High Level Computer Vision

CNN Architectures

@ May 10, 2023

Bernt Schiele

www.mpi-inf.mpg.de/hlcv/
cms.sic.saarland/hlcvss23/

Max Planck Institute for Informatics & Saarland University,
Saarland Informatics Campus Saarbrücken

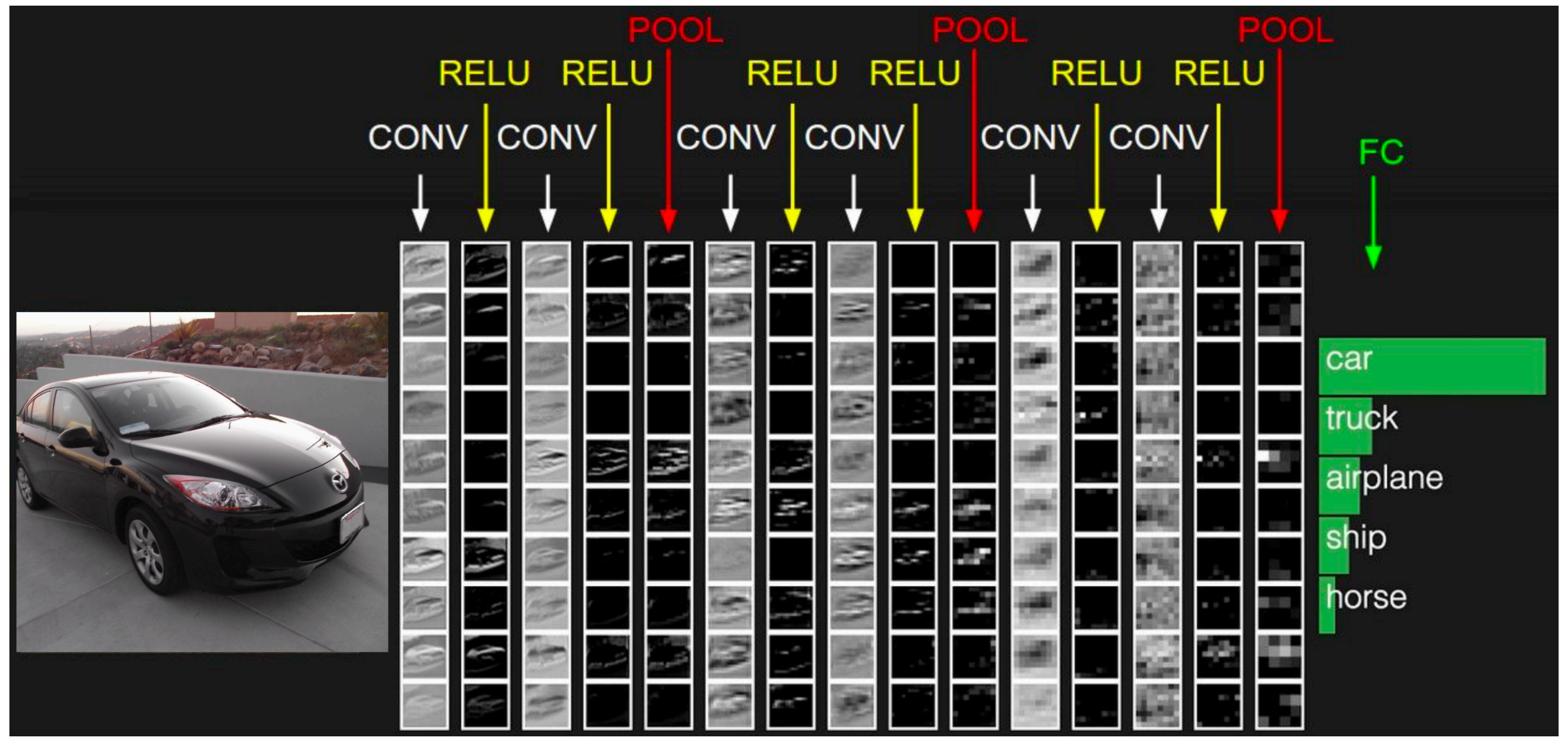
Overview Today's Lecture

- Convolutional Neural Network (CNN) Architectures
  - case studies
    - AlexNet,
    - VGG
    - GoogLeNet
    - ResNet
  - other architectures to know (briefly discussed today)
    - NiN (Network in Network), Wide ResNet, ResNeXT, Stochastic Depth, Squeeze-and-Excitation Network, DenseNet, NASNet

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Convolutional Neural Networks - recap...

- "Classic" Architecture:
  - { (CONV+RELU) * N + POOL } * M + {FC+RELU} * K + SOFTMAX
  - softmax function: $$\sigma(z_i) = \frac{e^{(-z_i)}}{\sum_{j=1}^{C} e^{(-z_j)}} = \frac{\exp(-z_i)}{\sum_{j=1}^{C} \exp(-z_j)}$$



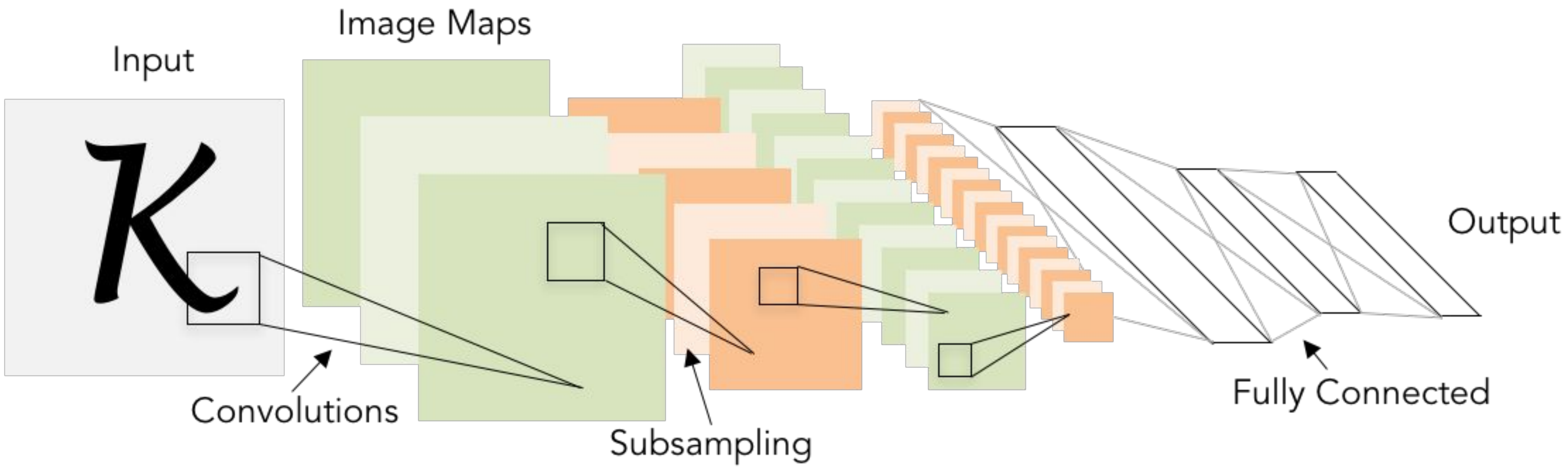
- CNNs stack CONV, POOL, FC layers
  - Trend towards smaller filters and deeper architectures
  - Trend towards getting rid of POOL / FC layers (just CONV)

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Review: LeNet-5

- cf. “Classic” Architecture:
  - { (CONV+RELU) * N + POOL } * M + {FC+RELU} * K + SOFTMAX

[LeCun et al., 1998]



Conv filters were 5x5, applied at stride 1
Subsampling (Pooling) layers were 2x2 applied at stride 2
i.e. architecture is [CONV-POOL-CONV-POOL-FC-FC]

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Case Study: AlexNet

[Krizhevsky et al. 2012]

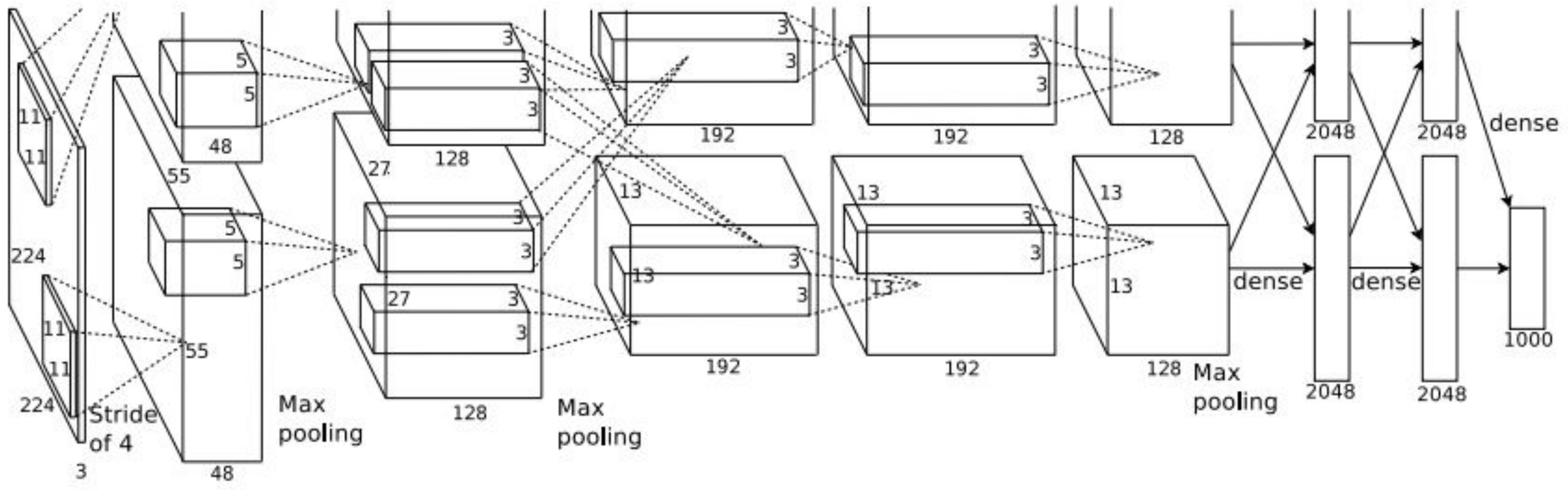


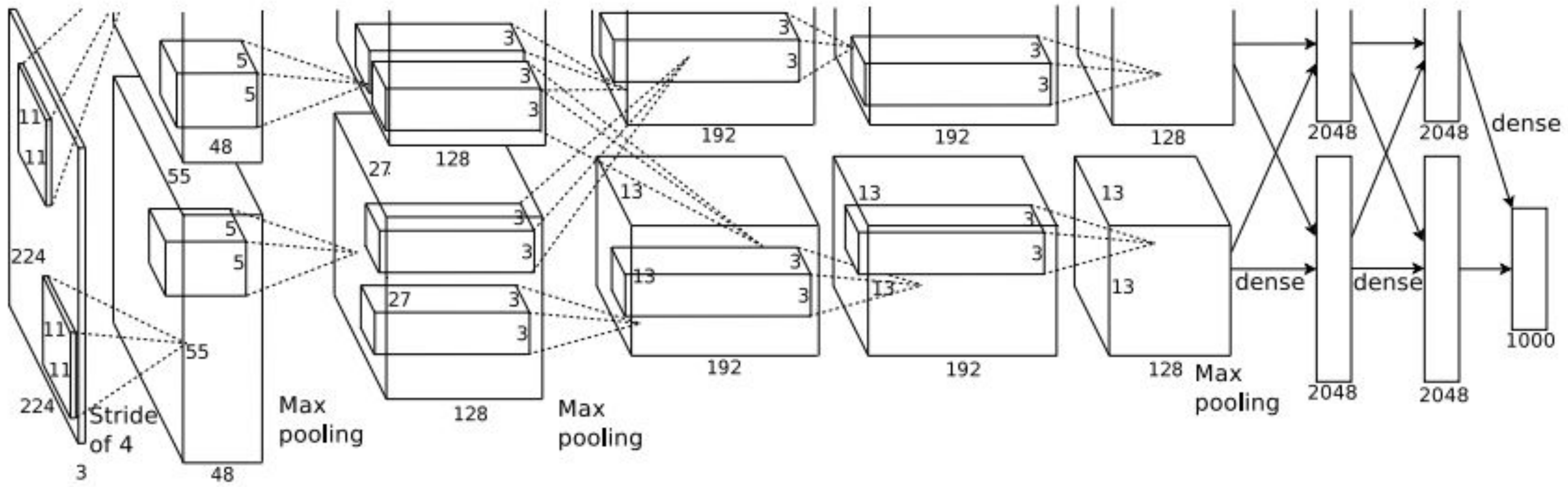
Figure copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.

Architecture:
CONV1
MAX POOL1
NORM1
CONV2
MAX POOL2
NORM2
CONV3
CONV4
CONV5
Max POOL3
FC6
FC7
FC8

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Case Study: AlexNet

[Krizhevsky et al. 2012]



Input: 227x227x3 images

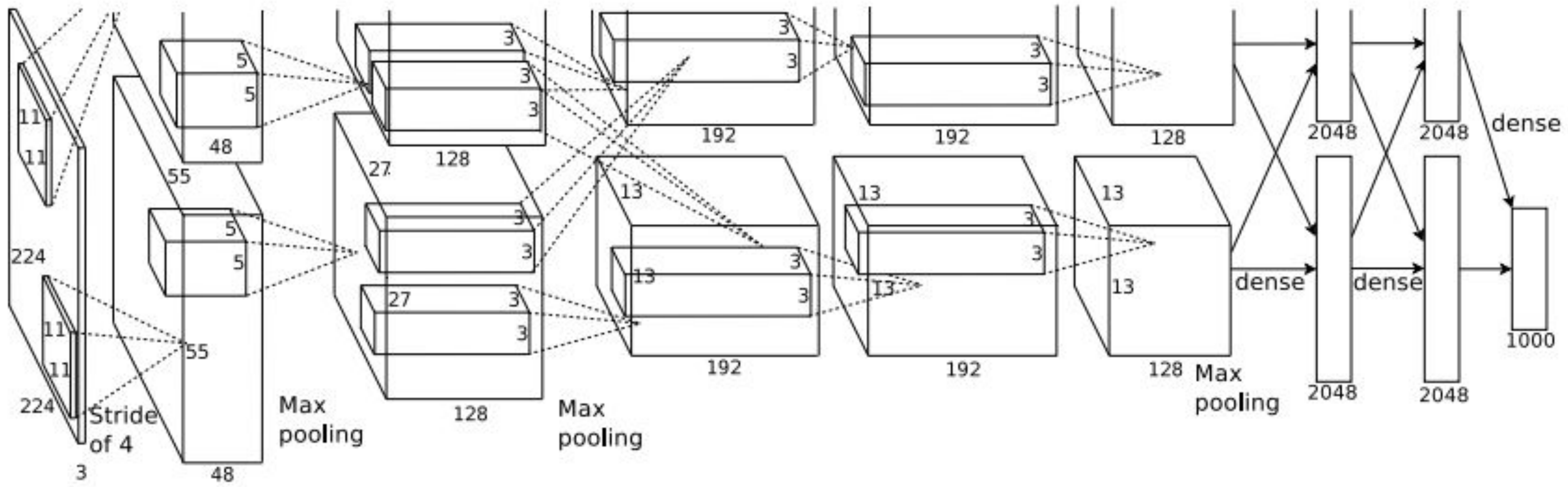
First layer (CONV1): 96 11x11 filters applied at stride 4
=>
Q: what is the output volume size? Hint: (227-11)/4+1 = 55

