENet		
	top-1 err.	top-5 err.	top-1 err.	top-5 err.	GFLOPs	top-1 err.	top-5 err.	GFLOPs
ResNet-50 [9]	24.7	7.8	24.80	7.48	3.86	$23.29_{(1.51)}$	$6.62_{(0.86)}$	3.87
ResNet-101 [9]	23.6	7.1	23.17	6.52	7.58	$22.38_{(0.79)}$	$6.07_{(0.45)}$	7.60
ResNet-152 [9]	23.0	6.7	22.42	6.34	11.30	$21.57_{(0.85)}$	$5.73_{(0.61)}$	11.32
ResNeXt-50 [43]	22.2	-	22.11	5.90	4.24	$21.10_{(1.01)}$	$5.49_{(0.41)}$	4.25
ResNeXt-101 [43]	21.2	5.6	21.18	5.57	7.99	$20.70_{(0.48)}$	$5.01_{(0.56)}$	8.00
BN-Inception [14]	25.2	7.82	25.38	7.89	2.03	$24.23_{(1.15)}$	$7.14_{(0.75)}$	2.04
Inception-ResNet-v2 [38]	$19.9^{\dagger}$	$4.9^{\dagger}$	20.37	5.21	11.75	$19.80_{(0.57)}$	$4.79_{(0.42)}$	11.76

Summary of CNN architectures



Summary of CNN architectures(cont.)



Thank you!

Results of my implementation

Accuracy of all my implementations

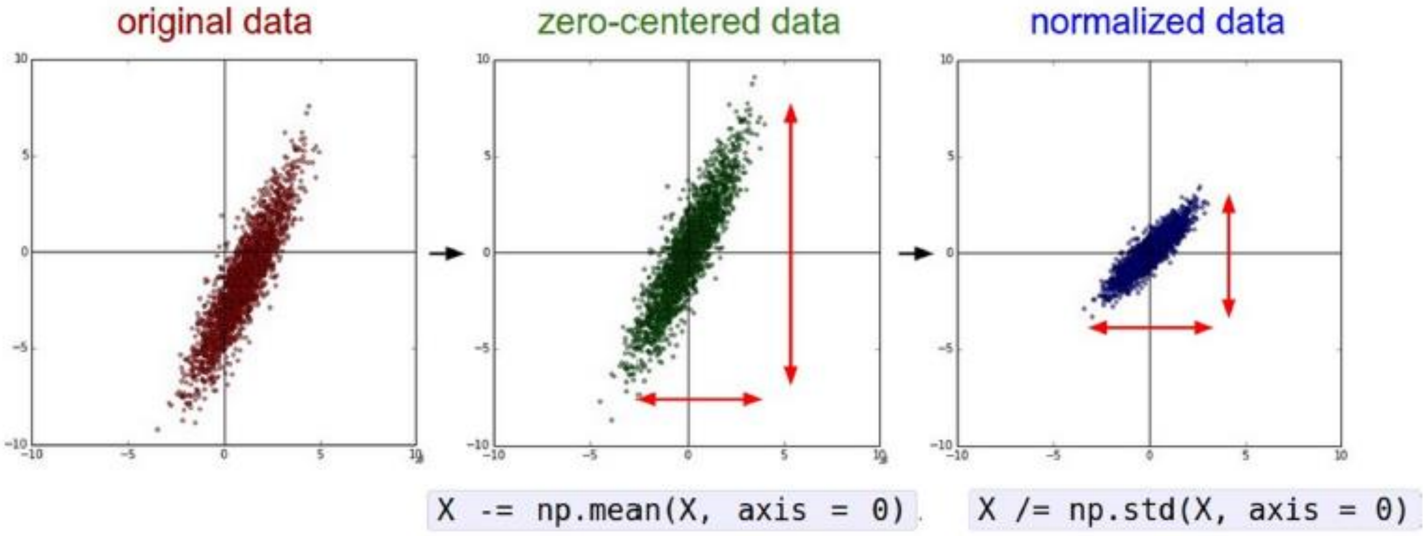
network	dropout	preprocess	GPU	params	training time	accuracy(%)
Lecun-Network	-	meanstd	GTX980TI	62k	30 min	76.27
Network-in-Network	0.5	meanstd	GTX1060	0.96M	1 h 30 min	91.25
Network-in-Network_bn	0.5	meanstd	GTX980TI	0.97M	2 h 20 min	91.75
Vgg19-Network	0.5	meanstd	GTX980TI	45M	4 hours	93.53
Residual-Network50	-	meanstd	GTX980TI	1.7M	8 h 58 min	94.10
Wide-resnet 16x8	-	meanstd	GTX1060	11.3M	11 h 32 min	95.14
ResNeXt-4x64d	-	meanstd	GTX1080TI	20M	22 h 50 min	95.51
DenseNet-100x12	-	meanstd	GTX980TI	0.85M	30 h 40 min	95.15

