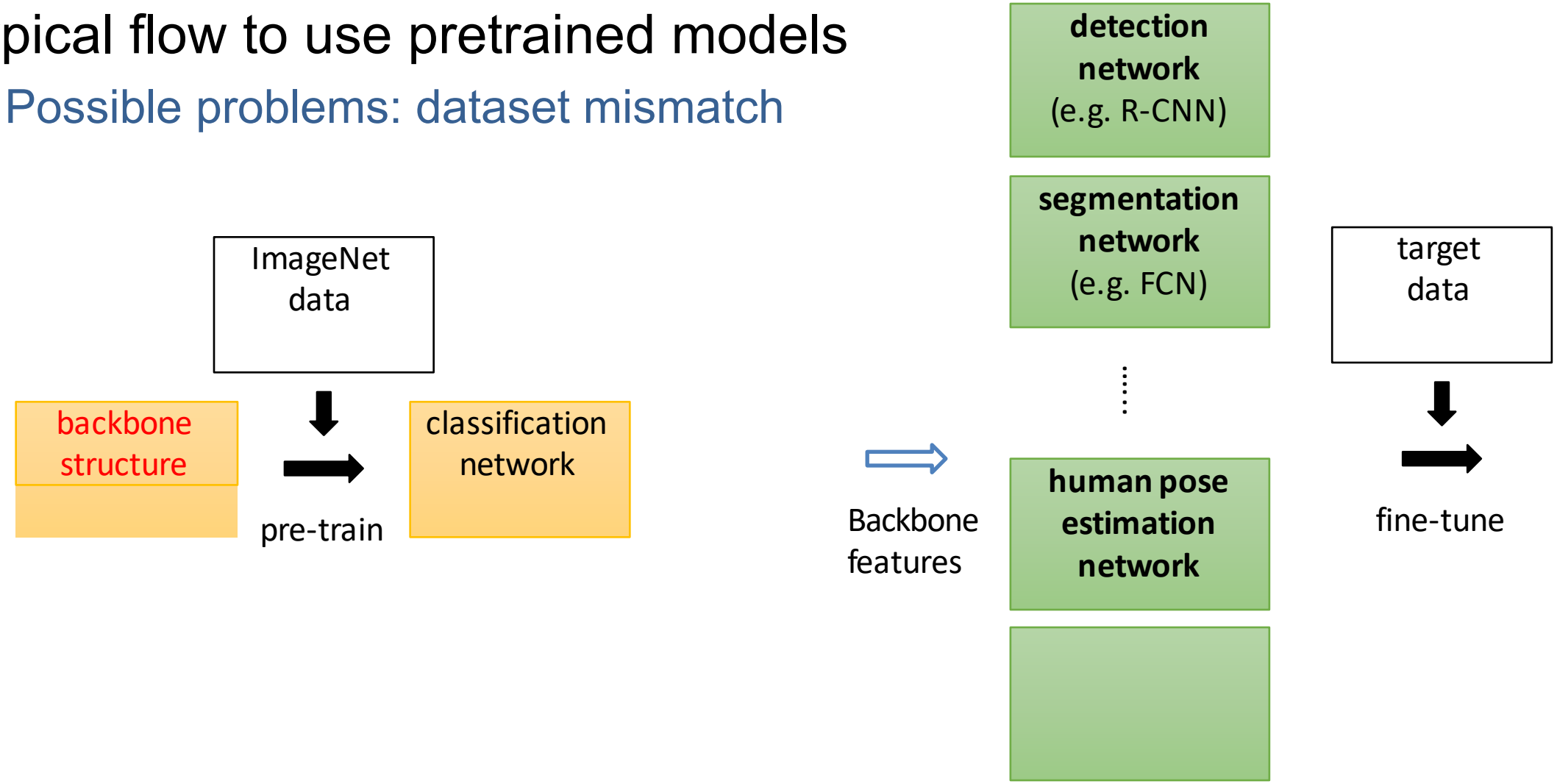
Lecture 5 Advanced CNN Architectures

Tian Sheuan Chang

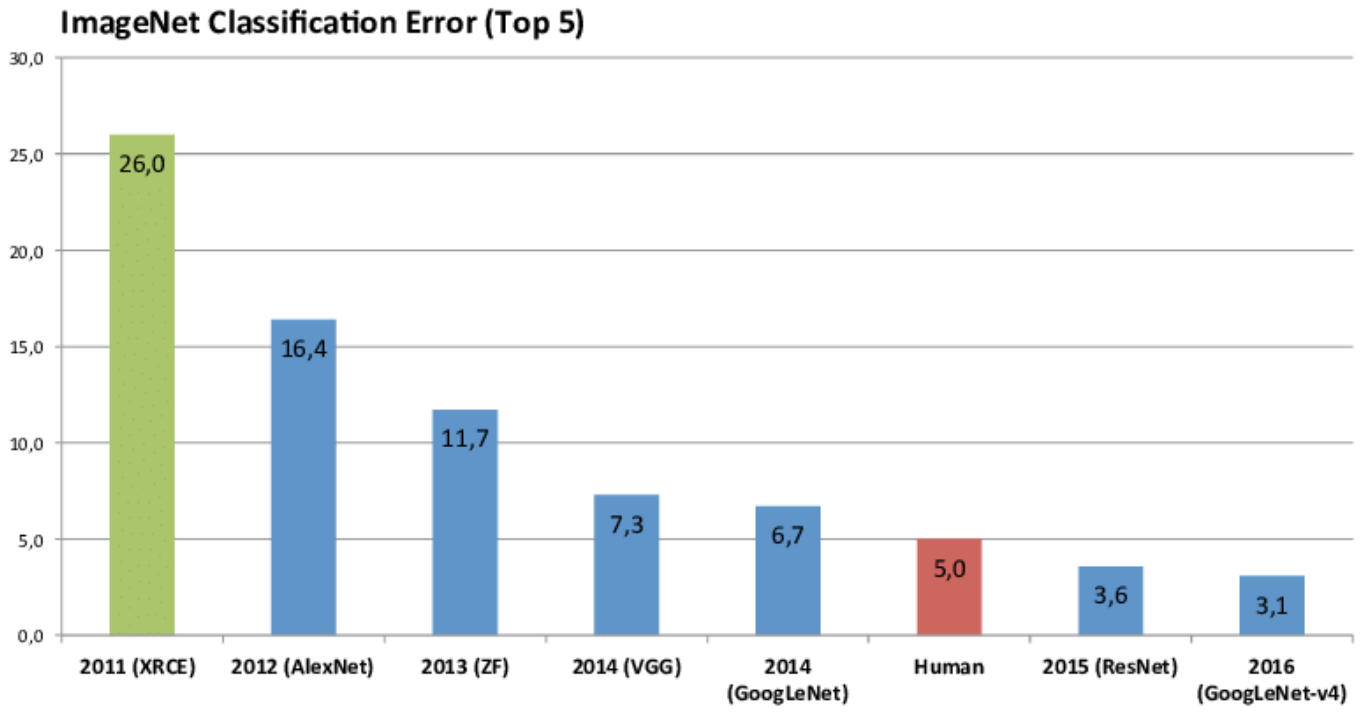
Deep Learning for Computer Vision

- Typical flow to use pretrained models
  - Possible problems: dataset mismatch



Outline

- LeNet
- AlexNet and its derivants
  - AlexNet, ZFNet, VGG
- GoogLeNet
- ResNet and its derivants
- SENet



演進脈絡

- 1990s–2012：可行性驗證到大規模成功
  - LeNet-5：局部感受野 + 權重共享 + 池化；在 MNIST 證明 CNN 可行。
  - AlexNet：靠 GPU 訓練、ReLU、Dropout、Data Aug 與大量數據在 ImageNet 取得決定性勝利。
- 2014–2016：加深、加寬與訊息流（避免退化）=> 準確度
  - VGG：用統一的小卷積（3×3）堆疊；結構規範但參數/計算重。
  - Inception：多尺度分支 (1x1, 3x3, 5x5) + 1×1 降維 提升效率。
  - ResNet：殘差（skip） 解決深度退化，使 100+ 層可訓練。
- 2016–2019：效率化與自動化 => 實際佈署效率
  - Xception/Depthwise Separable：把 3×3 conv 拆成 depthwise + pointwise，大減 FLOPs。
  - MobileNet/ShuffleNet：輕量模型針對edge/real-time。
  - NASNet/EfficientNet：NAS + 複合縮放（depth/width/resolution），效能/效率兼顧。
- 2019–至今：與 Transformer 互補、配方現代化 => 2.0
  - ViT 崛起（長距依賴強），CNN 陣營以 ConvNeXt、RegNet、RepVGG 等吸收 Transformer 訓練配方（例如 LayerNorm、GELU、較大感受野、資料配方），維持高效率競爭力。

模型	核心貢獻	優點	缺點	典型使用情境
LeNet-5 (1998)	卷積+池化奠基	結構簡潔、易教學	僅適合小資料	教學入門、MNIST
AlexNet (2012)	ReLU/Dropout/DataAug + GPU	首次大規模成功	參數龐大、易過擬合	ImageNet 里程碑
VGG (2014)	3×3 小卷積堆疊	結構規範、遷移性強	參數/計算量高	特徵抽取 backbone
Inception (2014–)	多尺度分支 + 1×1	高效、參數更少	模組複雜	伺服器端推論
ResNet (2015–)	殘差連接	易訓練超深網	計算/記憶仍重	通用強基線
DenseNet (2016)	密集連接	參數效率高	記憶/帶寬壓力	特徵重用場景
ResNeXt (2017)	基數（cardinality）	可擴展、表現佳	工程複雜度	高效能伺服器
SE/SK/CBAM (2017–)	通道/空間注意力	提升準確率	輕增成本	多任務骨幹升級
Xception (2017)	極致可分離卷積	效率/表現兼顧	需良好實作	高效主幹
MobileNet/ShuffleNet	深度可分離/通道洗牌	端邊即時	精度較低	手機/嵌入式
EfficientNet (2019)	複合縮放+NAS	SOTA 效率	搜索成本、黑箱	需精確效能/效率
RegNet (2020)	規律化設計空間	穩健、可擴展	需配方	大規模訓練
ConvNeXt (2022)	Conv × Transformer 配方	競爭力回升	配方較多	ViT 對標替代

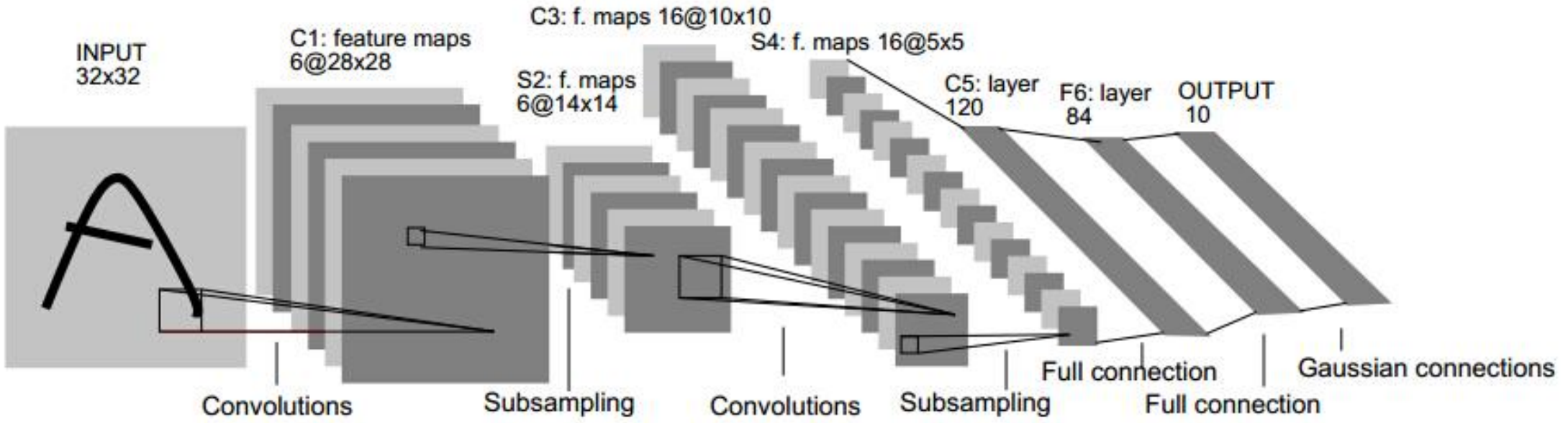
Evolution of CNN for Image Classification

- LeNet
  - 5x5, 2x3 avg pooling, sigmoid/tanh
- AlexNet
  - 3x3, max pooling, ReLU
  - Dropout
- Going for higher accuracy: (Vanished gradient, gradient combination)
  - Diverged kernel size
    - NIN: 1x1, GAP
    - Inception: 1x1, 3x3, 5x5,
    - Extremely large kernel: 31x31, 51x51
  - Deeper
    - VGG: 3x3 stack, vanished gradient
    - Batch normalization
    - ResNet: residual connection
  - Wider

- Parameter efficient network: (How to generate and combine features efficiently)
  - Reuse feature map
    - DenseNet
  - Gradient reuse
    - CSPNet
  - Channel/Kernel decomposition
    - Group convolution
    - mobileNet
  - Attention (space, channel, self attention)
    - SE-Net
    - ResneSt
  - Decouple training and inference
    - RepVGG

LeNet-5 [LeCun et al., 1998]

- 5x5 Conv@stride 1, 2x3 average pooling @stride 2
  - architecture is [CONV-POOL-CONV-POOL-FC-FC]
  - Sigmoid or tanh nonlinearity
- Trained on MNIST digit dataset with 60K training examples



$r_l = (r_{l-1} - 1) \times s_l + k_l$

5x5

10x10

(10-1)*1+5=14

28

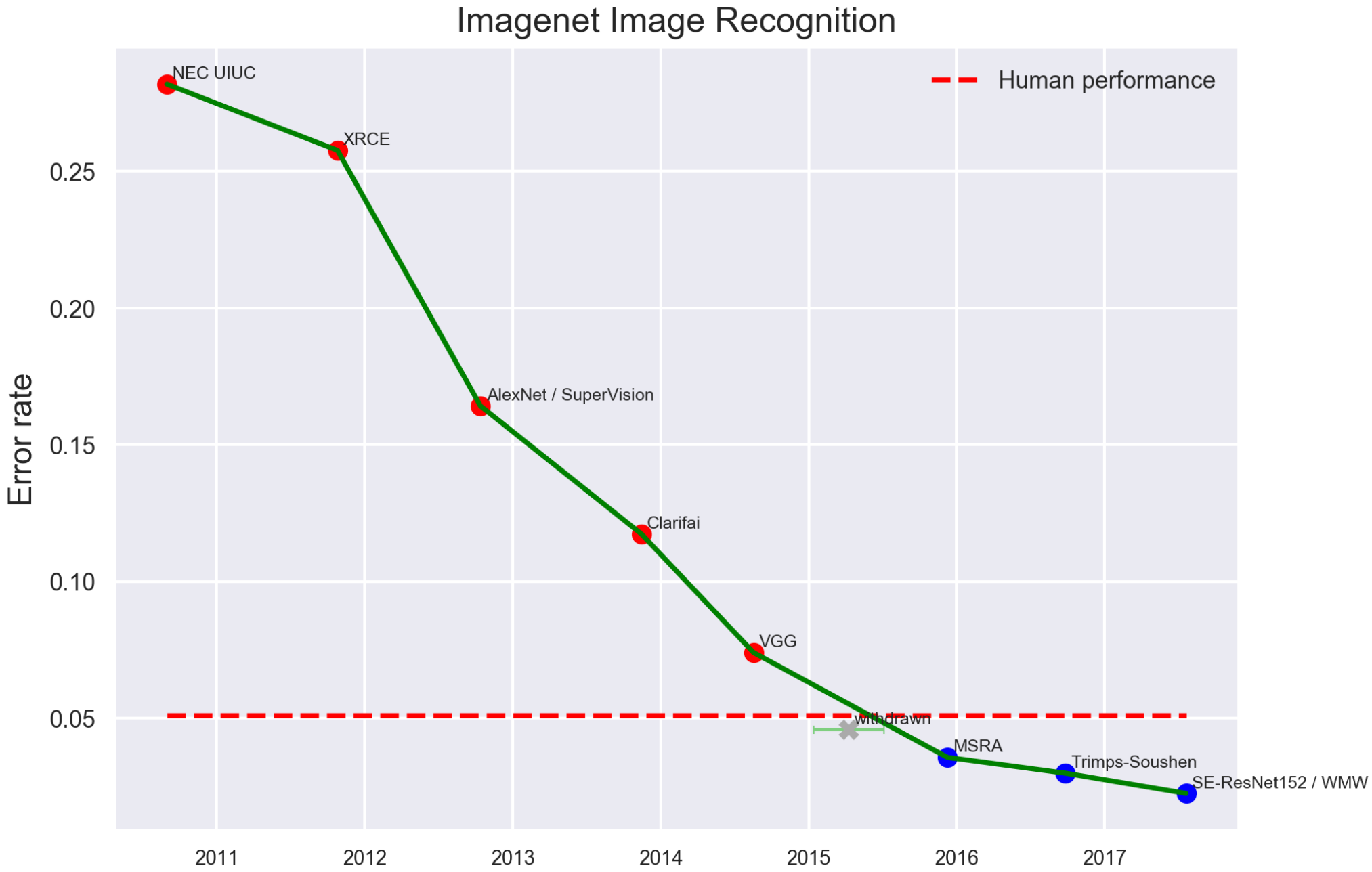
(28-1)*1+5 = 32

Source: cs231n

Fast forward to the arrival of big visual data…

- ~14 million labeled images, 20k classes
- Images gathered from Internet
- Human labels via Amazon MTurk
- ImageNet Large-Scale Visual Recognition Challenge (ILSVRC):
  1.2 million training images, 1000 classes

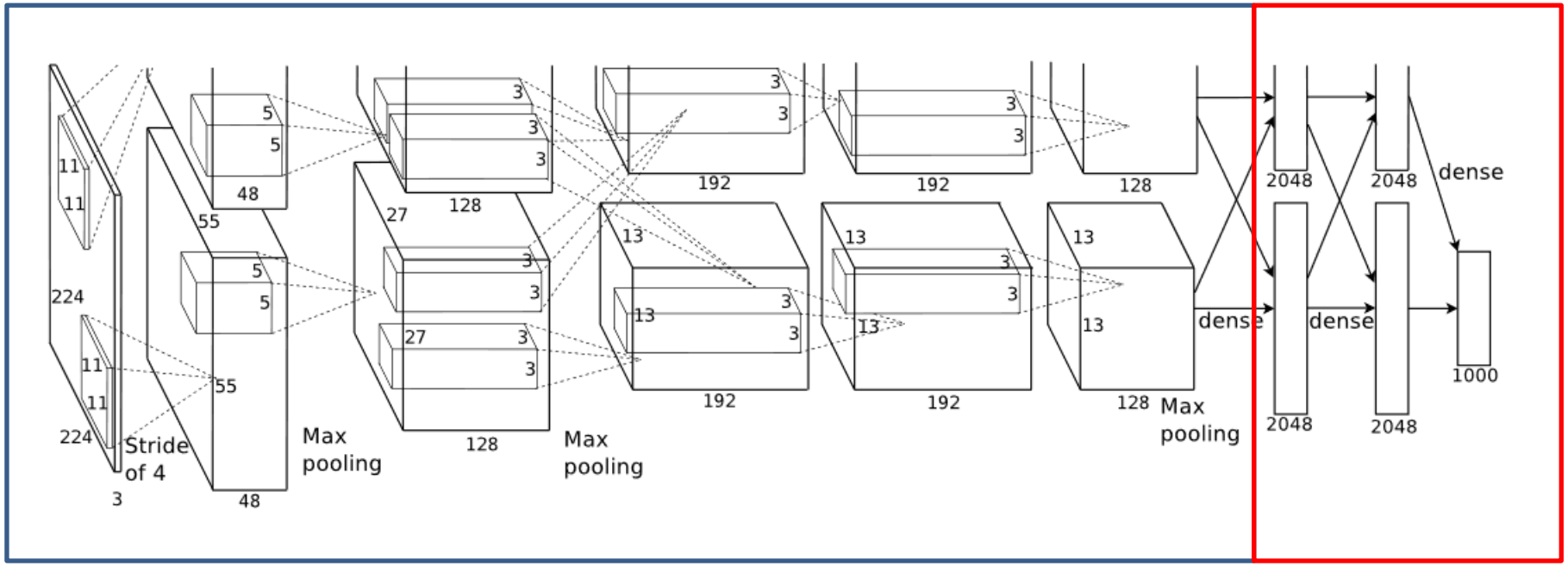
www.image-net.org/challenges/LSVRC/


Imagenet Image Recognition
Human performance
NEC UIUC
XRCE
AlexNet / SuperVision
Clarifai
VGG
withdrawn
MSRA
Trimps-Soushen
SE-ResNet152 / WMW
Error rate
0.25
0.20
0.15
0.10
0.05
2011
2012
2013
2014
2015
2016
2017

ALEXNET AND ITS DEVIANTS: ZFNET, VGGNET

AlexNet

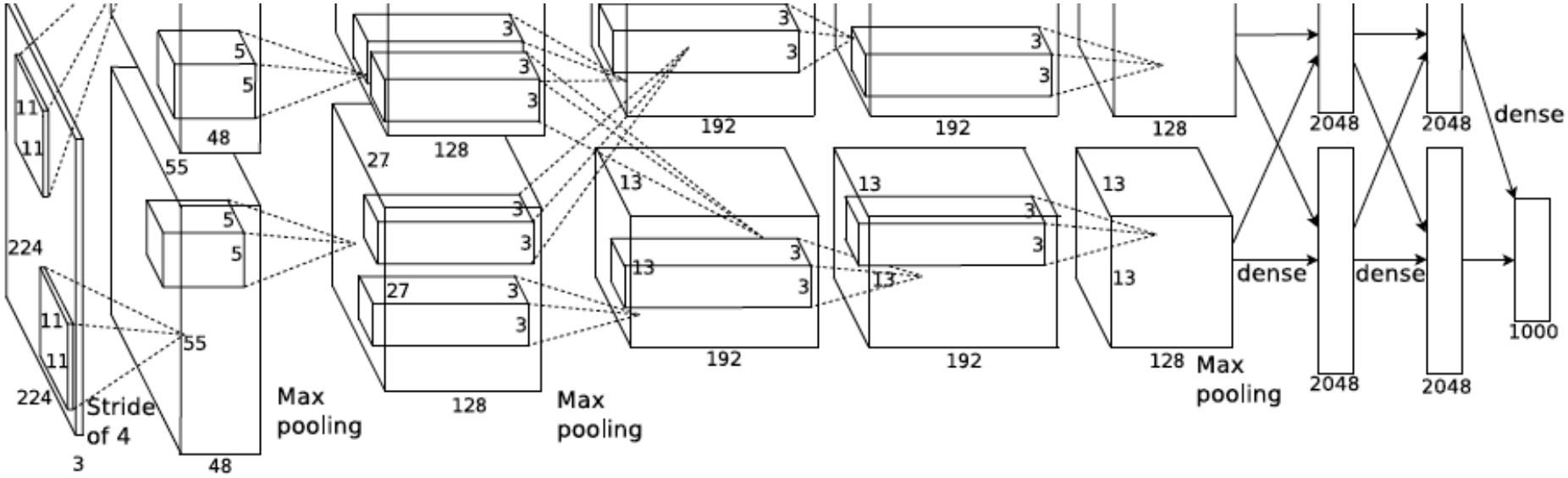
- 15.3% top-5 error rate
- Winner of ILSVRC2012



Convolutional layer
Feature extraction

Fully connected layer
classification

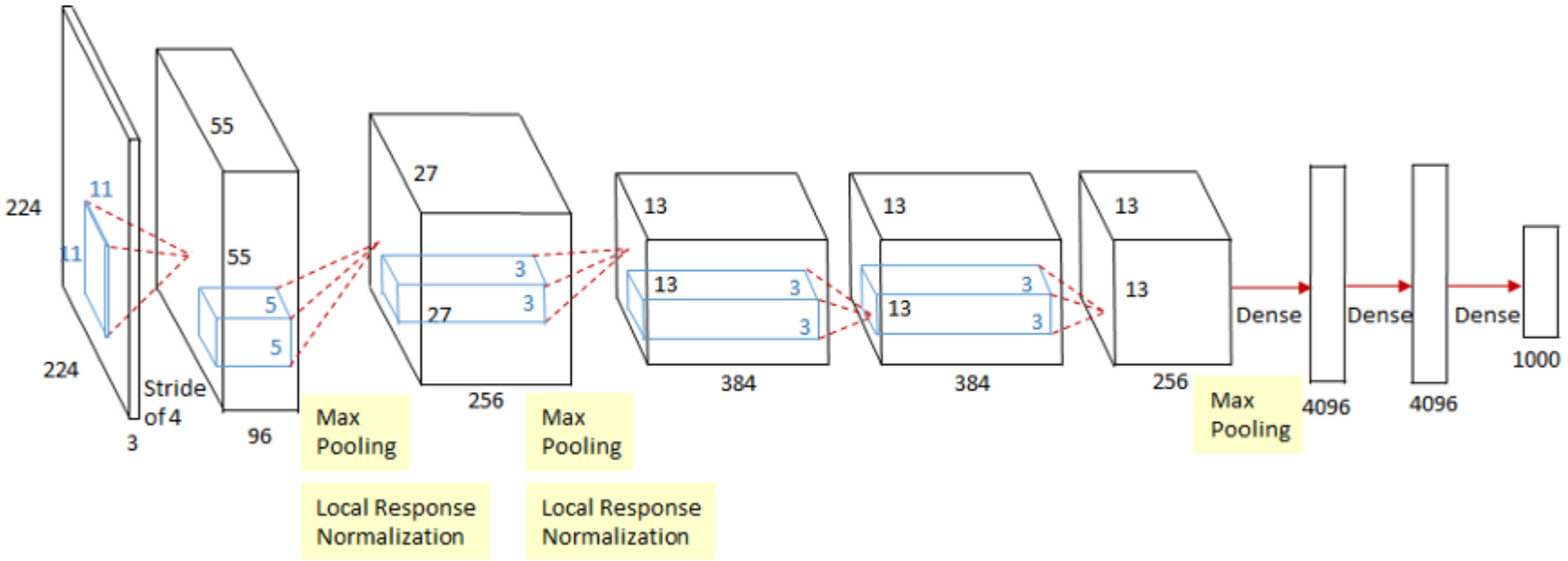
AlexNet: ILSVRC 2012 winner



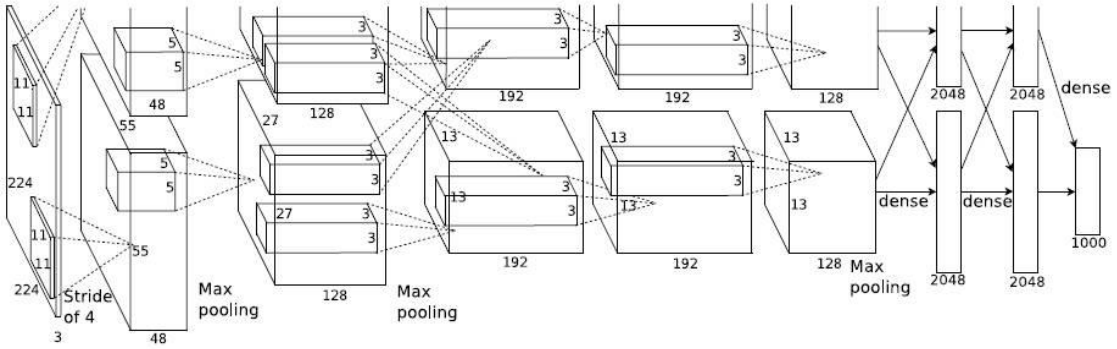
- Similar framework to LeNet but:
  - Max pooling, ReLU nonlinearity
  - More data and bigger model (7 hidden layers, 650K units, 60M params)
  - GPU implementation (50x speedup over CPU)
    - Trained on two GPUs for a week
  - Dropout regularization

A. Krizhevsky, I. Sutskever, and G. Hinton, ImageNet Classification with Deep Convolutional Neural Networks, NIPS 2012

CaffeNet: 1 GPU version of AlexNet



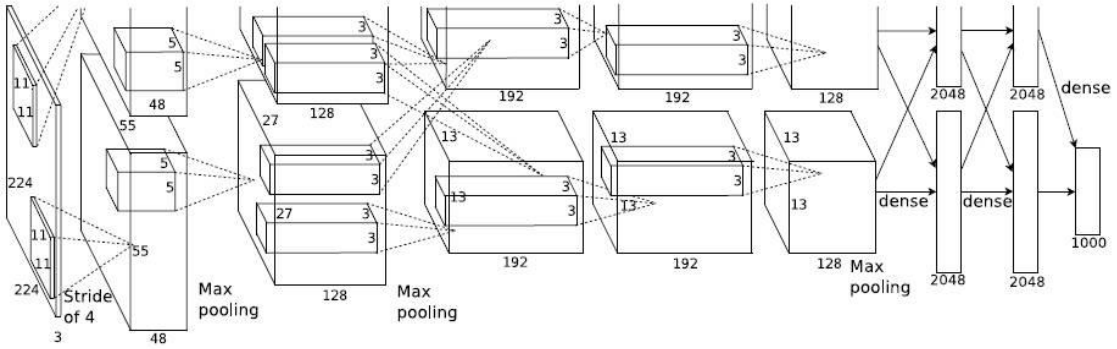
AlexNet: Parameters



- Input: 227x227x3 images
- First layer (CONV1): 96 11x11 filters applied at stride 4
- Q: what is the output volume size? Hint: (227-11)/4+1 = 55
  - Output volume [55x55x96]
- Q: What is the total number of parameters in this layer?
  - Parameters: (11*11*3)*96 = 35K

Source: cs231n

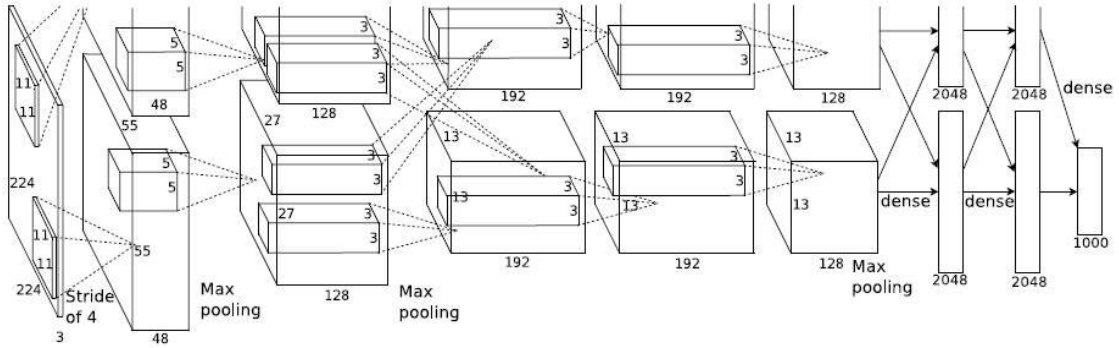
AlexNet



- Input: 227x227x3 images
- First layer (CONV1): 96 11x11 filters applied at stride 4
- Second layer (POOL1): 3x3 filters applied at stride 2
- Q: what is the output volume size? Hint: (55-3)/2+1 = 27
  - Output volume: 27x27x96
- Q: what is the number of parameters in this layer?
  - Parameters: 0!