(recall:)
(N - F) / stride + 1

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Case Study: AlexNet

[Krizhevsky et al. 2012]



Input: 227x227x3 images

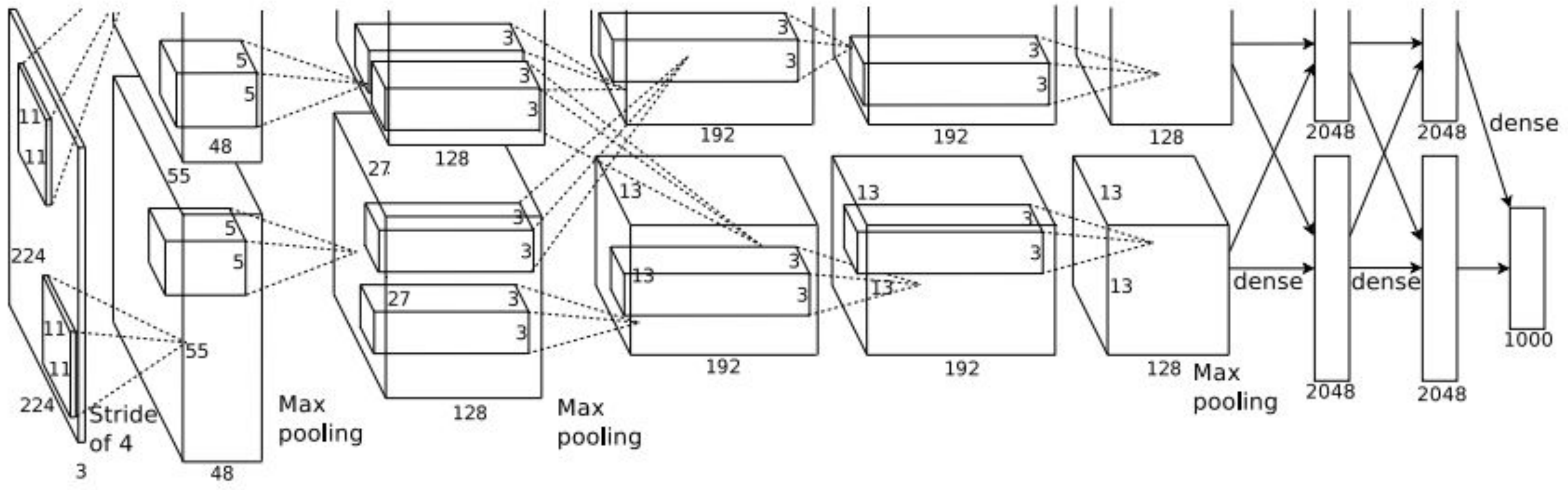
First layer (CONV1): 96 11x11 filters applied at stride 4
=>
Output volume **[55x55x96]**

Q: What is the total number of parameters in this layer?

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Case Study: AlexNet

[Krizhevsky et al. 2012]

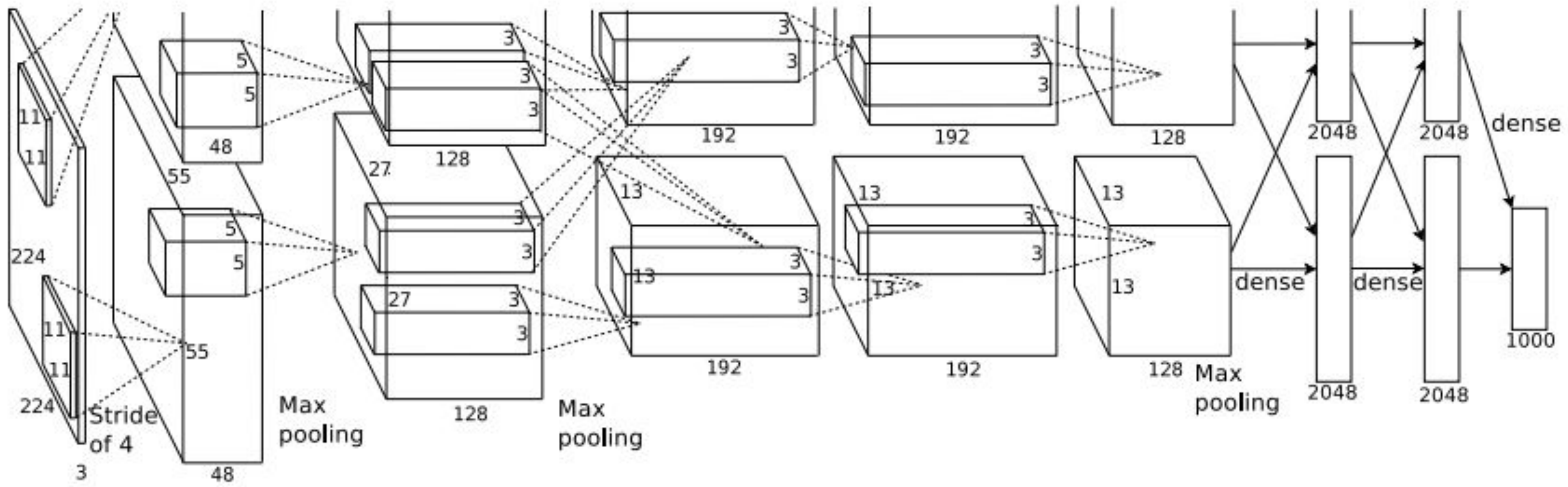


Input: 227x227x3 images

First layer (CONV1): 96 11x11 filters applied at stride 4
=>
Output volume **[55x55x96]**
Parameters: (11*11*3)*96 = **35K**

Case Study: AlexNet

[Krizhevsky et al. 2012]



Input: 227x227x3 images
After CONV1: 55x55x96

Second layer (POOL1): 3x3 filters applied at stride 2

Q: what is the output volume size? Hint: (55-3)/2+1 = 27

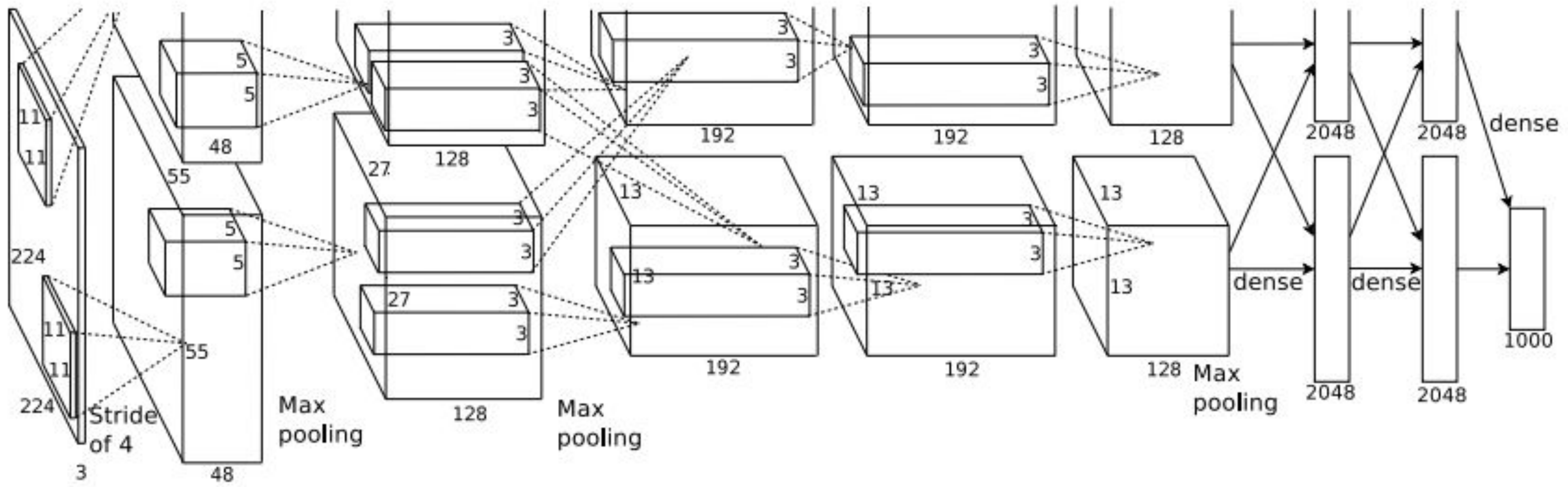
(recall:)
(N - F) / stride + 1

Figure copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Case Study: AlexNet

[Krizhevsky et al. 2012]



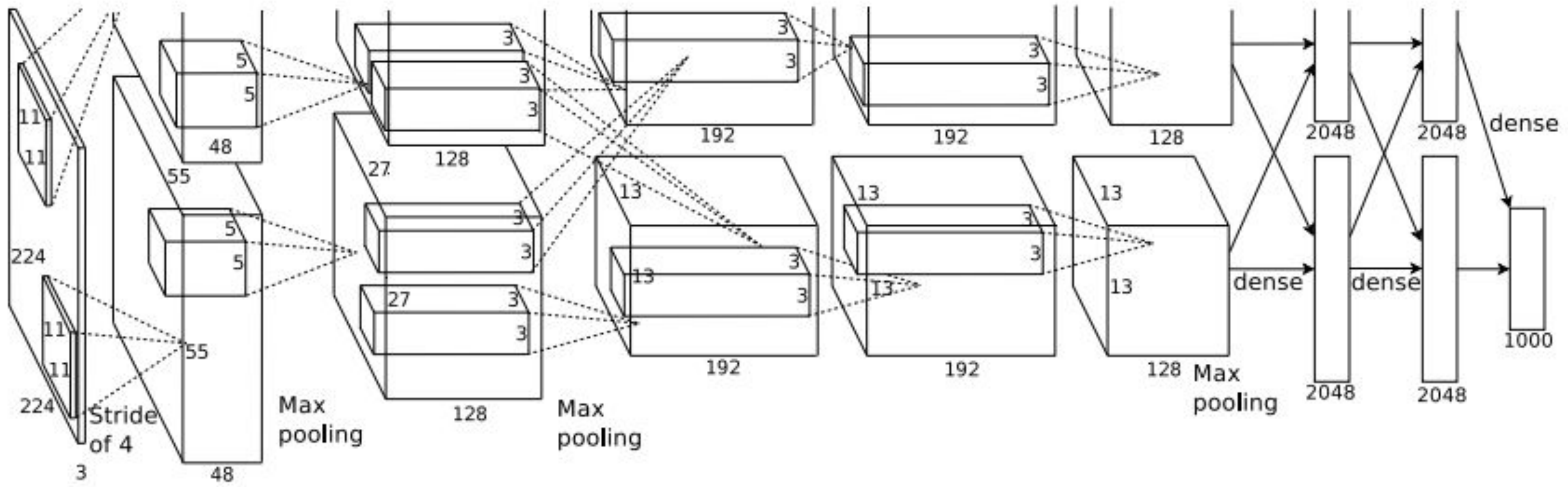
Input: 227x227x3 images
After CONV1: 55x55x96

Second layer (POOL1): 3x3 filters applied at stride 2
Output volume: 27x27x96

Q: what is the number of parameters in this layer?

Case Study: AlexNet

[Krizhevsky et al. 2012]



Input: 227x227x3 images
After CONV1: 55x55x96

Second layer (POOL1): 3x3 filters applied at stride 2
Output volume: 27x27x96
Parameters: 0!

Case Study: AlexNet

[Krizhevsky et al. 2012]

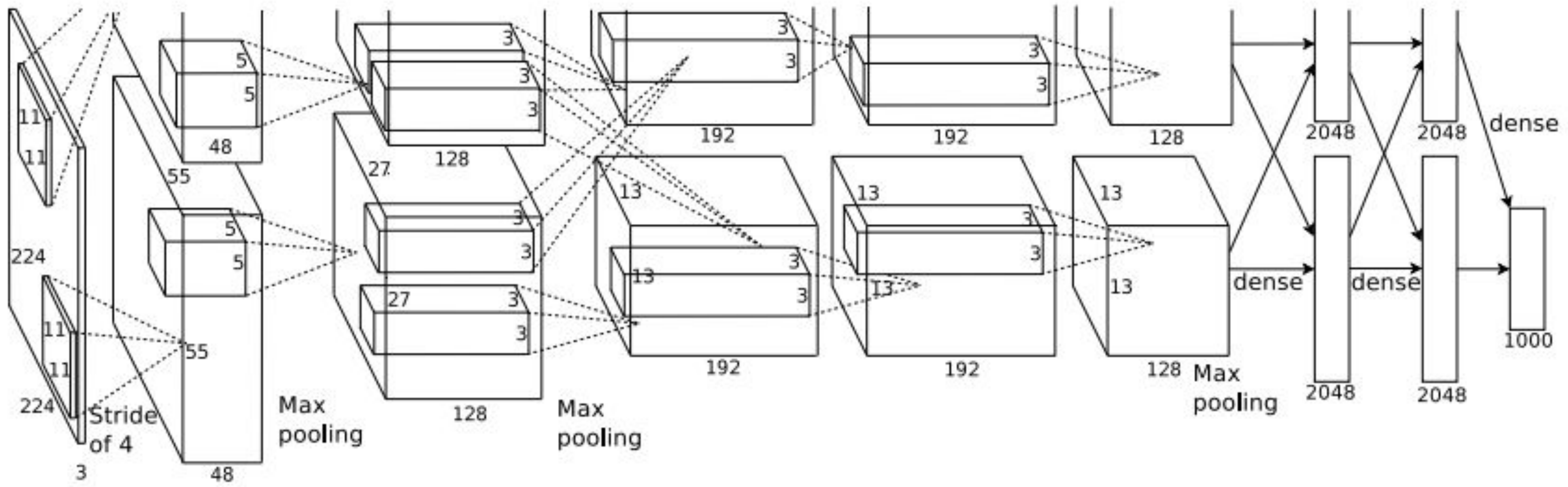


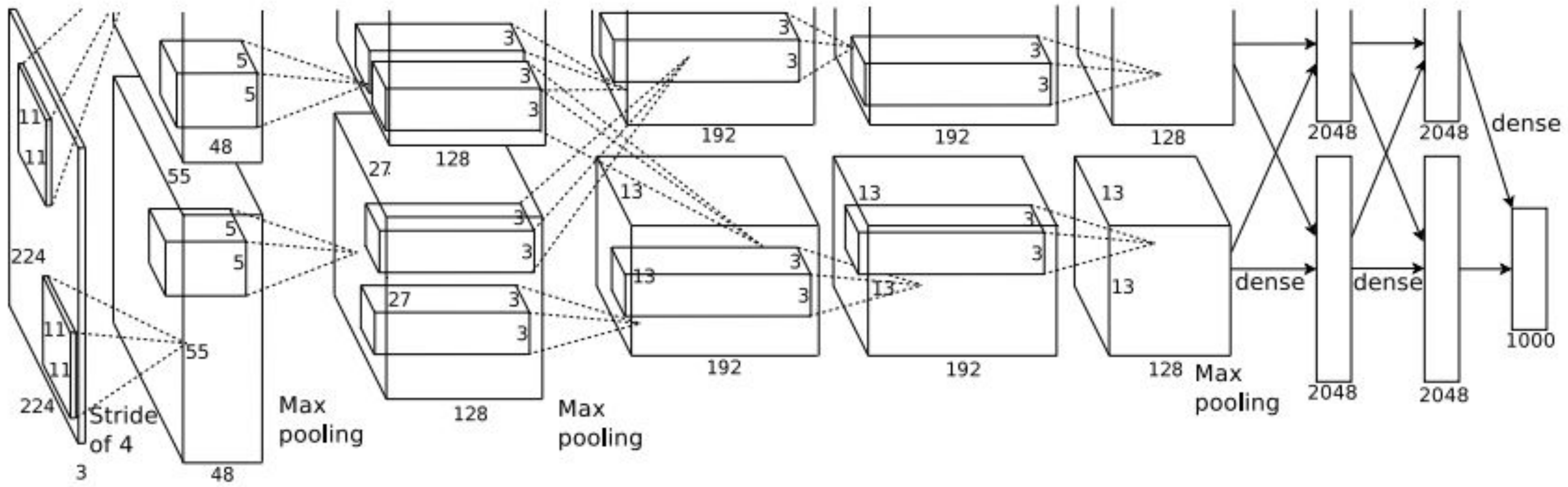
Figure copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.