- **About ResNeXt & DenseNet**
  - Because I don't have enough machines to train the larger networks.
    So I only trained the smallest network described in the paper.
    You can see the results in liuzhuang13/DenseNet and prlz77/ResNeXt.pytorch

Training Tricks

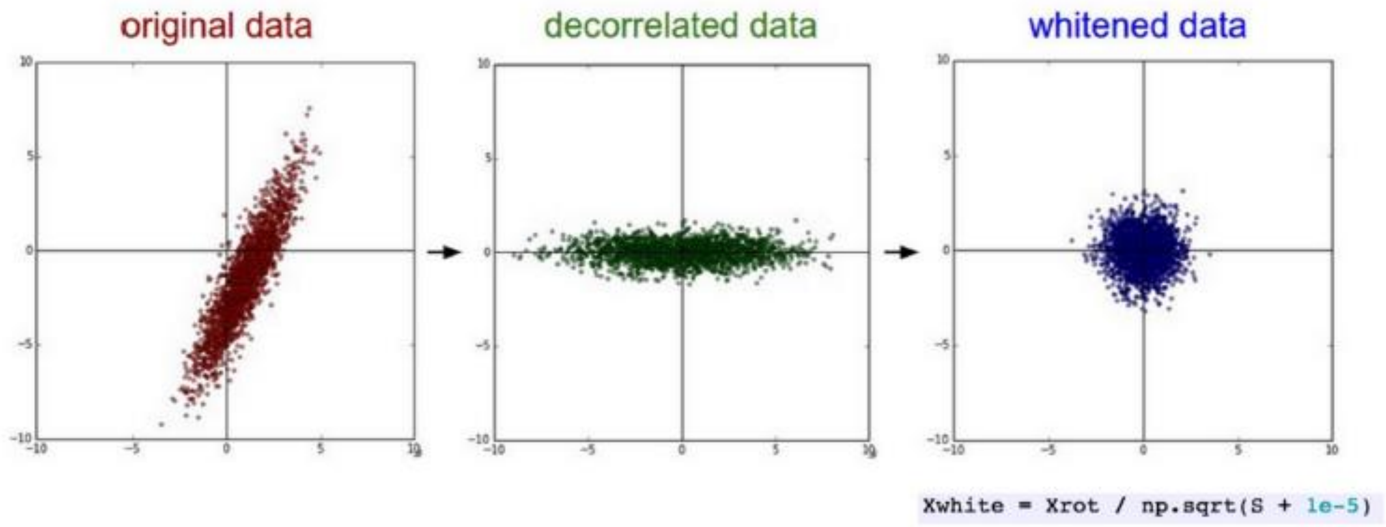
- Pre-Processing
- Data Augmentation
- Regularizations
- Initializations
- Fine-tune