Source: cs231n

AlexNet



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)
650K units, 60M params

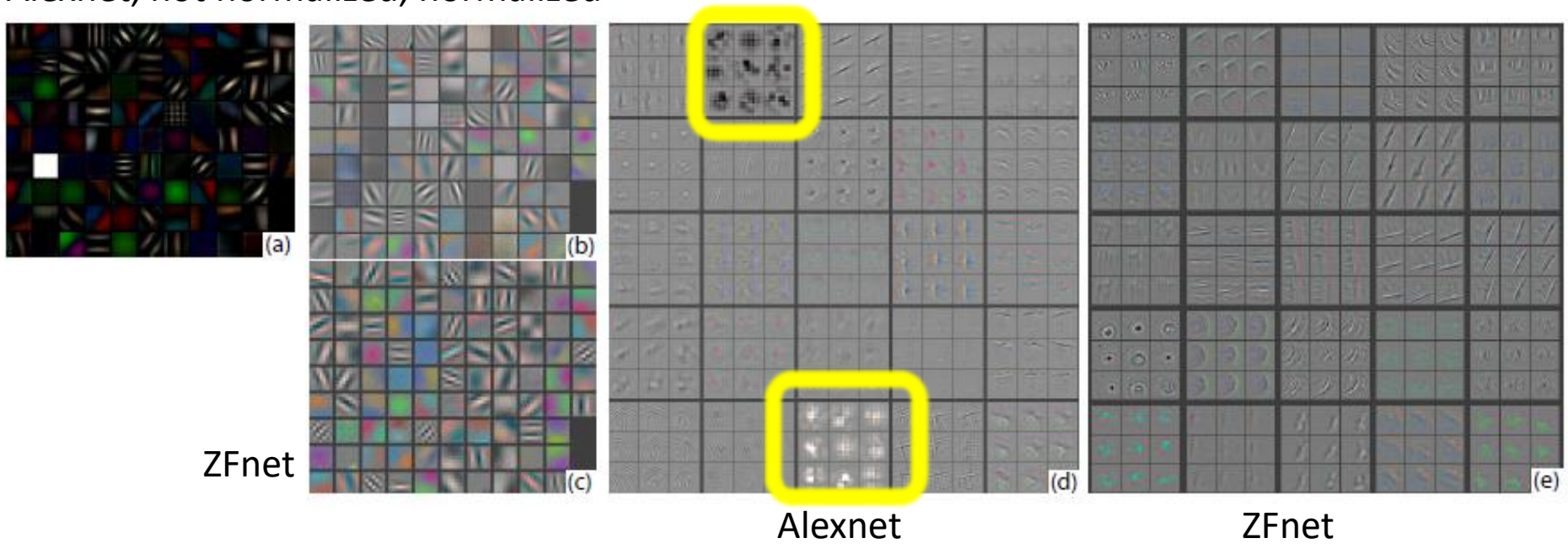
- **Details/Retrospectives**
  - first use of ReLU
  - used Norm layers (not common anymore)
  - heavy data augmentation
  - dropout 0.5
  - batch size 128
  - SGD Momentum 0.9
  - Learning rate 1e-2, reduced by 10 manually when val accuracy plateaus
  - L2 weight decay 5e-4
  - 7 CNN ensemble: 18.2% -> 15.4%

Source: cs231n

ZFNet - Architecture Selection

- Alexnet
  - Some of the features in the first layer do not converge well.
  - The second layer visualization shows aliasing artifacts.
- ZFNet Solutions
  - Reducing the 1st layer filter size from 11x11 to 7x7
  - Setting the stride of the convolution to be 2, rather than 4

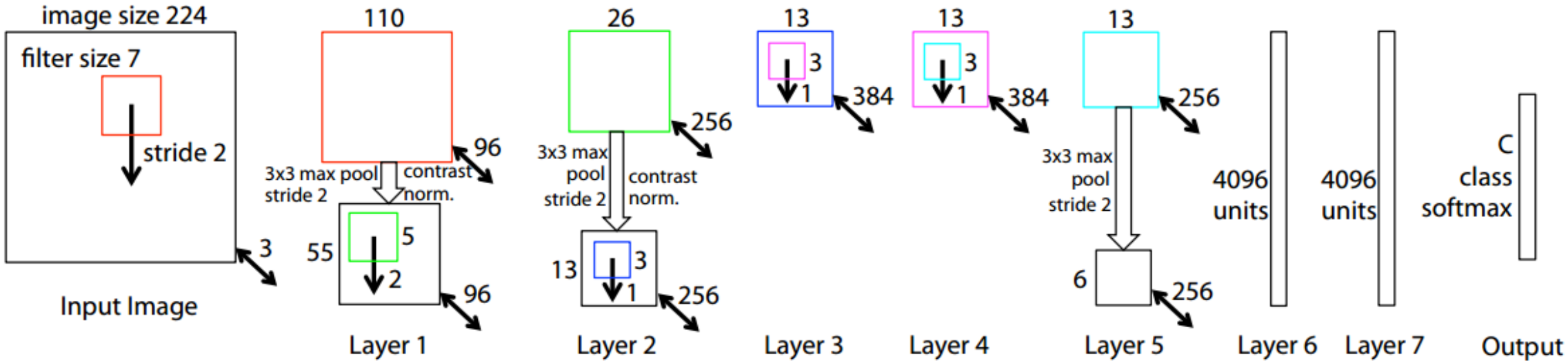
Alexnet, not normalized, normalized



Ref: D. Zeiler and R. Fergus “Visualizing and Understanding Convolutional Networks” arXiv:1311.2901

ZFNet: ILSVRC 2013 winner

- 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



M. Zeiler and R. Fergus, Visualizing and Understanding Convolutional Networks, ECCV 2014 (Best Paper Award winner)

ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners

Classification: ImageNet Challenge top-5 error

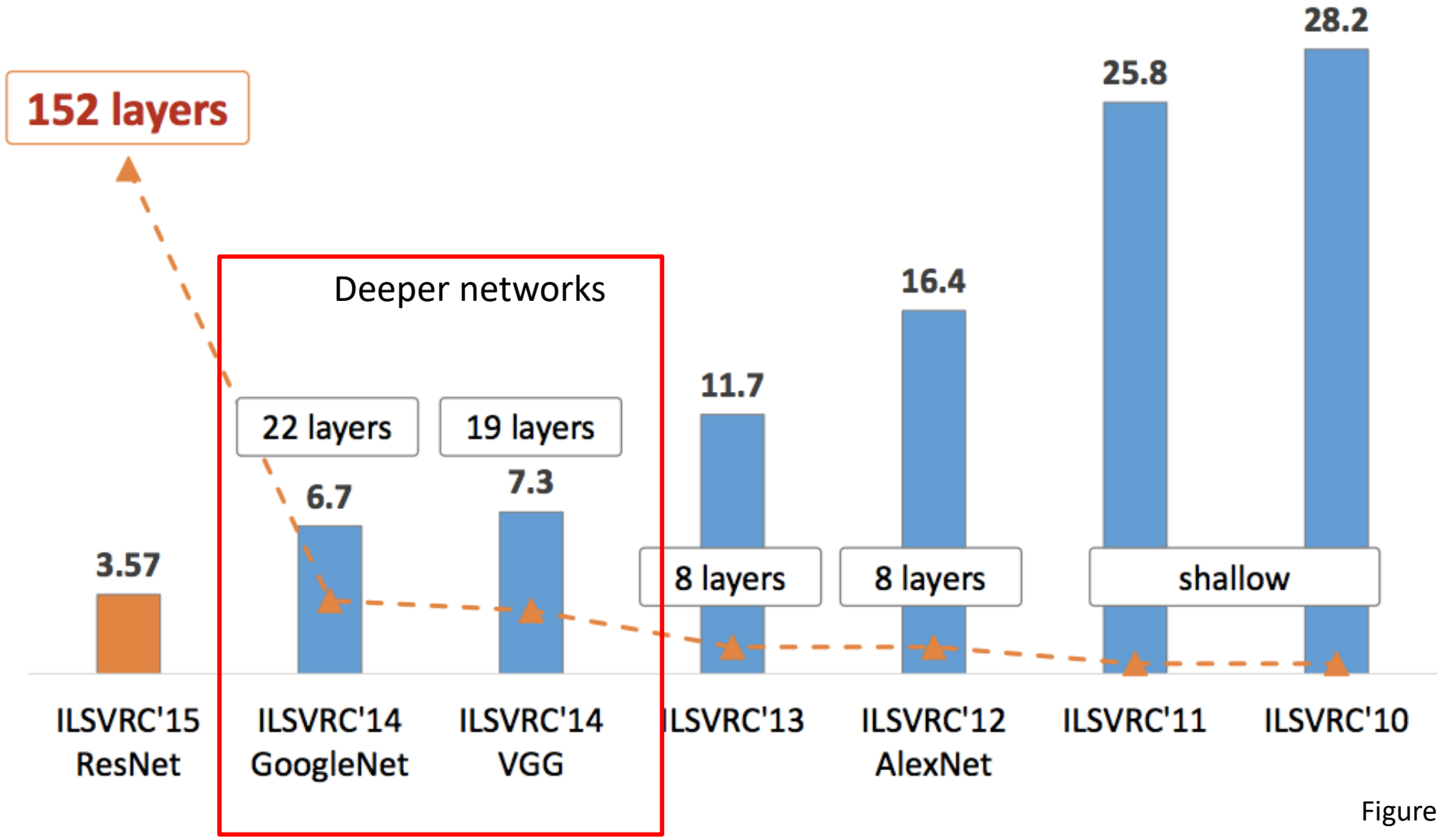
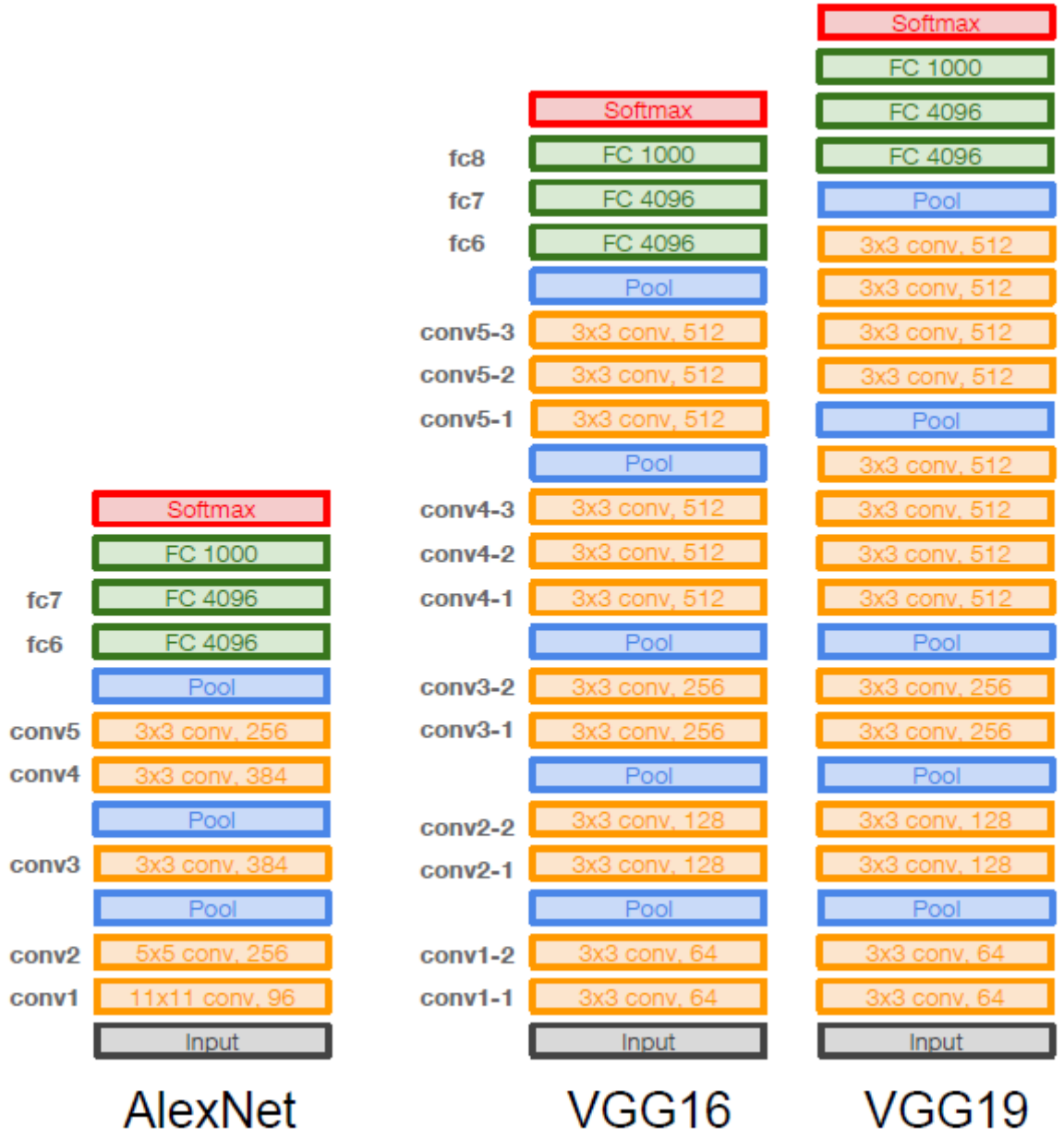


Figure source: Kaiming He

VGGNet

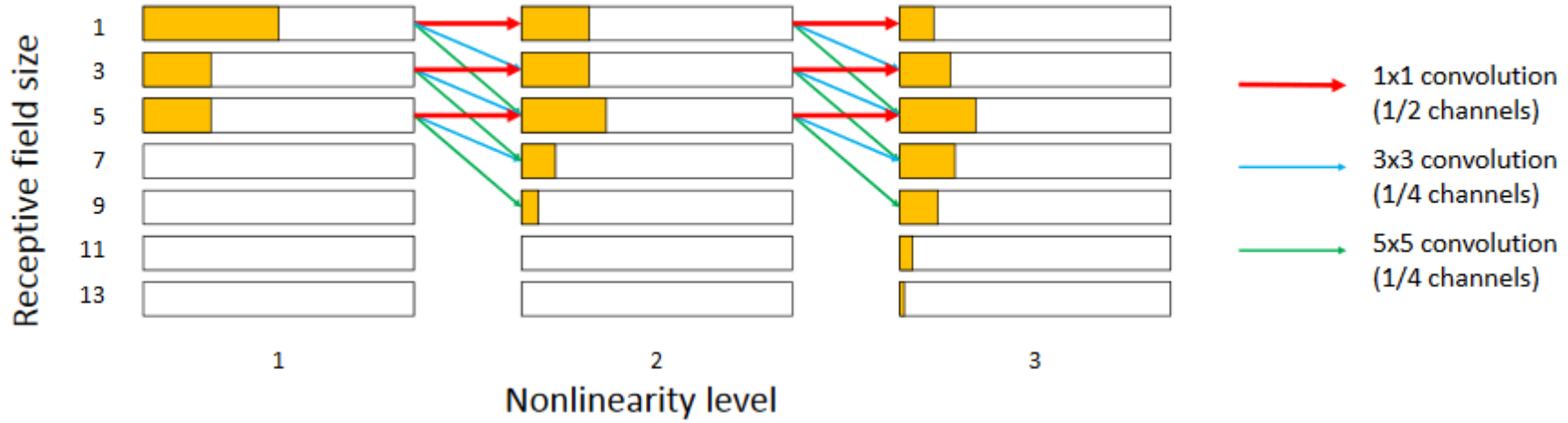
- Small filters, Deeper networks
- 8 layers (AlexNet) =>
  - 16 - 19 layers (VGG16Net)
- **Only 3x3 CONV** stride 1, pad 1 and 2x2 MAX POOL stride 2
- 11.7% top 5 error in ILSVRC'13 (ZFNet) =>
  - 7.3% top 5 error in ILSVRC'14



Source: cs231n

VGGNet

- Q: Why use smaller filters? (3x3 conv)
  - 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: $3*(3^2C^2) = 27C^2$ vs. $7^2C^2 = 49C^2$ for C channels per layer



Source: cs231n

VGG-16

[Conv->ReLU]*x->Maxpool

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

(not counting biases

Most memor
early CONV

Most params are
in late FC

VGG16

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

Source: cs231n

VGGNet

- Details:
  - ILSVRC'14 2nd in classification, 1st in localization
  - Similar training procedure as Krizhevsky 2012
  - No Local Response Normalisation (LRN)
  - Use VGG16 or VGG19 (VGG19 only slightly better, more memory)
  - Use ensembles for best results
  - FC7 features generalize well to other tasks

Heavy computation and parameters => need efficient network

VGG16 VGG19

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Table 2: **Number of parameters** (in millions).

Network	A,A-LRN	B	C	D	E
Number of parameters	133	133	134	138	144

Source: cs231n

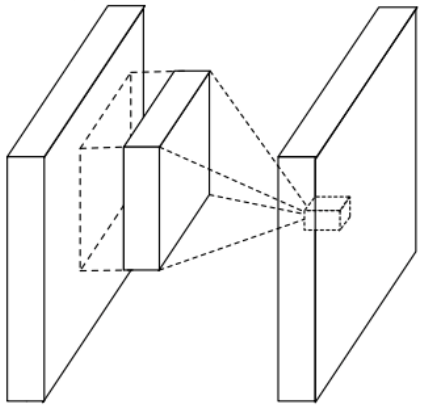
GOING DEEPER

NETWORK IN NETWORK, GOOGLENET, RESIDUAL NET

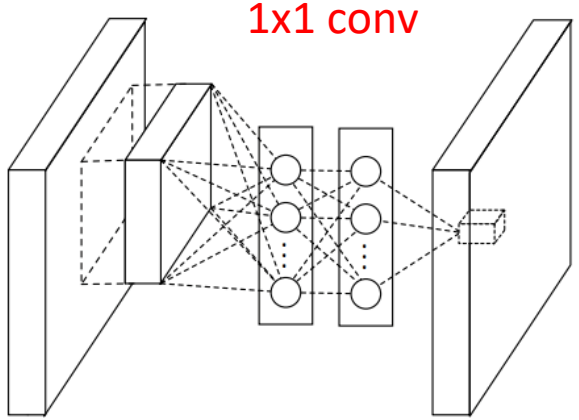
Network in network: 1x1 convolutions + GAV

- 1x1 convolution to add nonlinearity
- Global average pooling layer

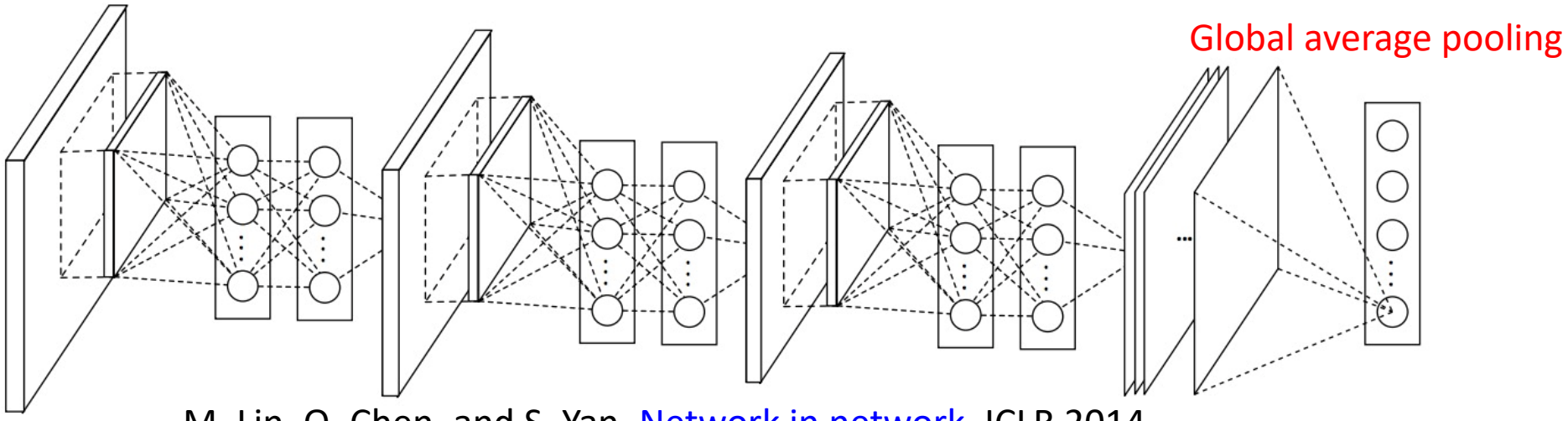
as part of the last classifier to save parameter numbers



(a) Linear convolution layer



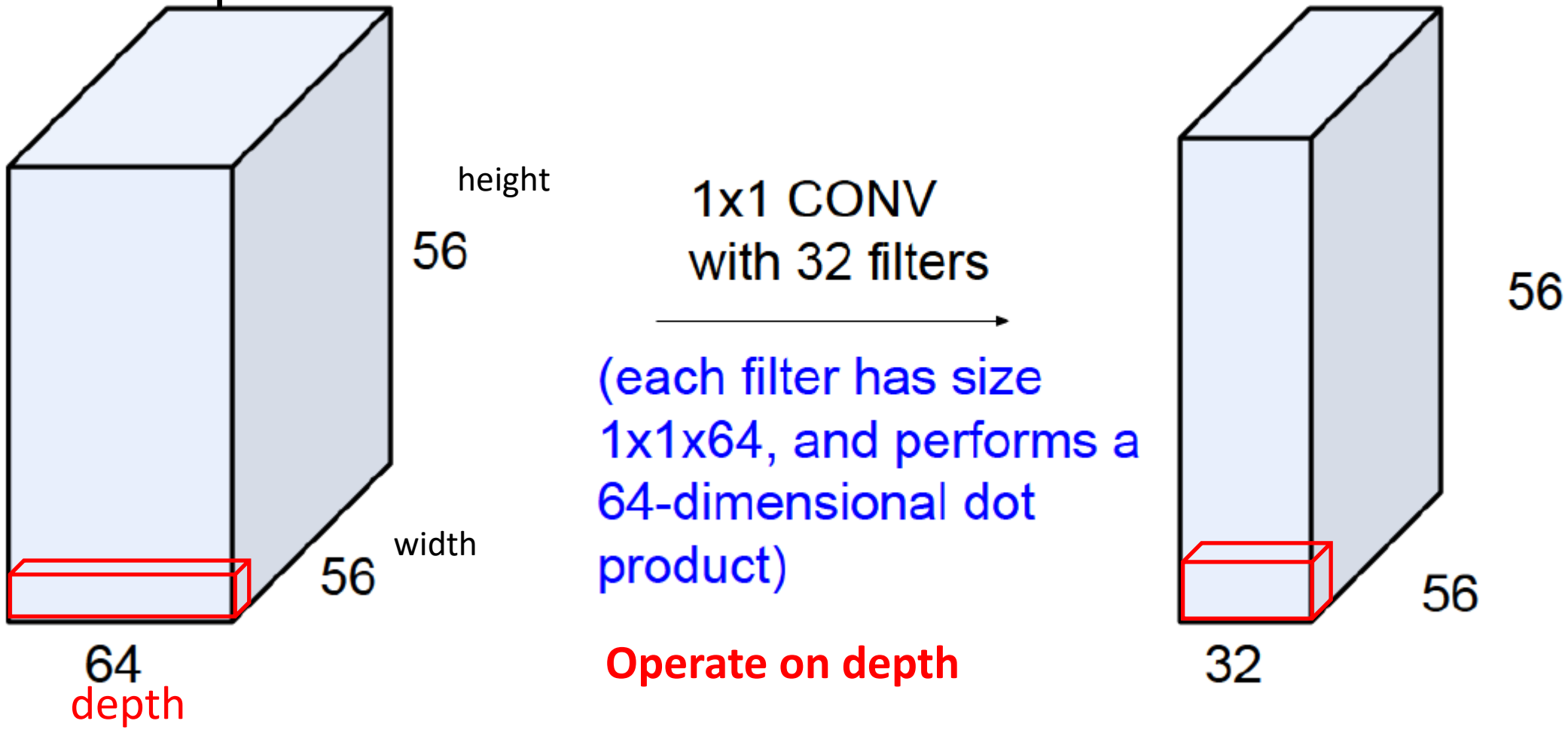
(b) Mlpconv layer



M. Lin, Q. Chen, and S. Yan, Network in network, ICLR 2014

1x1 Convolution

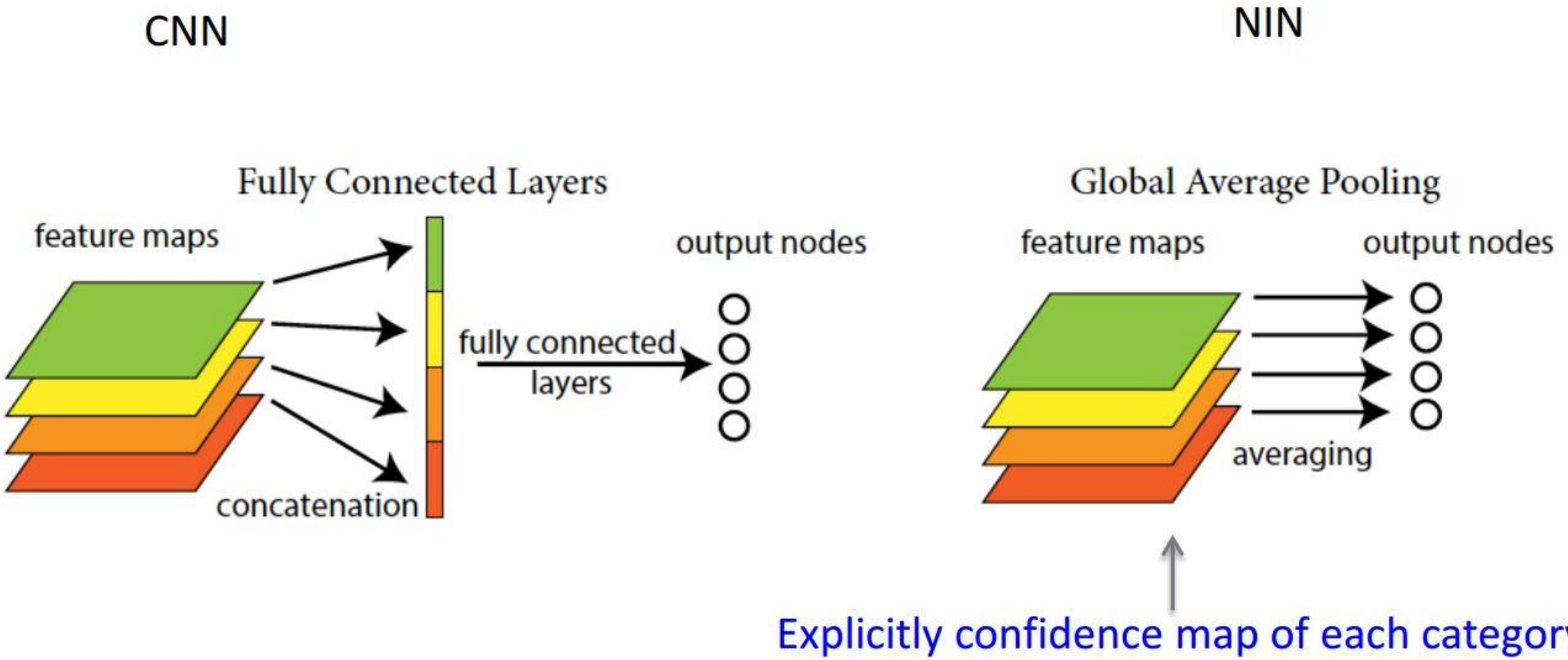
- Change channel number and add nonlinearity
- Fewer parameters



Source: cs231n

Global Average Pooling

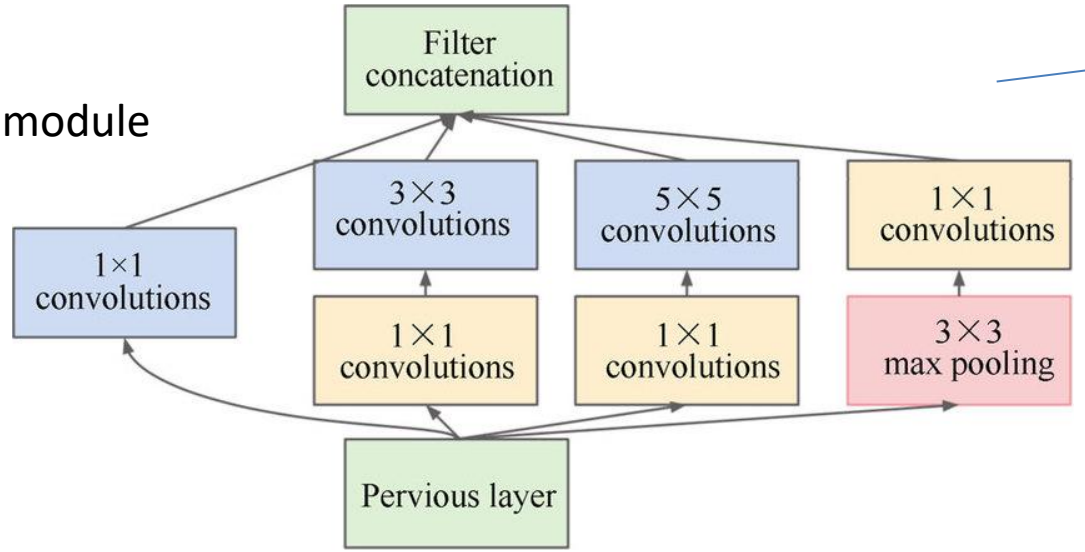
- Save a large portion of parameters



GoogLeNet

- Deeper networks, with computational efficiency
  - 22 layers
  - Efficient "Inception" module
  - No FC layers
  - Only 5 million parameters! 12x less than AlexNet
  - ILSVRC'14 classification winner(6.7% top 5 error)

Inception module



Source: cs231n

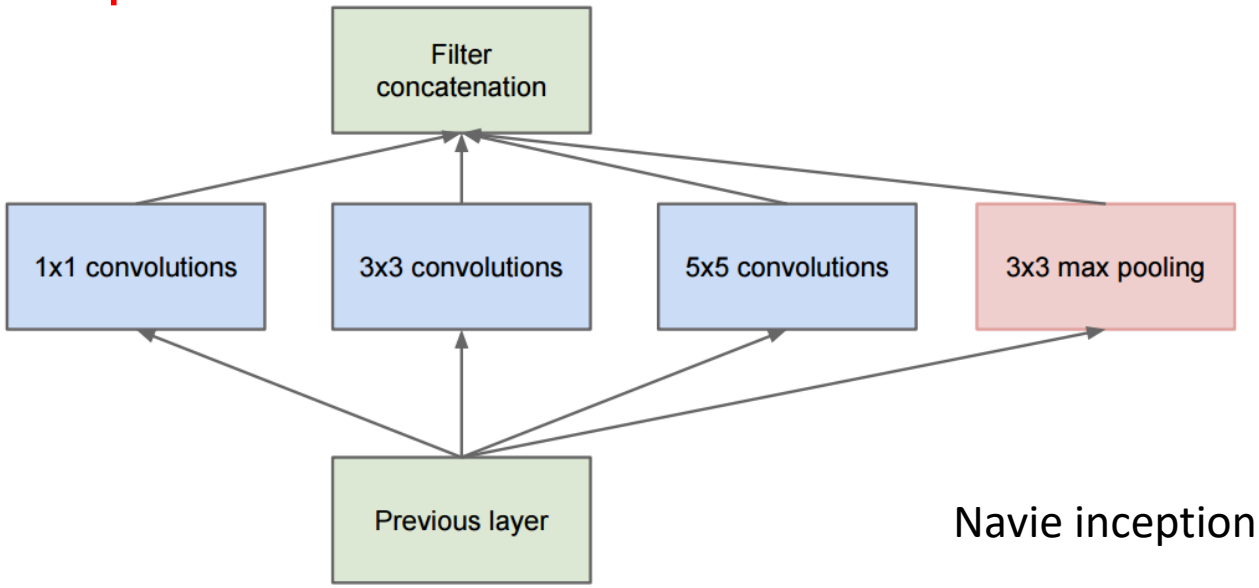
GoogLeNet

- Motivation
  - Uniform deeper and wider network for higher performance at great expense of computation/parameters/dataset
- Performance: deep and different filter size
  - Deep: vanishing gradient and exploding gradient problem
    - => auxiliary layer output
  - Different filter size: diverse filters, 3x3, 5x5, 1x1, max pooling
- Complexity
  - Filter level sparsity
  - FC layer => global average pooling layer
- The Inception Module: filter level sparsity
  - Parallel paths with different receptive field sizes and operations are meant to capture sparse patterns of correlations in the stack of feature maps

C. Szegedy et al., Going deeper with convolutions, CVPR 2015

GoogLeNet

- Motivation
  - Uniform deeper and wider network for higher performance at great expense of computation/parameters/dataset
- The Inception Module: filter level sparsity
  - Parallel paths with different receptive field sizes and operations are meant to capture sparse patterns of correlations in the stack of feature maps



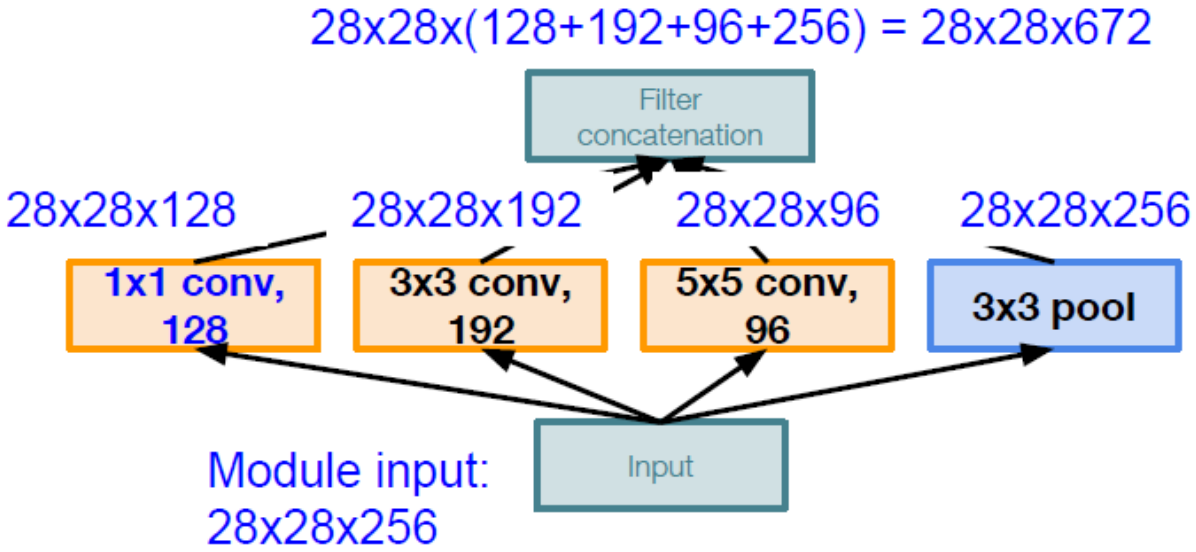
Navie inception module

C. Szegedy et al., Going deeper with convolutions, CVPR 2015

GoogLeNet: Problems of Naïve Inception Module

Example:

Q3:What is output size after filter concatenation?