Input: 227x227x3 images
After CONV1: 55x55x96
After POOL1: 27x27x96
...

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Case Study: AlexNet

[Krizhevsky et al. 2012]



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)

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Case Study: AlexNet

[Krizhevsky et al. 2012]

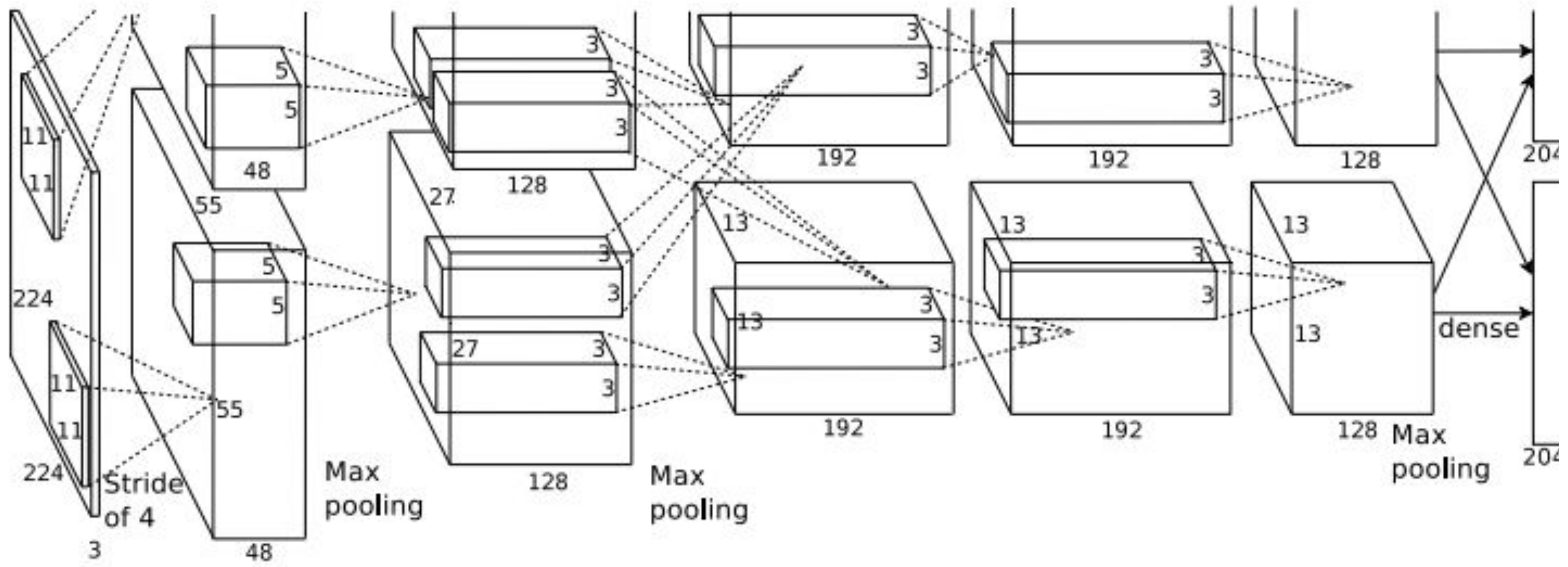
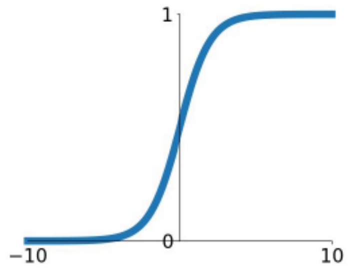


Figure copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.

Activation functions

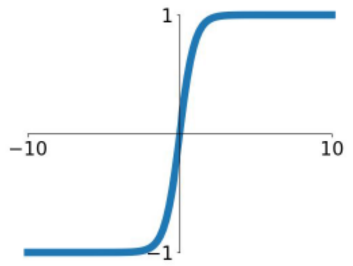
Sigmoid

$\sigma(x) = \frac{1}{1+e^{-x}}$



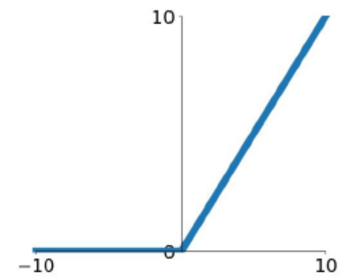
tanh

$\tanh(x)$



ReLU

$\max(0, x)$



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)

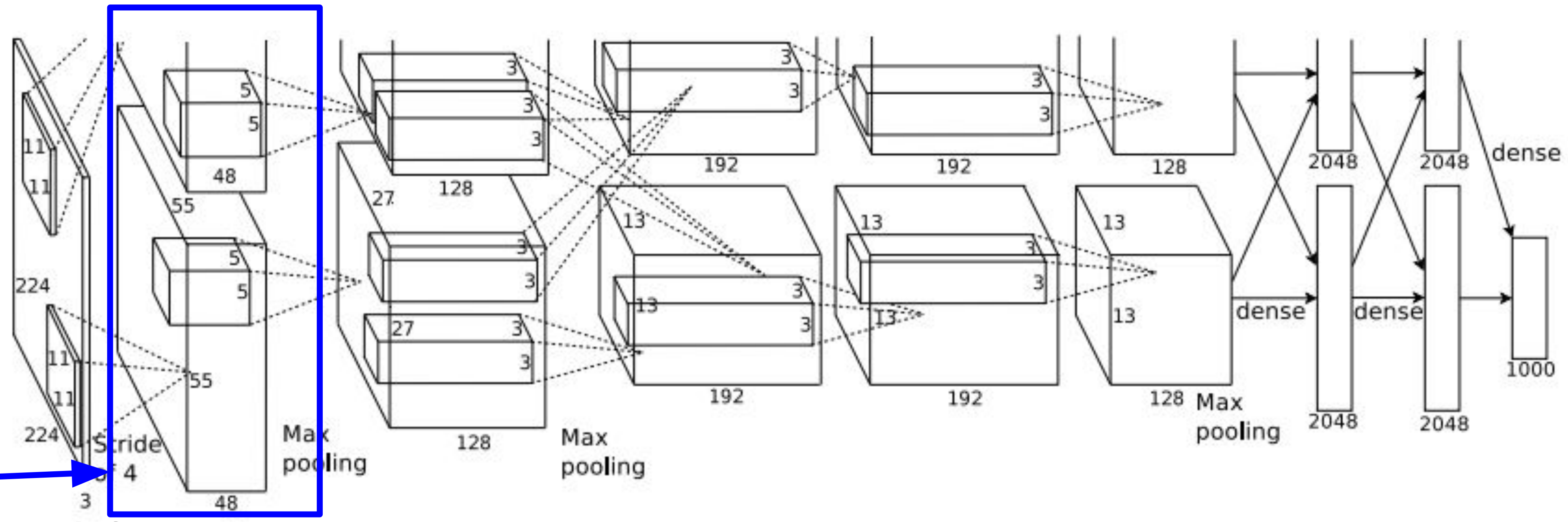
Details/Retrospectives:
- first use of ReLU
- used Norm layers (not common anymore)
- heavy data augmentation
- dropout 0.5
- batch size 128
- SGD Momentum 0.9
- Learning rate 1e-2, reduced by 10 manually when val accuracy plateaus
- L2 weight decay 5e-4
- 7 CNN ensemble: 18.2% -> 15.4%

AlexNet: Total parameters: 60 million

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Case Study: AlexNet

[Krizhevsky et al. 2012]



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)

[55x55x48] x 2

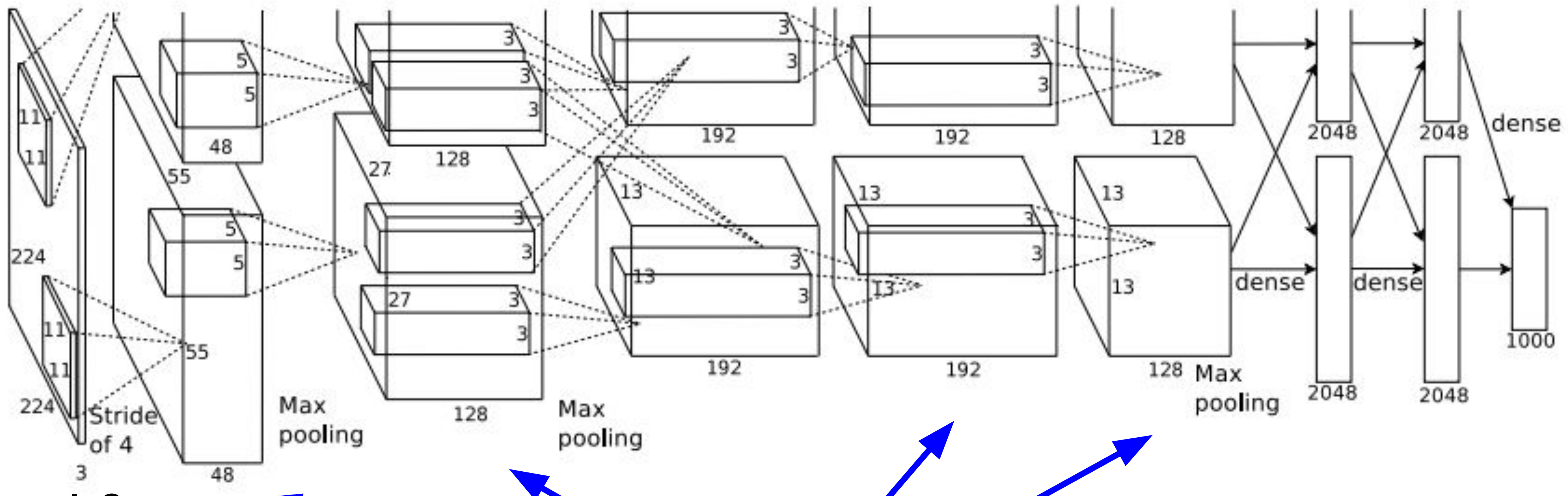
Historical note: Trained on GTX 580 GPU with only 3 GB of memory. Network spread across 2 GPUs, half the neurons (feature maps) on each GPU.

AlexNet: Total parameters: 60 million

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Case Study: AlexNet

[Krizhevsky et al. 2012]



CONV1, CONV2, CONV4, CONV5:
Connections only with feature maps on same GPU

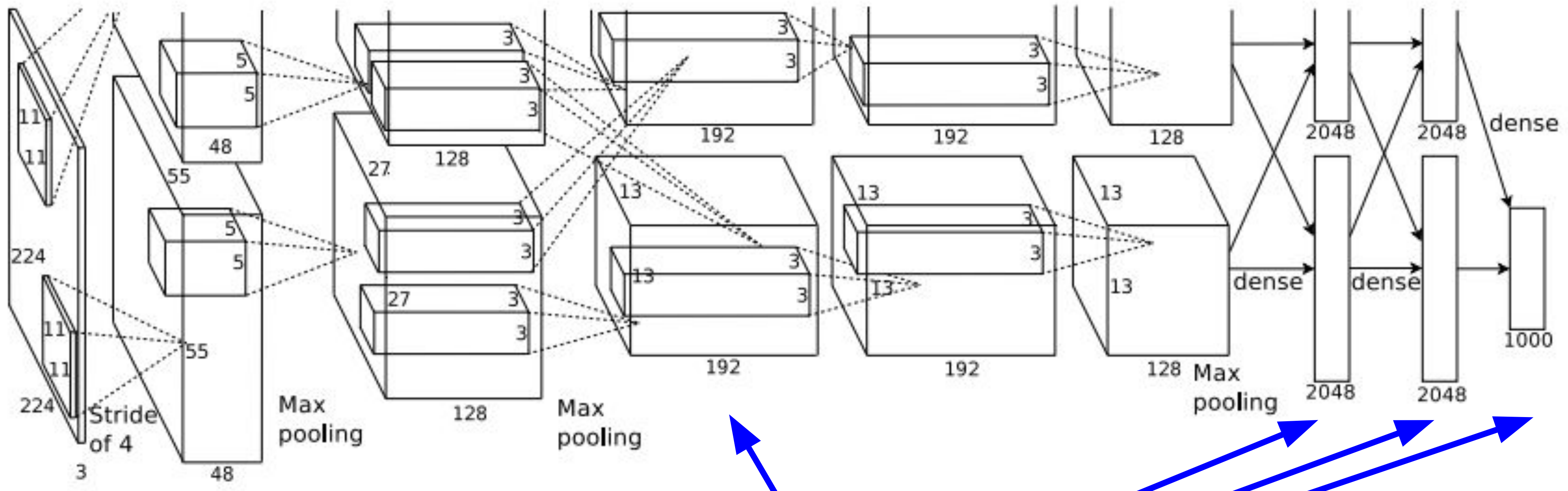
Full (simplified) AlexNet architecture:
[227x227x3] INPUT
[55x55x96] CONV1: 96 11x11 filters at stride 4, pad 0
[27x27x96] MAX POOL1: 3x3 filters at stride 2
[27x27x96] NORM1: Normalization layer
[27x27x256] CONV2: 256 5x5 filters at stride 1, pad 2
[13x13x256] MAX POOL2: 3x3 filters at stride 2
[13x13x256] 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)