Pre-Processing

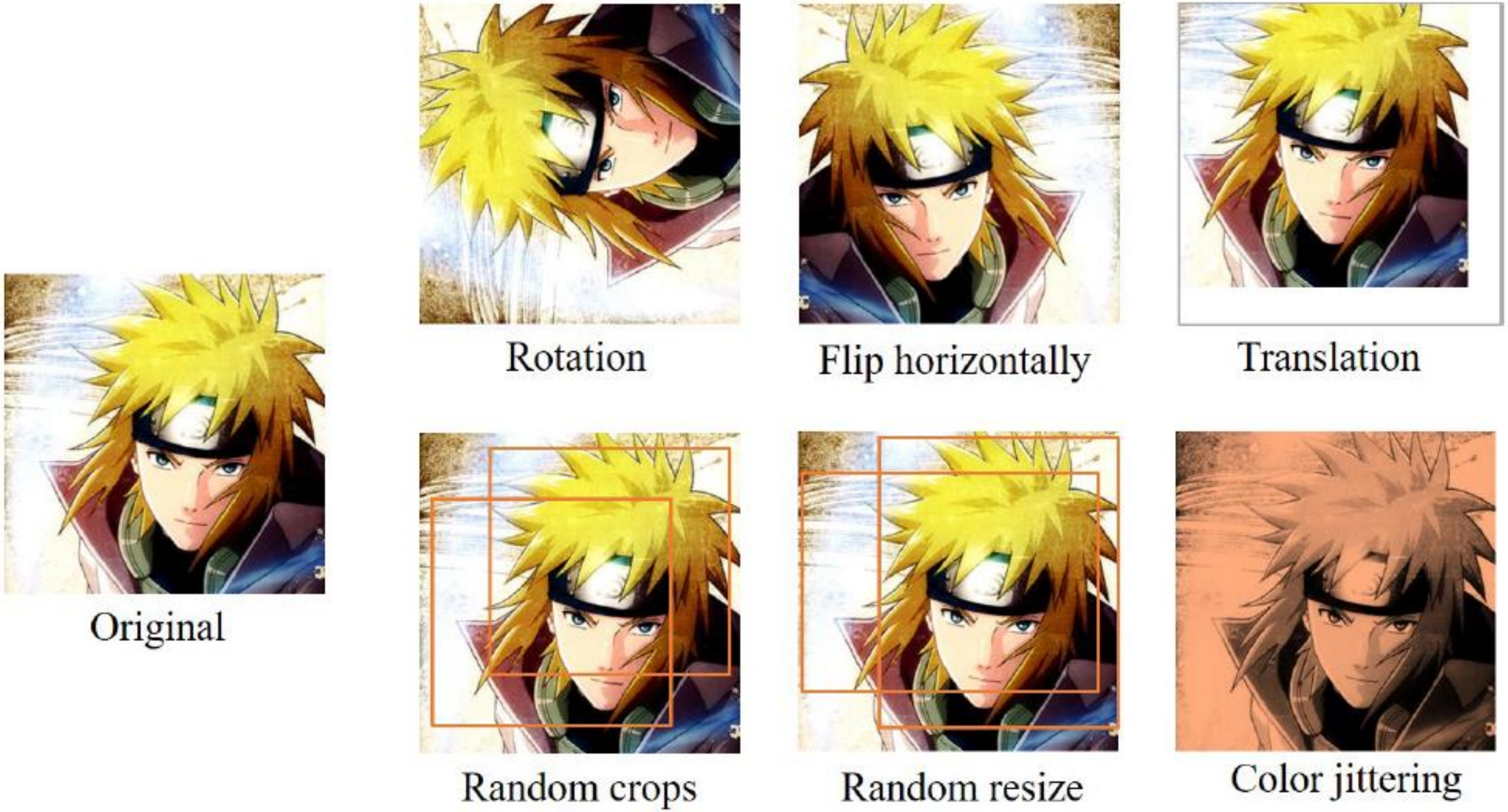


- Mean / Std

PCA Whitening

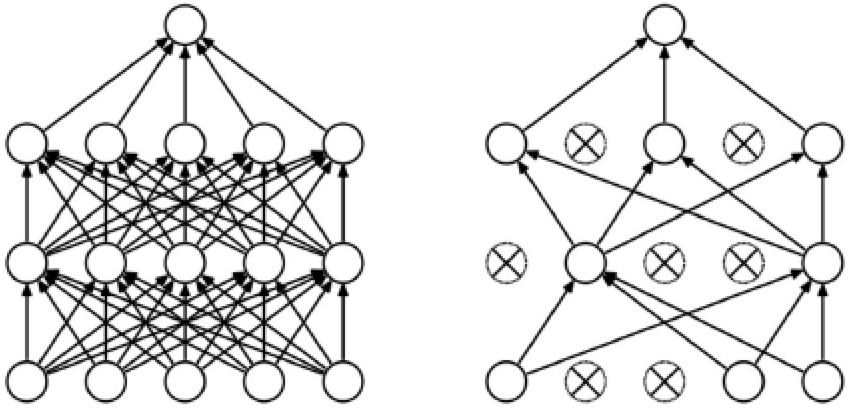


Data Augmentation

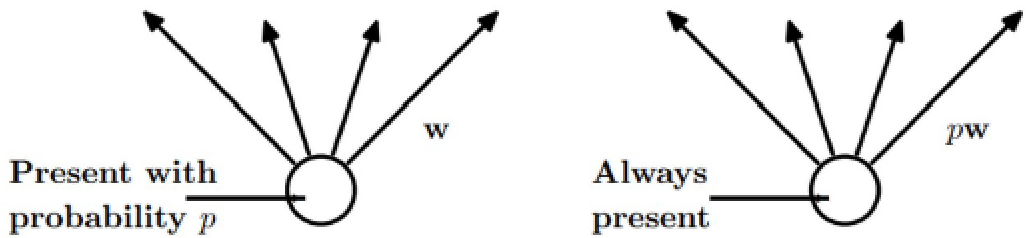


Regularizations

- L1 regularization
- L2 regularization (Weight Decay)
- Dropout



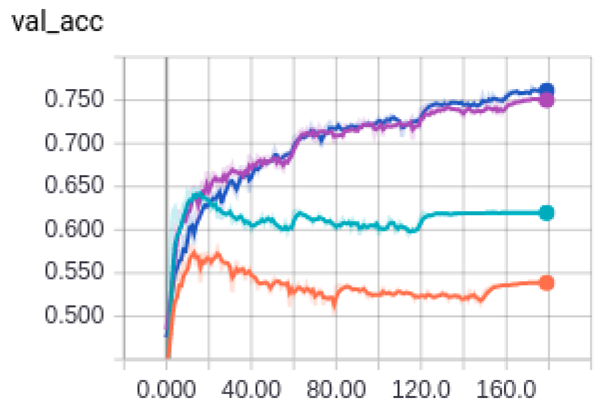
(a) Standard Neural Net (b) After applying dropout.



(a) At training time (b) At test time

Initializations

- Random Normal/Uniform
- He’s Weight Initial **(Kai-Ming He)**
    - Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification [arXiv:1502.01852]