Naive Inception module

Conv Ops:

[1x1 conv, 128] 28x28x128x1x1x256

[3x3 conv, 192] 28x28x192x3x3x256

[5x5 conv, 96] 28x28x96x5x5x256

Total: 854M ops

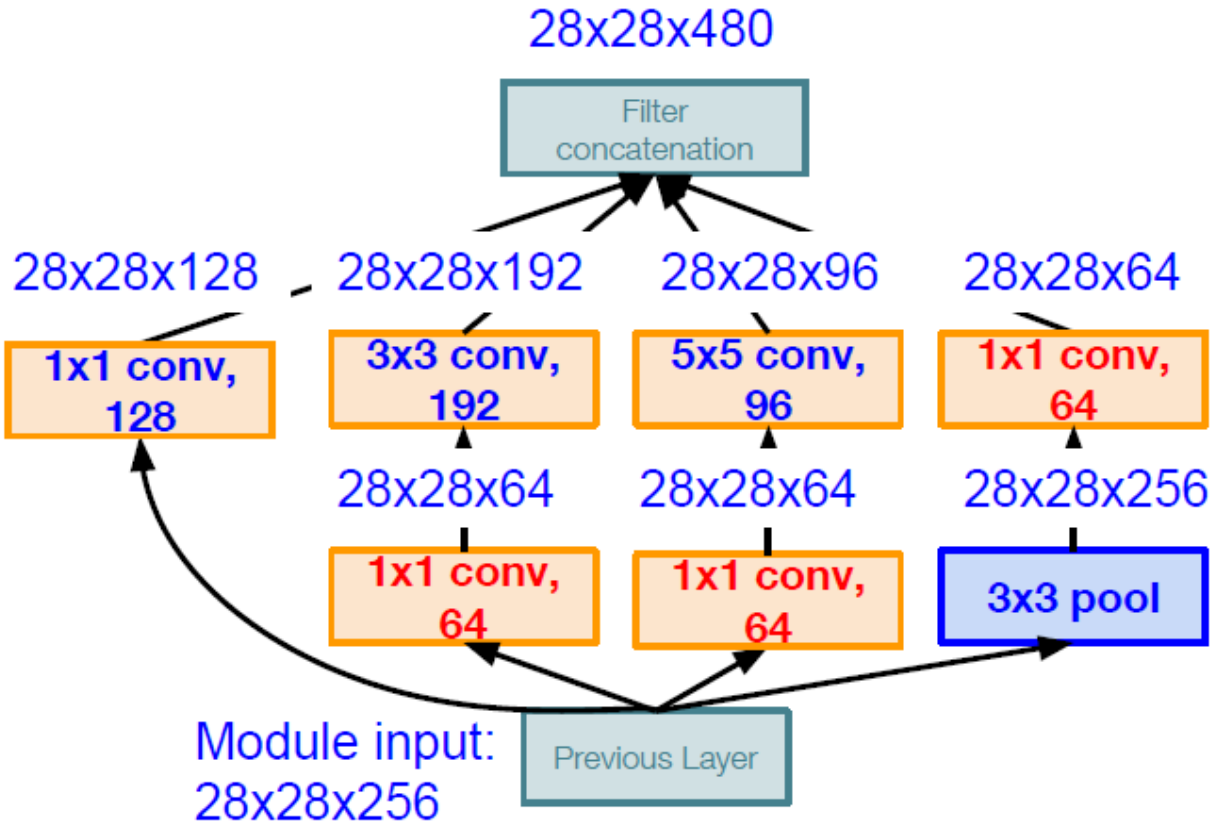
Very expensive compute

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

Source: cs231n

GoogLeNet



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

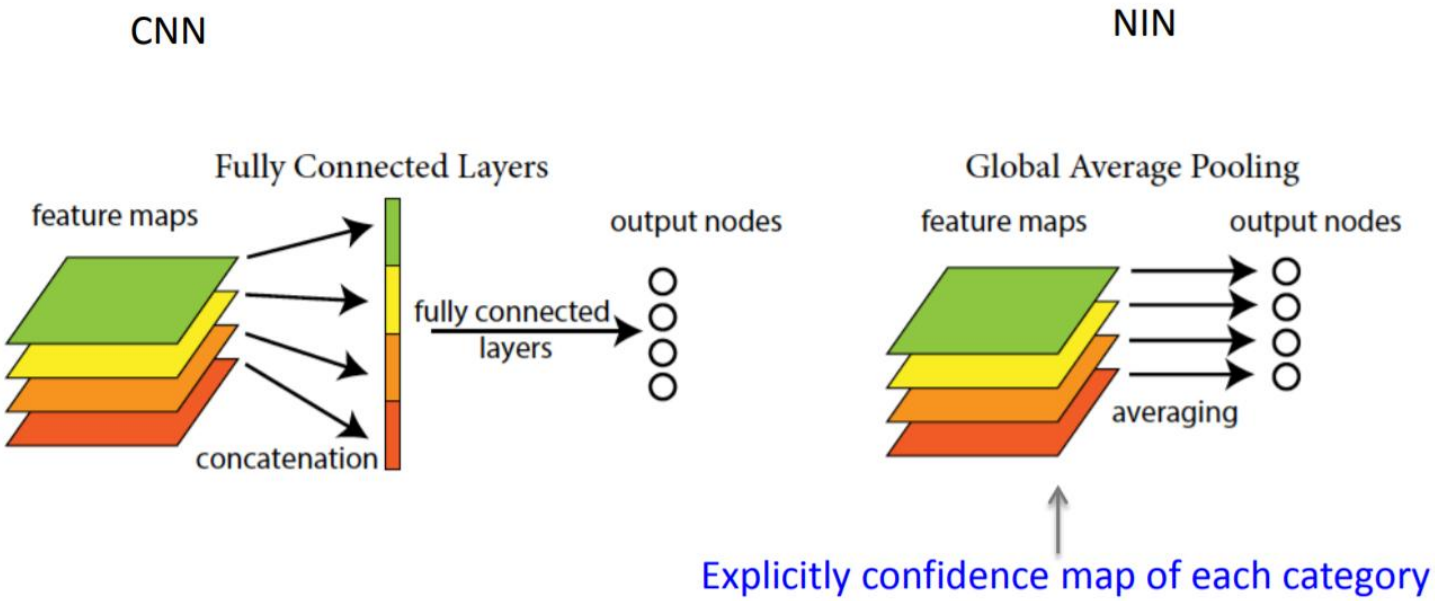
Total: 358M ops

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

Source: cs231n

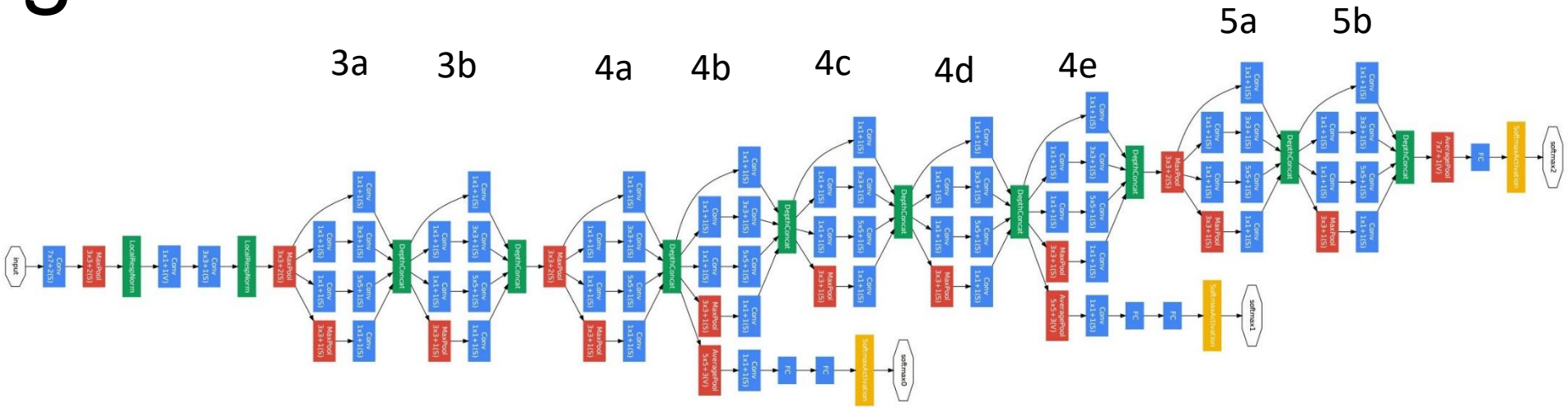
Googlenet-Global Average Pooling

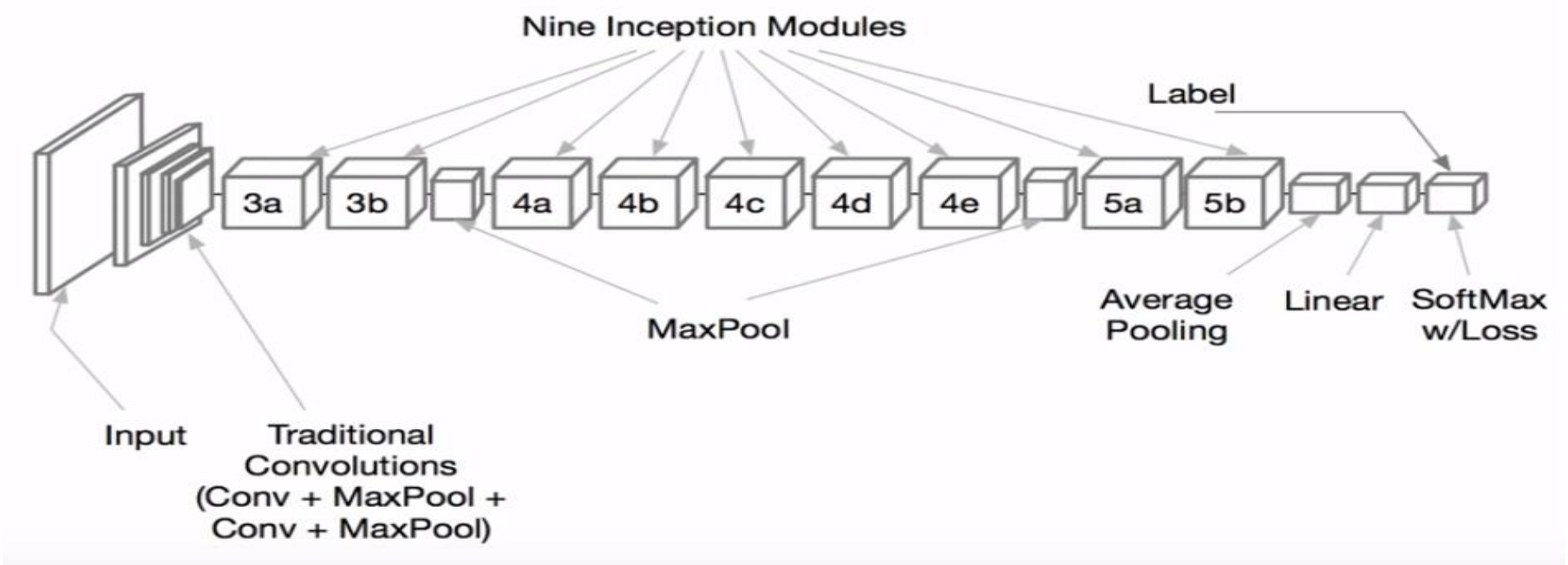
- Overcome overfitting
- Reduce parameters(about 100 millions in VGGnet)



Save a large portion of parameters

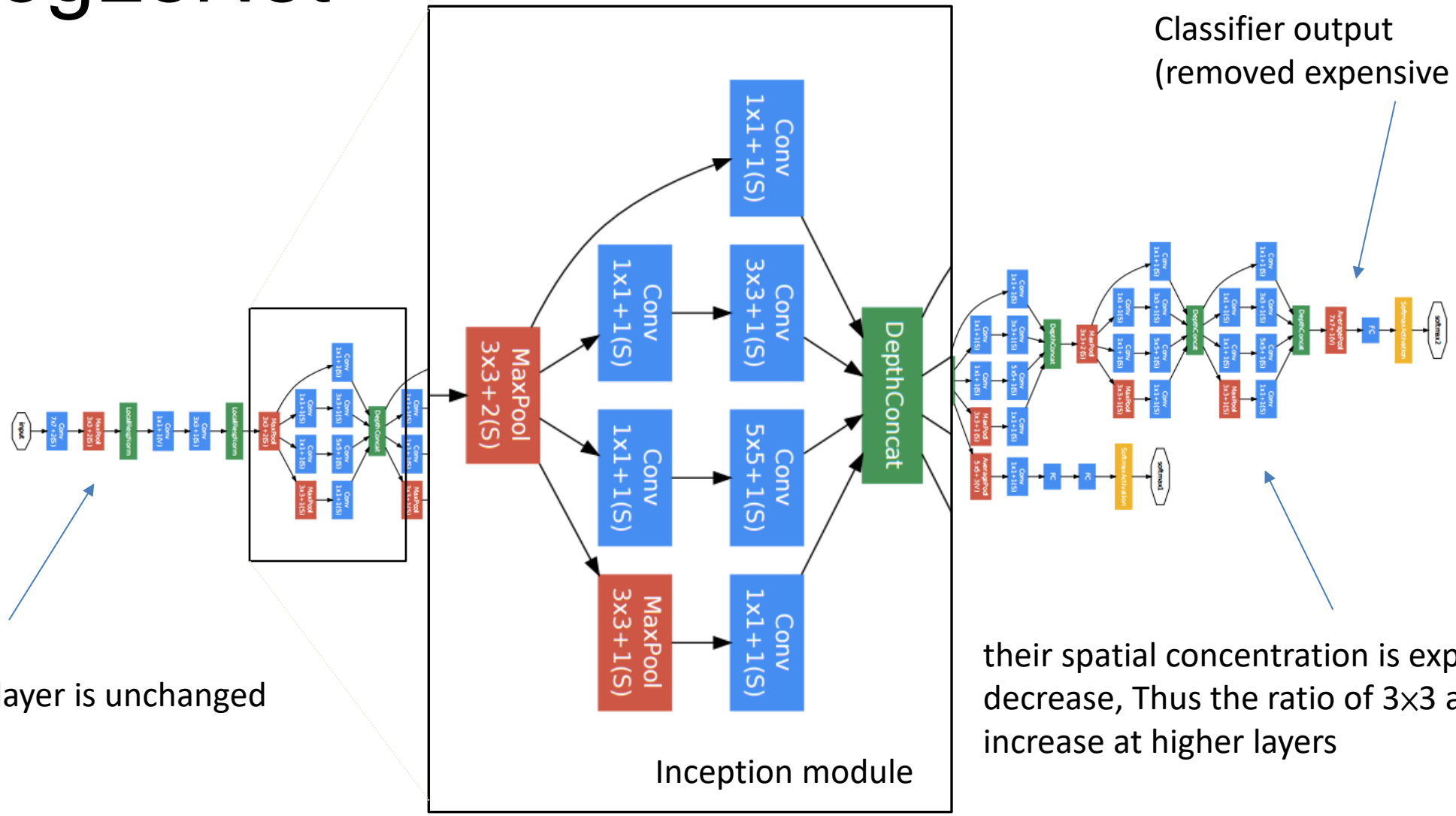
Google Net - The Architecture





Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, ”Going Deeper with Convolutions”, arXiv:1409.4842,Sep 2014.

GoogLeNet



Classifier output
(removed expensive FC layers!)

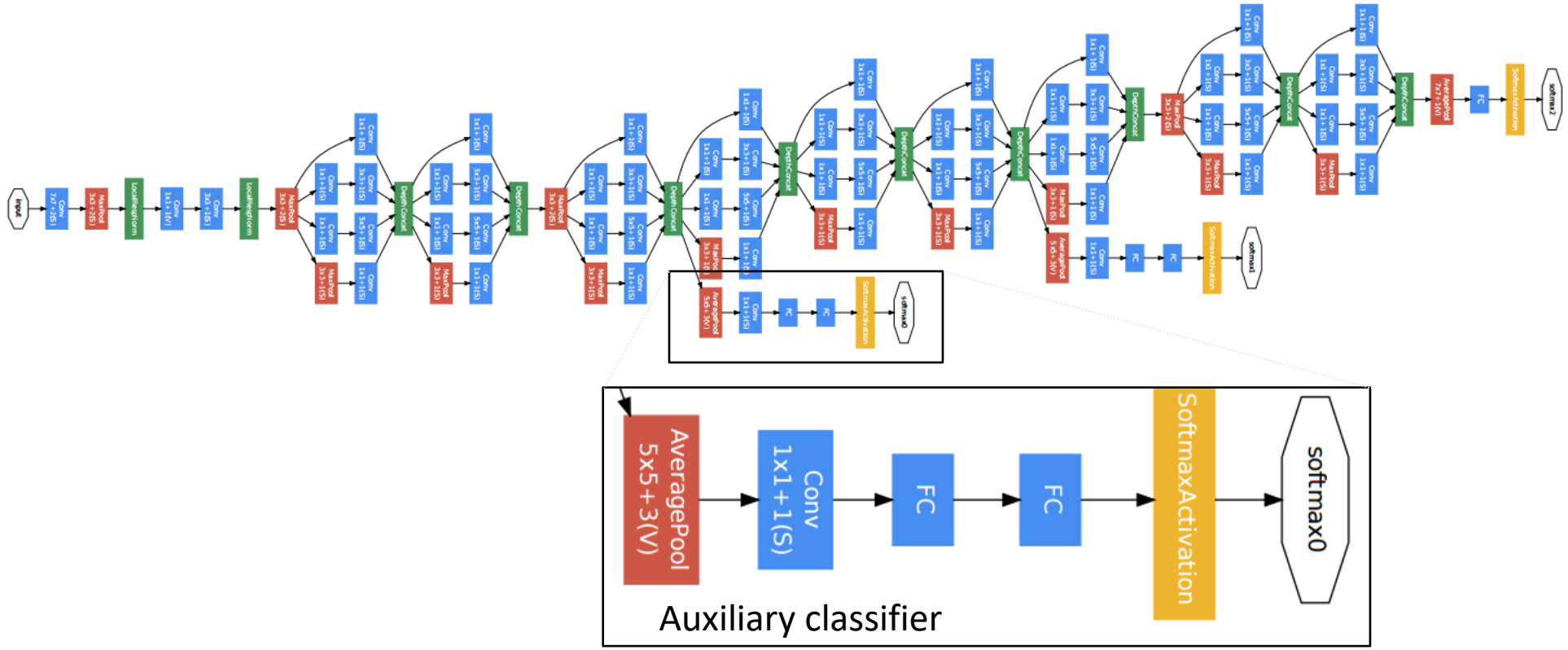
Initial layer is unchanged

their spatial concentration is expected to decrease, Thus the ratio of 3×3 and 5×5 should increase at higher layers

C. Szegedy et al., Going deeper with convolutions, CVPR 2015

GoogLeNet

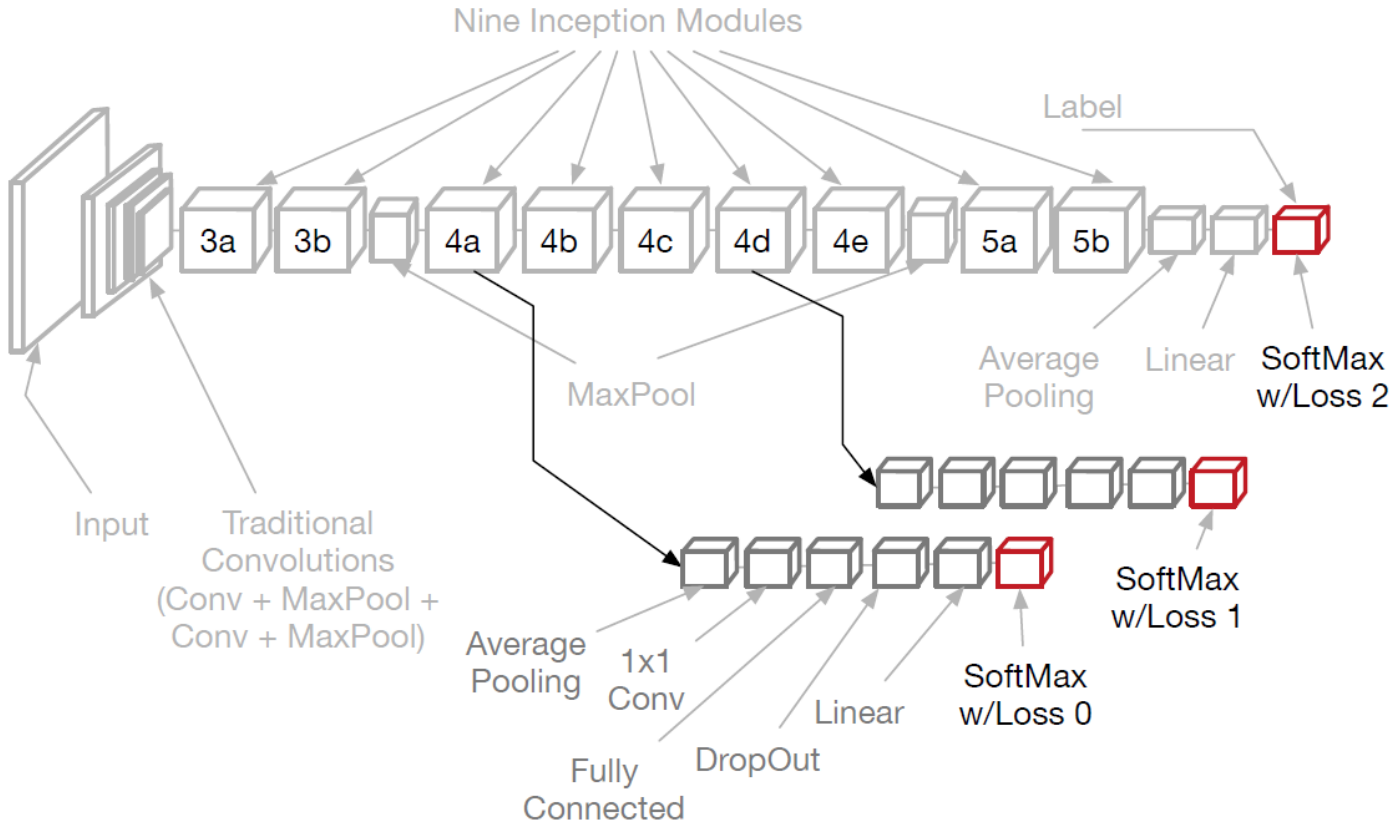
auxiliary classifiers injects additional gradient at lower layers
thought to combat the vanishing gradient problems



C. Szegedy et al., Going deeper with convolutions, CVPR 2015

Google Net – Vanishing Gradient

- Given relatively large depth of the network, the ability to propagate gradients back through all the layers in an effective manner was a concern.
- By adding auxiliary classifiers connected to these intermediate layers, combat the



GoogLeNet: Network Design and Result

- Achieve ILSVRC2014 1st
- 6.67% top-5 error rate

Team	Year	Place	Error (top-5)	Uses external data
SuperVision	2012	1st	16.4%	no
SuperVision	2012	1st	15.3%	Imagenet 22k
Clarifai	2013	1st	11.7%	no
Clarifai	2013	1st	11.2%	Imagenet 22k
MSRA	2014	3rd	7.35%	no
VGG	2014	2nd	7.32%	no
GoogLeNet	2014	1st	6.67%	no

Table 2: Classification performance.

Number of models	Number of Crops	Cost	Top-5 error	compared to base
1	1	1	10.07%	base
1	10	10	9.15%	-0.92%
1	144	144	7.89%	-2.18%
7	1	7	8.09%	-1.98%
7	10	70	7.62%	-2.45%
7	144	1008	6.67%	-3.45%

Table 3: GoogLeNet classification performance break down.

type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M
inception (3b)		28×28×480	2	128	128	192	32	96	64	380K	304M
max pool	3×3/2	14×14×480	0								
inception (4a)		14×14×512	2	192	96	208	16	48	64	364K	73M
inception (4b)		14×14×512	2	160	112	224	24	64	64	437K	88M
inception (4c)		14×14×512	2	128	128	256	24	64	64	463K	100M
inception (4d)		14×14×528	2	112	144	288	32	64	64	580K	119M
inception (4e)		14×14×832	2	256	160	320	32	128	128	840K	170M
max pool	3×3/2	7×7×832	0								
inception (5a)		7×7×832	2	256	160	320	32	128	128	1072K	54M
inception (5b)		7×7×1024	2	384	192	384	48	128	128	1388K	71M
avg pool	7×7/1	1×1×1024	0								
dropout (40%)		1×1×1024	0								
linear		1×1×1000	1							1000K	1M
softmax		1×1×1000	0								

GoogLeNet

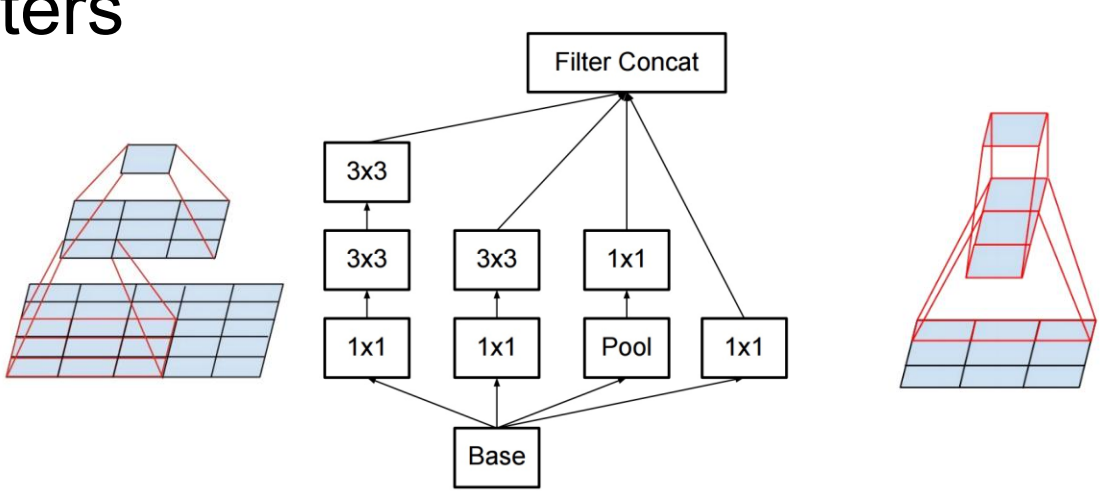
- An alternative view:

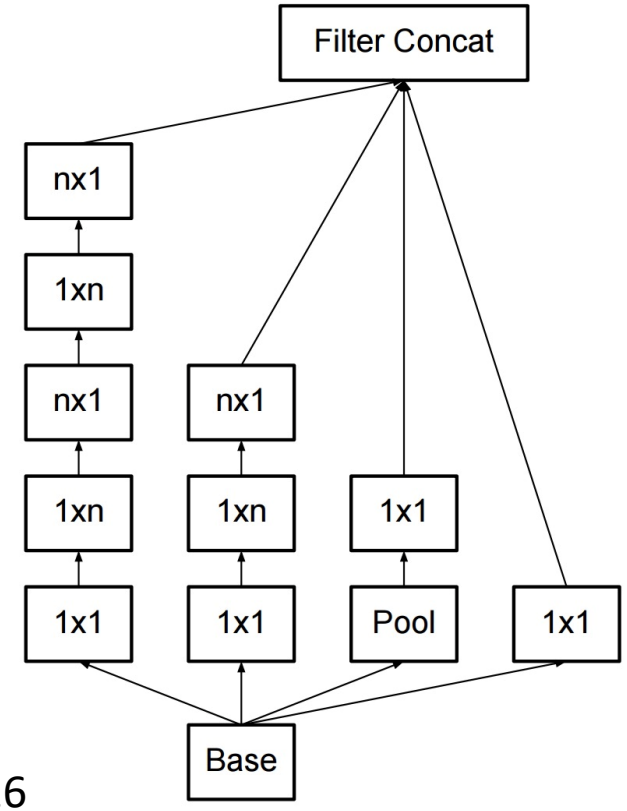
type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M
inception (3b)		28×28×480	2	128	128	192	32	96	64	380K	304M
max pool	3×3/2	14×14×480	0								
inception (4a)		14×14×512	2	192	96	208	16	48	64	364K	73M
inception (4b)		14×14×512	2	160	112	224	24	64	64	437K	88M
inception (4c)		14×14×512	2	128	128	256	24	64	64	463K	100M
inception (4d)		14×14×528	2	112	144	288	32	64	64	580K	119M
inception (4e)		14×14×832	2	256	160	320	32	128	128	840K	170M
max pool	3×3/2	7×7×832	0								
inception (5a)		7×7×832	2	256	160	320	32	128	128	1072K	54M
inception (5b)		7×7×1024	2	384	192	384	48	128	128	1388K	71M
avg pool	7×7/1	1×1×1024	0								
dropout (40%)		1×1×1024	0								
linear		1×1×1000	1							1000K	1M
softmax		1×1×1000	0								

C. Szegedy et al., Going deeper with convolutions, CVPR 2015

Inception v2, v3

- Regularize training with batch normalization, reducing importance of auxiliary classifiers
- More variants of inception modules with aggressive factorization of filters





C. Szegedy et al., Rethinking the inception architecture for computer vision, CVPR 2016

RESIDUAL NET AND ITS VARIANTS

ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners

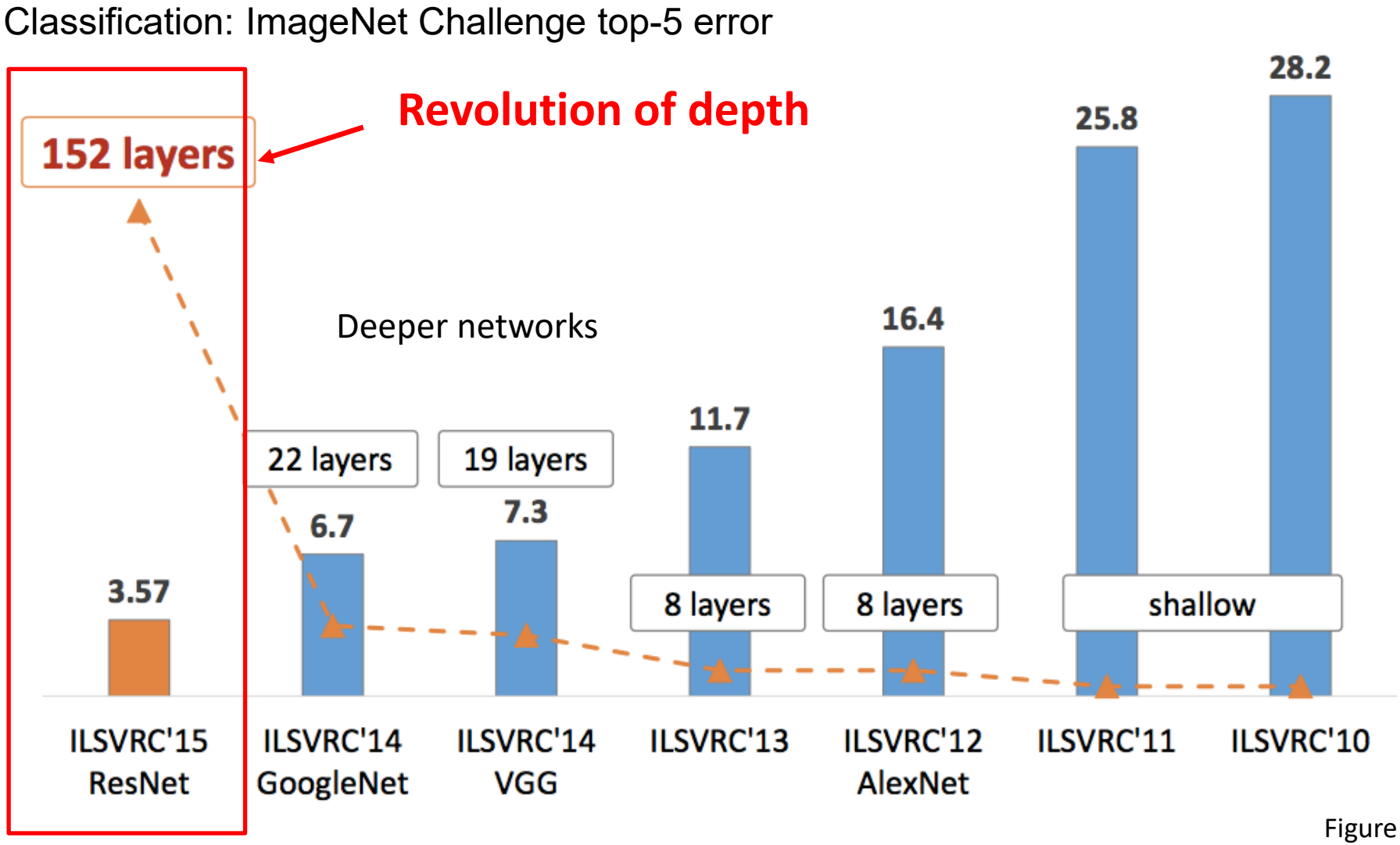
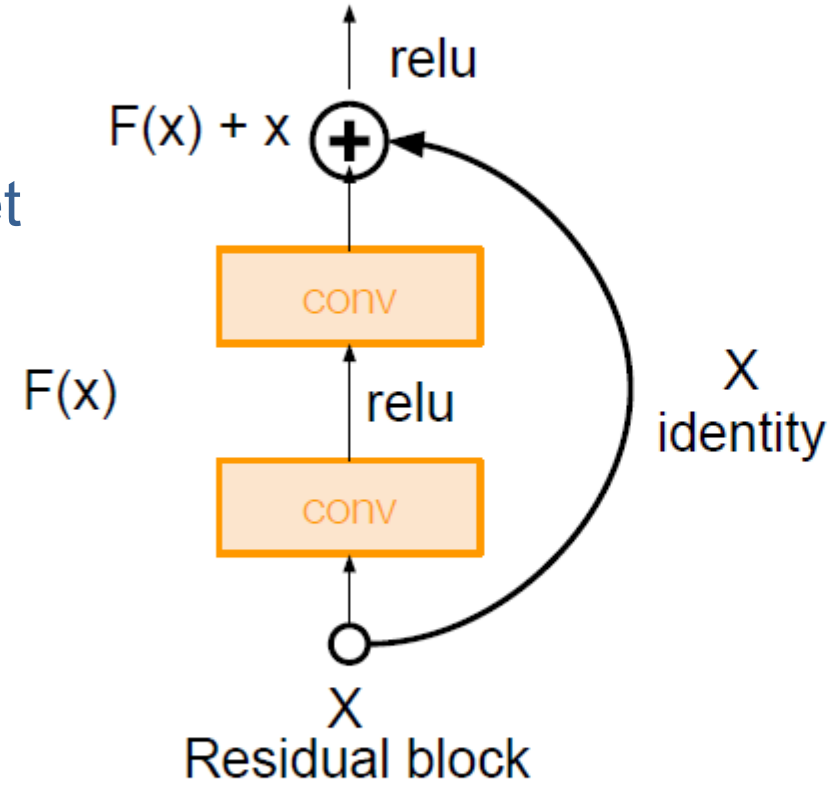


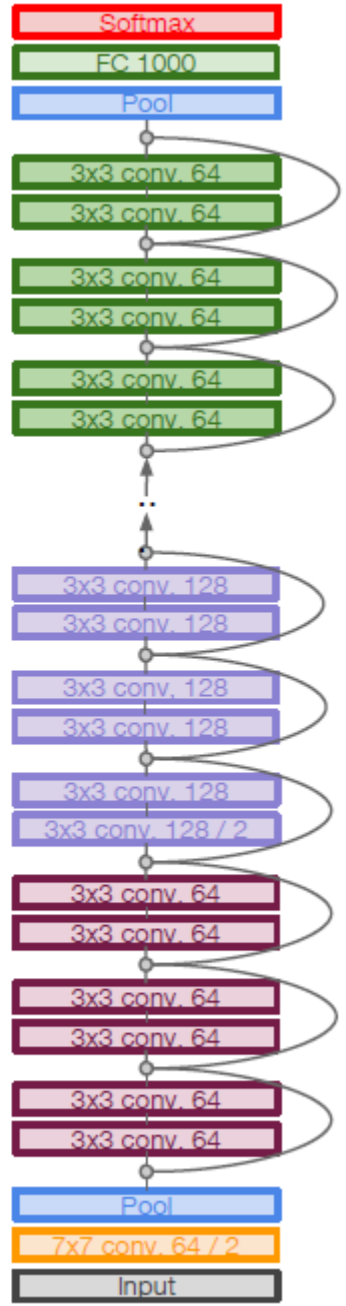
Figure source: Kaiming He

Source: cs231n

ResNet: Residual Net

- Very deep networks using residual connections
  - 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!