AlexNet: Total parameters: 60 million

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Case Study: AlexNet

[Krizhevsky et al. 2012]



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)

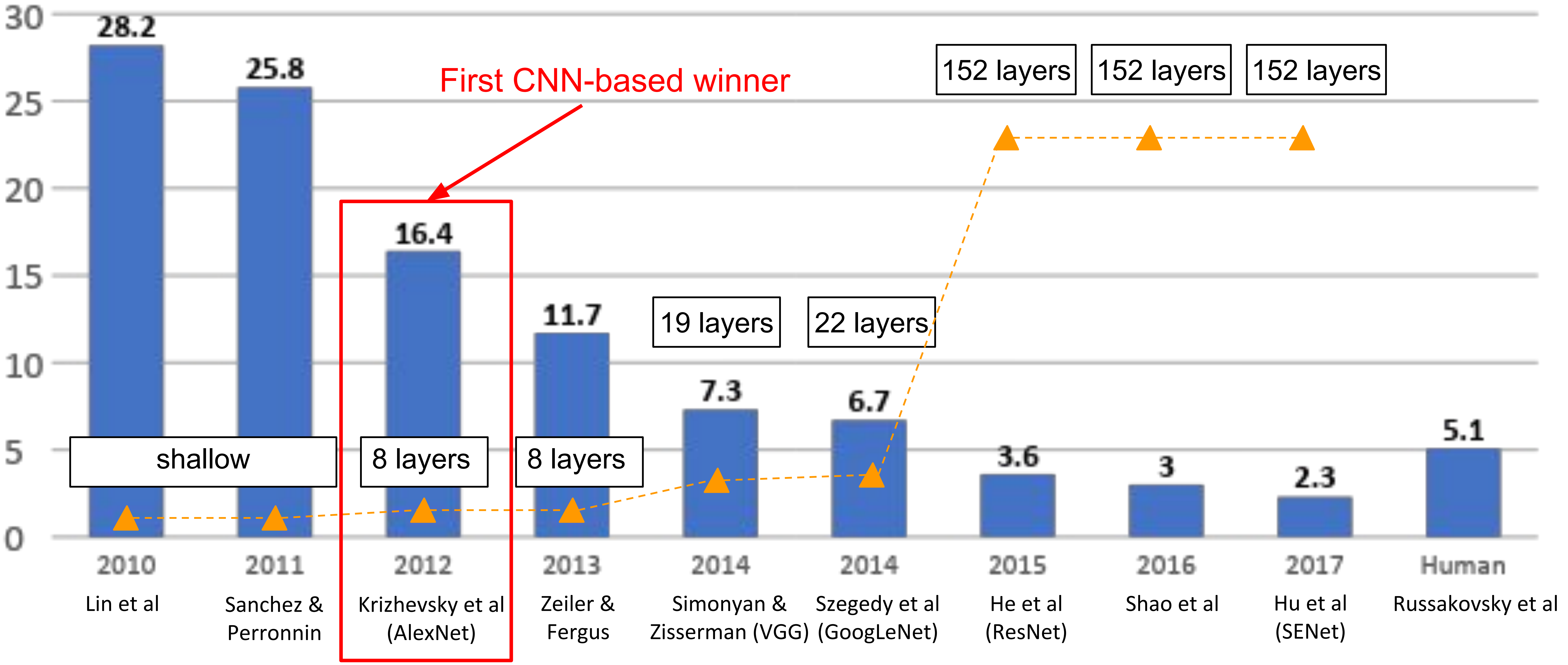
CONV3, FC6, FC7, FC8:
Connections with all feature maps in preceding layer, communication across GPUs

AlexNet: Total parameters: 60 million

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

ImageNet Large Scale Visual Recognition Challenge (ILSVRC)

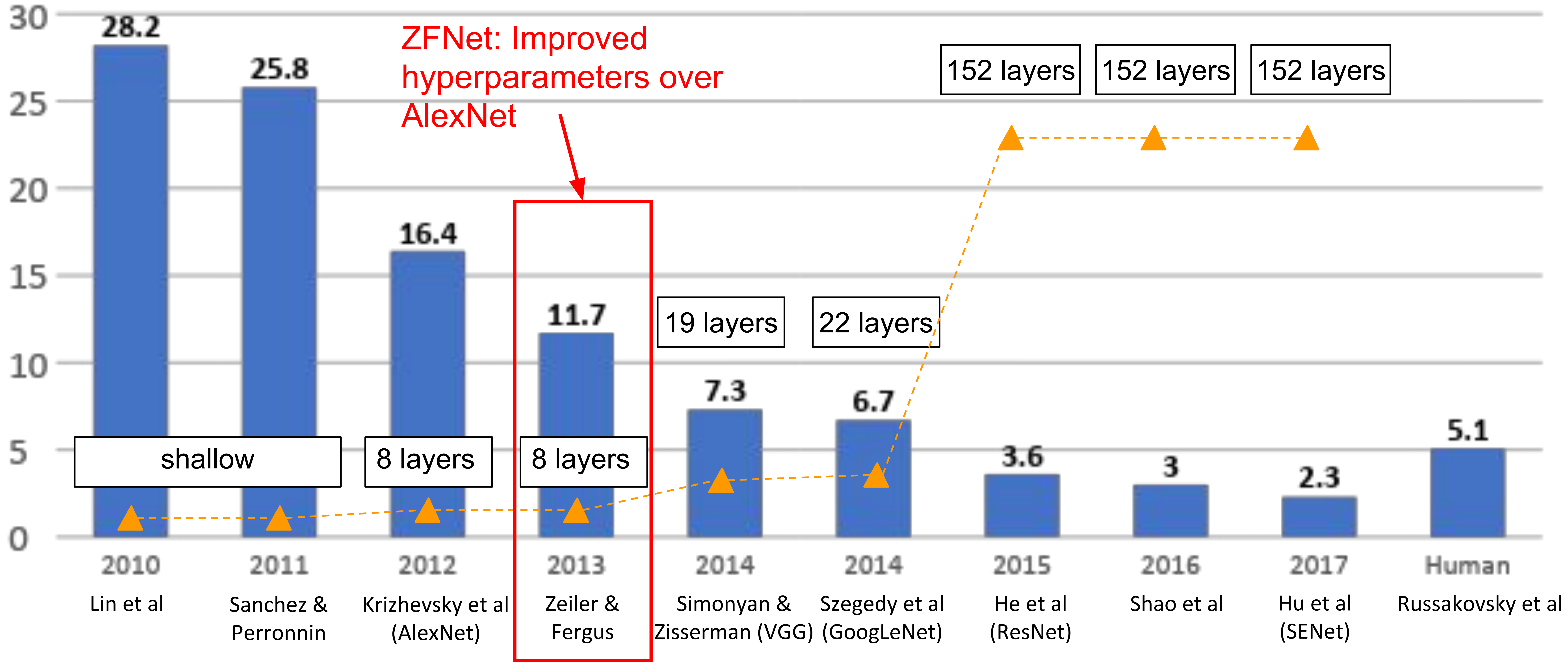
- Winners over the years (top-5 error):



slide credit: Fei-Fei, Justin Johnson, Serena Yeung

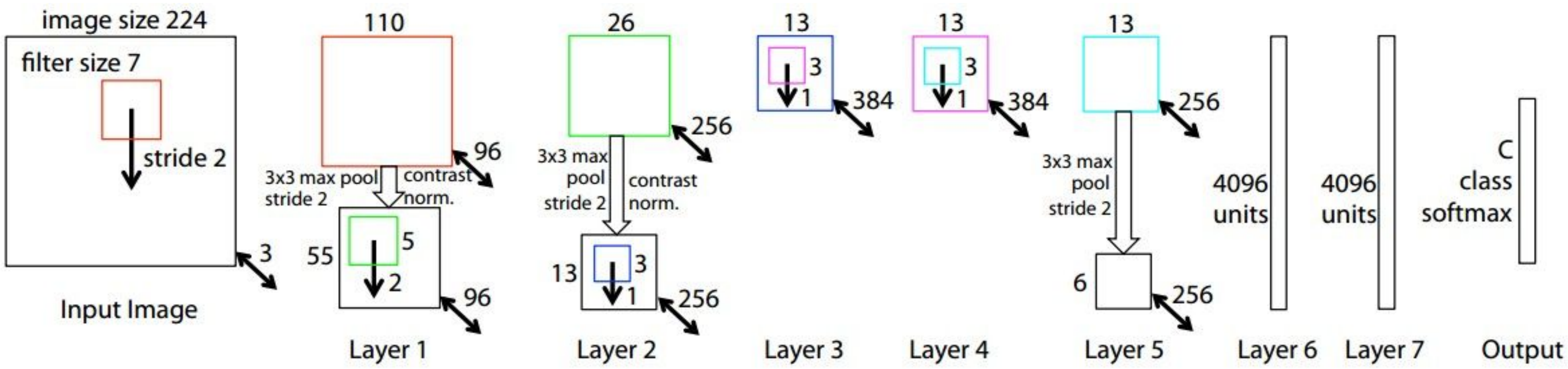
ImageNet Large Scale Visual Recognition Challenge (ILSVRC)

- Winners over the years (top-5 error)



slide credit: Fei-Fei, Justin Johnson, Serena Yeung

ZFNet [Zeiler & Fergus, 2013]



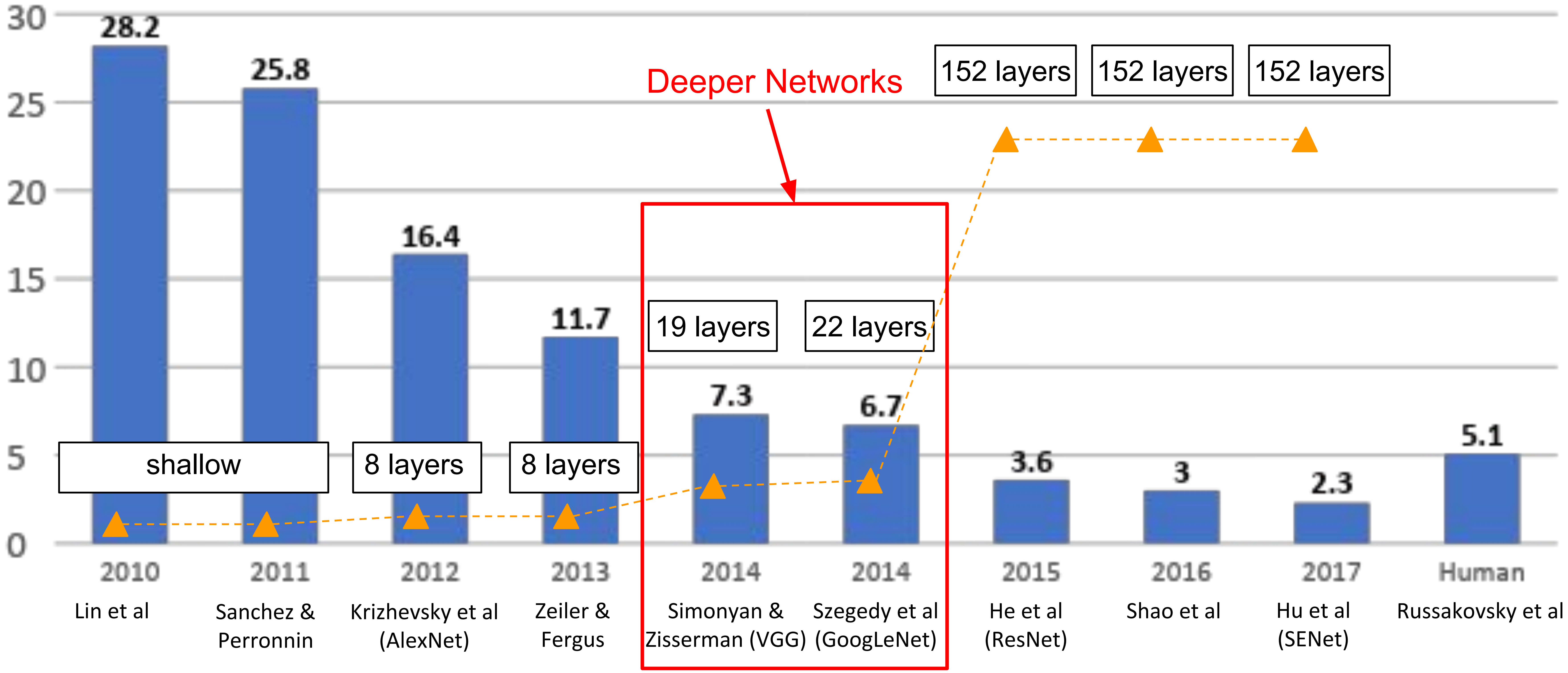
AlexNet but:
CONV1: change from (11x11 stride 4) to (7x7 stride 2)
CONV3,4,5: instead of 384, 384, 256 filters use 512, 1024, 512

ImageNet top 5 error: 16.4% -> 11.7%

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

ImageNet Large Scale Visual Recognition Challenge (ILSVRC)