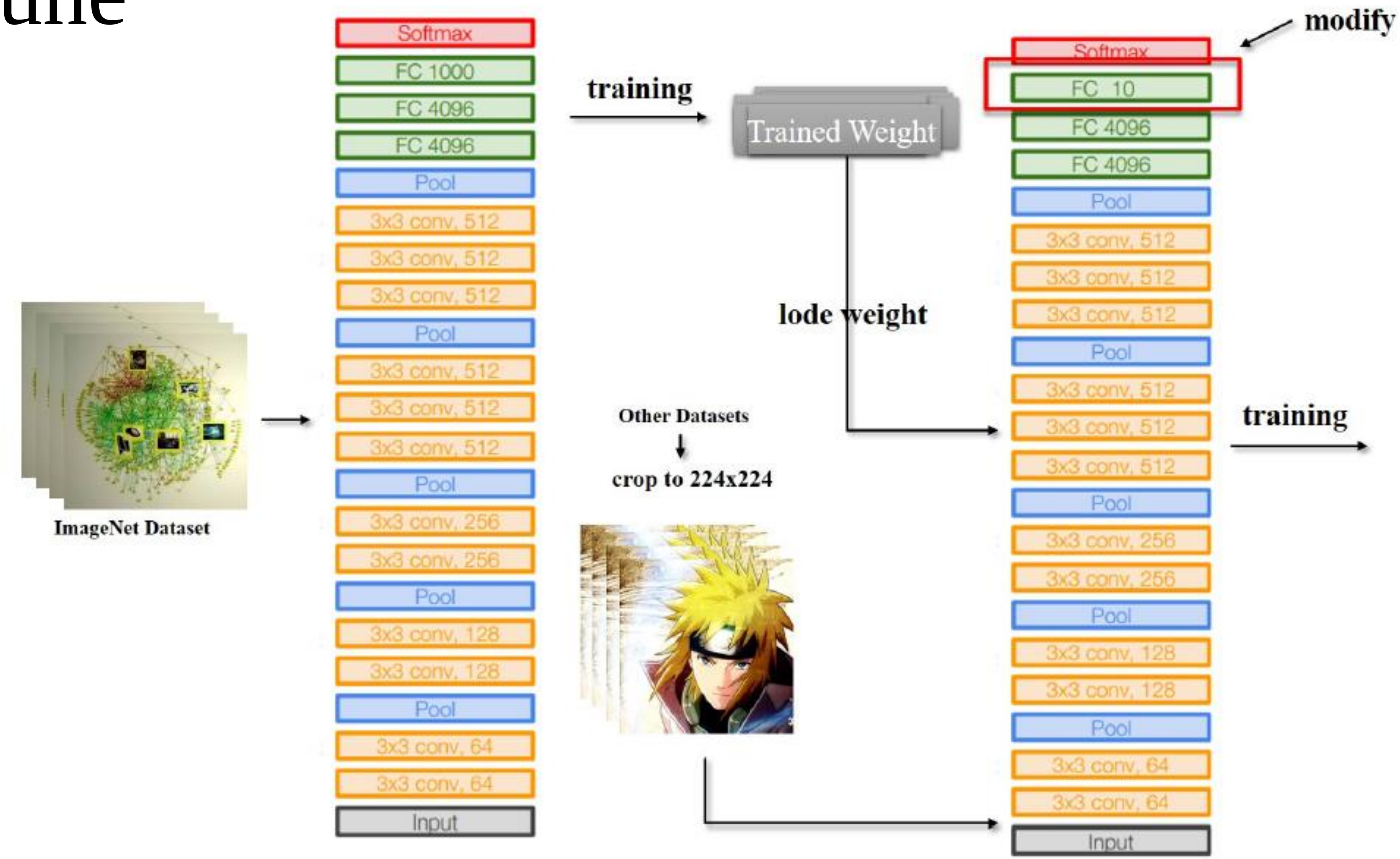
	Name	Smoothed	Value	Step	Time	Relative
	lenet	0.5387	0.5384	179.0	Sat Aug 5, 14:51:07	6m 36s
	lenet_dp	0.6197	0.6198	179.0	Sat Aug 5, 14:52:16	8m 51s
	lenet_dp_da	0.7502	0.7507	179.0	Sat Aug 5, 15:16:49	31m 42s
	lenet_dp_da_wd	0.7611	0.7599	179.0	Sat Aug 5, 15:22:30	30m 31s

Table 6: Test accuracy for Retrain + WI + WD + BN

Retrain+WI+BN	**WD=0.0001**	**92.99%**
Retrain+WI+BN	**WD=0.0005**	**93.74%**
Retrain+WI+BN	**WD=0.0010**	**93.94%**
Retrain+WI+BN	**WD=0.0013**	**93.83%**
Retrain+WI+BN	**WD=0.0015**	**94.14%**

Test accuracy of my Vgg19 on cifar-10 dataset

Fine-tune


Softmax
FC 1000
FC 4096
FC 4096
Pool
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
Pool
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
Pool
3x3 conv, 256
3x3 conv, 256
Pool
3x3 conv, 128
3x3 conv, 128
Pool
3x3 conv, 64
3x3 conv, 64
Input
ImageNet Dataset
training
Trained Weight
lode weight
Other Datasets
crop to 224x224
modify
Softmax
FC 10
FC 4096
FC 4096
Pool
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
Pool
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
Pool
3x3 conv, 256
3x3 conv, 256
Pool
3x3 conv, 128
3x3 conv, 128
Pool
3x3 conv, 64
3x3 conv, 64
Input
training

Other materials

- **CS231n** Lecture 5: Convolutional Neural Networks
- **CS231n** Lecture 9: CNN Architectures
- Deep Learning Gets Way Deeper
  - ICML 2016 tutorial **(Kai-Ming He)**
- Learning Deep Features for Visual Recognition
  - CVPR 2017 tutorial **(Kai-Ming He)**
- Must Know Tips/Tricks in Deep Neural Networks
  - LAMDA Group **(Xiu-Shen Wei)**