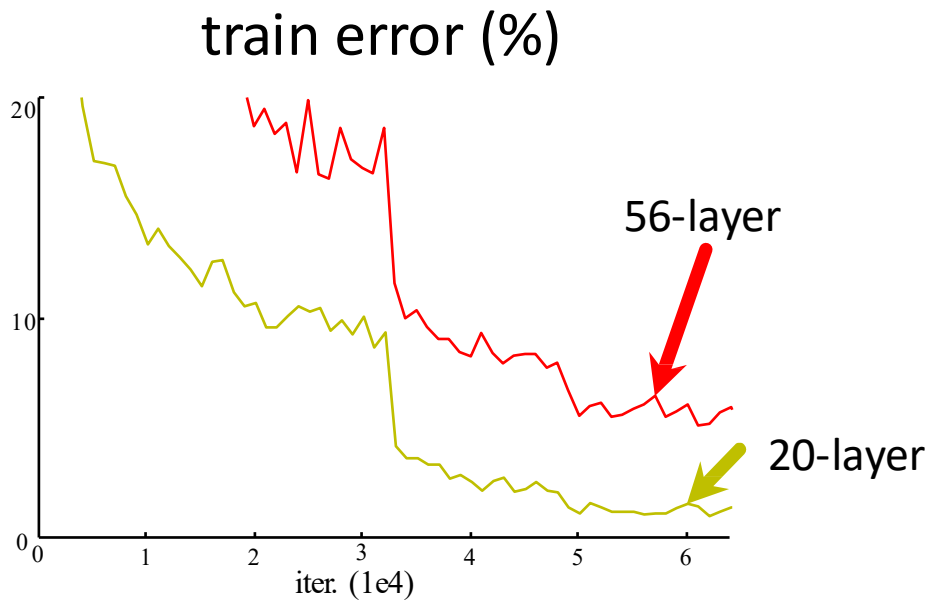


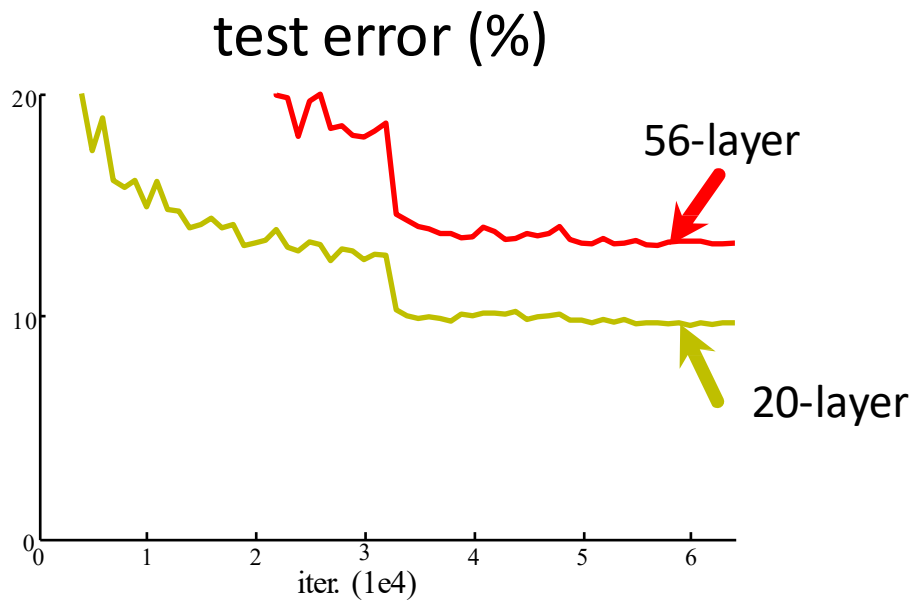
Source: cs231n

Simply Stacking Layers?

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

CIFAR-10





This is not overfitting. This is an optimization problem, deeper model hard to optimize

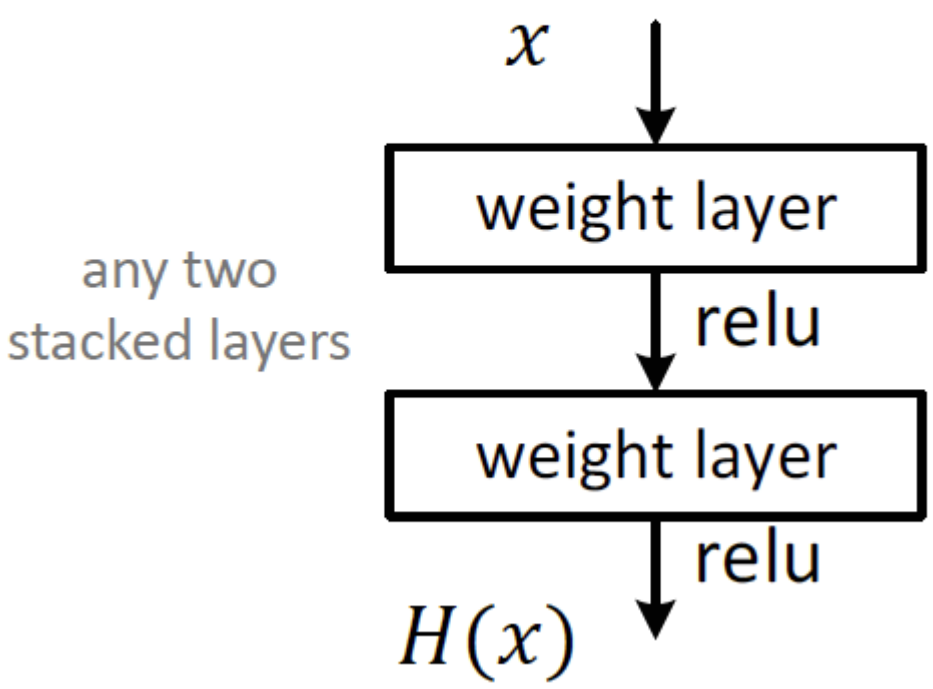
ResNet

- Hypothesis:
  - the problem is an optimization problem, deeper models are harder to optimize

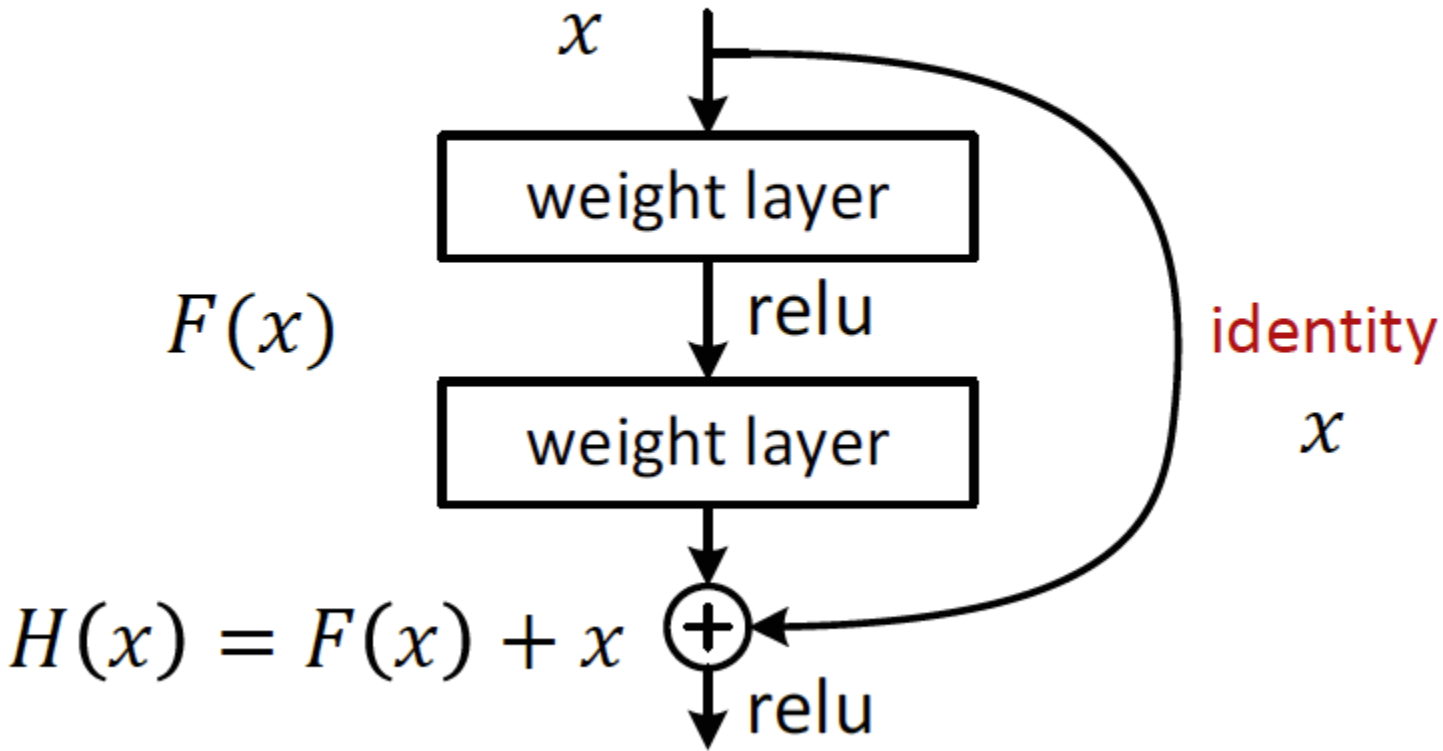
- The deeper model should be able to perform at least as well as the shallower model.
- A solution by construction is copying the learned layers from the shallower model and setting additional layers to identity mapping.

- Use network layers to fit a residual mapping instead of directly trying to fit a desired underlying mapping
  - If identity were optimal, easy to set weights as 0
  - If optimal mapping is closer to identity, easier to find small fluctuations

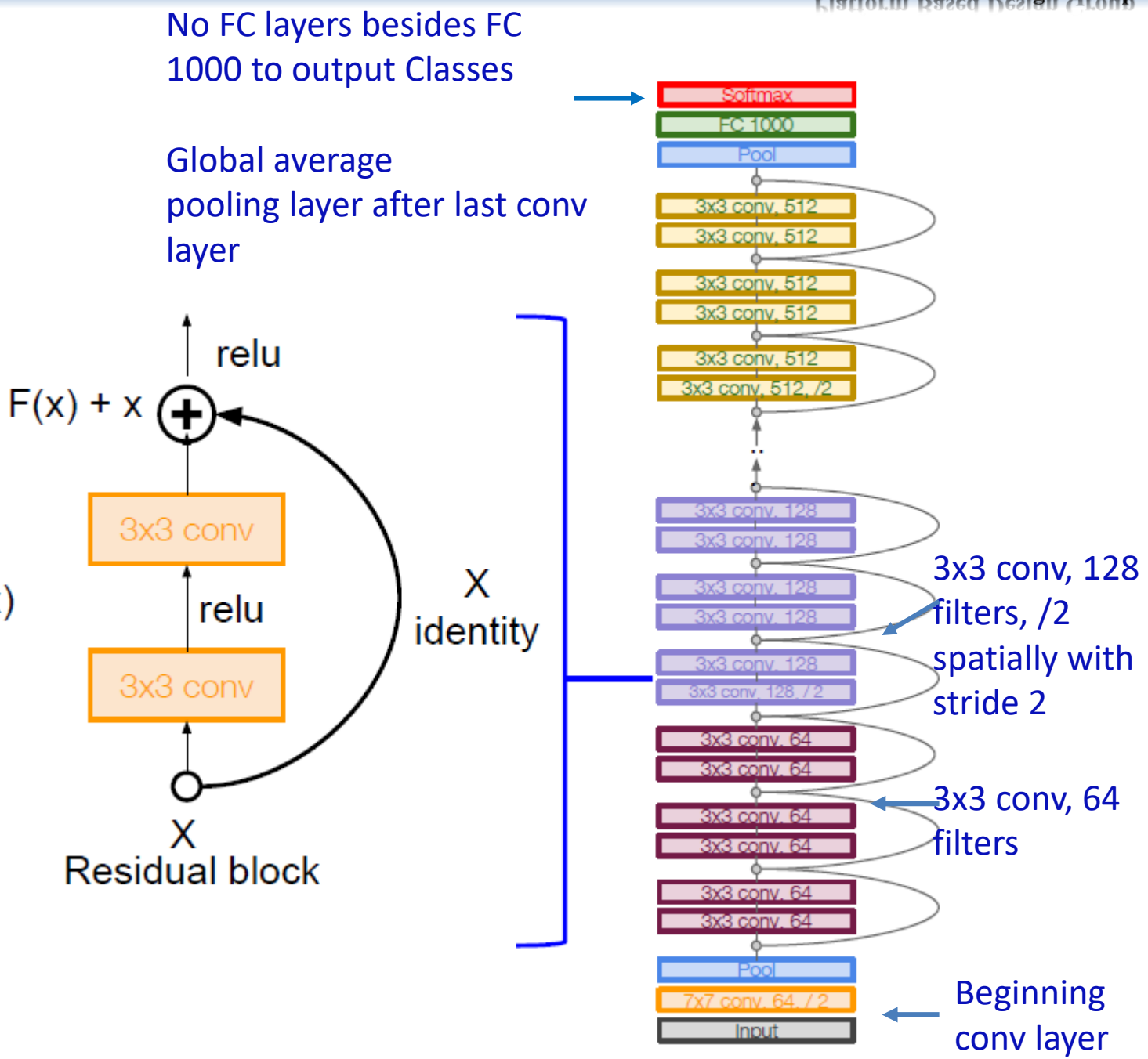
- Plaint net



- Residual net



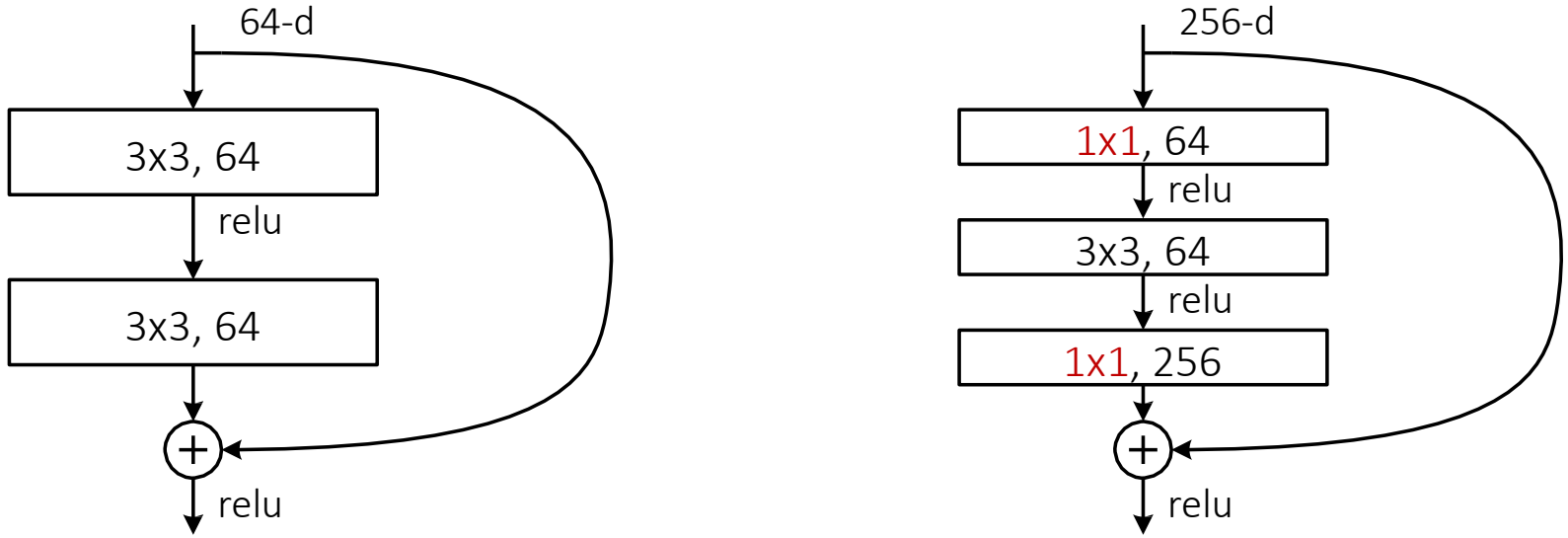
- Full ResNet architecture:
  - Stack residual blocks
  - Every residual block has two 3x3 conv layers
  - Periodically, double # of filters and downsample spatially using stride 2 (/2 in each dimension)
  - Additional conv layer at the beginning
  - No FC layers at the end(only FC 1000 to output classes)



Source: cs231n

ImageNet experiments

- A practical design of going deeper
  - For deeper networks (ResNet-50+), use “bottleneck”layer to improve efficiency(similar to GoogLeNet)

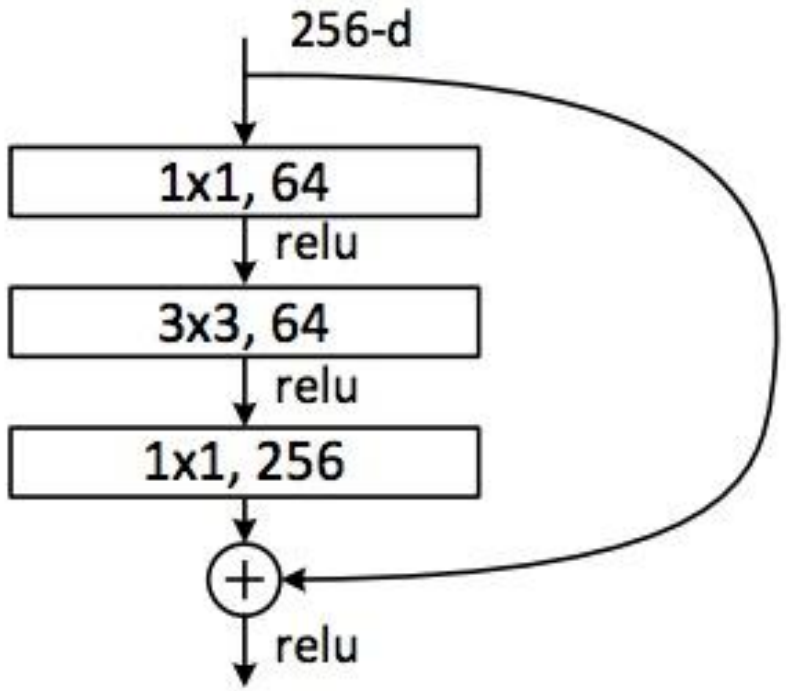


Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. “Deep Residual Learning for Image Recognition”. CVPR 2016.

ResNet

Deeper residual module (bottleneck)

A practical design of going deeper



- Directly performing 3x3 convolutions with 256 feature maps at input and output:
  256 x 256 x 3 x 3 ~ 600K operations
- Using 1x1 convolutions to reduce 256 to 64 feature maps, followed by 3x3 convolutions, followed by 1x1 convolutions to expand back to 256 maps:
  256 x 64 x 1 x 1 ~ 16K
  64 x 64 x 3 x 3 ~ 36K
  64 x 256 x 1 x 1 ~ 16K
  Total: ~70K

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, Deep Residual Learning for Image Recognition, CVPR 2016 (Best Paper)

ResNet

- Architectures for ImageNet:

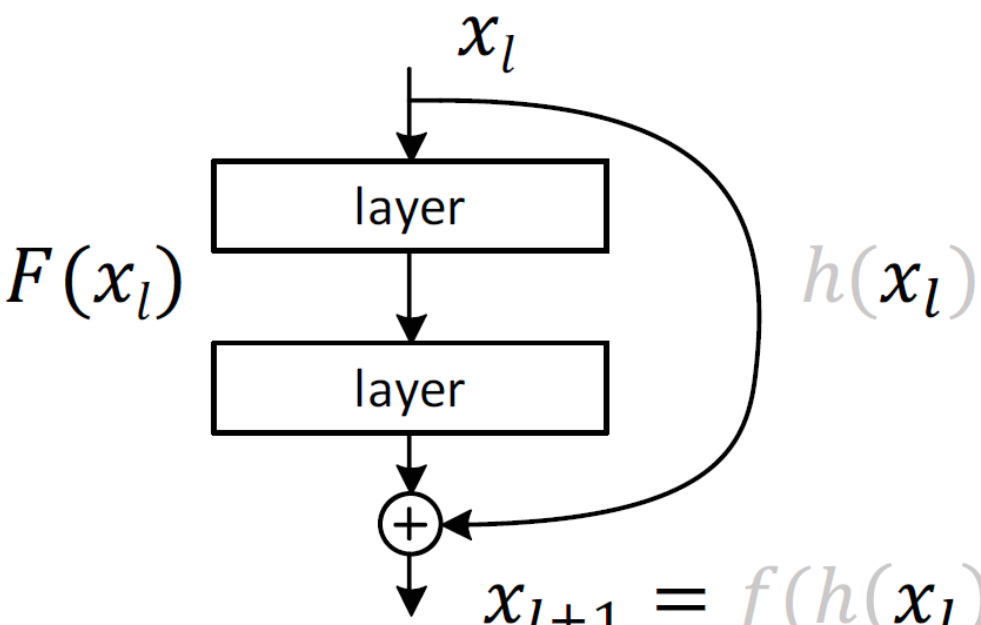
layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
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

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, Deep Residual Learning for Image Recognition, CVPR 2016 (Best Paper)

ResNet

- Training ResNet in practice:
  - Batch Normalization after every CONV layer
  - Xavier/2 initialization from He et al.
  - SGD + Momentum (0.9)
  - Learning rate: 0.1, divided by 10 when validation error plateaus
  - Mini-batch size 256
  - Weight decay of 1e-5
  - No dropout used

On the Importance of Identity Mapping From 100 layers to 1000 layers



- shortcut mapping: h = identity
- ~~after-add mapping: f = ReLU~~
- What if f = identity?

Identity Mapping: Smooth Forward and Backward Propagation

Very smooth forward propagation

$$x_{l+1} = x_l + F(x_l)$$

$$x_{l+2} = x_{l+1} + F(x_{l+1})$$

$$x_{l+2} = x_l + F(x_l) + F(x_{l+1})$$

Additive outcome

$$x_L = x_l + \sum_{i=l}^{L-1} F(x_i)$$

- in contrast to multiplicative: $x_L = \prod_{i=l}^{L-1} W_i x_l$

backward propagation

$$x_L = x_l + \sum_{i=l}^{L-1} F(x_i)$$

Additive gradients

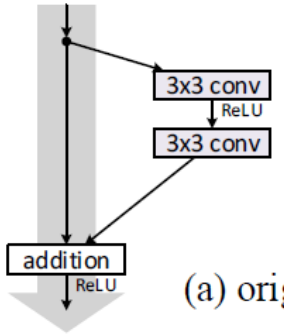
$$\frac{\partial E}{\partial x_l} = \frac{\partial E}{\partial x_L}\frac{\partial x_L}{\partial x_l} = \frac{\partial E}{\partial x_L}\left(1 + \frac{\partial}{\partial x_l}\sum_{i=1}^{L-1} F(x_i)\right)$$

- in contrast to multiplicative: $\frac{\partial E}{\partial x_l} = \prod_{i=l}^{L-1} W_i \frac{\partial E}{\partial x_L}$

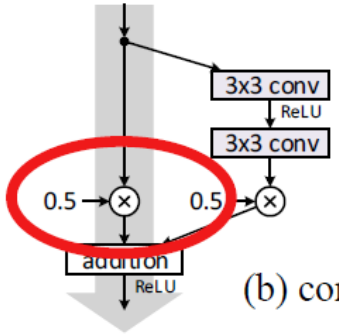
what if shortcut mapping $h \neq$ identity?

* ResNet-110 on CIFAR-10

$h(x) = x$
error: 6.6%



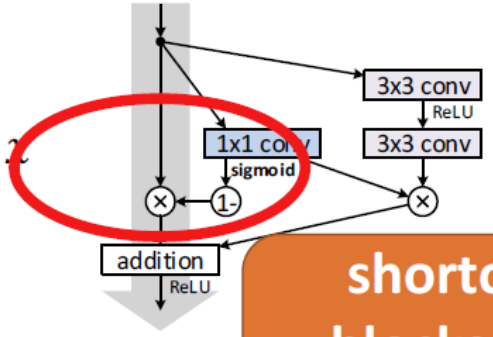
(a) original

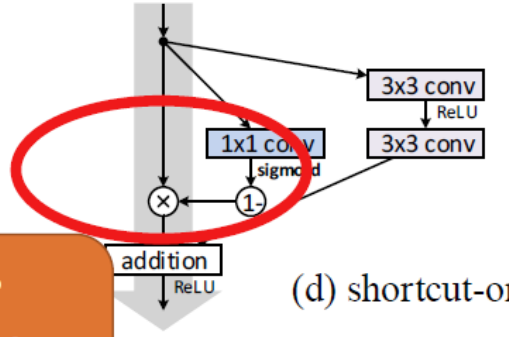


(b) constant scaling

$h(x) = 0.5x$
error: 12.4%

$h(x) = \text{gate} \cdot x$
error: 8.7%



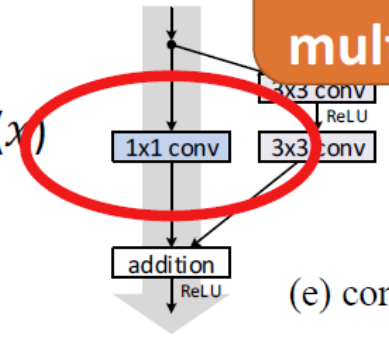


(d) shortcut-only gating

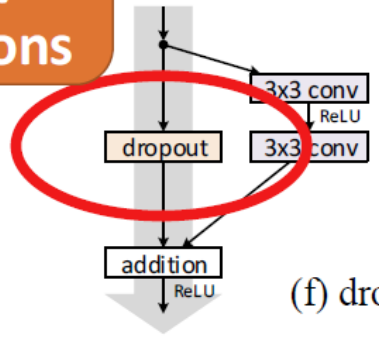
$h(x) = \text{gate} \cdot x$
error: 12.9%

shortcuts blocked by multiplications

$h(x) = \text{conv}(x)$
error: 12.2%



(e) conv shortcut

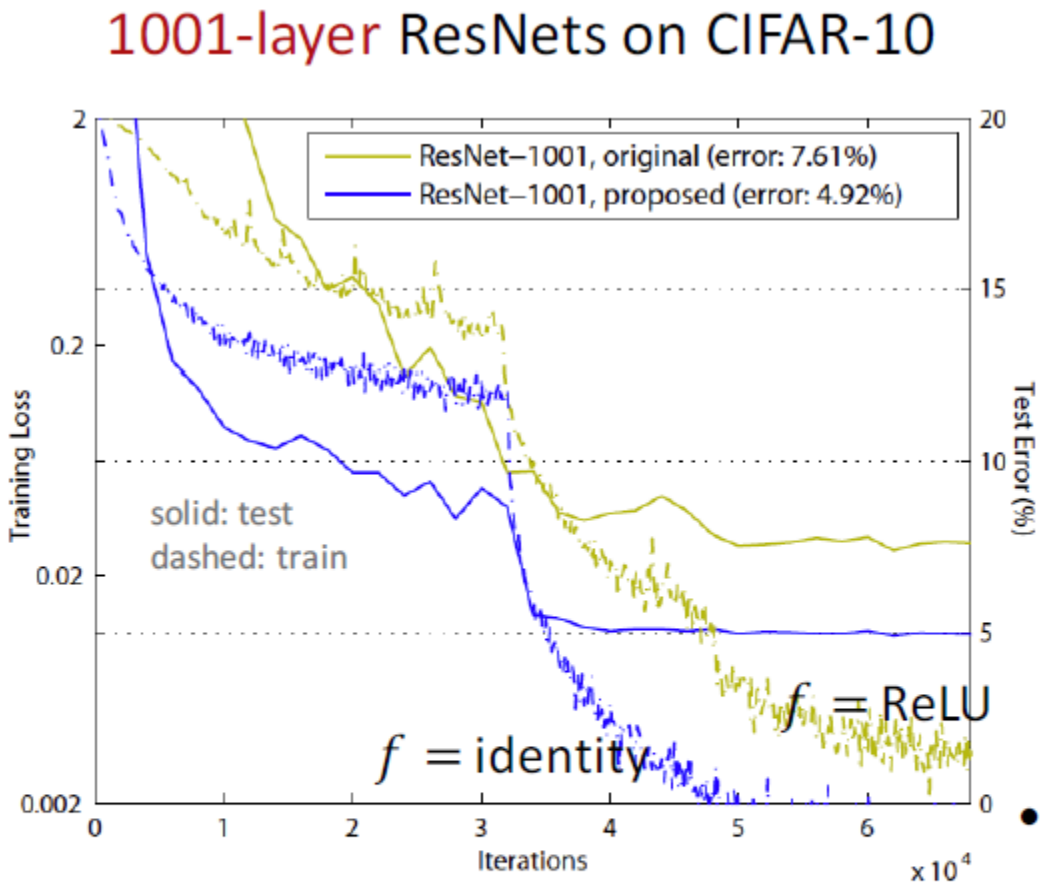


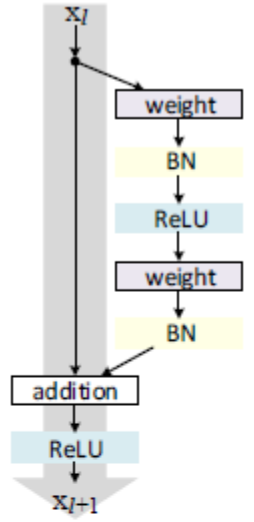
(f) dropout shortcut

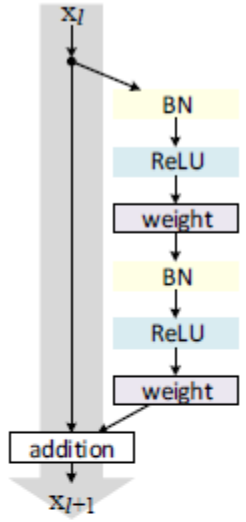
$h(x) = \text{dropout}(x)$
error: > 20%

Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. "Identity Mappings in Deep Residual Networks". arXiv 2016.

what if after-add mapping f is identity







moves activation to residual mapping pathway

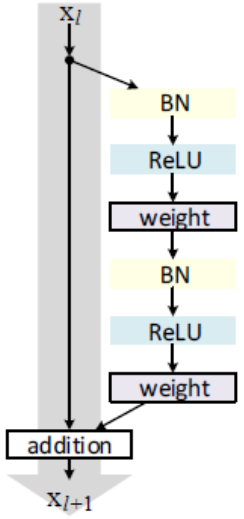
f = ReLU f = identity

- ReLU could block prop when there are 1000 layers
- pre-activation design eases optimization (and improves generalization; see paper)

Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. "Identity Mappings in Deep Residual Networks". arXiv 2016.