- Winners over the years (top-5 error):



slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Case Study: VGGNet

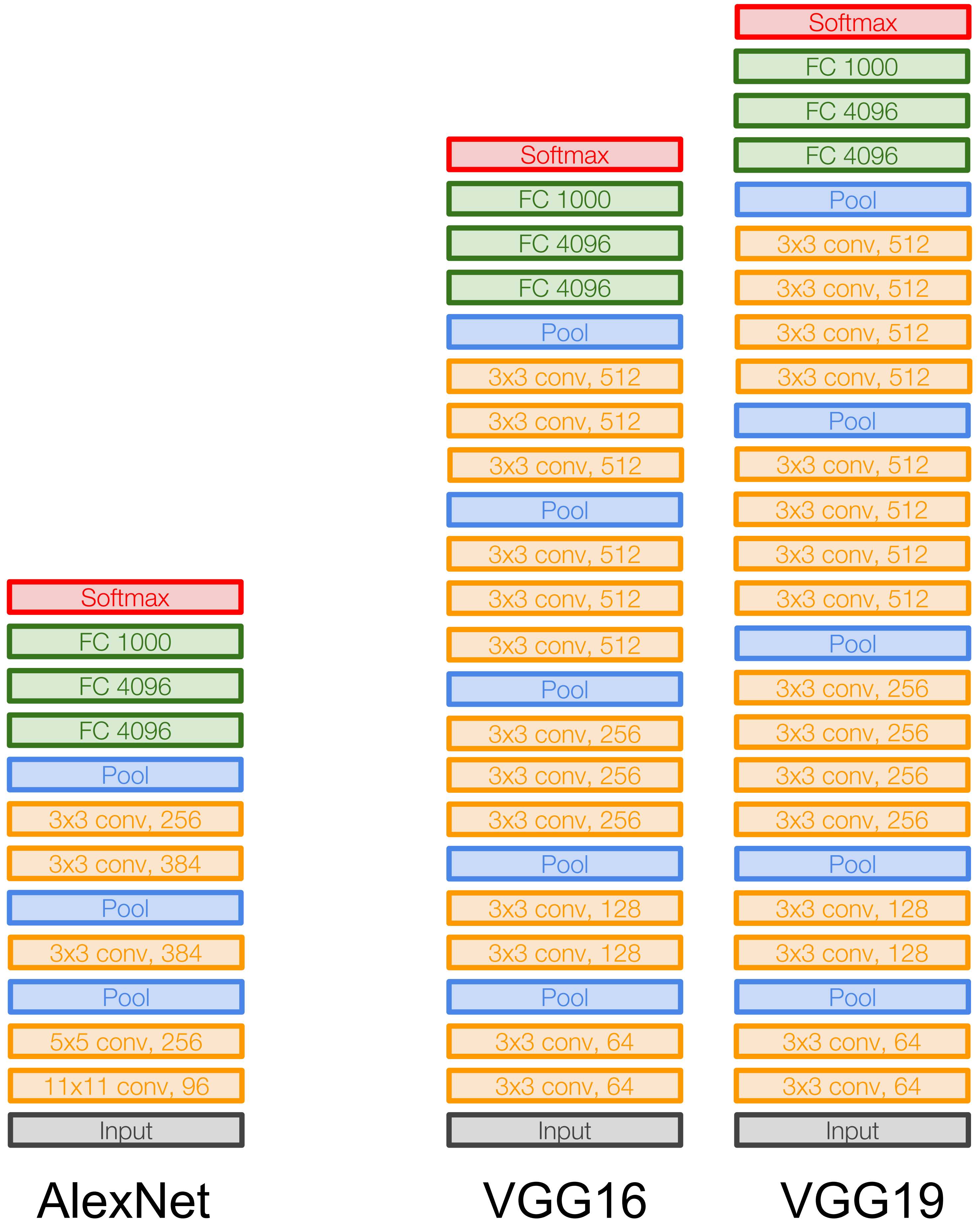
[Simonyan and Zisserman, 2014]

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

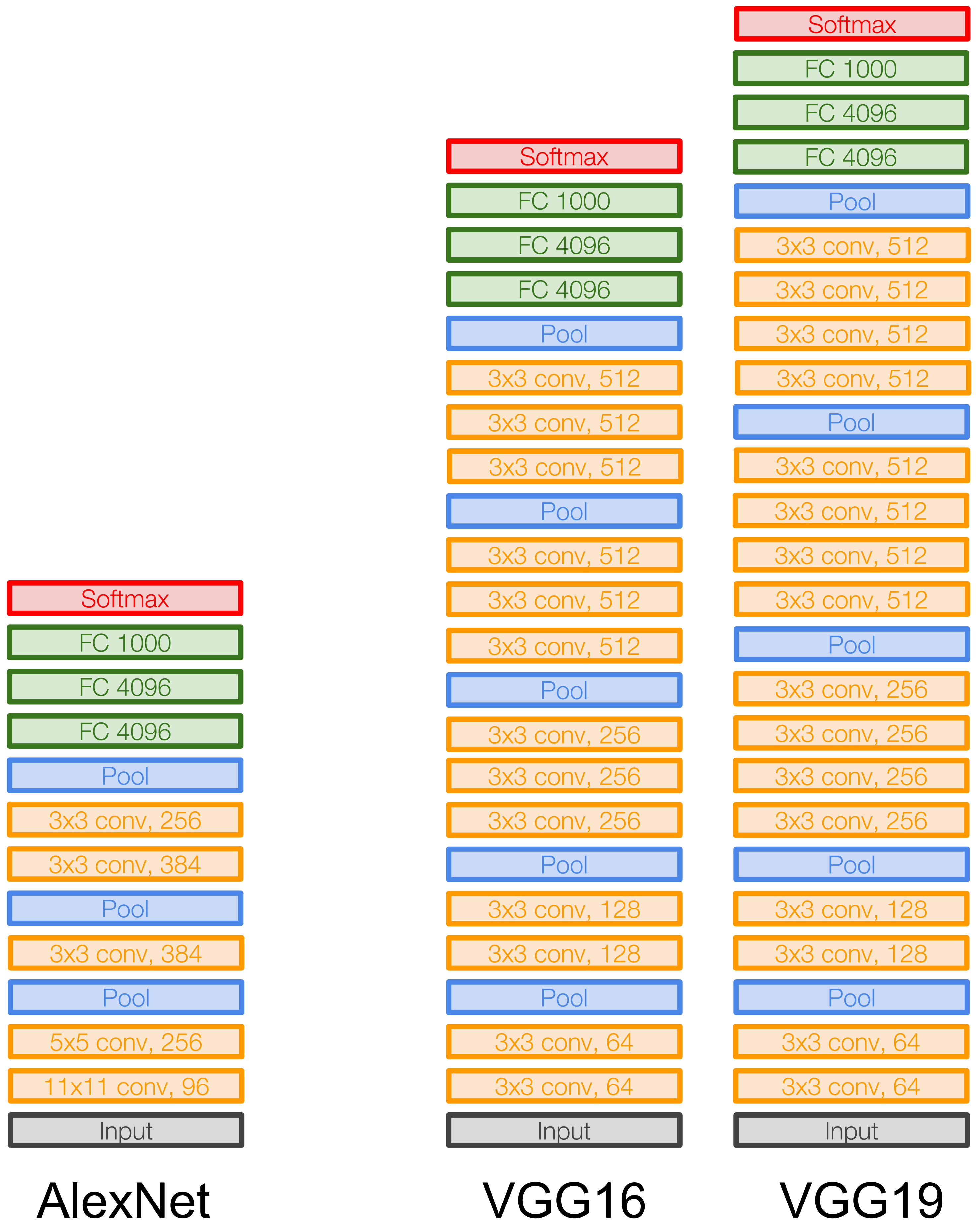


slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Case Study: VGGNet

[Simonyan and Zisserman, 2014]

Q: Why use smaller filters? (3x3 conv)

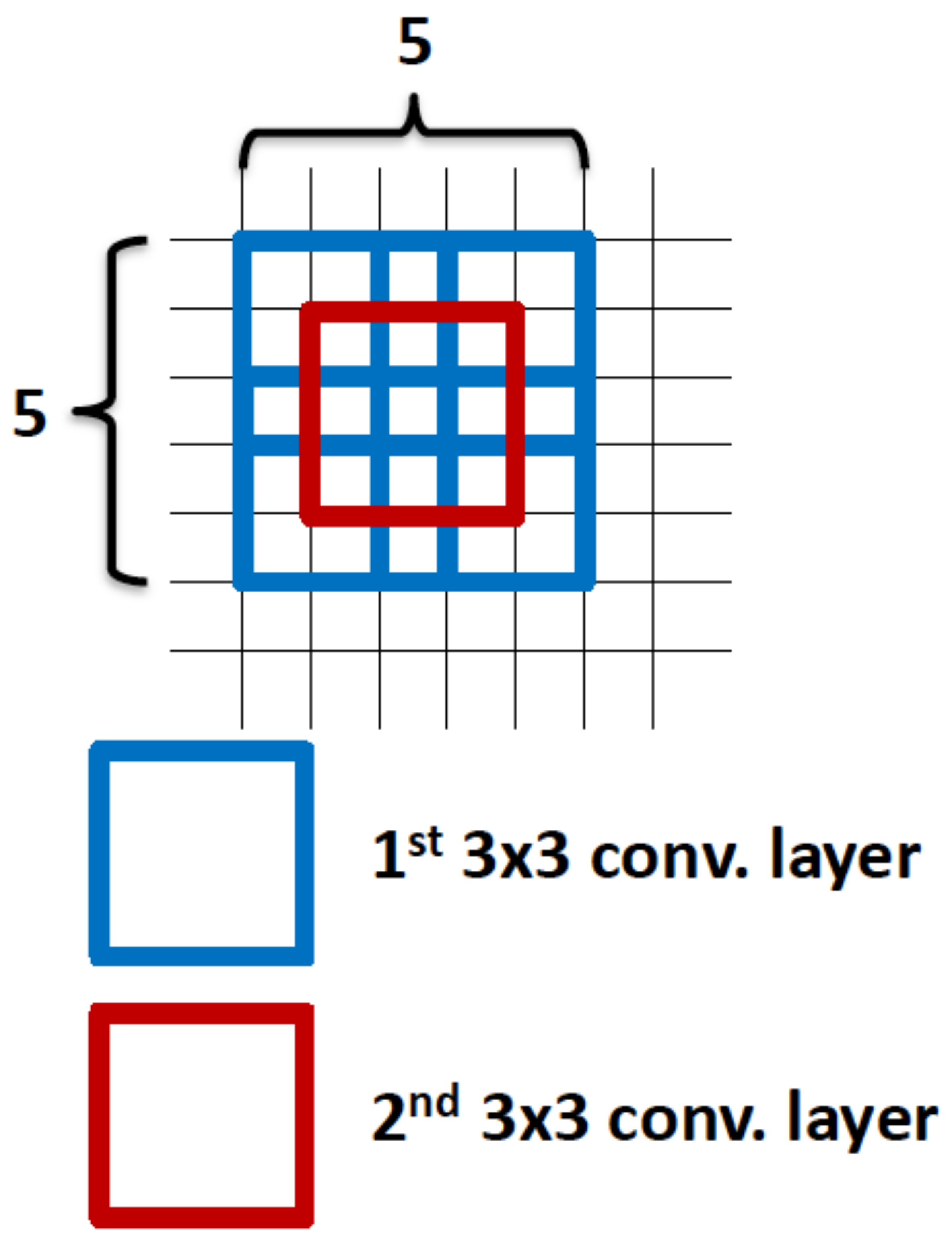


slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Case Study: VGGNet

Why 3x3 layers?

- Stacked conv. layers have a large receptive field
  - two 3x3 layers – 5x5 receptive field
  - three 3x3 layers – 7x7 receptive field
- More non-linearity
- Less parameters to learn



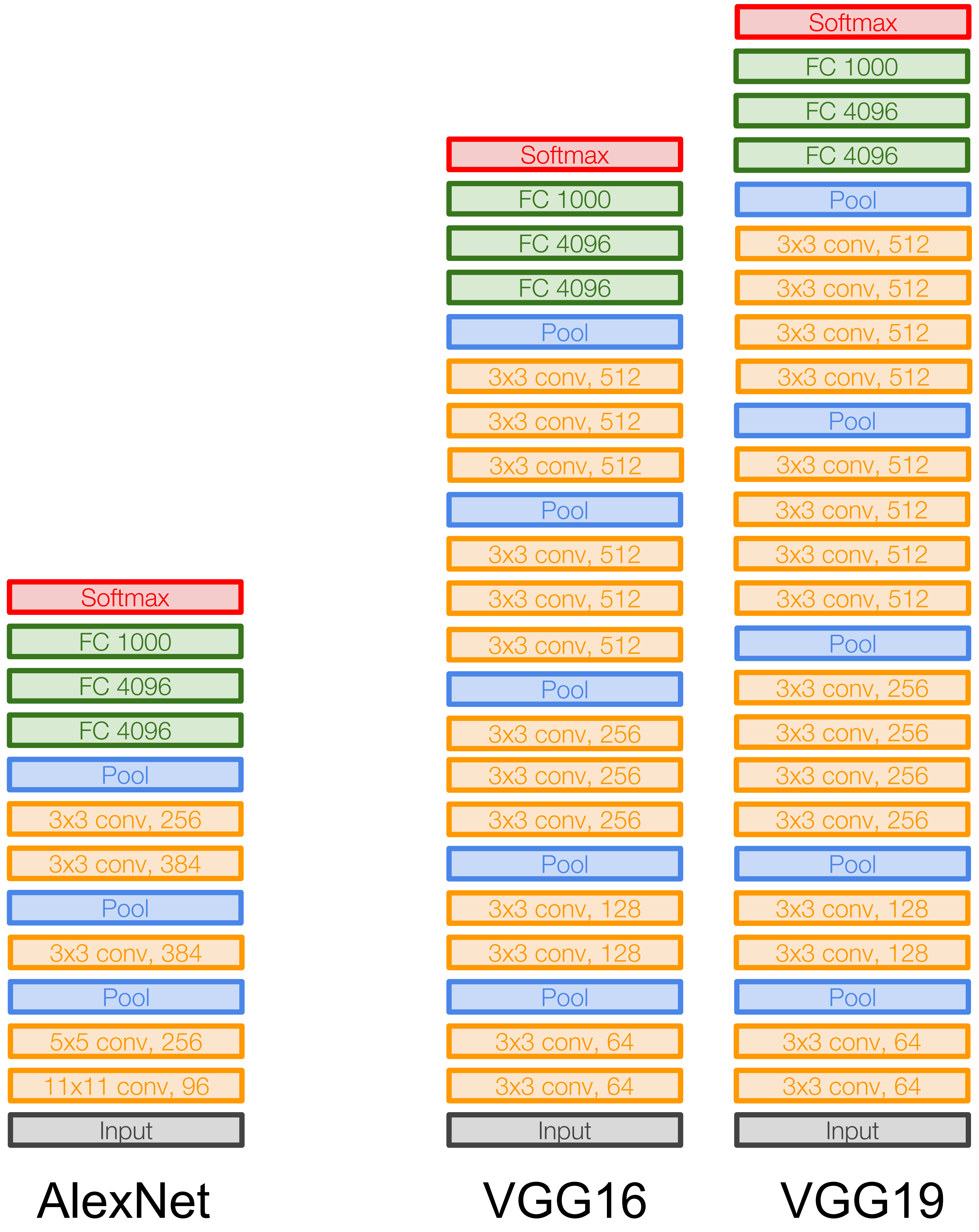
Case Study: VGGNet

[Simonyan and Zisserman, 2014]

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

[7x7]



slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Case Study: VGGNet

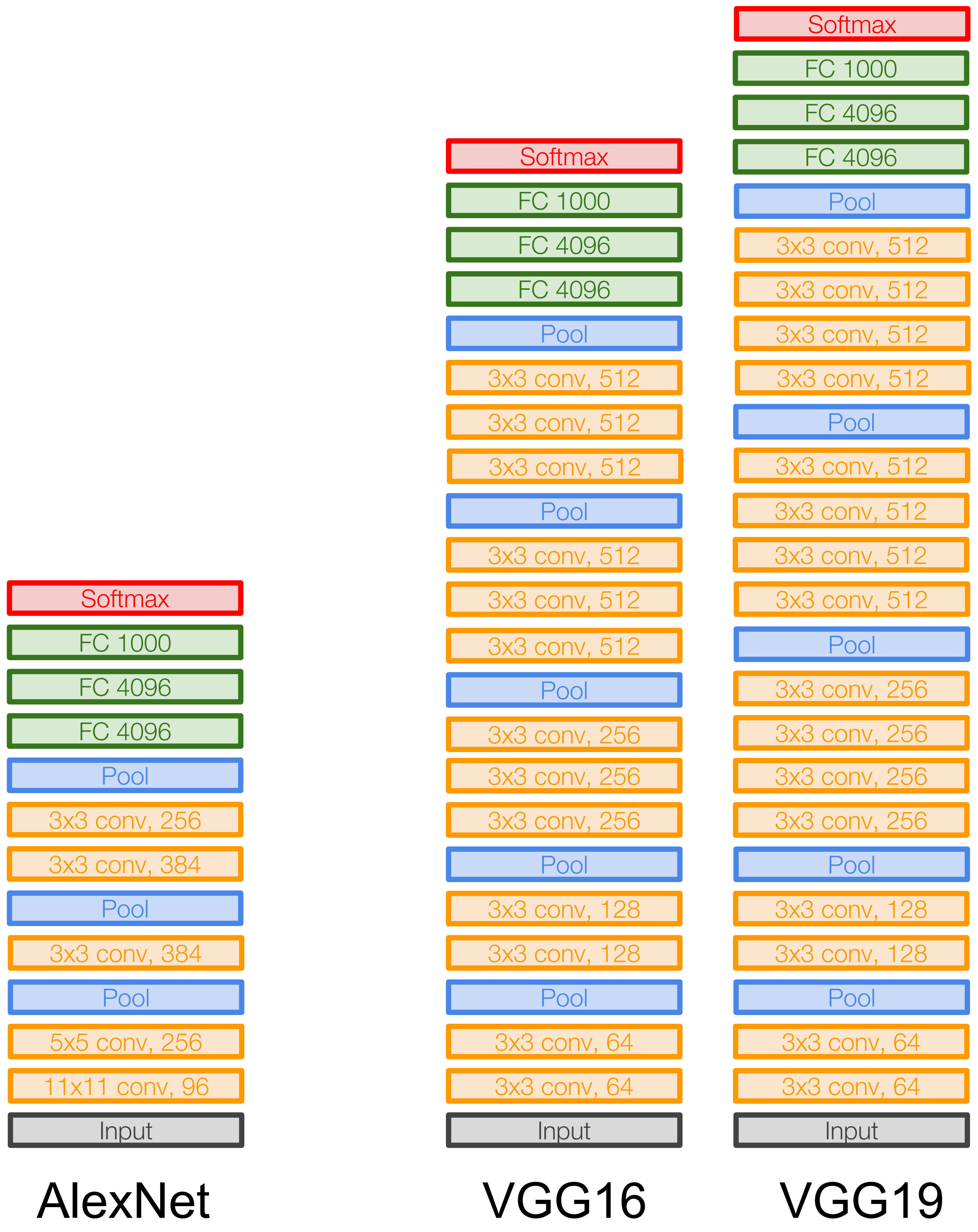
[Simonyan and Zisserman, 2014]

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: $3 * (3^2C^2)$ vs. 7^2C^2 for C channels per layer



slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Case Study: VGGNet

(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

TOTAL memory: 24M * 4 bytes ~= 96MB / image (for a forward pass)
TOTAL params: 138M parameters

VGG16

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Case Study: VGGNet

INPUT: [224x224x3] memory: 224*224*3=150K params: 0 (not counting biases)
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

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

Note:

Most memory is in early CONV

Most params are in late FC

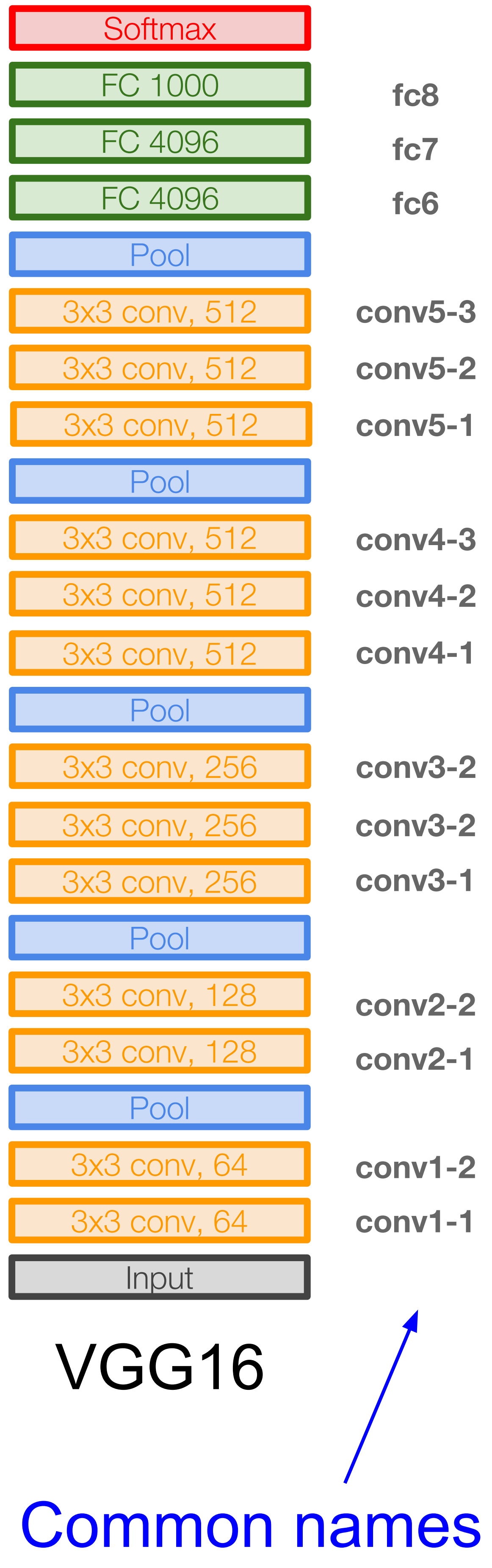
slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Case Study: VGGNet

(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

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

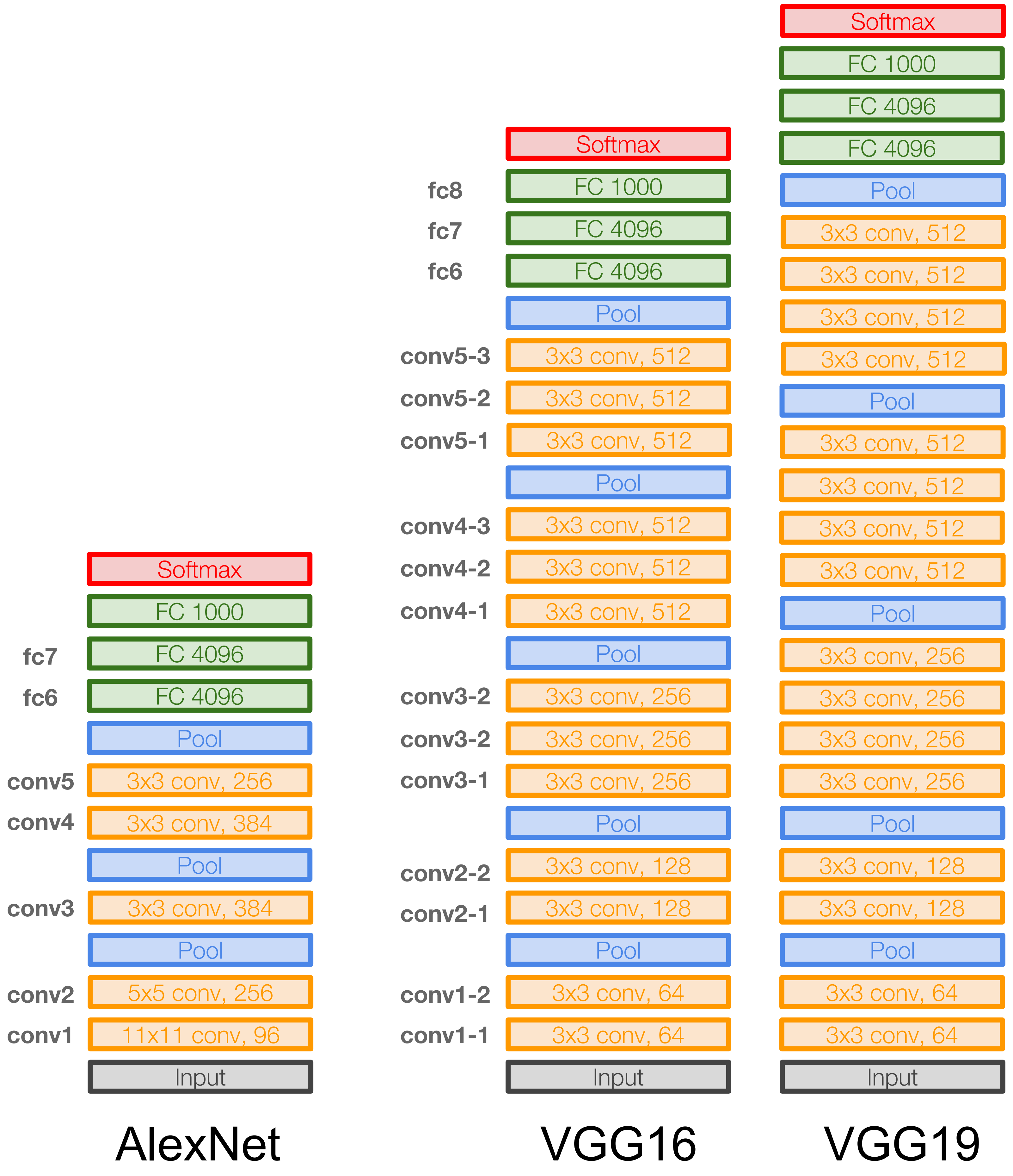


slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Case Study: VGGNet

Details:

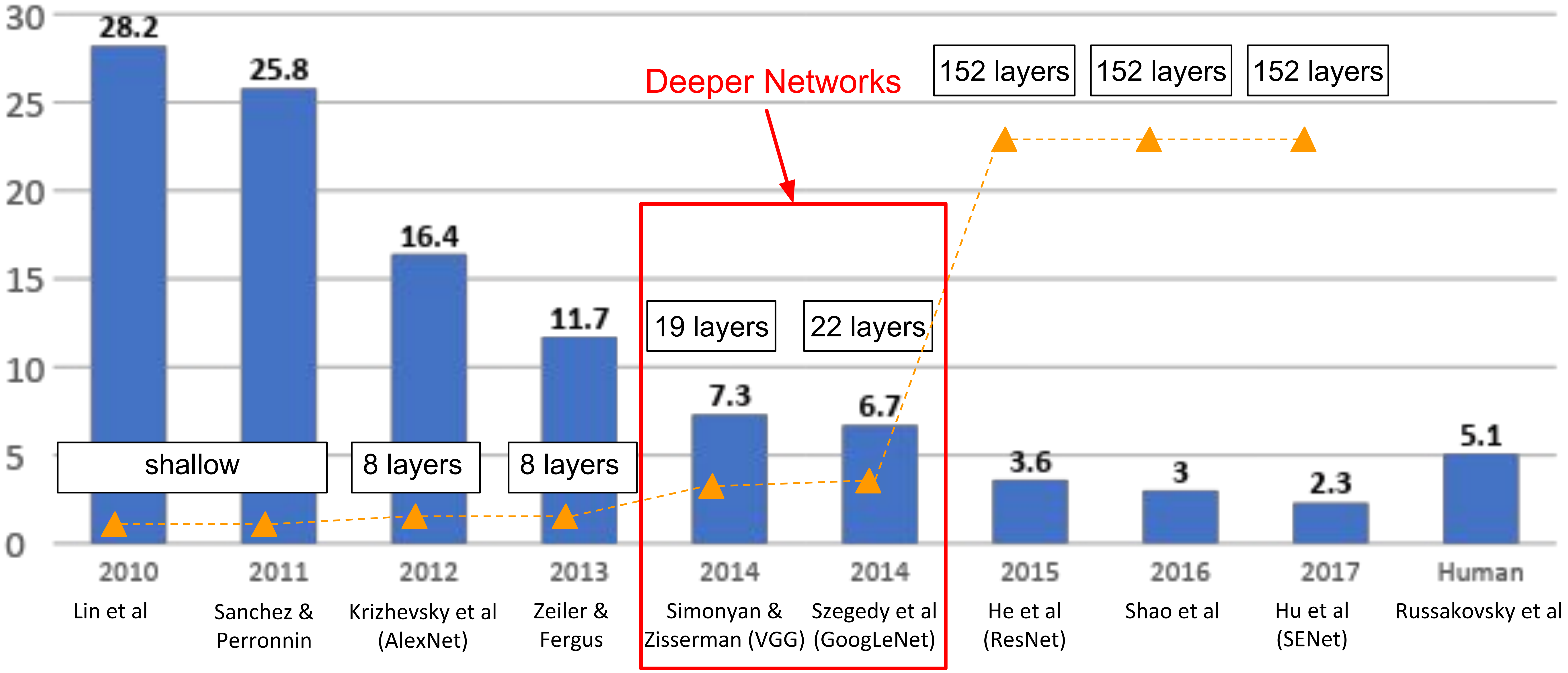
- ILSVRC’14 2nd in classification, 1st in localization
- Similar training procedure as Krizhevsky 2012
- No Local Response Normalisation (LRN)
- Use VGG16 or VGG19 (VGG19 only slightly better, more memory)
- Use ensembles for best results
- FC7 features generalize well to other tasks



slide credit: Fei-Fei, Justin Johnson, Serena Yeung

ImageNet Large Scale Visual Recognition Challenge (ILSVRC)

- Winners over the years (top-5 error):



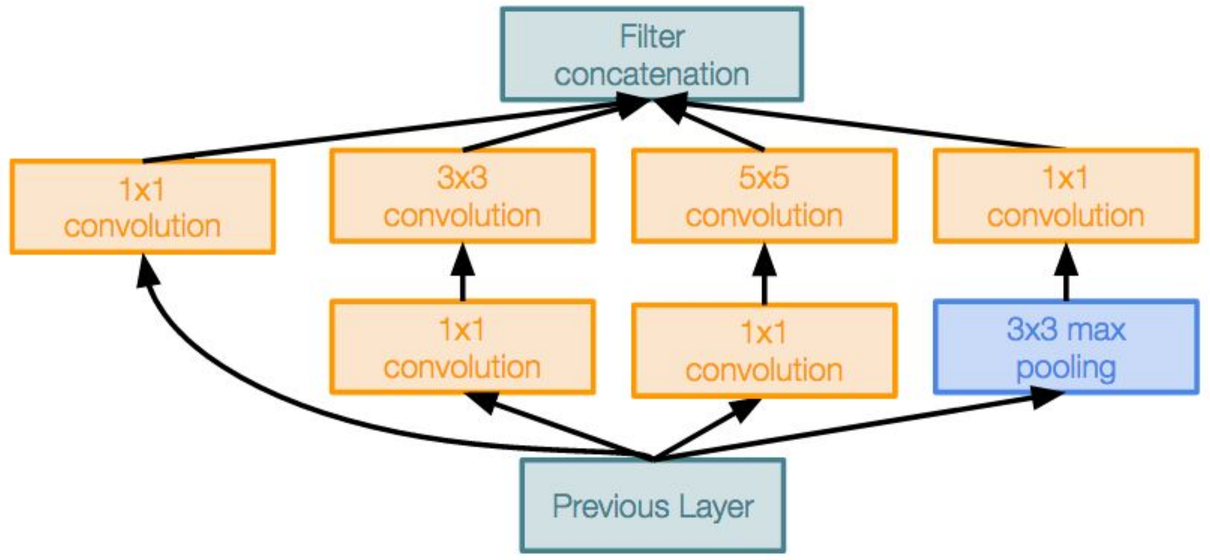
slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Case Study: GoogLeNet