Observations on 1000-Layer Training

- Keep the shortest path as smooth as possible
  - by making h and f identity
  - forward/backward signals directly flow through this path
  - Features of any layers are additive outcomes
- 1000-layer ResNets can be easily trained and have better accuracy



WHY DO RESNET WORK?

Key takeaways

Residual networks contain many paths.

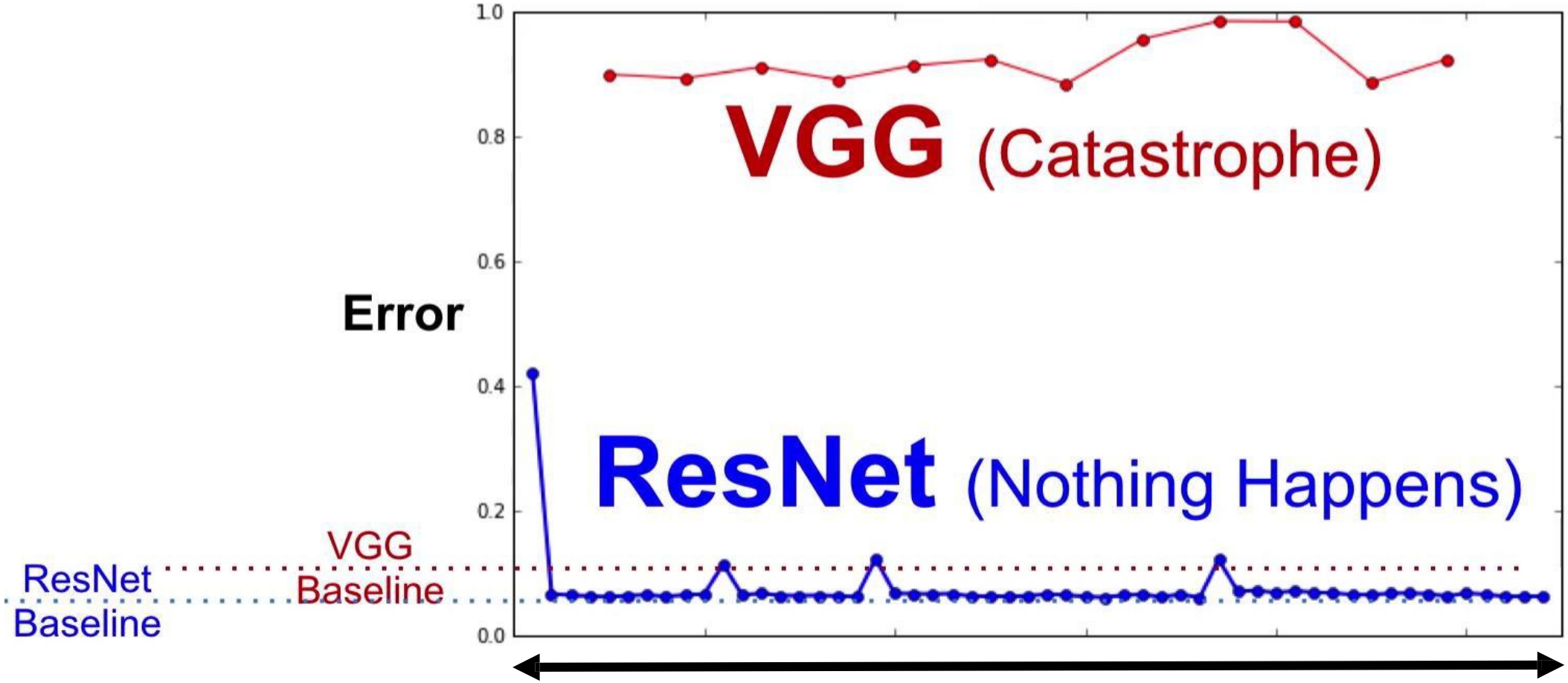
Previous networks have a single path.

Only short paths contribute gradient during training.

Vanishing gradient suppresses gradient from long paths.

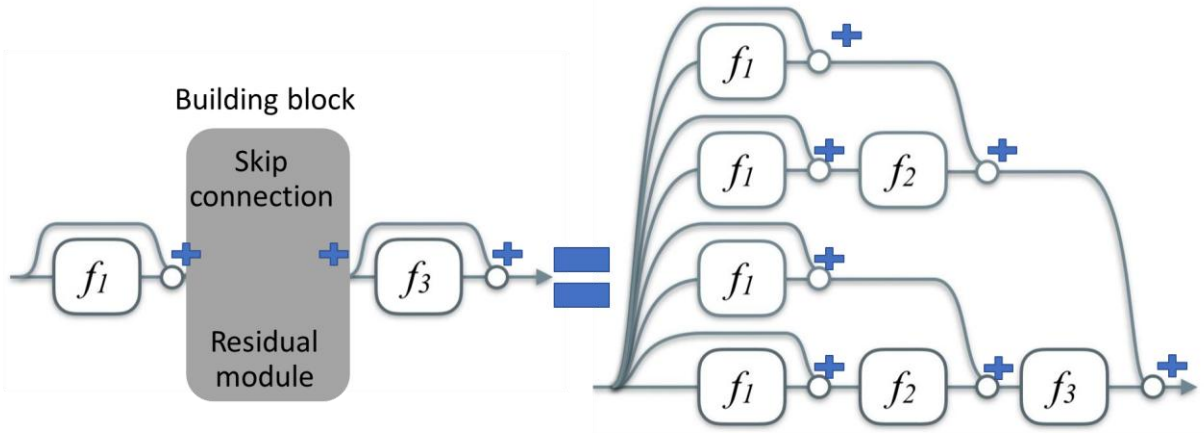
Residual networks contain many paths

For example, **what happens when we delete layers at test time?**



Slide from NIPS 2016 Spotlight Video

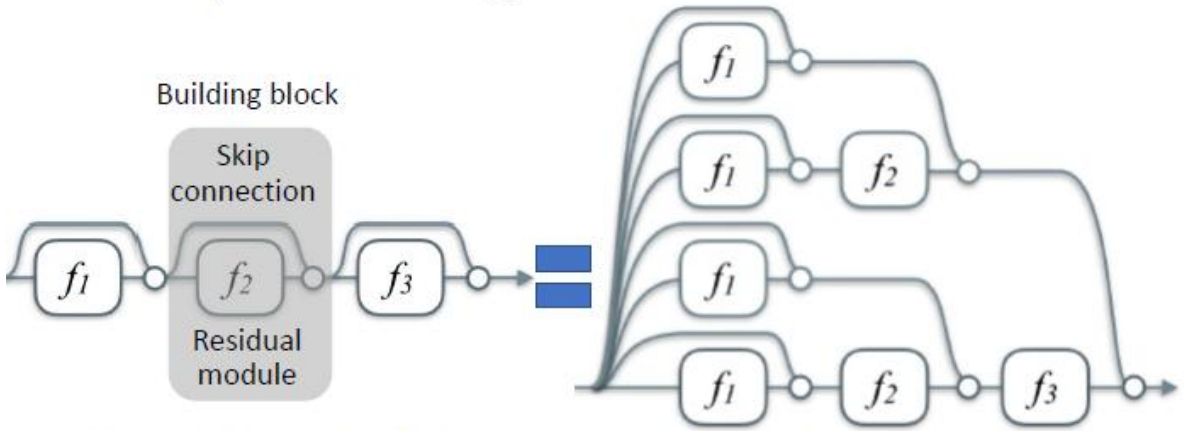
Why does this happen? **The «unraveled view»**



(a) Conventional 3-block residual network

Unraveled view of (a)

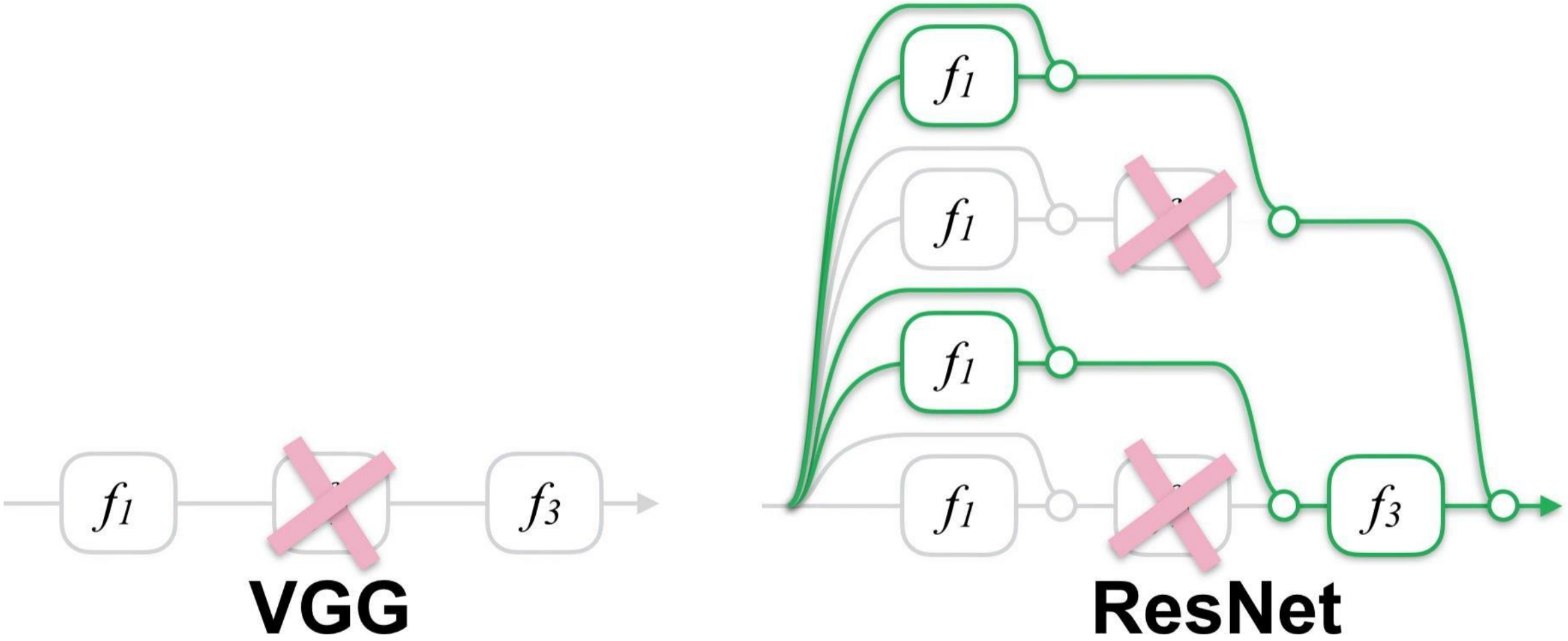
Slide adapted from NIPS 2016 Spotlight Video



(a) Conventional 3-block residual network

Unraveled view of (a)

Deletion of one layer

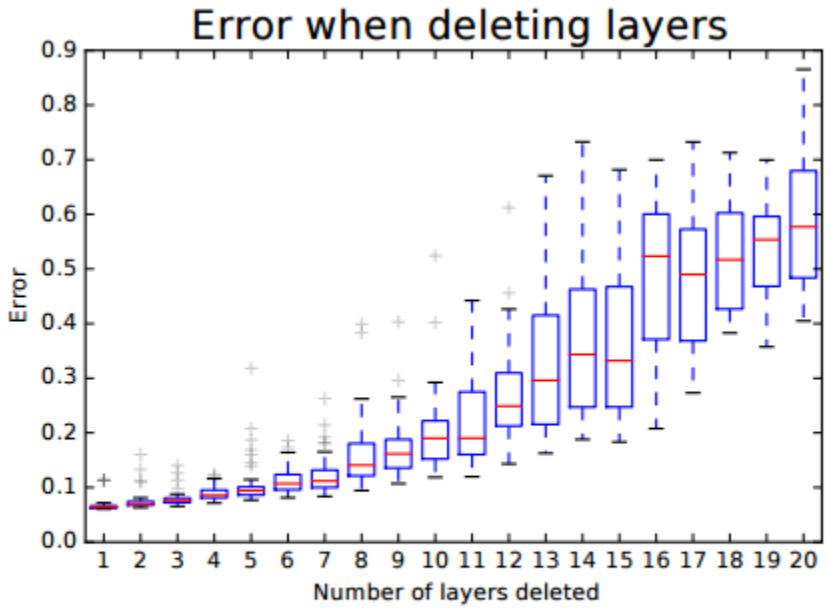


All paths are affected

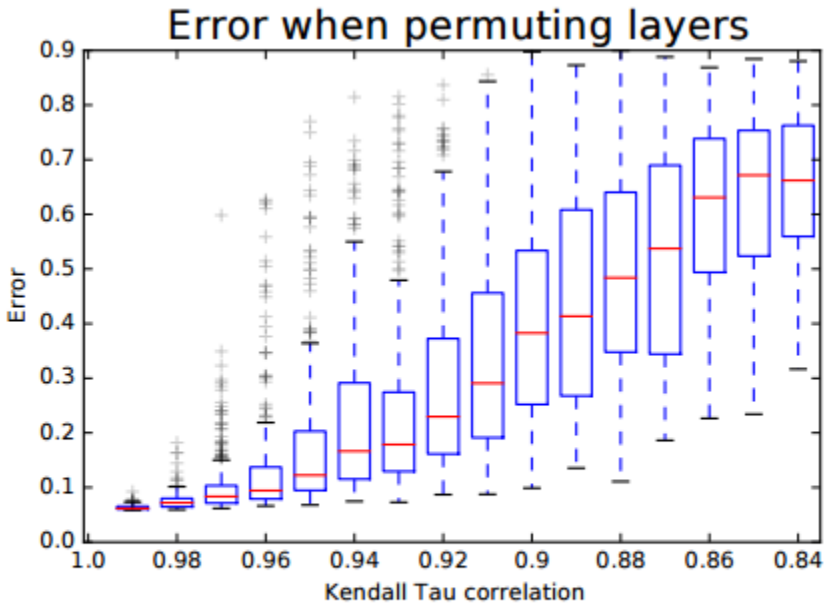
Only **half** of the paths are affected

Slide from NIPS 2016 Spotlight Video

Performance varies smoothly when deleting **several** layers or permuting layers



(a)



(b)

Figure 5: (a) Error increases smoothly when randomly deleting several modules from a residual

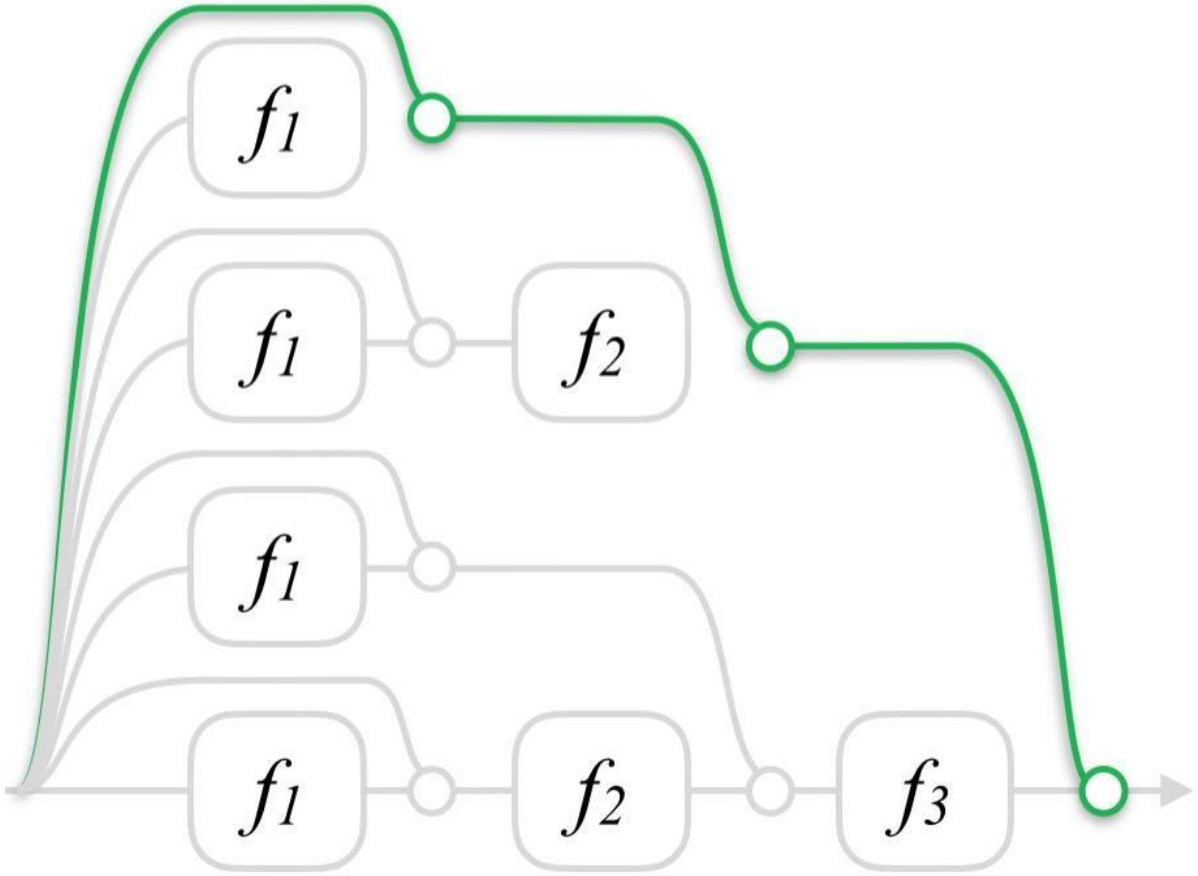
Slide from NIPS 2016 Spotlight Video

Summary

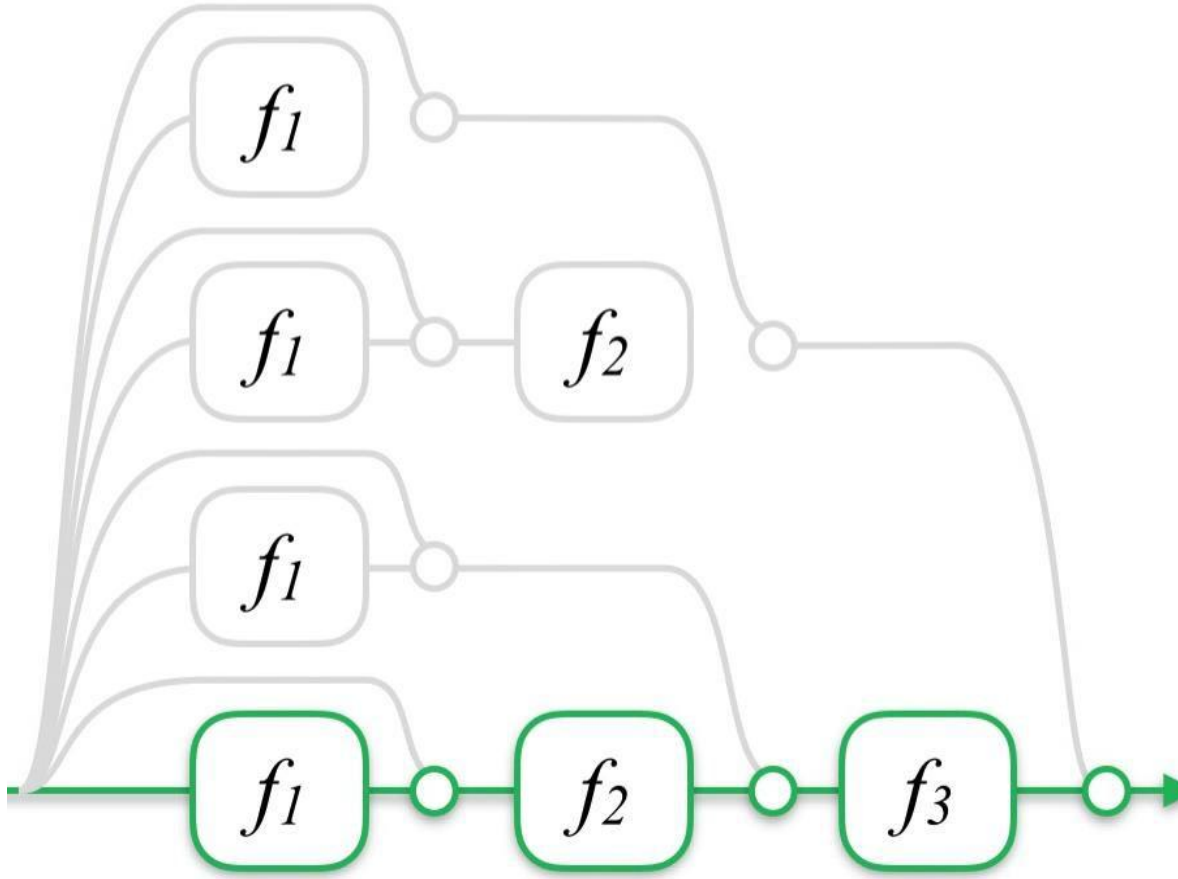
- residual networks
  - viewed as a collection of many paths, instead of a single ultra-deep network
    - Deeper model, more emsembled model
  - these paths do not strongly depend on each other, even though they are trained jointly: Ensemble-like behavior
  - only the short paths contribute gradient during training. Deep paths are not required during training.

Distribution of path length

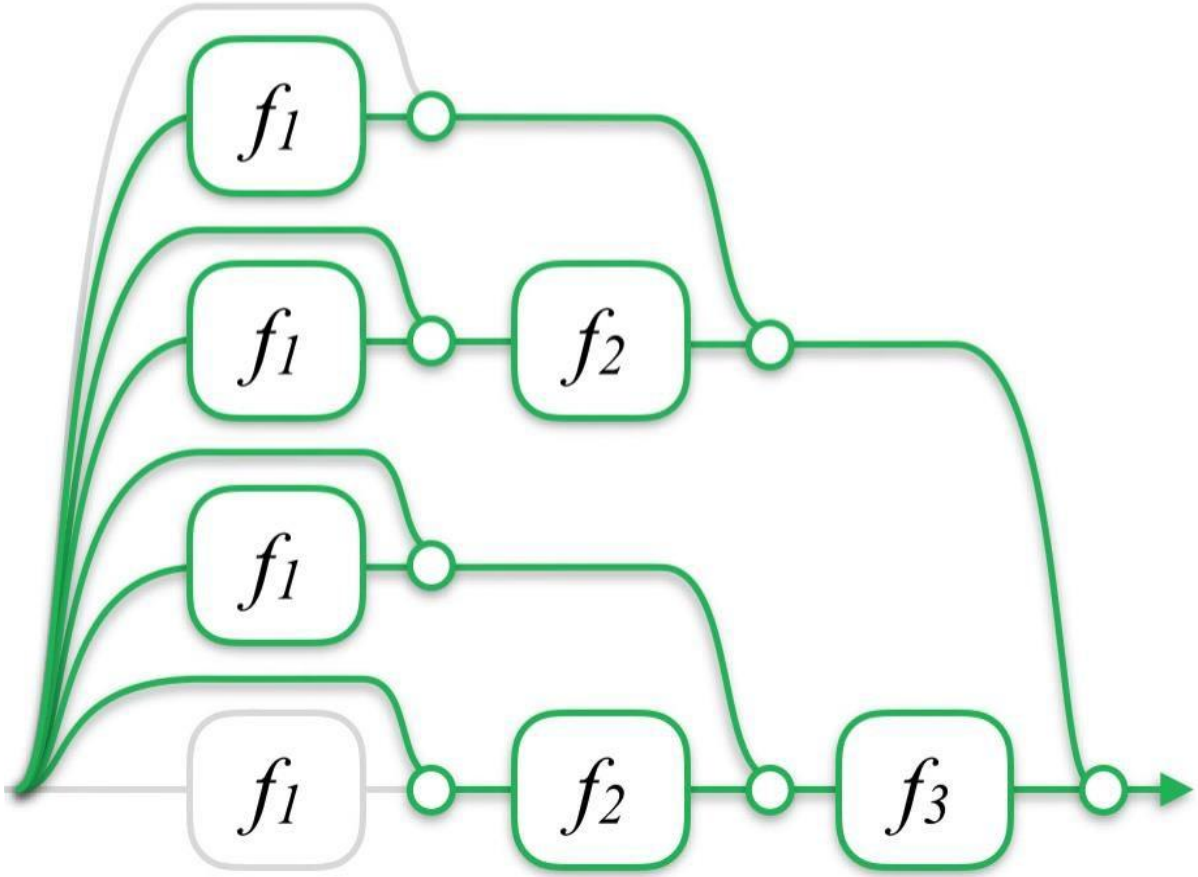
There are very few **short paths…**



And very few **long paths…**



Distribution of path length



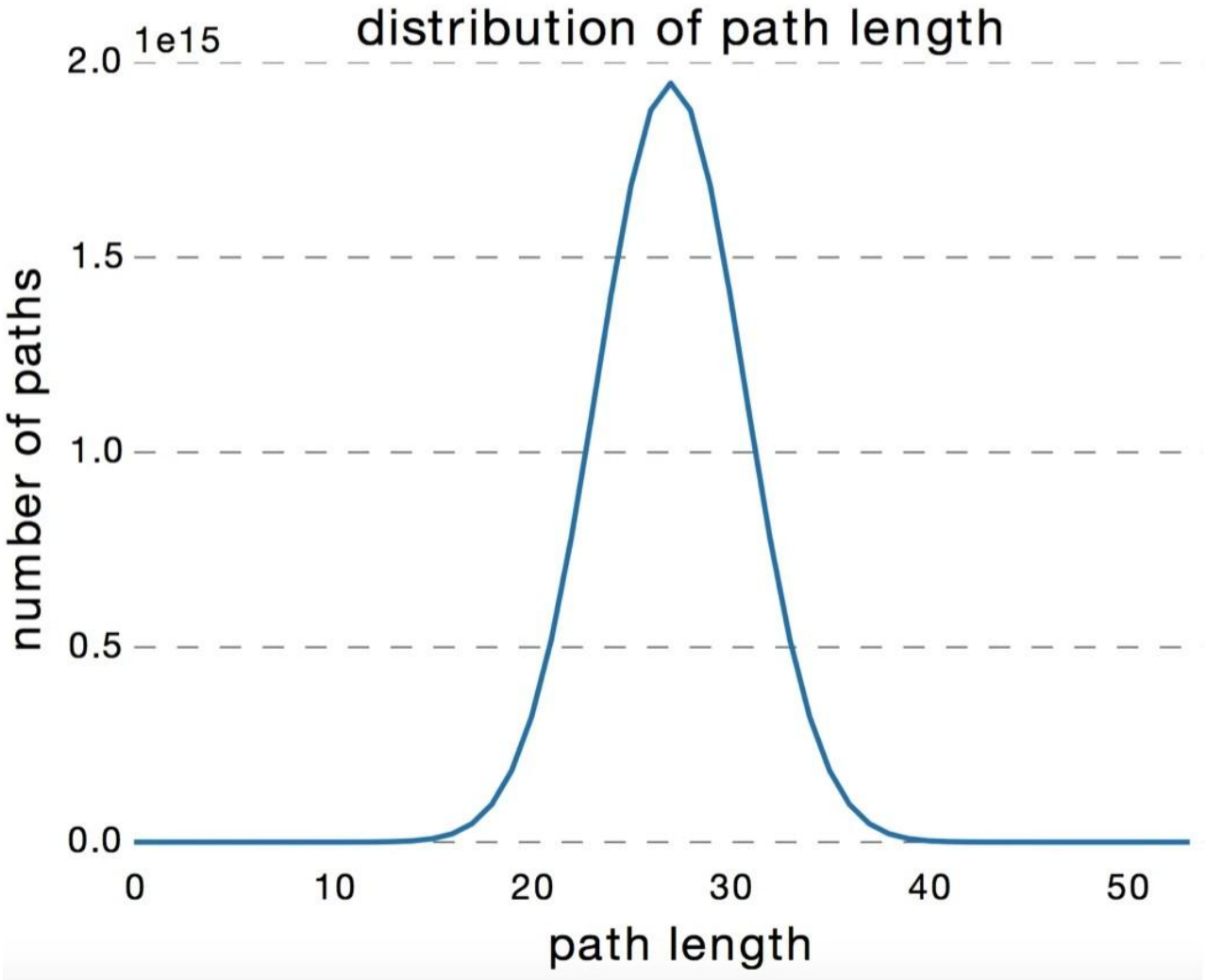
There are very few **short paths…**

And very few **long paths…**

Most paths are **medium length!**

Slide from NIPS 2016 Spotlight Video

Distribution of path length



There are very few **short paths…**

And very few **long paths…**

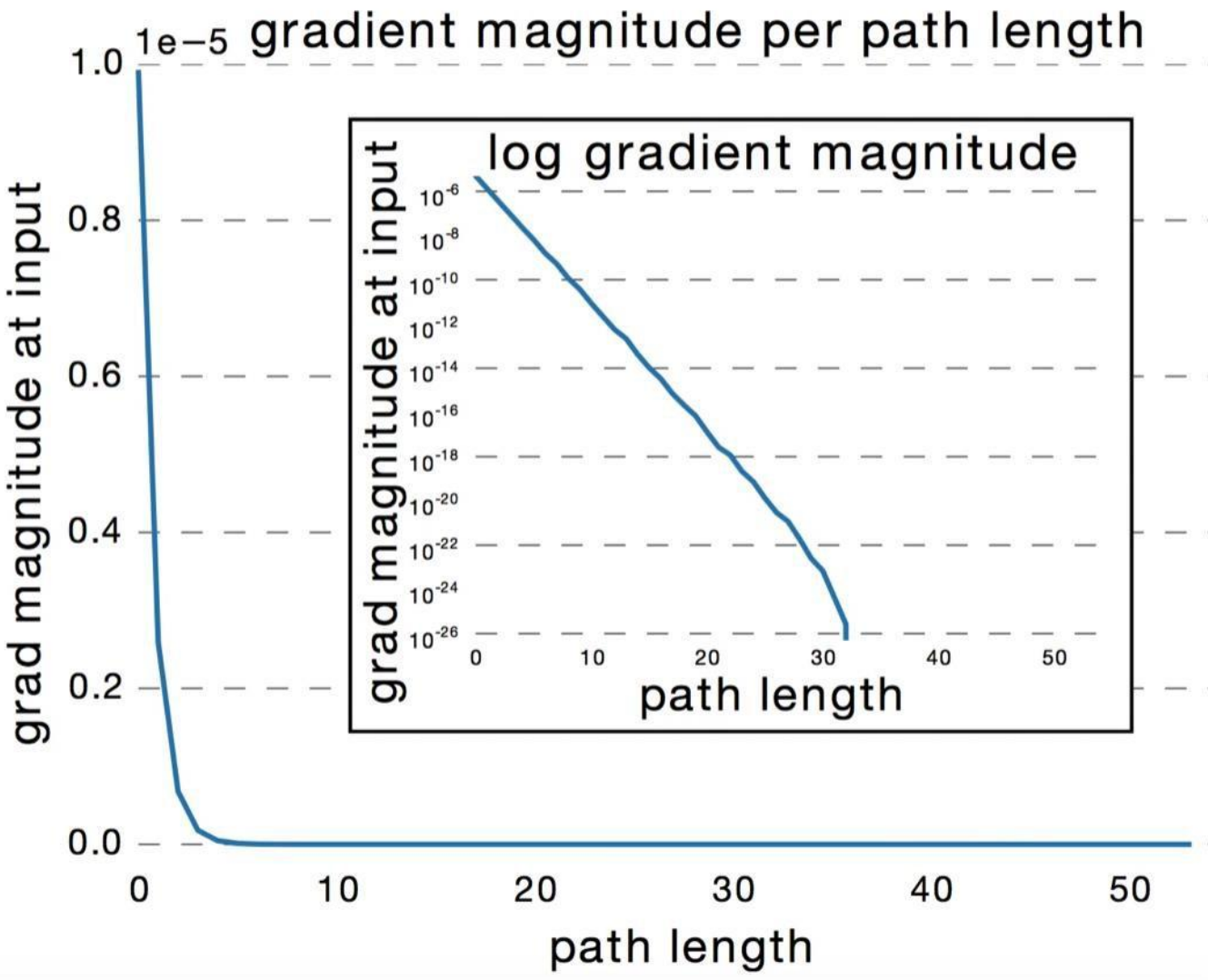
Most paths are **medium length!**

Paths length follows a **binomial distribution.**

Slide from NIPS 2016 Spotlight Video

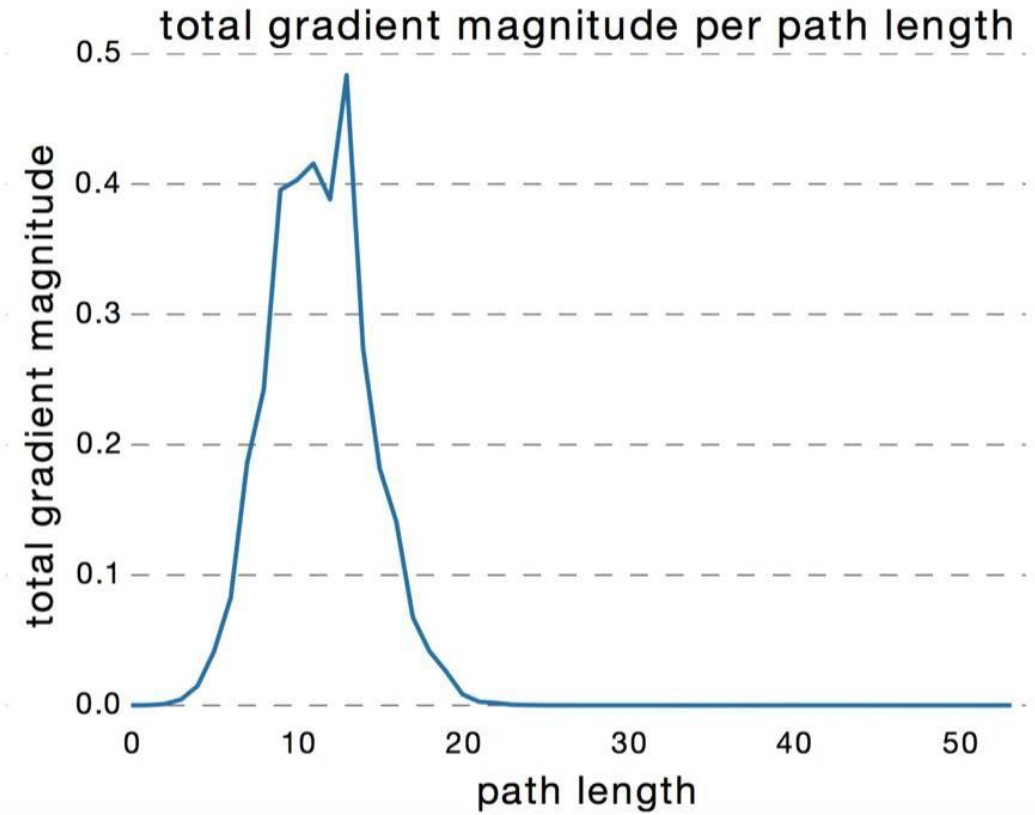
Vanishing gradient

The gradient magnitude **decreases exponentially** with increasing path length.