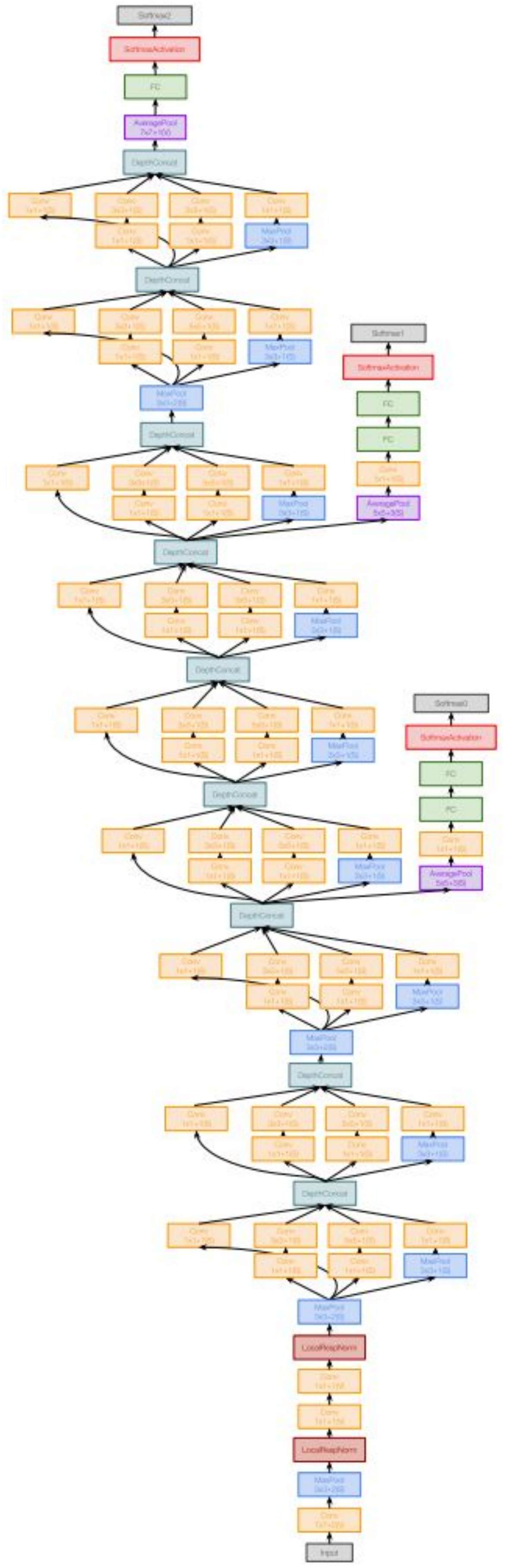
[Szegedy et al., 2014]

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

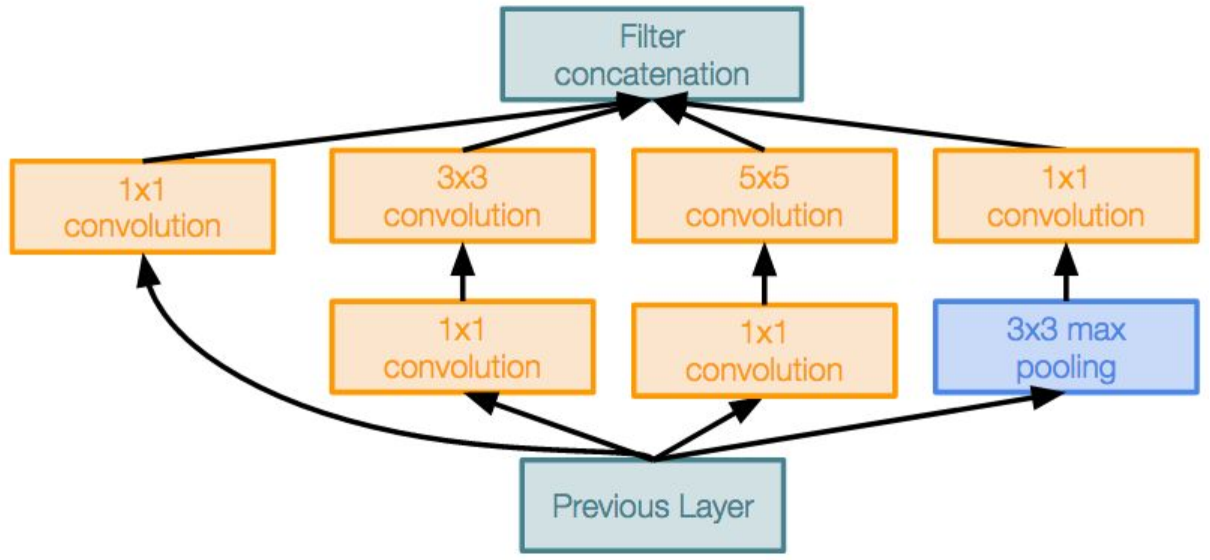


slide credit: Fei-Fei, Justin Johnson, Serena Yeung

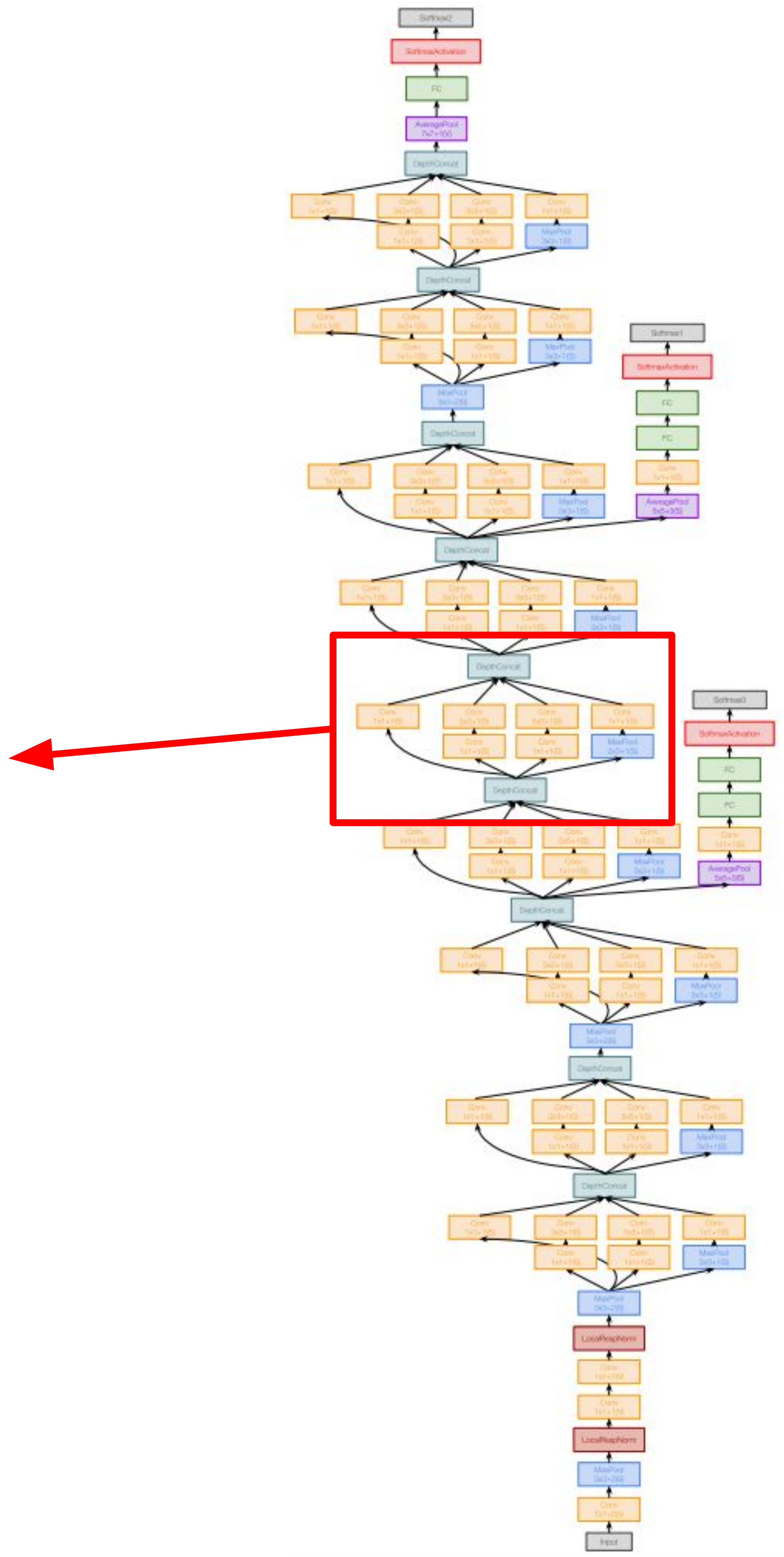
Case Study: GoogLeNet

[Szegedy et al., 2014]

“Inception module”: design a good local network topology (network within a network) and then stack these modules on top of each other



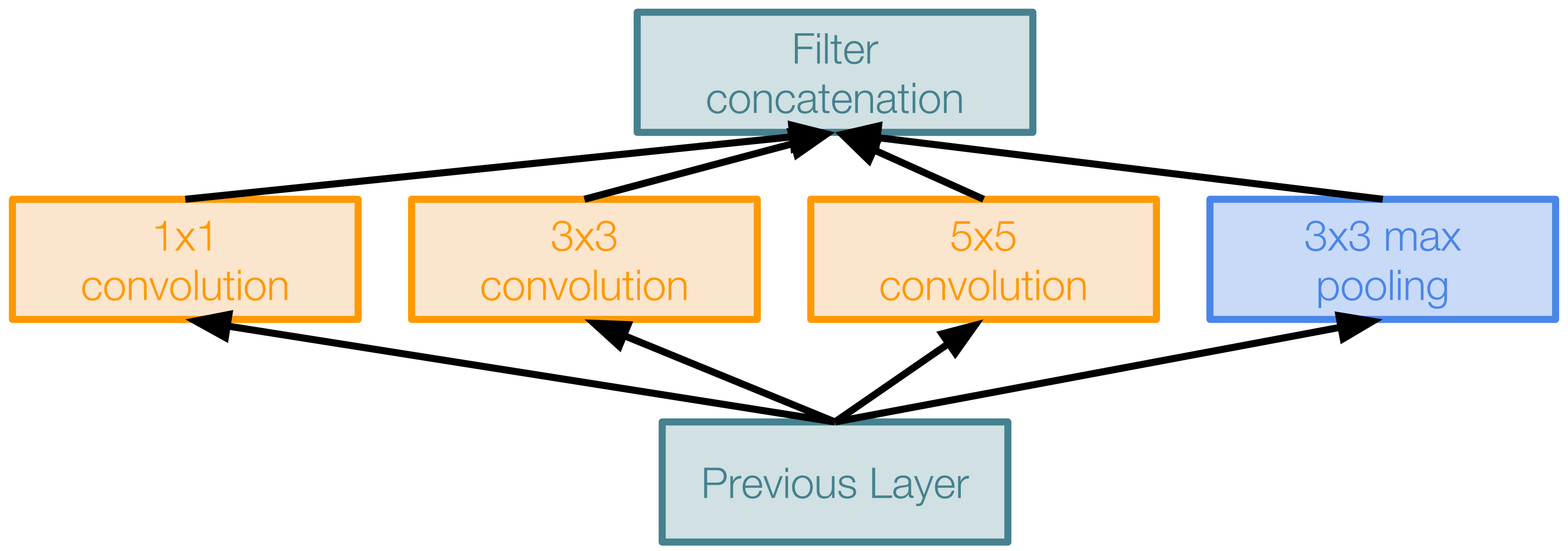
Inception module



slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Case Study: GoogLeNet

[Szegedy et al., 2014]



Naive Inception module

Apply parallel filter operations on the input from previous layer:

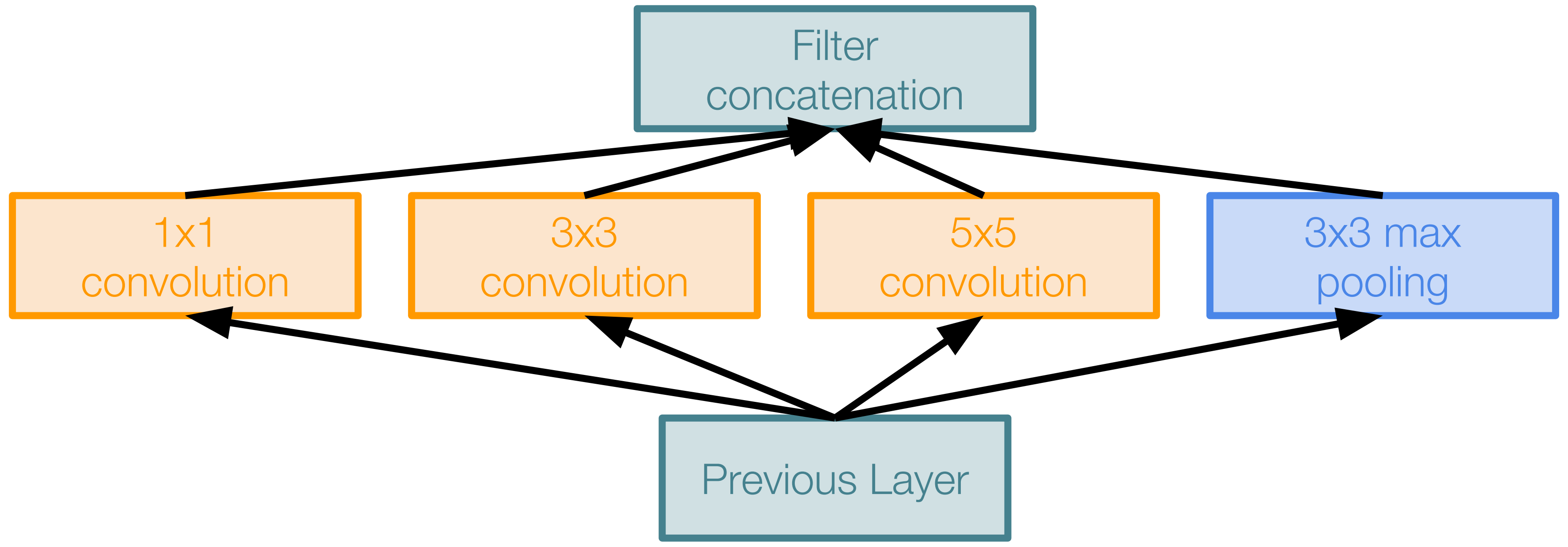
- Multiple receptive field sizes for convolution (1x1, 3x3, 5x5)
- Pooling operation (3x3)