Slide from NIPS 2016 Spotlight Video

Gradient during training with respect to path lengths



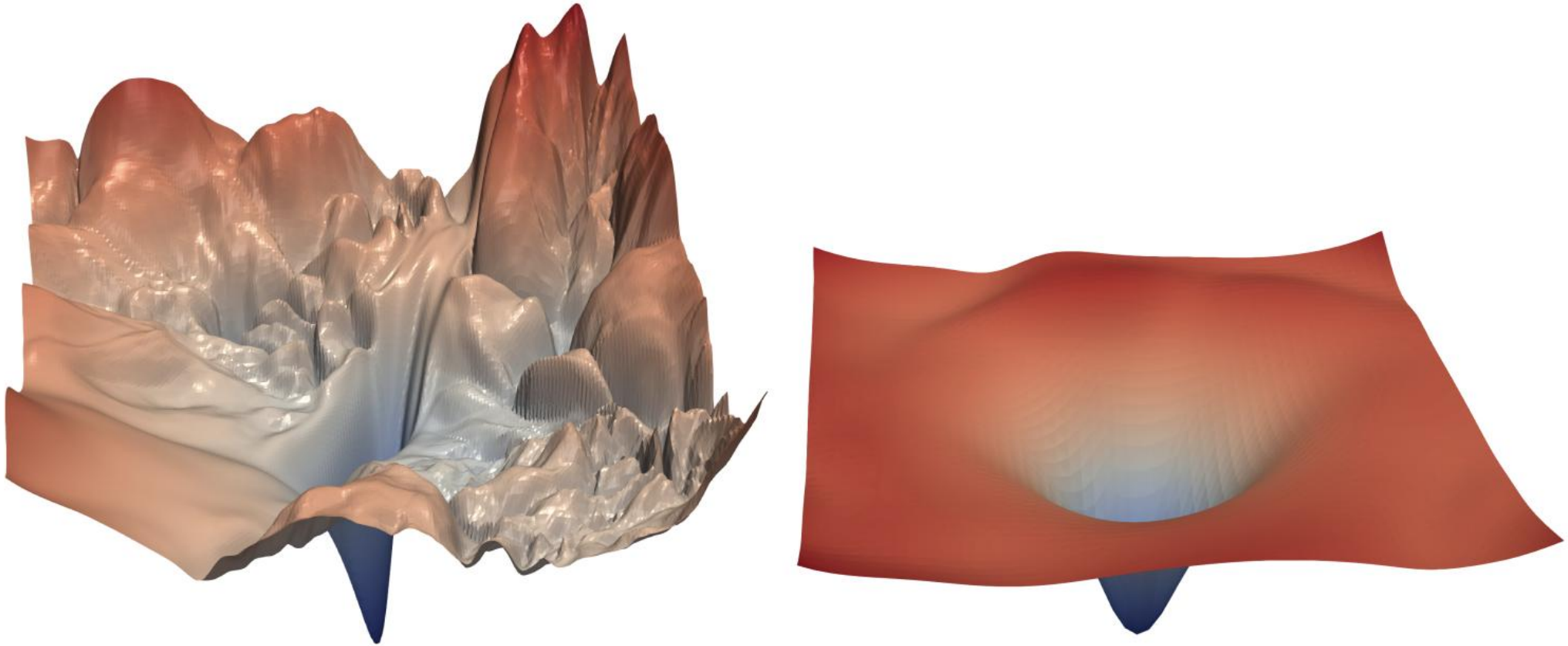
Combining the path length distribution and the vanishing gradients, one can observe that most of the gradient comes from relatively short paths.

Slide from NIPS 2016 Spotlight Video

Summary

- residual networks
  - viewed as a collection of many paths, instead of a single ultra-deep network
  - these paths do not strongly depend on each other, even though they are trained jointly: Ensemble-like behavior
  - only the short paths contribute gradient during training. Deep paths are not required during training.
- It is **not depth, but the ensemble** that makes residual networks strong
  - Residual networks push the limits of network multiplicity, not network depth
  - that **multiplicity**, the network's expressability in the terms of the number of paths, plays a key role

Loss Surface



(a) without skip connections

(b) with skip connections

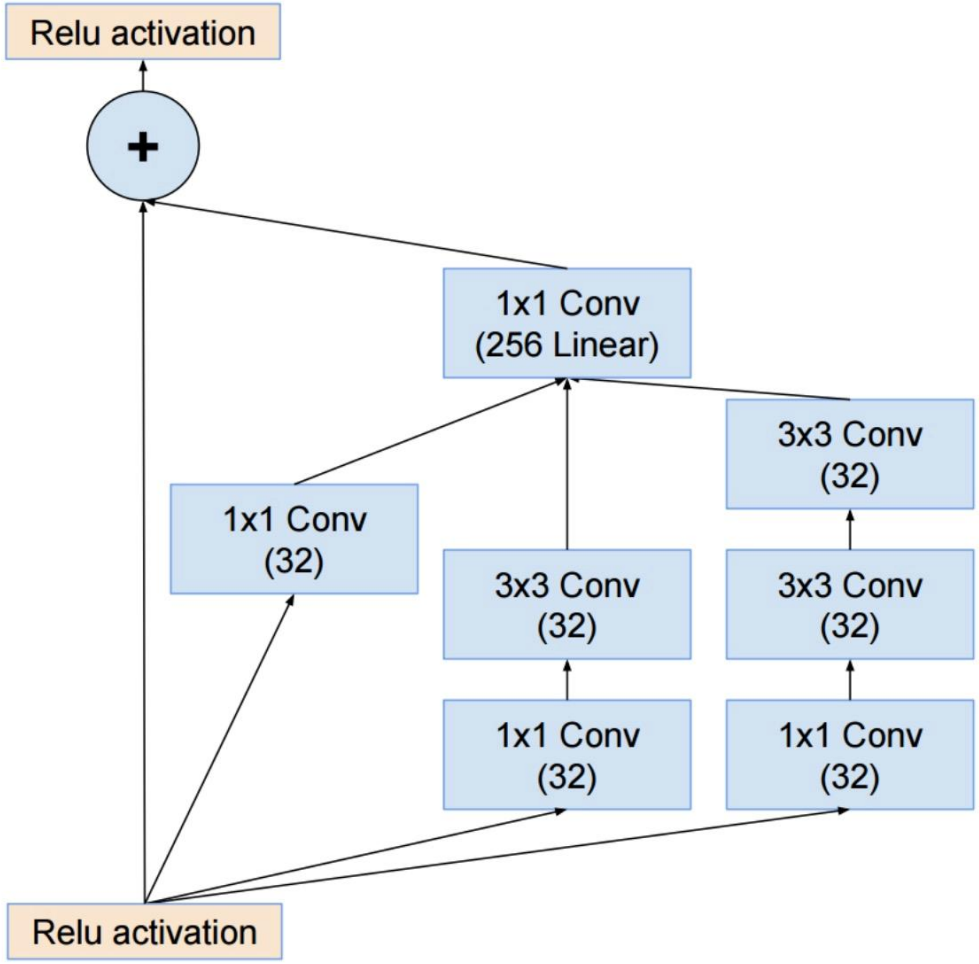
Figure 1: The loss surfaces of ResNet-56 with/without skip connections. The proposed filter normalization scheme is used to enable comparisons of sharpness/flatness between the two figures.

IMPROVING RESNET AND OTHER CNN ARCHITECTURES

Improving ResNets

- ResNets + X
  - ResNets + Inception
  - ResNets + Group convolution
    - Aggregated Residual Transformations for Deep Neural Networks (ResNeXt)
  - ResNets + dropout
    - Deep Networks with Stochastic Depth
  - ResNets + self-attention: transformer
  - Wide Residual Networks
    - Instead of increasing depth only
- Extreme skip connection
  - DenseNet

Inception v4: Inception + ResNet



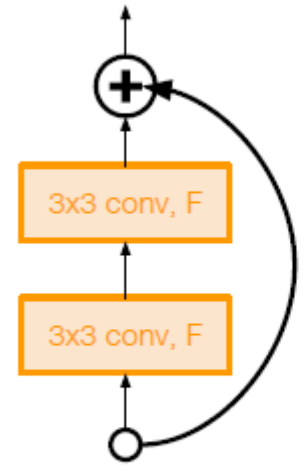
C. Szegedy et al., Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning, arXiv 2016

Wide Residual Network

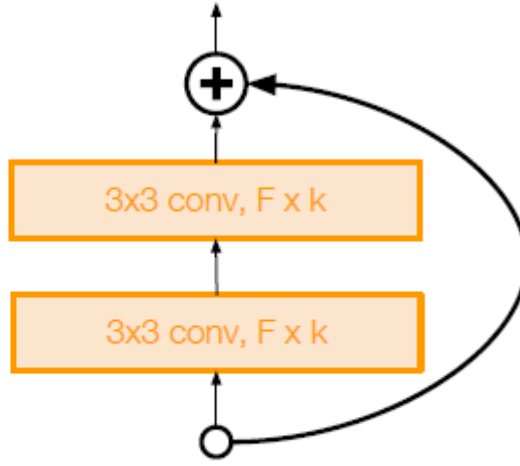
- Problems of over 1000 layers of residual network
  - Little improvement but at double number of layers => slow to train
- Argues that residuals are the important factor, not depth
- Solution: decrease depth and increase width of residual network
  - Wider layer is more suitable for parallel GPU computation
  - Use dropout for regularization
  - 16-layer WRN is better than over 1000 ResNet
  - 50 layer WRN is better than 153 layer ResNet

Model	top-1 err, %	top-5 err, %	#params	time/batch 16
ResNet-50	24.01	7.02	25.6M	49
ResNet-101	22.44	6.21	44.5M	82
ResNet-152	22.16	6.16	60.2M	115
WRN-50-2-bottleneck	21.9	6.03	68.9M	93
pre-ResNet-200	21.66	5.79	64.7M	154

imagetnet



Basic residual block



Wide residual block

B(3, 3)

group name	output size	block type = $B(3,3)$
conv1	32 × 32	[3×3, 16]
conv2	32×32	$\begin{bmatrix} 3\times3, 16\times k \\ 3\times3, 16\times k \end{bmatrix} \times N$
conv3	16×16	$\begin{bmatrix} 3\times3, 32\times k \\ 3\times3, 32\times k \end{bmatrix} \times N$
conv4	8×8	$\begin{bmatrix} 3\times3, 64\times k \\ 3\times3, 64\times k \end{bmatrix} \times N$
avg-pool	1 × 1	[8 × 8]

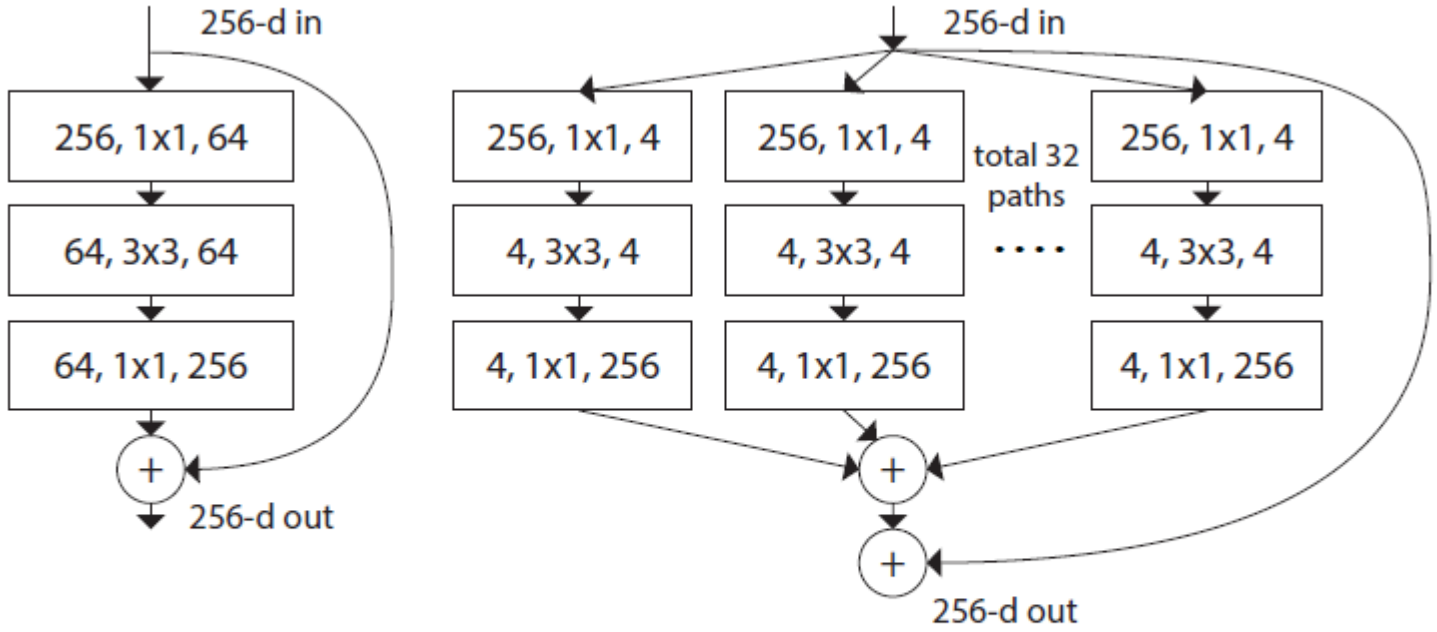
	depth-k	# params	CIFAR-10	CIFAR-100
NIN [20]			8.81	35.67
DSN [19]			8.22	34.57
FitNet [24]			8.39	35.04
Highway [28]			7.72	32.39
ELU [5]			6.55	24.28
original-ResNet[11]	110	1.7M	6.43	25.16
	1202	10.2M	7.93	27.82
stoc-depth[14]	110	1.7M	5.23	24.58
	1202	10.2M	4.91	-
pre-act-ResNet[13]	110	1.7M	6.37	-
	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**

1. 寬度的增加提高了性能
2.增加深度和寬度都有好處，直到參數太大，regularization不夠
3.相同參數時，寬度比深度好訓練

Aggregated Residual Transformations for Deep Neural Networks (ResNeXt)

- Also from creators of ResNet
- Concept: VGG/ResNet(repeat) +
  - Googlenet (split-transform-merge):
    - too many hyperparameters, hard to adapt to other tasks
    - Parallel same residual block to reduce hyperparameters
- Better performance than ResNet under same no. of parameters

$C \cdot (256 \cdot d + 3 \cdot 3 \cdot d \cdot d + d \cdot 256)$



Equivalent Building Block

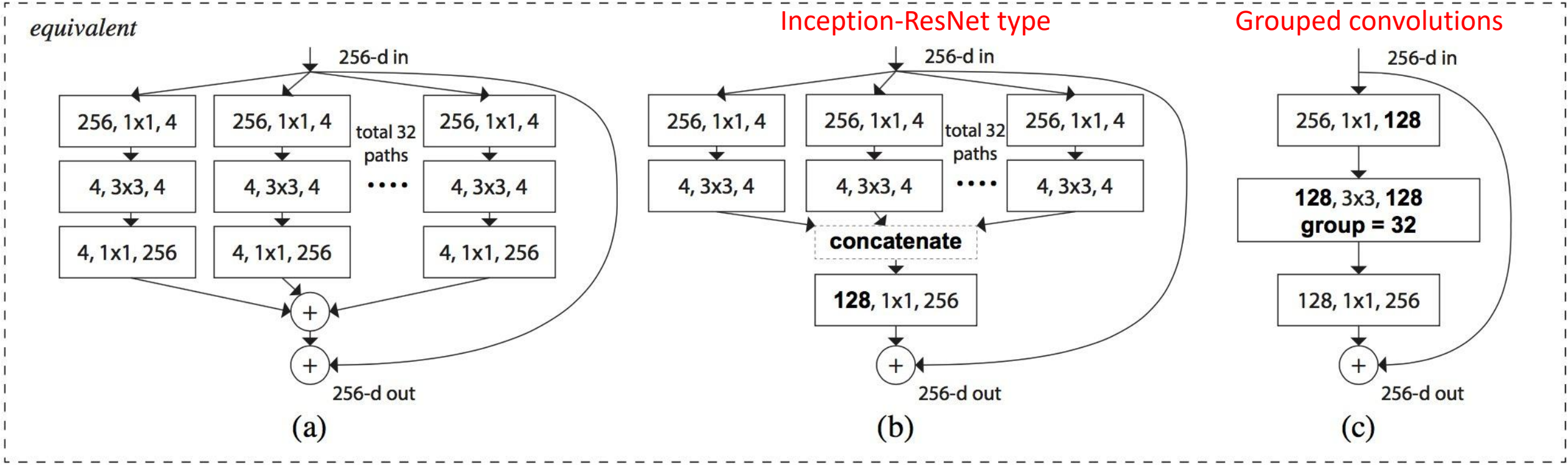


Figure 3. Equivalent building blocks of ResNeXt. **(a)**: Aggregated residual transformations, the same as Fig. 1 right. **(b)**: A block equivale to (a), implemented as early concatenation. **(c)**: A block equivalent to (a,b), implemented as grouped convolutions [23]. Notations in **bo** text highlight 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.		**25.5**×10^6	**25.0**×10^6
FLOPs		**4.1**×10^9	**4.2**×10^9

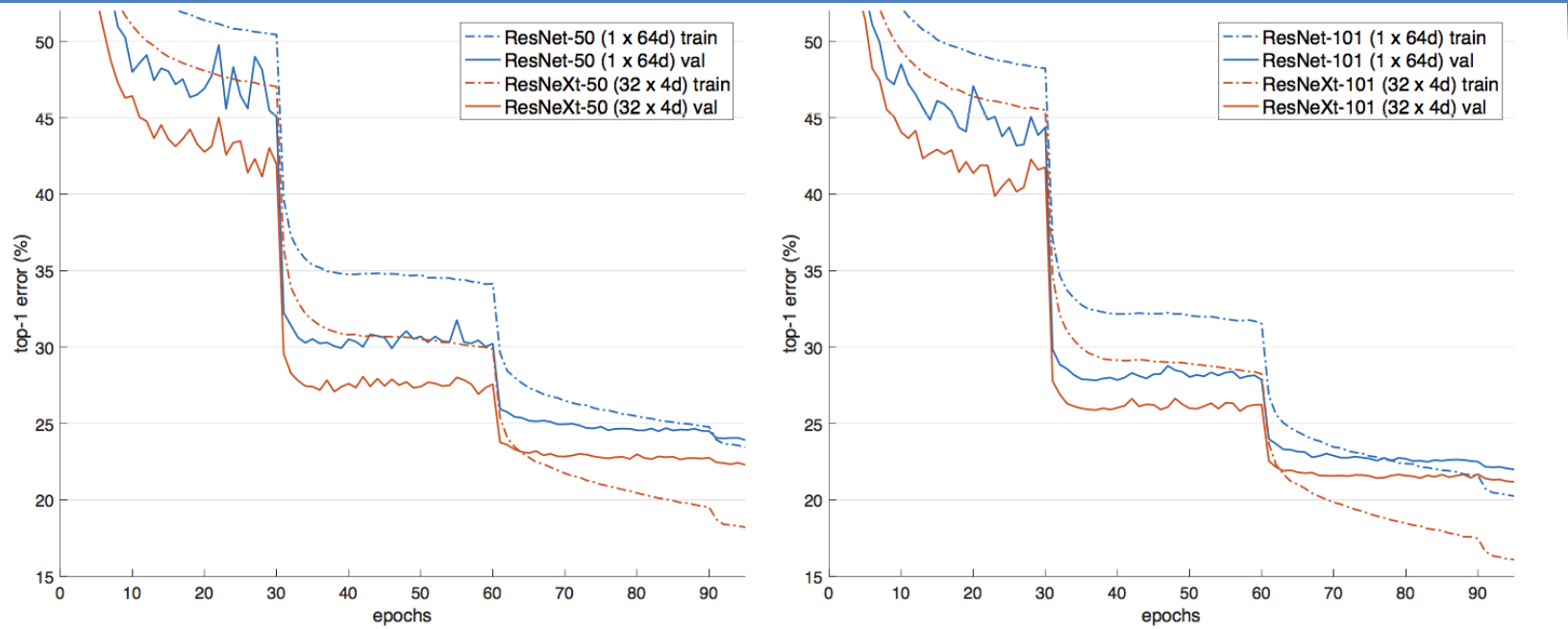


Figure 5. Training curves on ImageNet-1K. (**Left**): ResNet/ResNeXt-50 with preserved complexity (∼4.1 billion FLOPs, ∼25 million parameters); (**Right**): ResNet/ResNeXt-101 with preserved complexity (∼7.8 billion FLOPs, ∼44 million parameters).

	setting	top-1 error (%)
ResNet-50	1 × 64d	23.9
ResNeXt-50	2 × 40d	23.0
ResNeXt-50	4 × 24d	22.6
ResNeXt-50	8 × 14d	22.3
ResNeXt-50	32 × 4d	**22.2**
ResNet-101	1 × 64d	22.0
ResNeXt-101	2 × 40d	21.7
ResNeXt-101	4 × 24d	21.4
ResNeXt-101	8 × 14d	21.3
ResNeXt-101	32 × 4d	**21.2**

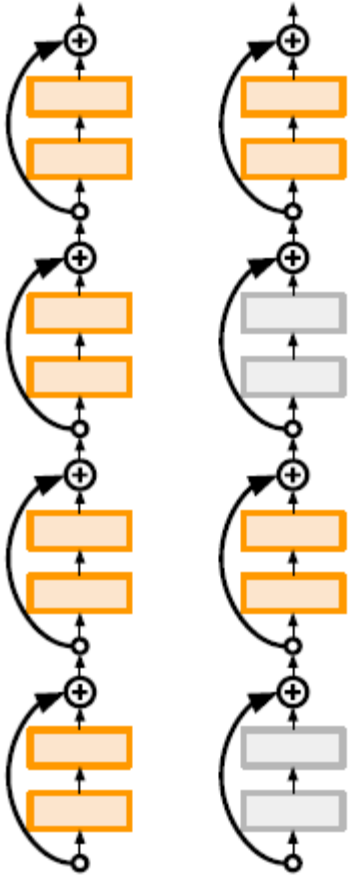
Table 3. Ablation experiments on ImageNet-1K. (**Top**): ResNet-50 with preserved complexity (∼4.1 billion FLOPs); (**Bottom**): ResNet-101 with preserved complexity (∼7.8 billion FLOPs). The error rate is evaluated on the single crop of 224×224 pixels.

	setting	top-1 err (%)	top-5 err (%)
1× complexity references:			
ResNet-101	1 × 64d	22.0	6.0
ResNeXt-101	32 × 4d	21.2	5.6
2× complexity models follow:			
ResNet-**200** [14]	1 × 64d	21.7	5.8
ResNet-101, wider	1 × **100**d	21.3	5.7
ResNeXt-101	**2** × 64d	20.7	5.5
ResNeXt-101	**64** × 4d	**20.4**	**5.3**

Table 4. Comparisons on ImageNet-1K when the number of FLOPs is increased to 2× of ResNet-101's. The error rate is evaluated on the single crop of 224×224 pixels. The highlighted factors are the factors that increase complexity.

Deep Networks with Stochastic Depth

- Motivation
  - reduce vanishing gradients and training time through short networks during training
- How
  - Randomly drop a subset of layers during each training pass
  - Bypass with identity function
  - Use full deep network at test time
- Implicit ensembles



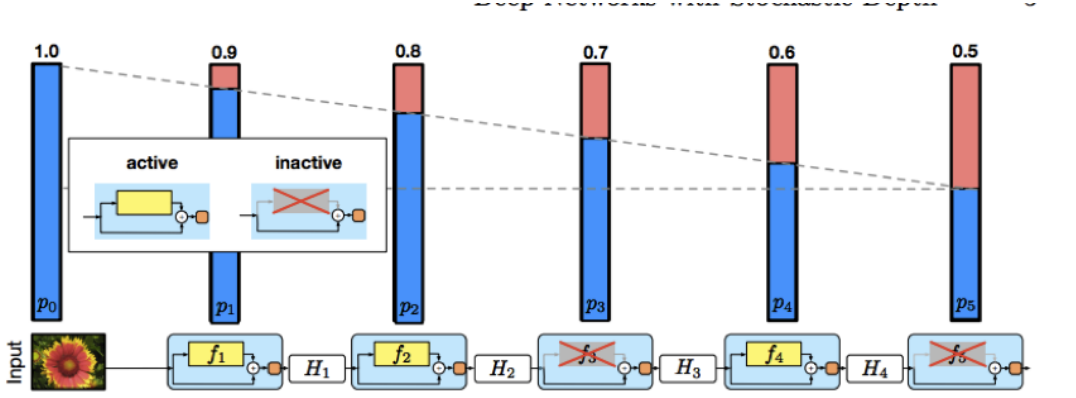


Fig. 2. The linear decay of p_ℓ illustrated on a ResNet with stochastic depth for $p_0 = 1$ and $p_L = 0.5$. Conceptually, we treat the input to the first ResBlock as H_0, which is

DenseNet

- Densely Connected Convolutional Networks
  - Dense blocks
    - where each layer is connected to every other layer in feedforward fashion
  - Alleviates vanishing gradient, strengthens feature propagation, encourages feature reuse, substantially reduce the number of parameters

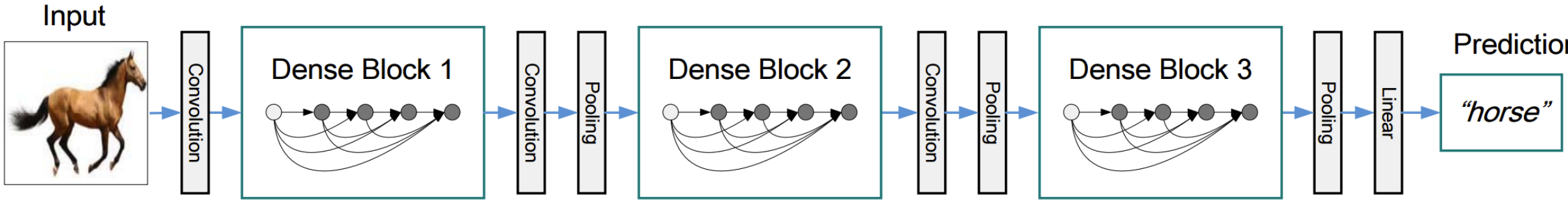


Figure 2. A deep DenseNet with three dense blocks. The layers between two adjacent blocks are referred to as transition layers and chang feature map sizes via convolution and pooling.

Source: 2017, CVPR, Densely Connected Convolutional Networks

DenseNet

- Dense block

- Every layer is connected to all other layers.
- For each layer, the feature-maps of all preceding layers are used as inputs, and its own feature-maps are used as inputs into all subsequent layers.

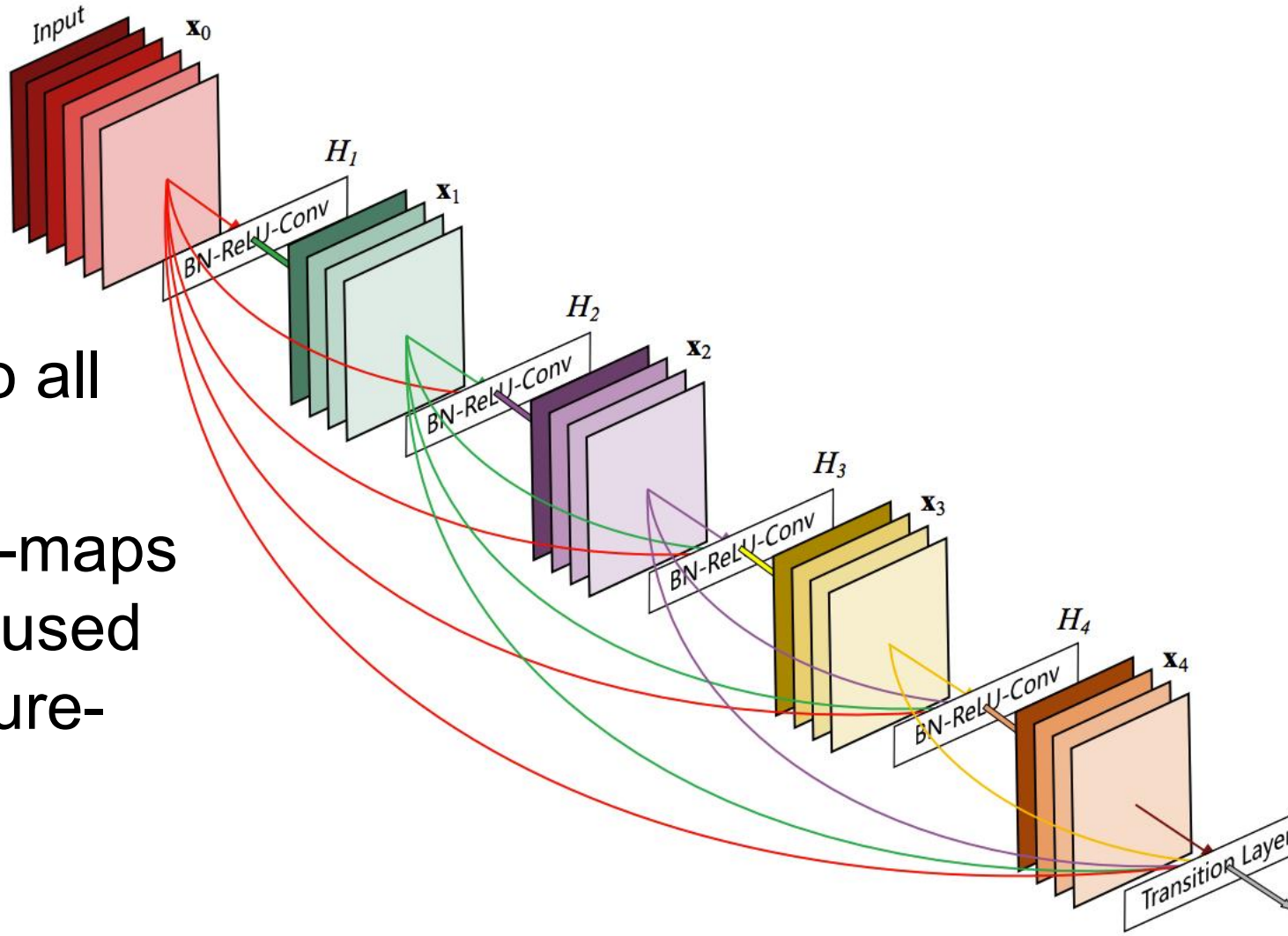
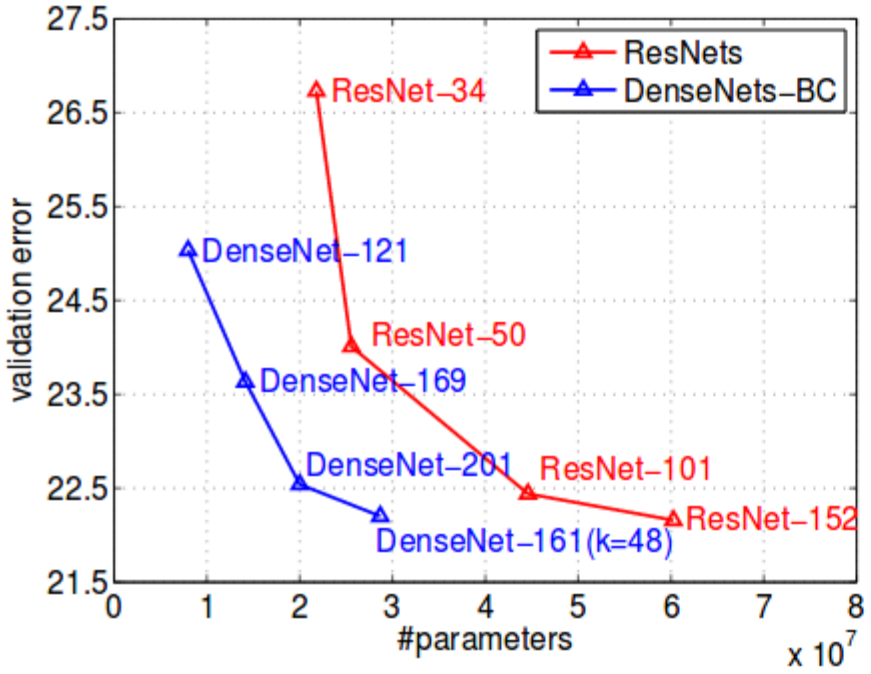
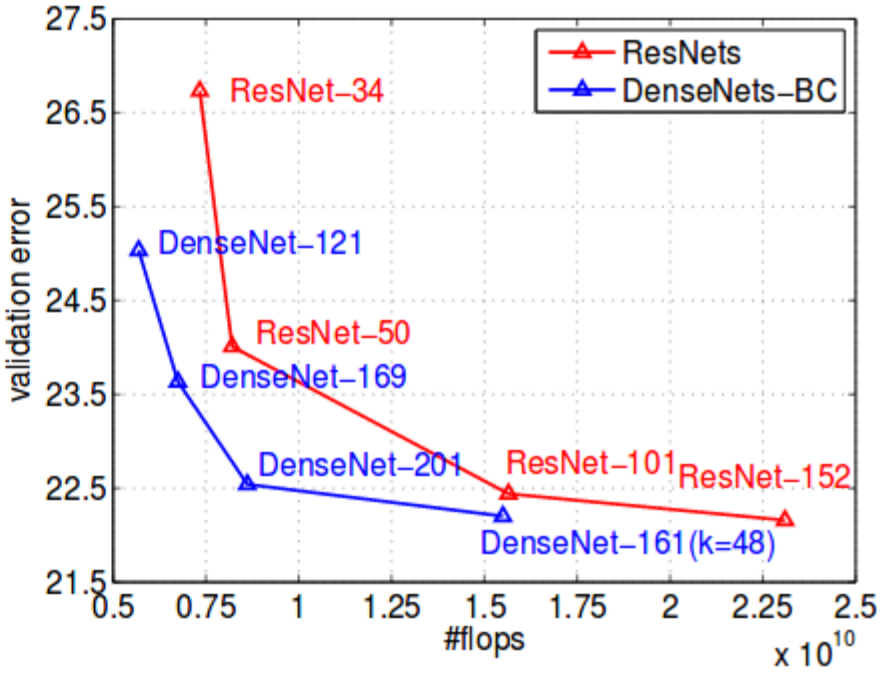


Figure 1. A 5-layer dense block with a growth rate of $k = 4$. Each layer takes all preceding feature-maps as input.

DenseNet

- Half of parameters but similar performance as ResNet





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**	-

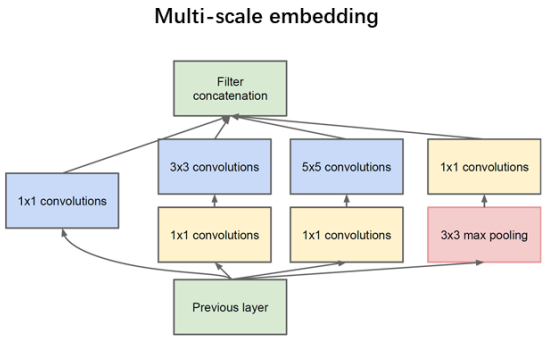
ATTENTION IN CNN

Attention in CNN

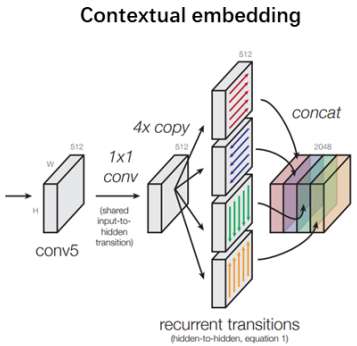
- Attention
  - Enhance or suppress information
- How
  - Channel attention
    - SE-Net, ResNeSt
  - Spatial attention (self attention)
    - Pixel attention
  - Combine of these twos
    - CBAM

How to Combine and Enhance these Information

- Exploration on spatial enhancement
  - Multi-scale filters
  - Contextural embedding
  - Attention



Inception [9]

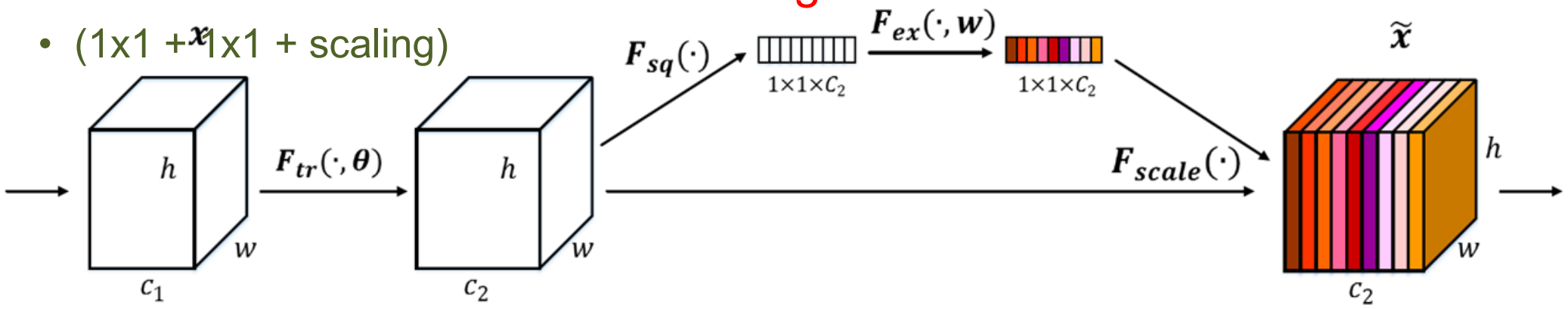


Inside-outside Network [13]

- How about enhanced from the aspect of channel?
  - Explicitly model channel interdependencies within models
  - Feature recalibration
    - Selectively enhance useful features and suppress less useful ones

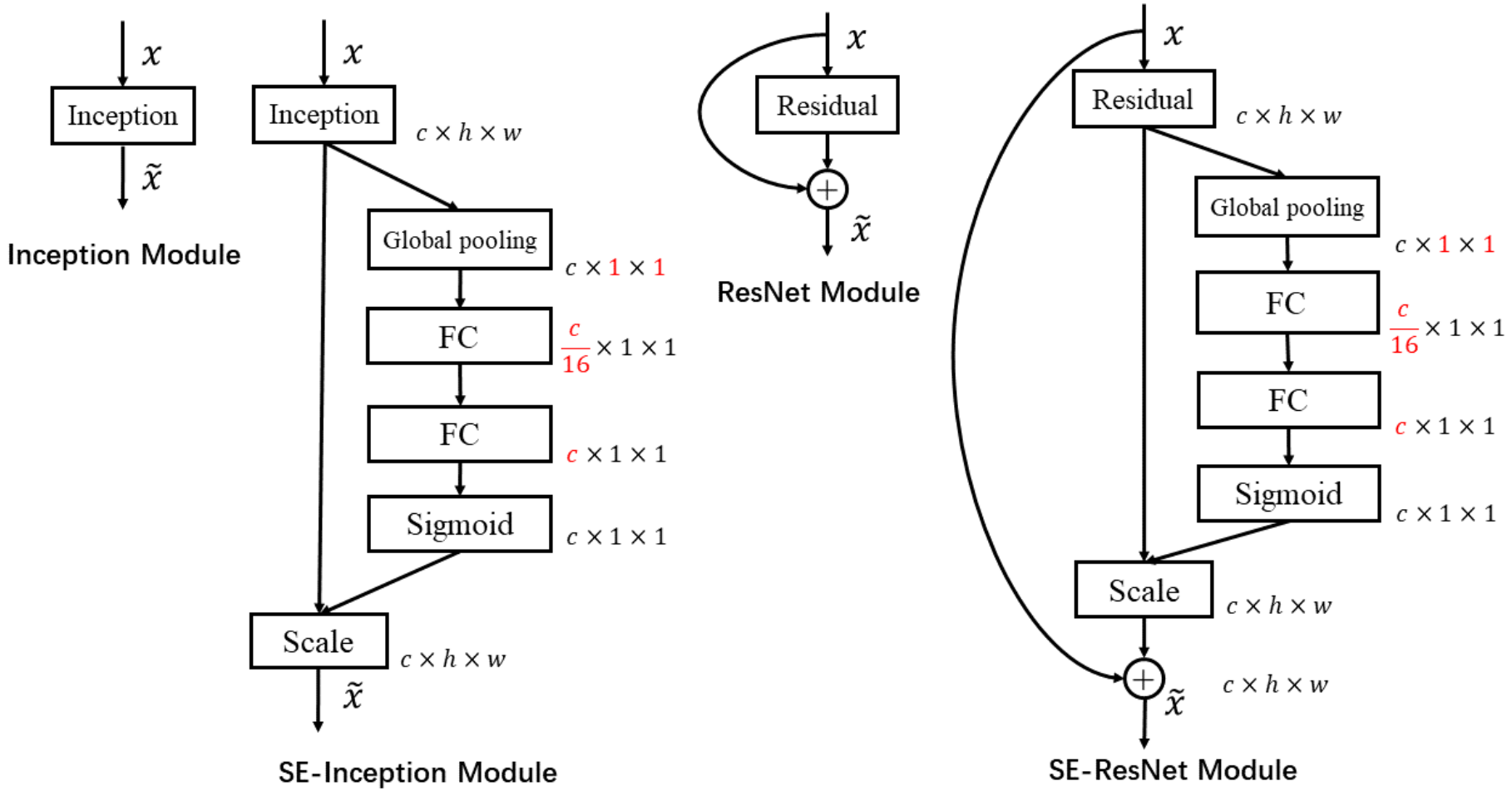
Squeeze-Excitation Network

- Squeeze-excitation module
  - Squeeze: Shrink feature map to get global information (Global Avg Pooling)
  - Excitation: Learn a channel-wise weights to combine channel information
    - (1x1 + 1x1 + scaling)



Squeeze	Excitation	Scale
• Shrinking feature maps $\in \mathbb{R}^{w\times h\times c_2}$ through spatial dimensions $(w \times h)$ • Global distribution of channel-wise responses	• Learning $W \in \mathbb{R}^{c_2\times c_2}$ to explicitly model channel-association • Gating mechanism to produce channel-wise weights	• Reweighting the feature maps $\in \mathbb{R}^{w\times h\times c_2}$

SE + Inception or ResNet



SE-ResNet-50 vs. ResNet-50

- Parameters: 2~10% extra
- GPU inference time: 10% extra
- CPU inference time: < 2% extra

Comparison with State-of-the-art

	224 × 224		320 × 320 / 299 × 299	
	top-1 err.	top-5 err.	top-1 err.	top-5 err.
ResNet-152 [1]	23.0	6.7	21.3	5.5
ResNet-200 [3]	21.7	5.8	20.1	4.8
Inception-v3 [10]	-	-	21.2	5.6
Inception-v4 [8]	-	-	20.0	5.0
Inception-ResNet-v2 [8]	-	-	19.9	4.9
ResNeXt-101 [11] (64 × 4d)	20.4	5.3	19.1	4.4
DenseNet-161 [4] (k = 48)	22.2	-	-	-
Very Deep PolyNet [12]	-	-	18.71	4.25
SENet	**18.68**	**4.47**	**17.28**	**3.79**

Table 4. Single-crop error of state-of-the-art CNNs on ImageNet-1k validation set. The shorter edge is resized to 224 and 320 (299 for Inception models) respectively and then conduct center-crop testing as in [11]. The **SENet** is our well-structured model whose error rates are remarkably lower than previous models.

SENet is a SE-ResNeXt-152 (64 × 4d)

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**	**6.62**
ResNet-101 [1]	23.6	7.1	23.17	6.52	**22.38**	**6.07**
ResNet-152 [1]	23.0	6.7	22.42	6.34	**21.57**	**5.73**

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

Incorporation with Modern Architectures

	Original		Our re-implementation		SE-module	
	top-1 err.	top-5 err.	top-1 err.	top-5 err.	top-1 err.	top-5 err.
ResNeXt-50 [7]	22.2	-	22.11	5.90	**21.10**	**5.49**
ResNeXt-101 [7]	21.2	5.6	21.18	5.57	**20.70**	**5.01**
BN-Inception [4]	25.2	7.82	25.38	7.89	**24.23**	**7.14**
Inception-ResNet-v2 [5]	19.9†	4.9†	20.37	5.21	**19.80**	**4.79**

Table 2. Error rates (%) of single-crop results on the ImageNet-1k validation set. Error rate followed by † means that the image size for center crop is not clear and it evaluates on the non-blacklisted subset of validation set [5], which may lead to slightly better results.

ECANet

- 能否以更有效的方式學習有效的通道注意力?
- SENet
  - GAP
  - FC1 + FC2
    - Cross channels + dimension reduction
      - 降維對通道注意預測帶來了副作用，捕獲所有通道之間的依賴是低效的，也是不必要的
  - sigmoid

ECANet: GAP + 1-D Conv + Sigmoid

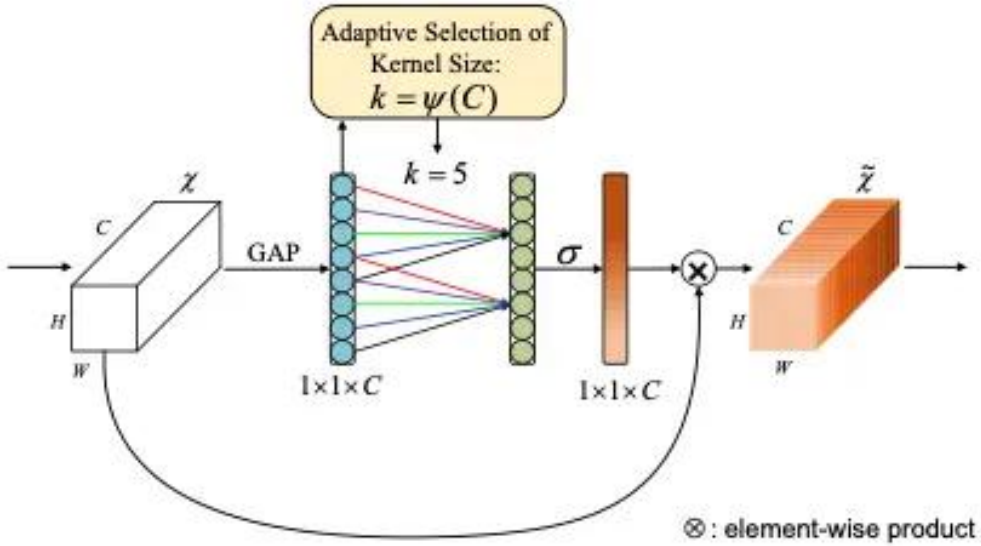


Figure 2. Diagram of our efficient channel attention (ECA) module. Given the aggregated features obtained by global average pooling (GAP), ECA generates channel weights by performing a fast $1D$ convolution of size k, where k is adaptively determined via a mapping of channel dimension C.

Model	No DR	Cross-channel Interaction	Lightweight
SENet [14]	×	√	–
CBAM [33]	×	√	×
GE-θ^- [13]	√	×	√
GE-θ [13]	√	×	×
GE-θ^+ [13]	×	√	×
A^2-Net [4]	×	√	×
GSoP-Net [9]	×	√	×
ECA-Net (Ours)	√	√	√

Table 1. Comparison of existing attention modules in terms of whether no channel dimensionality reduction (No DR), cross-channel interaction and less parameters than SE (indicated by lightweight) or not.

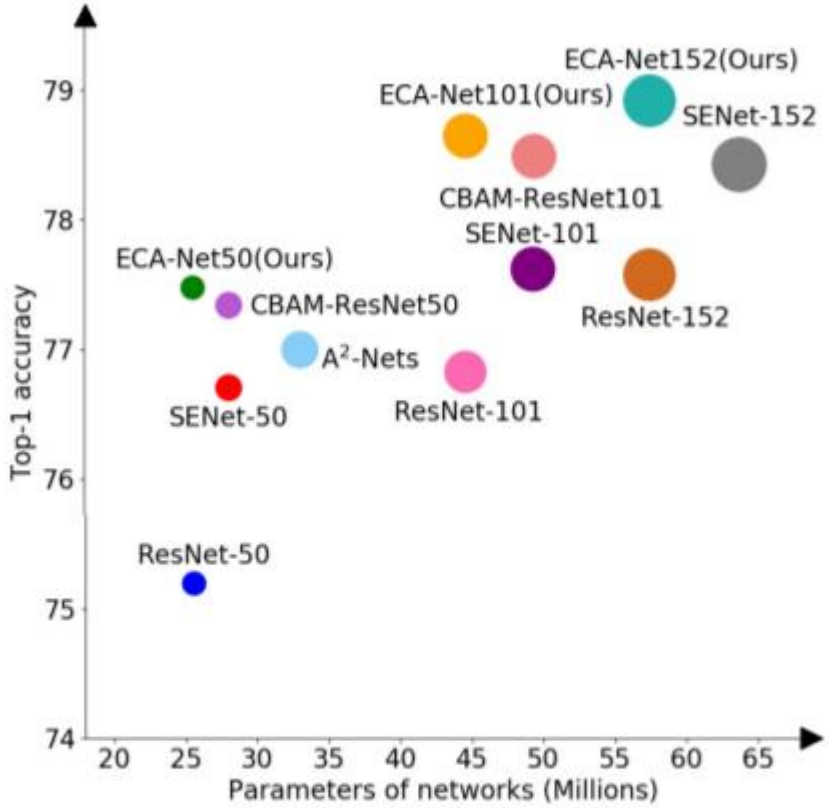


Figure 1. Comparison of various attention modules (i.e., SENet [14], CBAM [33], A^2-Nets [4] and ECA-Net) using ResNets [11] as backbone models in terms of classification accuracy, network parameters and FLOPs, indicated by radiuses of circles. Note that our ECA-Net obtains higher accuracy while having less model complexity.

Pixel Attention

- generates attention coefficients for all pixels of the feature map
  - use a 1×1 convolution layer and a sigmoid function

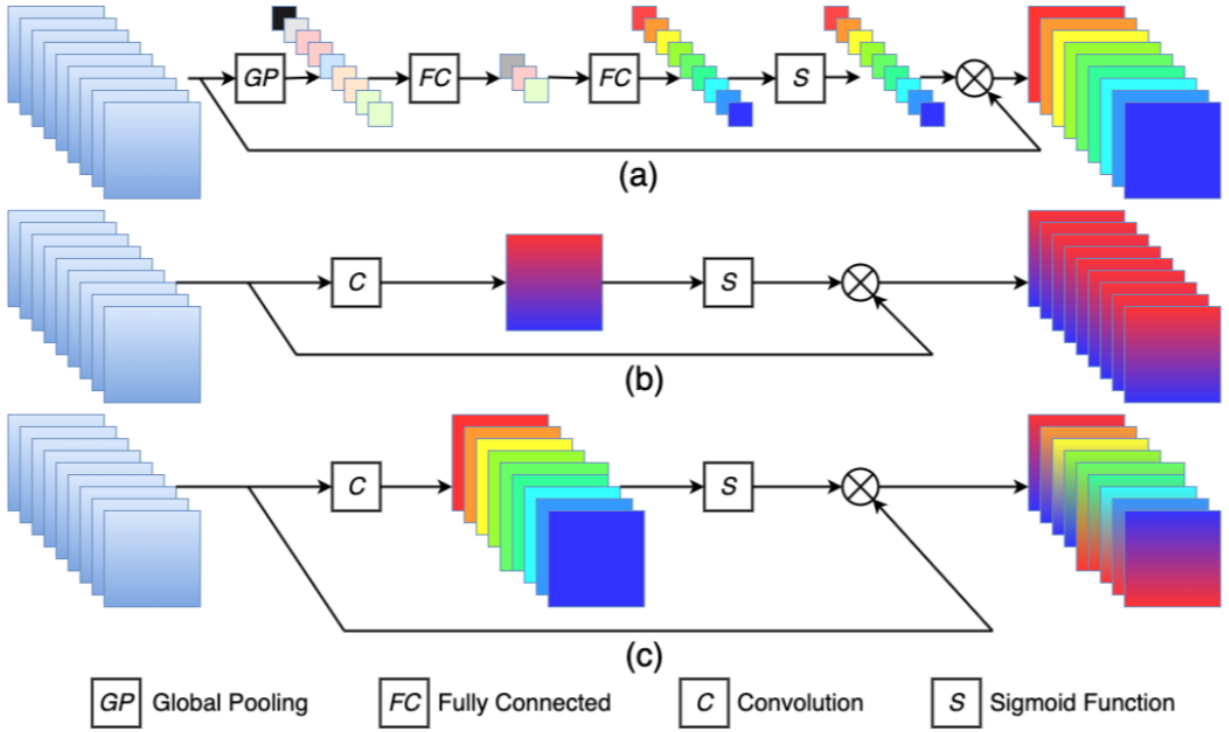
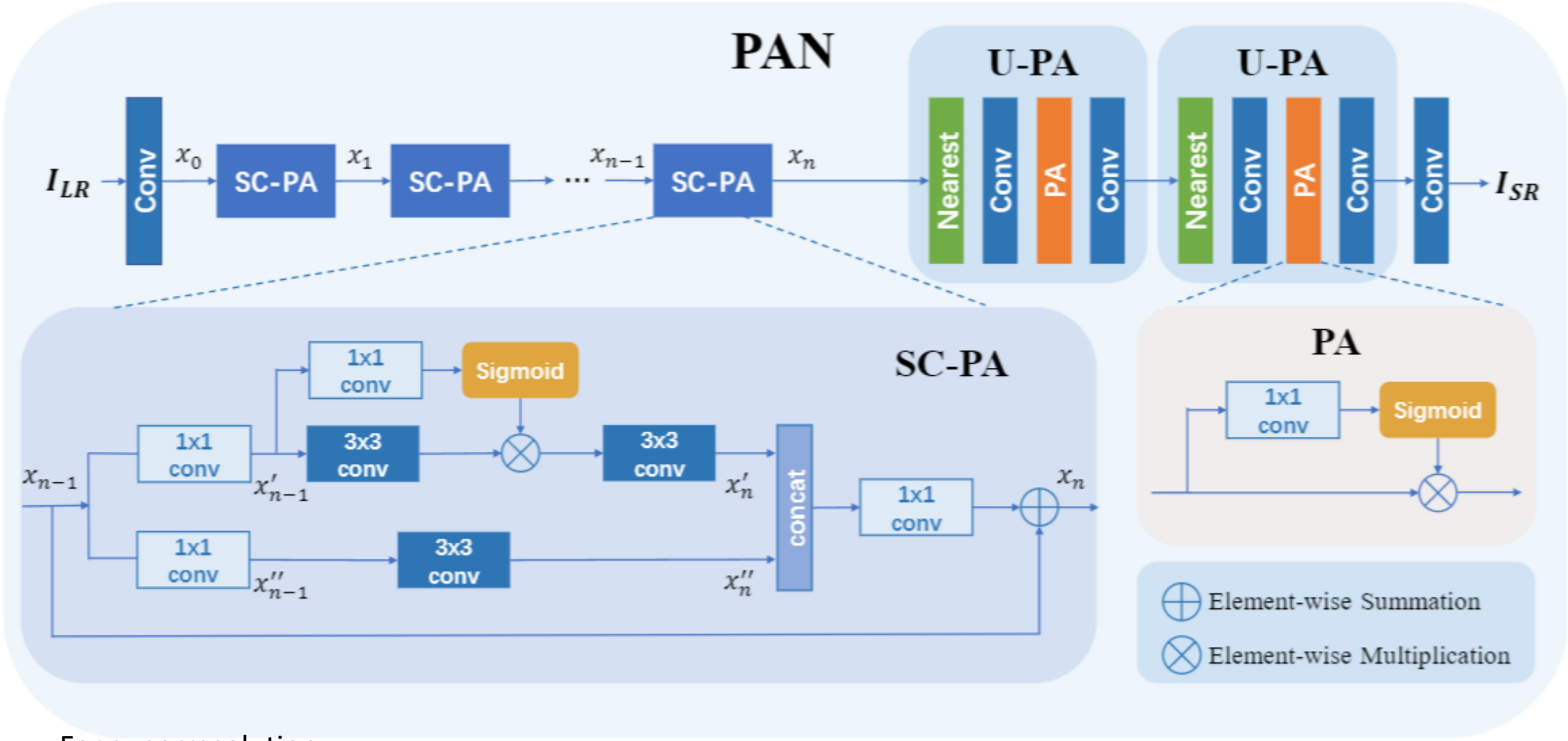
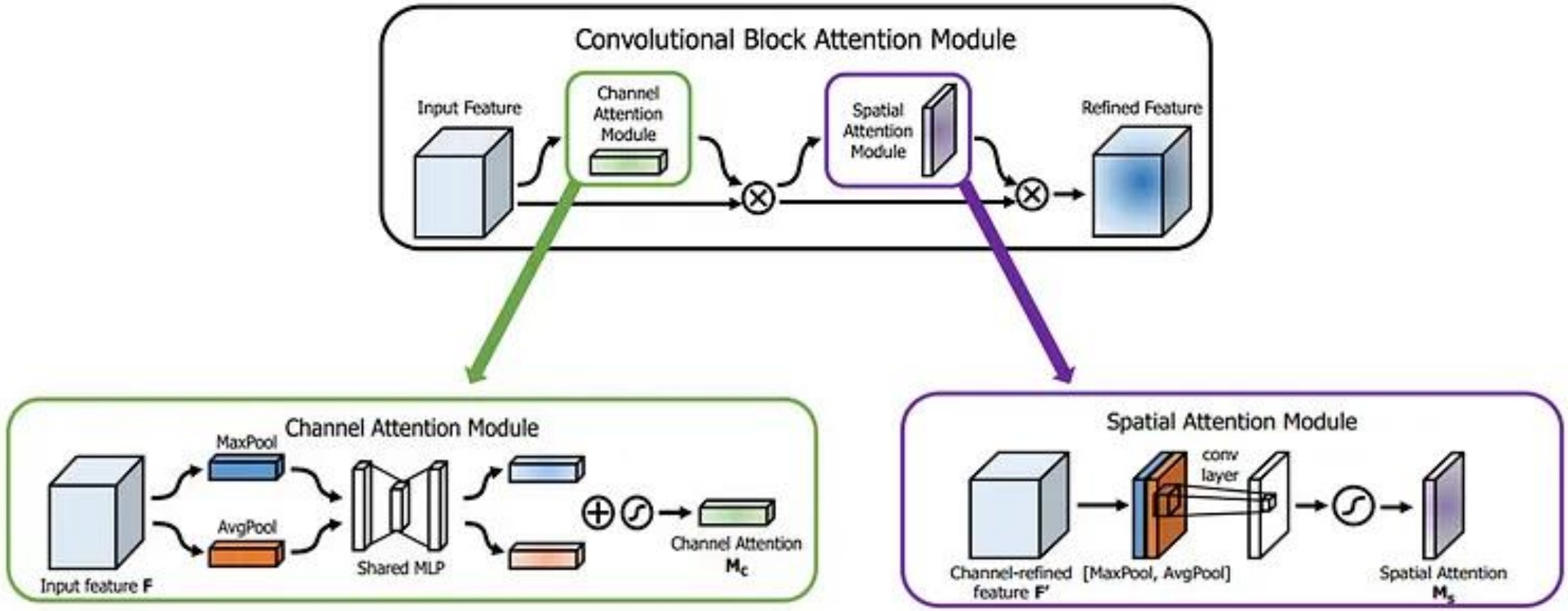


Fig. 3. (a) CA: Channel Attention; (b) SA: Spatial Attention; (c) PA: Pixel Attention.



For superresolution

CBAM: Convolutional Block Attention Module

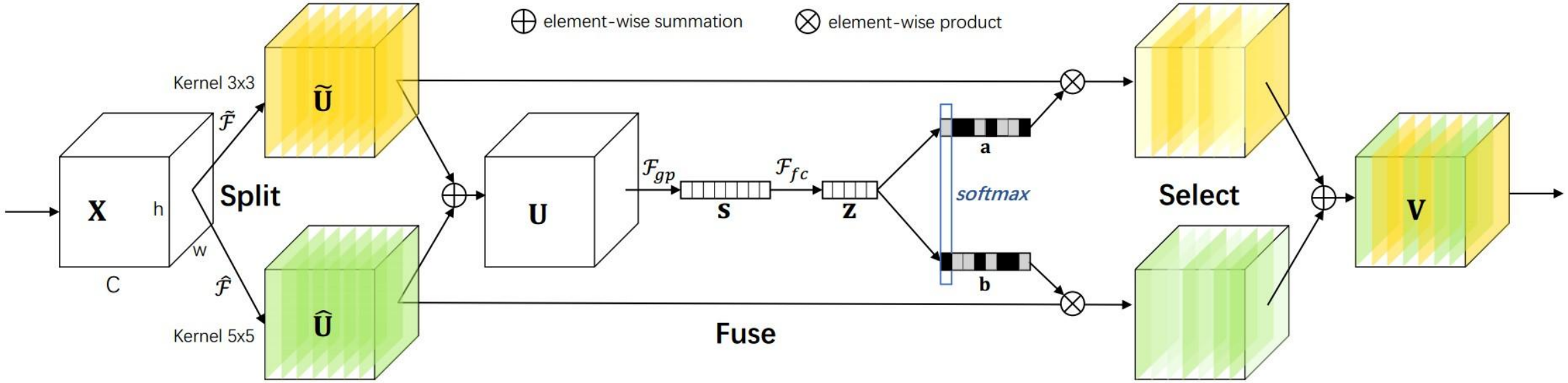


$$\mathbf{M_c}(\mathbf{F}) = \sigma(MLP(AvgPool(\mathbf{F})) + MLP(MaxPool(\mathbf{F})))$$
$$= \sigma(\mathbf{W_1}(\mathbf{W_0}(\mathbf{F^c_{avg}})) + \mathbf{W_1}(\mathbf{W_0}(\mathbf{F^c_{max}}))),$$

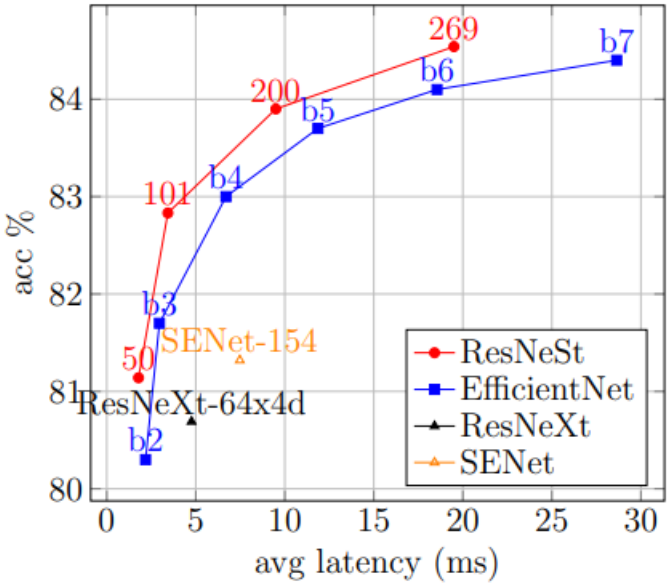
$$\mathbf{M_s}(\mathbf{F}) = \sigma(f^{7\times7}([AvgPool(\mathbf{F}); MaxPool(\mathbf{F})]))$$
$$= \sigma(f^{7\times7}([\mathbf{F^s_{avg}}; \mathbf{F^s_{max}}])),$$

SKNet: Selective Kernel Networks

- SE-Net + split kernel (3x3, 5x5)



ResNeSt : Split-Attention Networks

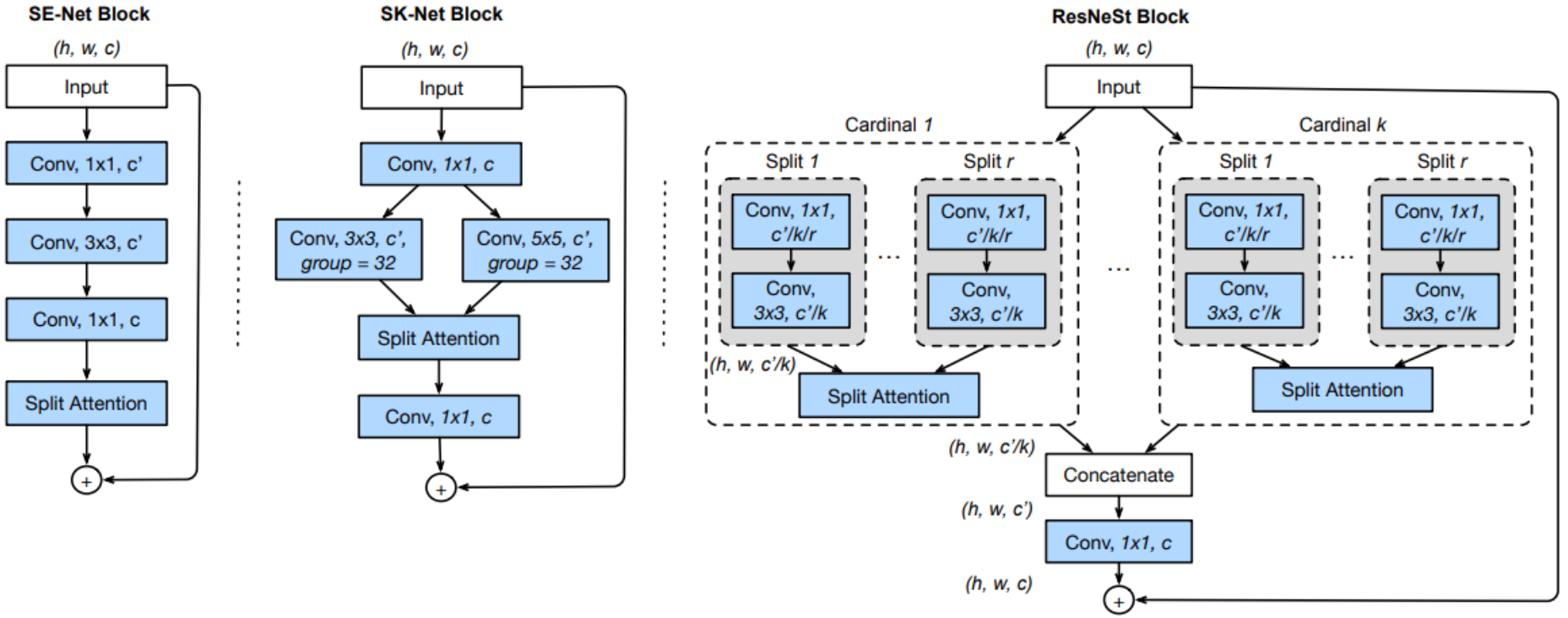


	Crop	#P	Acc%
ResNeSt-50 (ours)	224	27.5M	81.1
ResNeSt-101 (ours)	256	48.3M	82.8
ResNeSt-200 (ours)	320	70.2M	83.9
ResNeSt-269 (ours)	416	111M	84.5

	Backbone	#Params	Score%
FasterRCNN [46]	ResNet-50 [57]	34.9M	39.25
	ResNeSt-50 (ours)	36.8M	42.33
DeeplabV3 [7]	ResNet-50 [57]	42.2M	42.10
	ResNeSt-50 (ours)	44.0M	45.12

SE-Net + ResNext

Table 1: (Left) Accuracy and latency trade-off on GPU using official code implementation (details in Section 5). (Right-Top) Top-1 accuracy on ImageNet using ResNeSt. (Right-Bottom) Transfer learning results: object detection mAP on MS-COCO [42] and semantic segmentation mIoU on ADE20K [71].



G=KR
R=1, => SENet

Fig. 1: Comparing our ResNeSt block with SE-Net [30] and SK-Net [38]. A detailed view of Split-Attention unit is shown in Figure 2. For simplicity, we show ResNeSt block in cardinality-major view (the featuremap groups with same cardinal group index reside next to each other). We use radix-major in the real implementation, which can be modularized and accelerated by group convolution and standard CNN layers (see supplementary material).

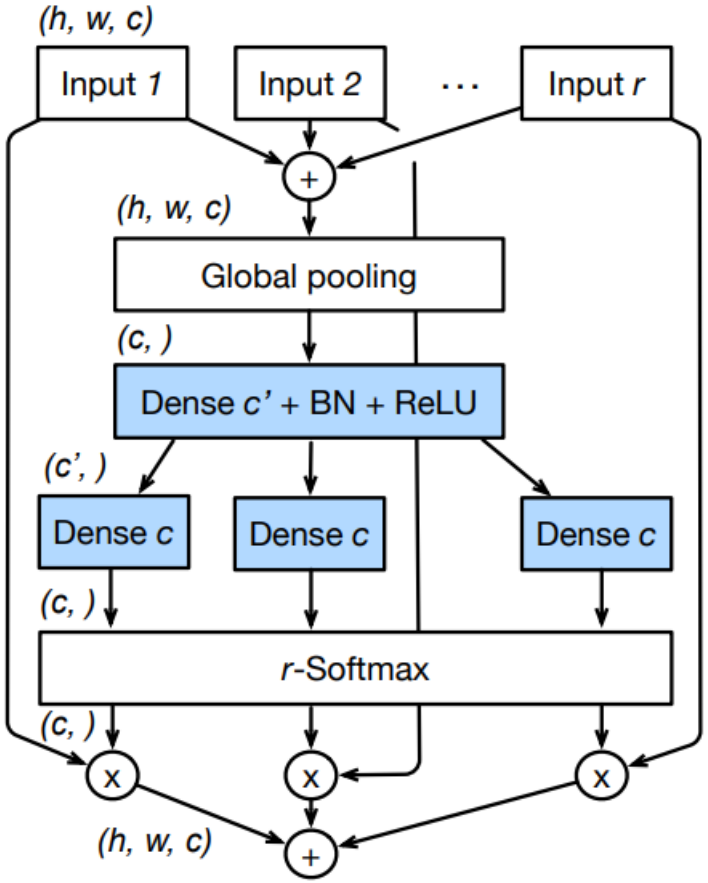
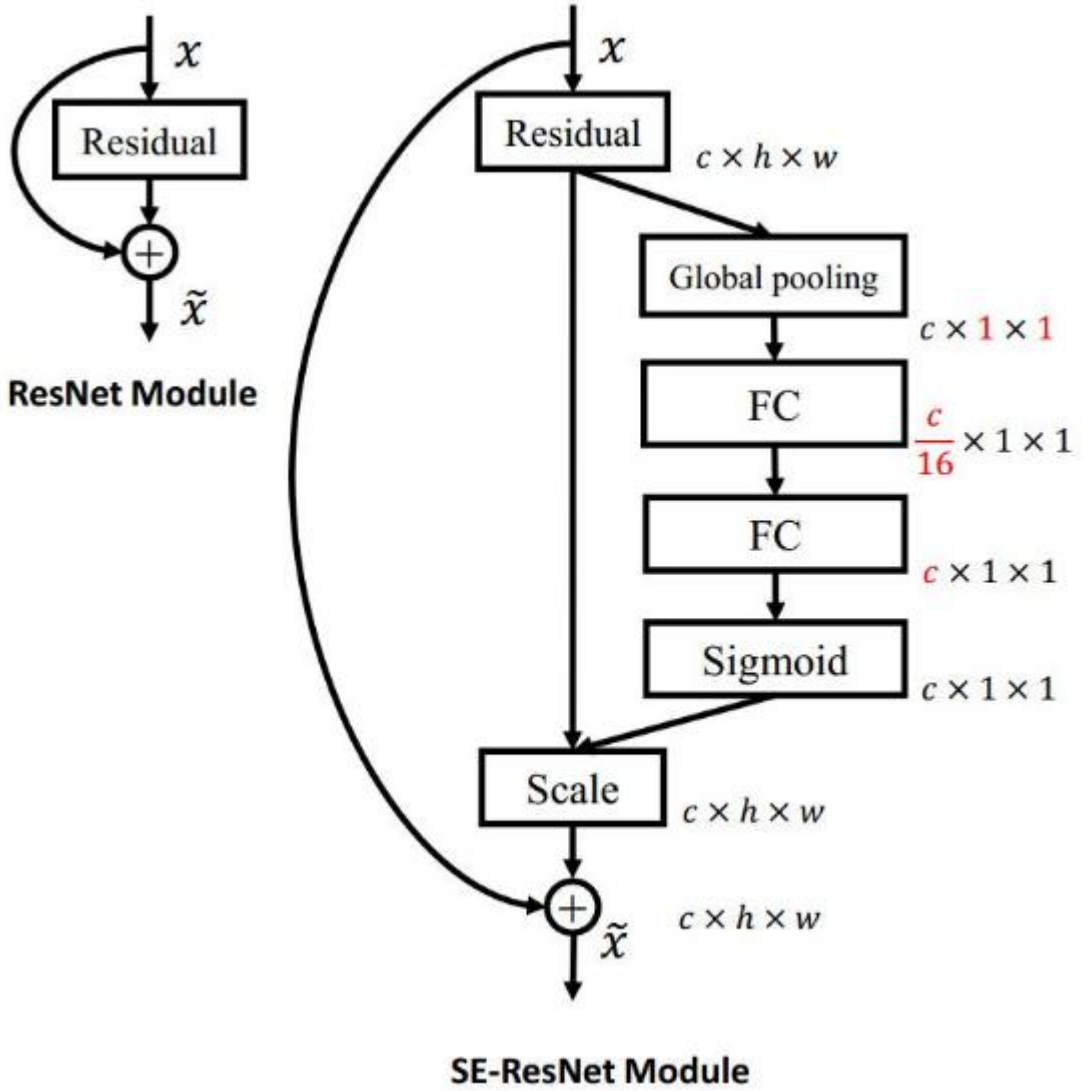


Fig. 2: Split-Attention within a cardinal group. For easy visualization in the figure, we use $c = C/K$ in this figure.

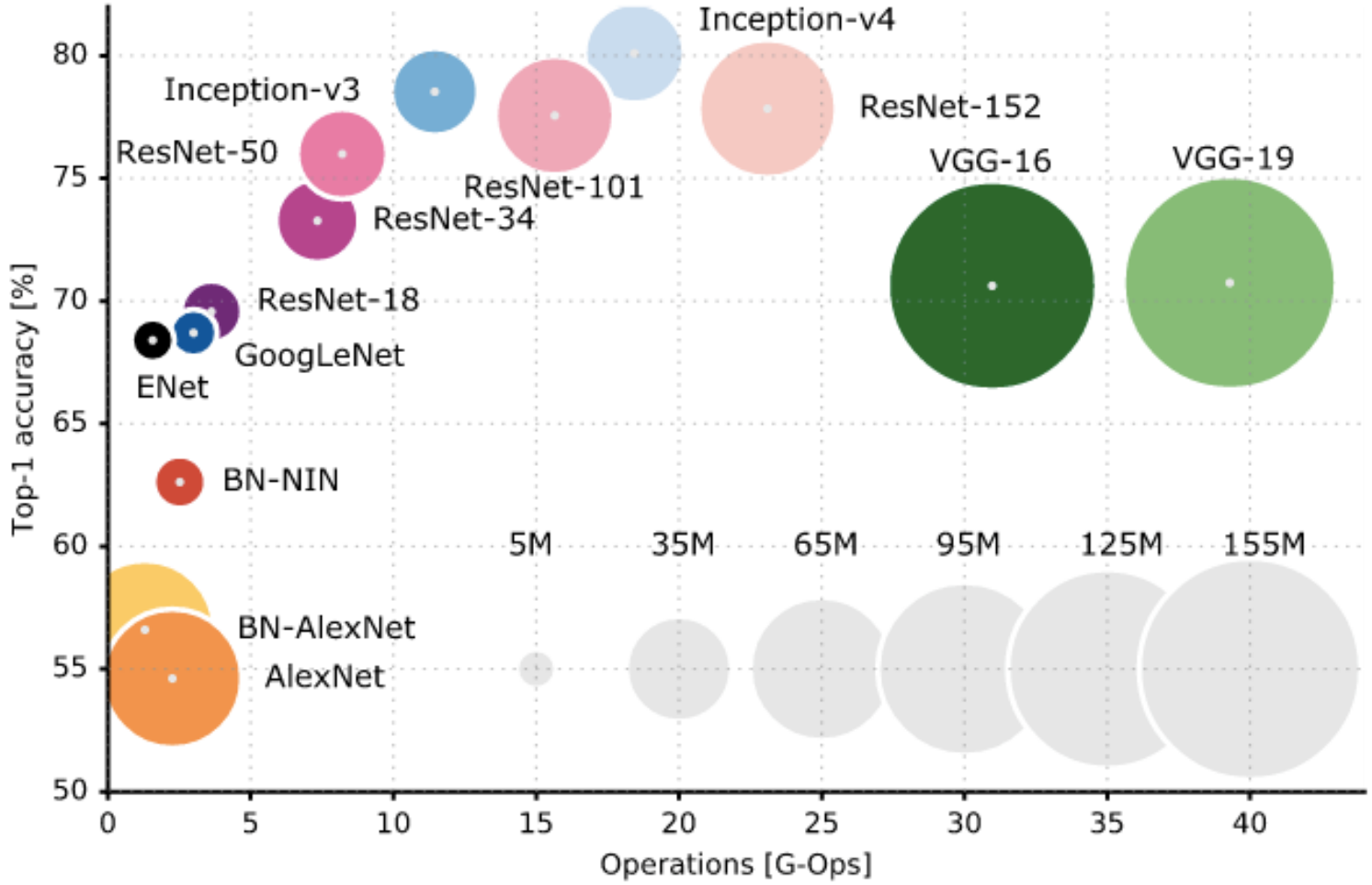


	#P	GFLOPs	top-1 acc (%)	
			224×	320×
ResNet-50 [23]	25.5M	4.14	76.15	76.86
ResNeXt-50 [60]	25.0M	4.24	77.77	78.95
SENet-50 [29]	27.7M	4.25	78.88	80.29
ResNetD-50 [26]	25.6M	4.34	79.15	79.70
SKNet-50 [38]	27.5M	4.47	79.21	80.68
ResNeSt-50-fast(ours)	27.5M	4.34	**80.64**	**81.43**
ResNeSt-50(ours)	27.5M	5.39	81.13	81.82
ResNet-101 [23]	44.5M	7.87	77.37	78.17
ResNeXt-101 [60]	44.3M	7.99	78.89	80.14
SENet-101 [29]	49.2M	8.00	79.42	81.39
ResNetD-101 [26]	44.6M	8.06	80.54	81.26
SKNet-101 [38]	48.9M	8.46	79.81	81.60
ResNeSt-101-fast(ours)	48.2M	8.07	**81.97**	**82.76**
ResNeSt-101(ours)	48.3M	10.2	82.27	83.00

Table 3: Image classification results on ImageNet, comparing our proposed ResNeSt with other ResNet variants of similar complexity in 50-layer and 101-layer configurations. We report top-1 accuracy using crop sizes 224 and 320.

COMPLEXITY

Accuracy vs. efficiency



https://culurciello.github.io/tech/2016/06/04/nets.html

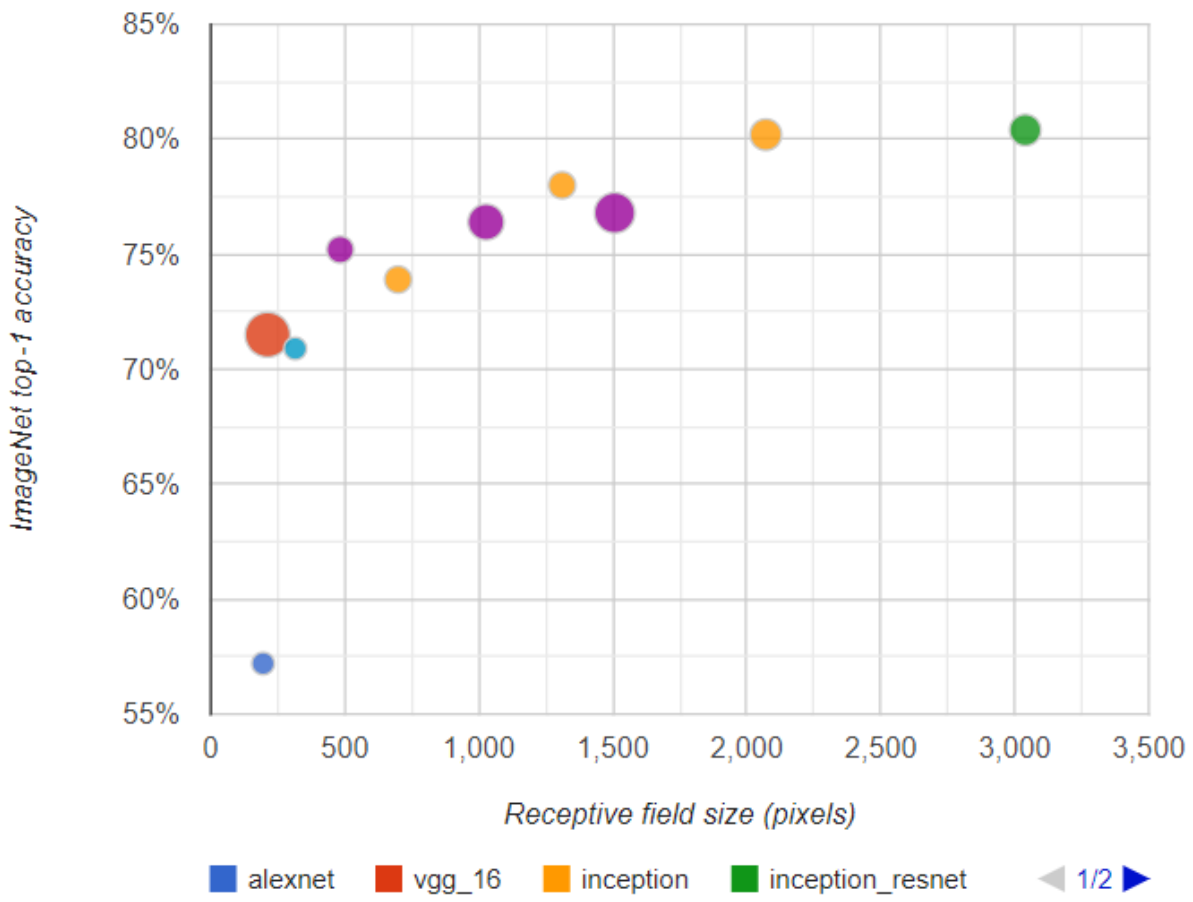
Metrics	AlexNet	VGG-16	GoogLeNet (v1)	ResNet-50
Accuracy (top-5 error)*	19.8	8.80	10.7	7.02
Input	227x227	224x224	224x224	224x224
# of CONV Layers	**5**	**16**	**21**	**49**
Filter Sizes	3, 5,11	3	1, 3 , 5, 7	1, 3, 7
# of Channels	3 - 256	3 - 512	3 - 1024	3 - 2048
# of Filters	96 - 384	64 - 512	64 - 384	64 - 2048
Stride	1, 4	1	1, 2	1, 2
# of Weights	2.3M	14.7M	6.0M	23.5M
# of MACs	666M	15.3G	1.43G	3.86G
# of FC layers	**3**	**3**	**1**	**1**
# of Weights	58.6M	124M	1M	2M
# of MACs	58.6M	124M	1M	2M
Total Weights	**61M**	**138M**	**7M**	**25.5M**
Total MACs	**724M**	**15.5G**	**1.43G**	**3.9G**

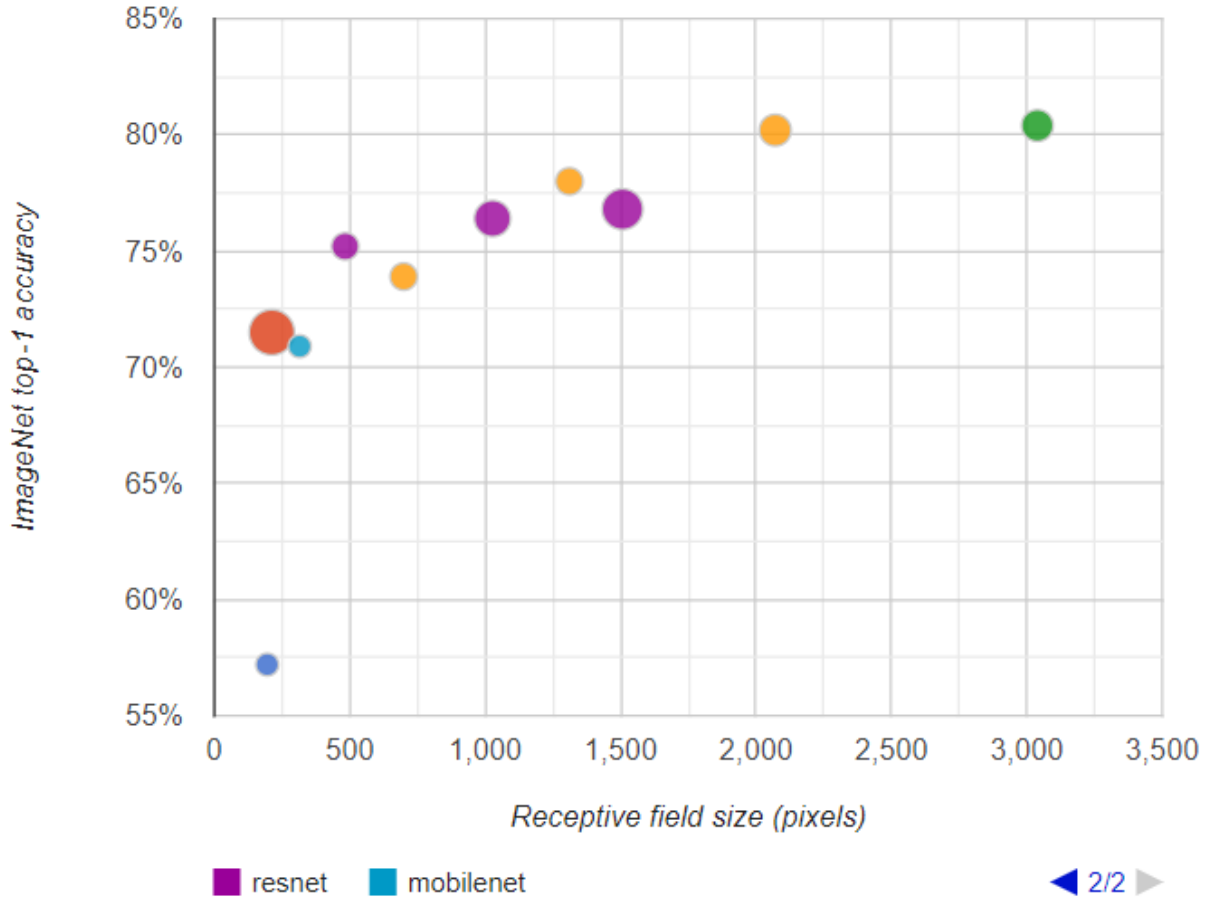
*Single crop results: https://github.com/jcjohnson/cnn-benchmarks

Source: Tutorial on Hardware Architectures for Deep Neural Networks

ConvNet Model	Receptive Field (r)	Effective Stride (S)	Effective Padding (P)	Model Year
alexnet_v2	195	32	64	2014
vgg_16	212	32	90	2014
mobilenet_v1	315	32	126	2017
mobilenet_v1_075	315	32	126	2017
resnet_v1_50	483	32	239	2015
inception_v2	699	32	318	2015
resnet_v1_101	1027	32	511	2015
inception_v3	1311	32	618	2015
resnet_v1_152	1507	32	751	2015
resnet_v1_200	1763	32	879	2015
inception_v4	2071	32	998	2016
inception_resnet_v2	3039	32	1482	2016

Receptive field v.s. Accuracy





Effective Receptive Field (ERF)

- not all pixels in a receptive field contribute equally to an output unit's response
  - Center pixels contribute more
  - distribution of impact in a receptive field distributes as a Gaussian

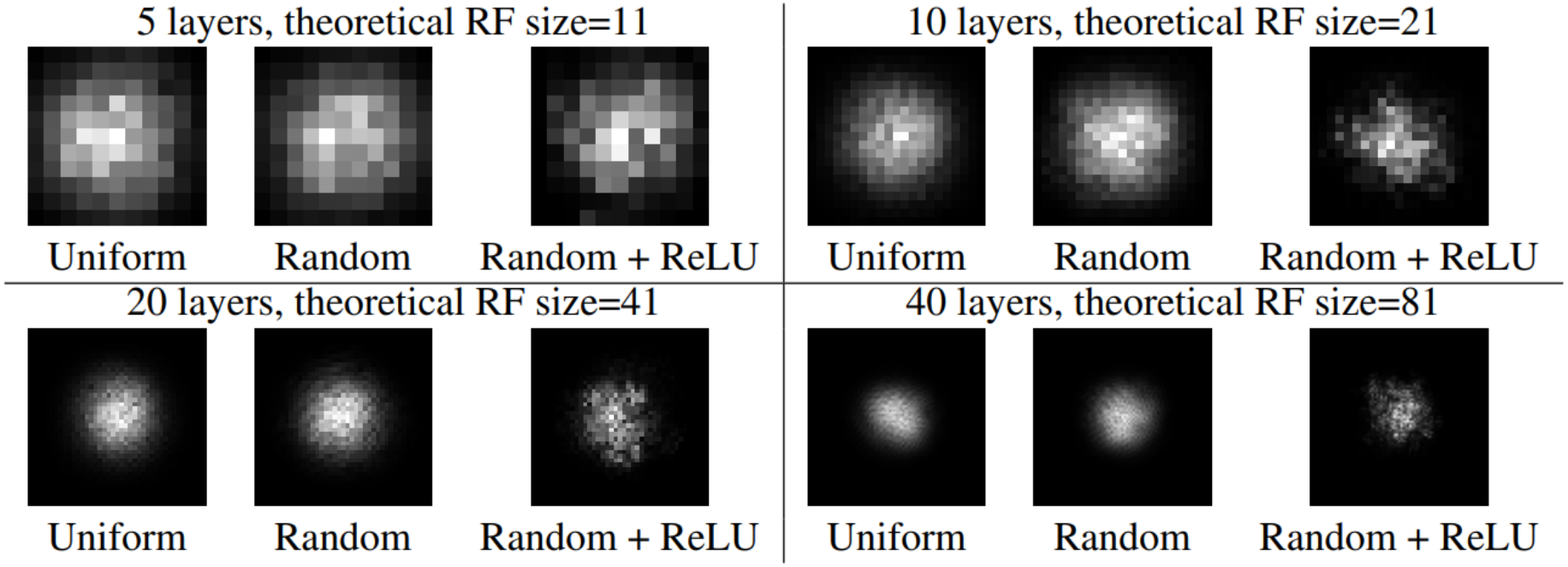


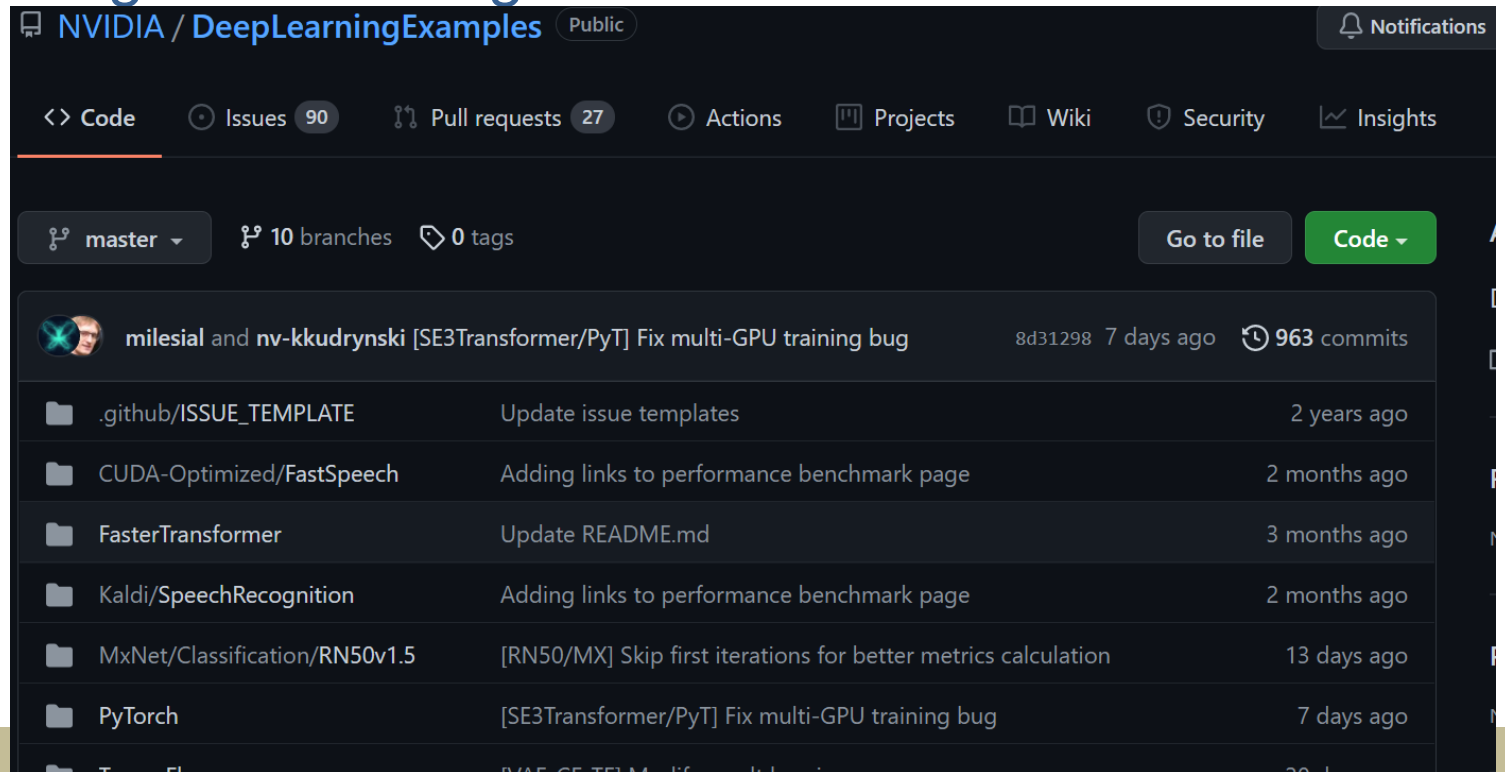
Figure 1: Comparing the effect of number of layers, random weight initialization and nonlinear activation on the ERF. Kernel size is fixed at 3 × 3 for all the networks here. Uniform: convolutional

Summary

- VGG, GoogLeNet, ResNet all in wide use, available in model zoos
- ResNet current best default
- Trend towards extremely deep networks
- Significant research centers around design of layer / skip connections and improving gradient flow
- Even more recent trend towards examining necessity of depth vs. width and residual connections

Get the SOTA Models?

- Find it on examples of different DL framework
- From Nvidia
  - https://github.com/NVIDIA/DeepLearningExamples
  - ImageNet Training in one hour



模型	核心貢獻	優點	缺點	典型使用情境
LeNet-5 (1998)	卷積+池化奠基	結構簡潔、易教學	僅適合小資料	教學入門、MNIST
AlexNet (2012)	ReLU/Dropout/DataAug + GPU	首次大規模成功	參數龐大、易過擬合	ImageNet 里程碑
VGG (2014)	3×3 小卷積堆疊	結構規範、遷移性強	參數/計算量高	特徵抽取 backbone
Inception (2014–)	多尺度分支 + 1×1	高效、參數更少	模組複雜	伺服器端推論
ResNet (2015–)	殘差連接	易訓練超深網	計算/記憶仍重	通用強基線
DenseNet (2016)	密集連接	參數效率高	記憶/帶寬壓力	特徵重用場景
ResNeXt (2017)	基數（cardinality）	可擴展、表現佳	工程複雜度	高效能伺服器
SE/SK/CBAM (2017–)	通道/空間注意力	提升準確率	輕增成本	多任務骨幹升級
Xception (2017)	極致可分離卷積	效率/表現兼顧	需良好實作	高效主幹
MobileNet/ShuffleNet	深度可分離/通道洗牌	端邊即時	精度較低	手機/嵌入式
EfficientNet (2019)	複合縮放+NAS	SOTA 效率	搜索成本、黑箱	需精確效能/效率
RegNet (2020)	規律化設計空間	穩健、可擴展	需配方	大規模訓練
ConvNeXt (2022)	Conv × Transformer 配方	競爭力回升	配方較多	ViT 對標替代

演進脈絡

- 1990s–2012：可行性驗證到大規模成功
  - LeNet-5：局部感受野 + 權重共享 + 池化；在 MNIST 證明 CNN 可行。
  - AlexNet：靠 GPU 訓練、ReLU、Dropout、Data Aug 與大量數據在 ImageNet 取得決定性勝利。
- 2014–2016：加深、加寬與訊息流（避免退化）
  - VGG：用統一的小卷積（3×3）堆疊；結構規範但參數/計算重。
  - Inception：多尺度分支 + 1×1 降維 提升效率。
  - ResNet：殘差（skip） 解決深度退化，使 100+ 層可訓練。
- 2016–2019：效率化與自動化
  - Xception/Depthwise Separable：把 3×3 conv 拆成 depthwise + pointwise，大減 FLOPs。
  - MobileNet/ShuffleNet：輕量模型針對端邊/即時。
  - NASNet/EfficientNet：NAS + 複合縮放（depth/width/resolution），效能/效率兼顧。
- 2019–至今：與 Transformer 互補、配方現代化
  - ViT 崛起（長距依賴強），CNN 陣營以 ConvNeXt、RegNet、RepVGG 等吸收 Transformer 訓練配方（例如 LayerNorm、GELU、較大感受野、資料配方），維持高效率競爭力。