Concatenate all filter outputs together depth-wise

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Case Study: GoogLeNet

[Szegedy et al., 2014]



Naive Inception module

Apply parallel filter operations on the input from previous layer:

- Multiple receptive field sizes for convolution (1x1, 3x3, 5x5)
- Pooling operation (3x3)

Concatenate all filter outputs together depth-wise

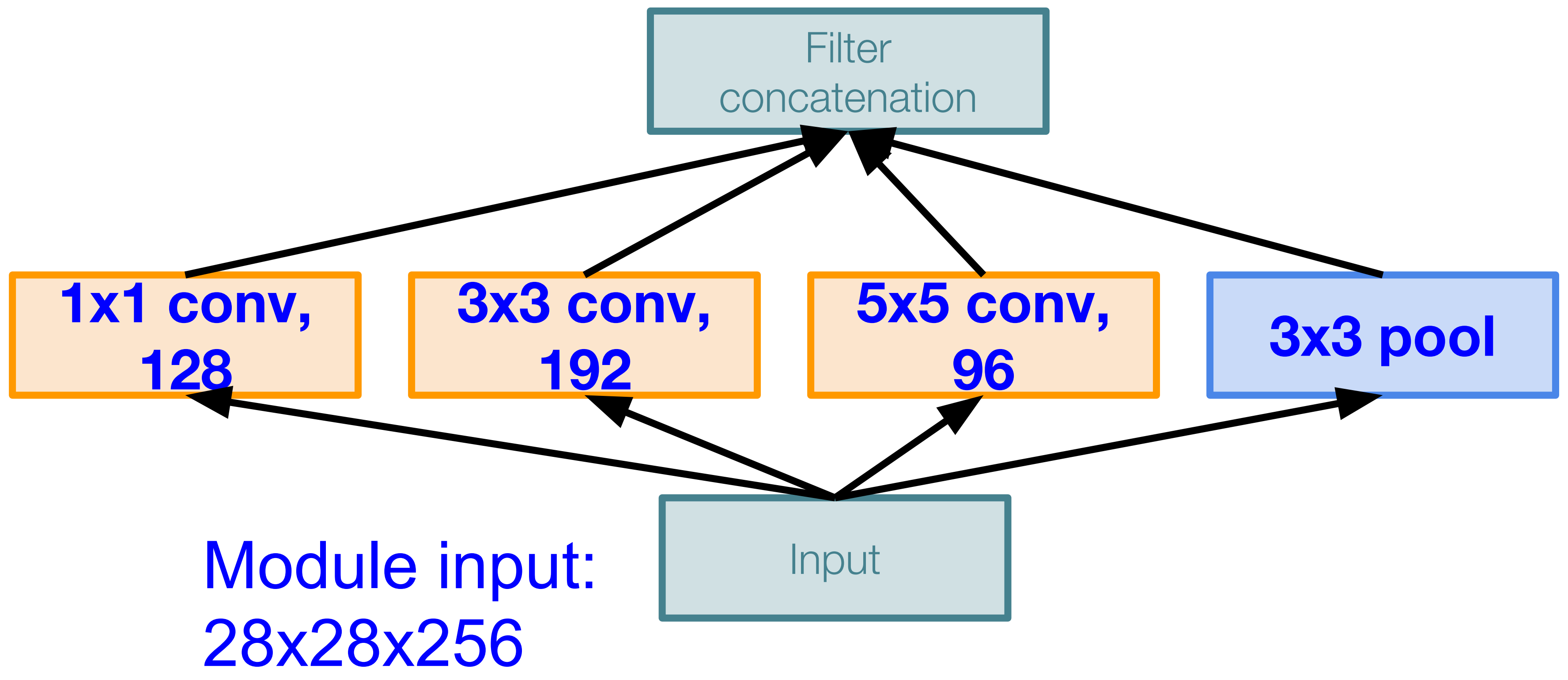
Q: What is the problem with this?
[Hint: Computational complexity]

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Case Study: GoogLeNet

[Szegedy et al., 2014]

Example:



Naive Inception module

Q: What is the problem with this?
[Hint: Computational complexity]

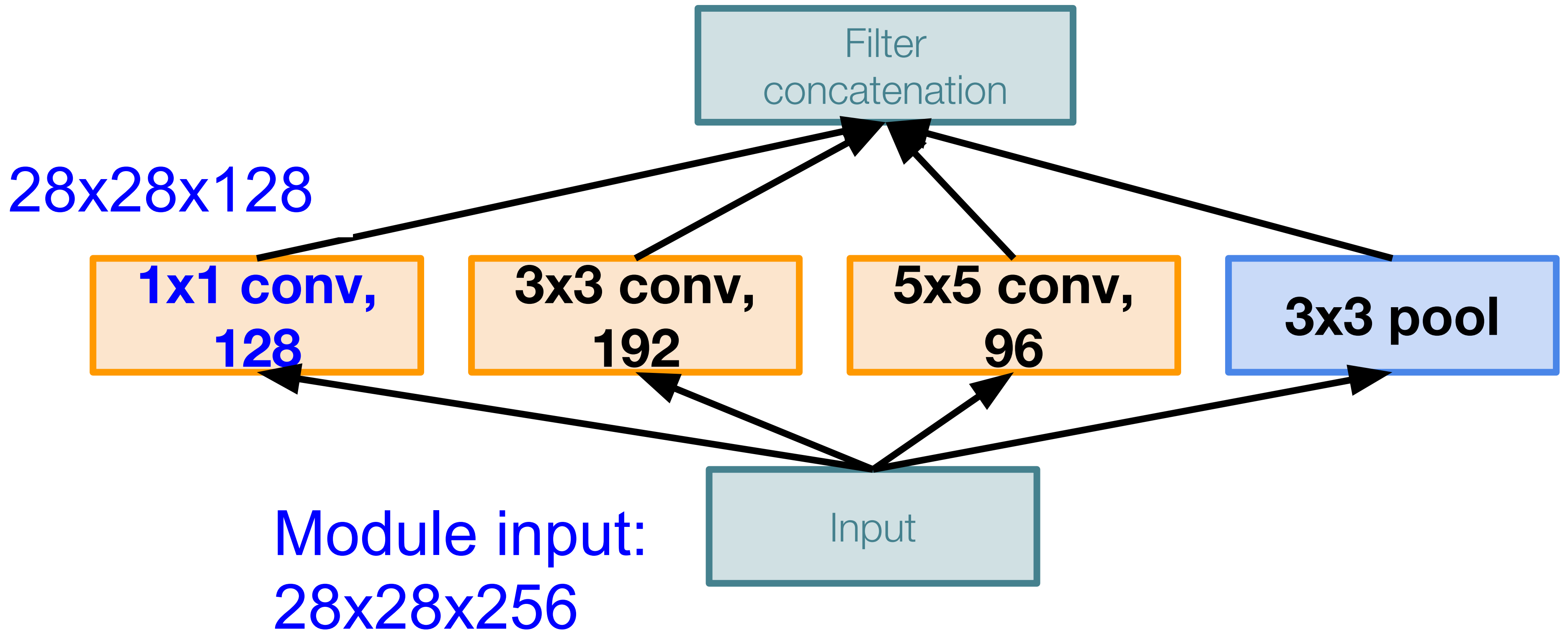
slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Case Study: GoogLeNet

[Szegedy et al., 2014]

Example:

Q1: What is the output size of the
1x1 conv, with 128 filters?



Naive Inception module

Q: What is the problem with this?
[Hint: Computational complexity]

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

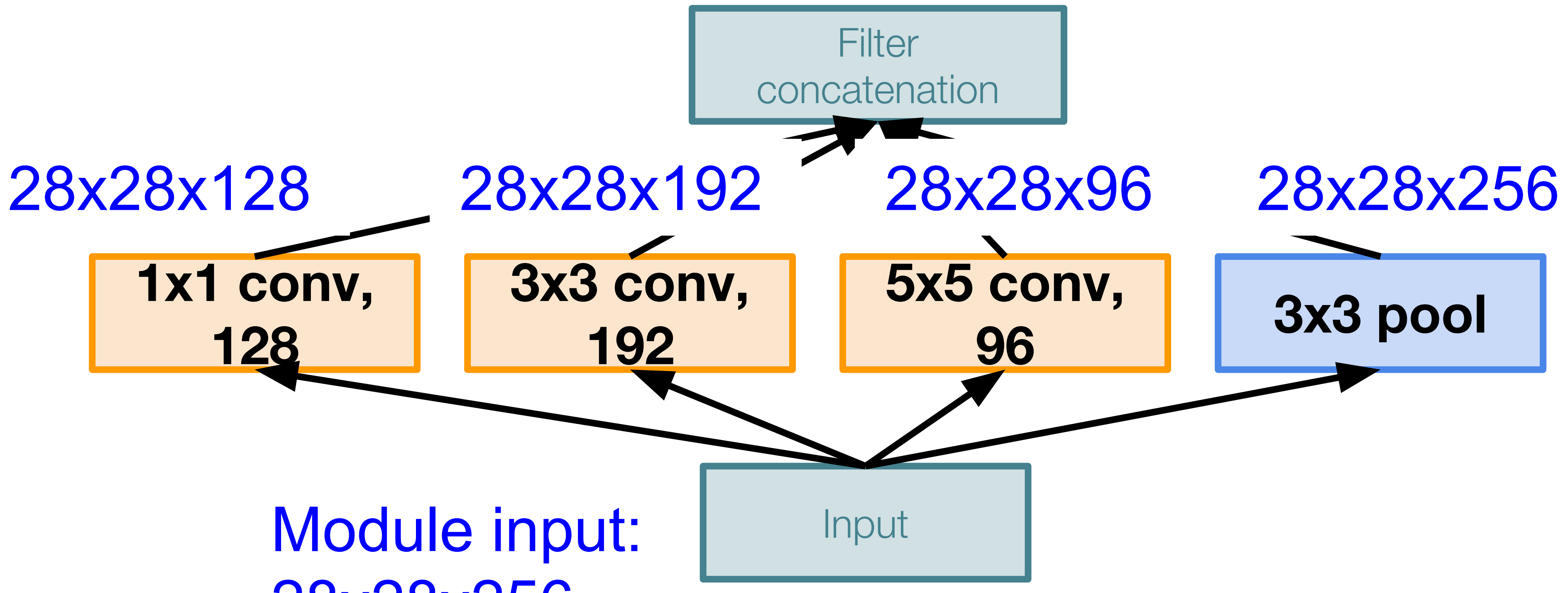
Case Study: GoogLeNet

[Szegedy et al., 2014]

Example:

Q2: What are the output sizes of all different filter operations?

Q: What is the problem with this?
[Hint: Computational complexity]



Module input: 28x28x256

Naive Inception module

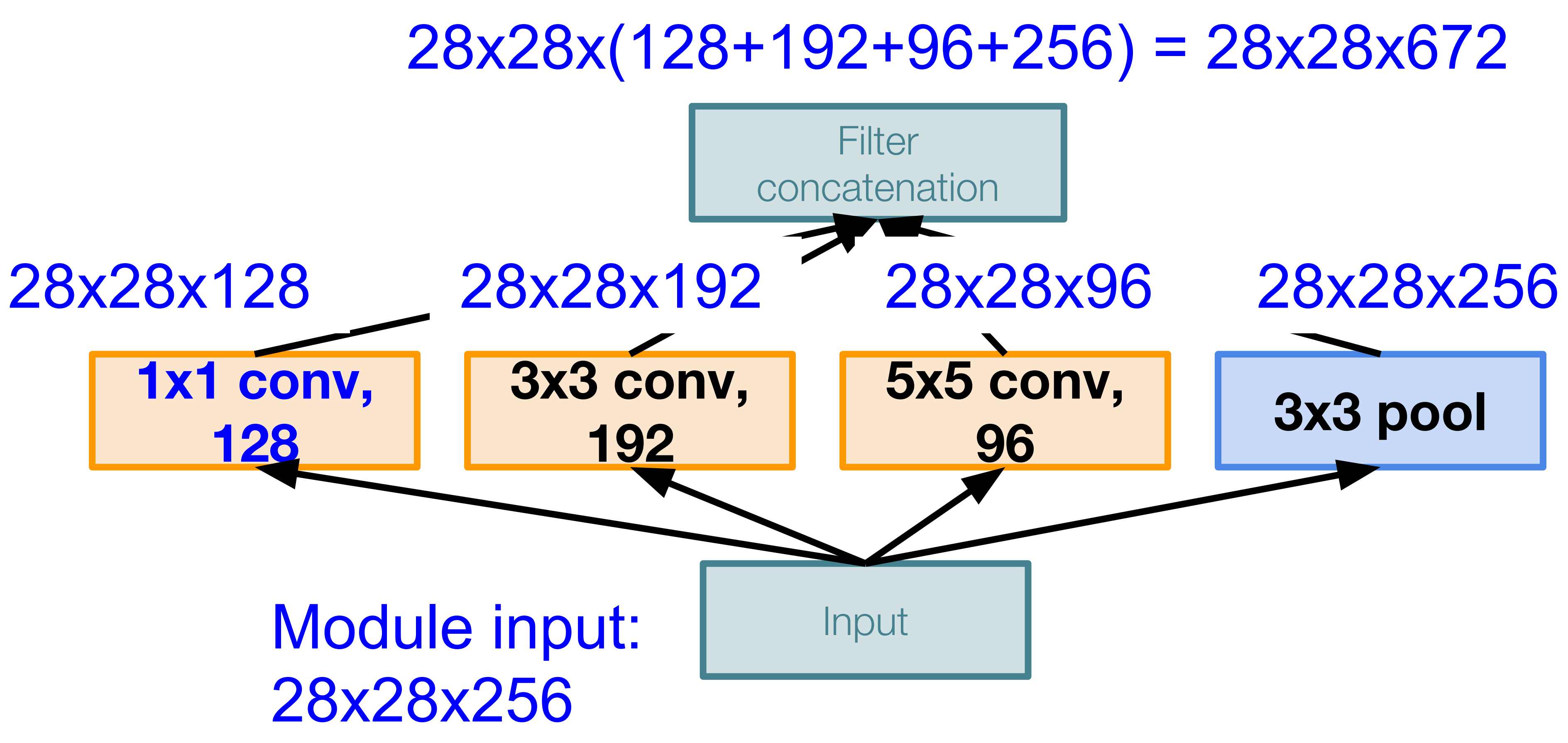
slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Case Study: GoogLeNet

[Szegedy et al., 2014]

Example:

Q3:What is output size after
filter concatenation?



Naive Inception module

Q: What is the problem with this?
[Hint: Computational complexity]

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!

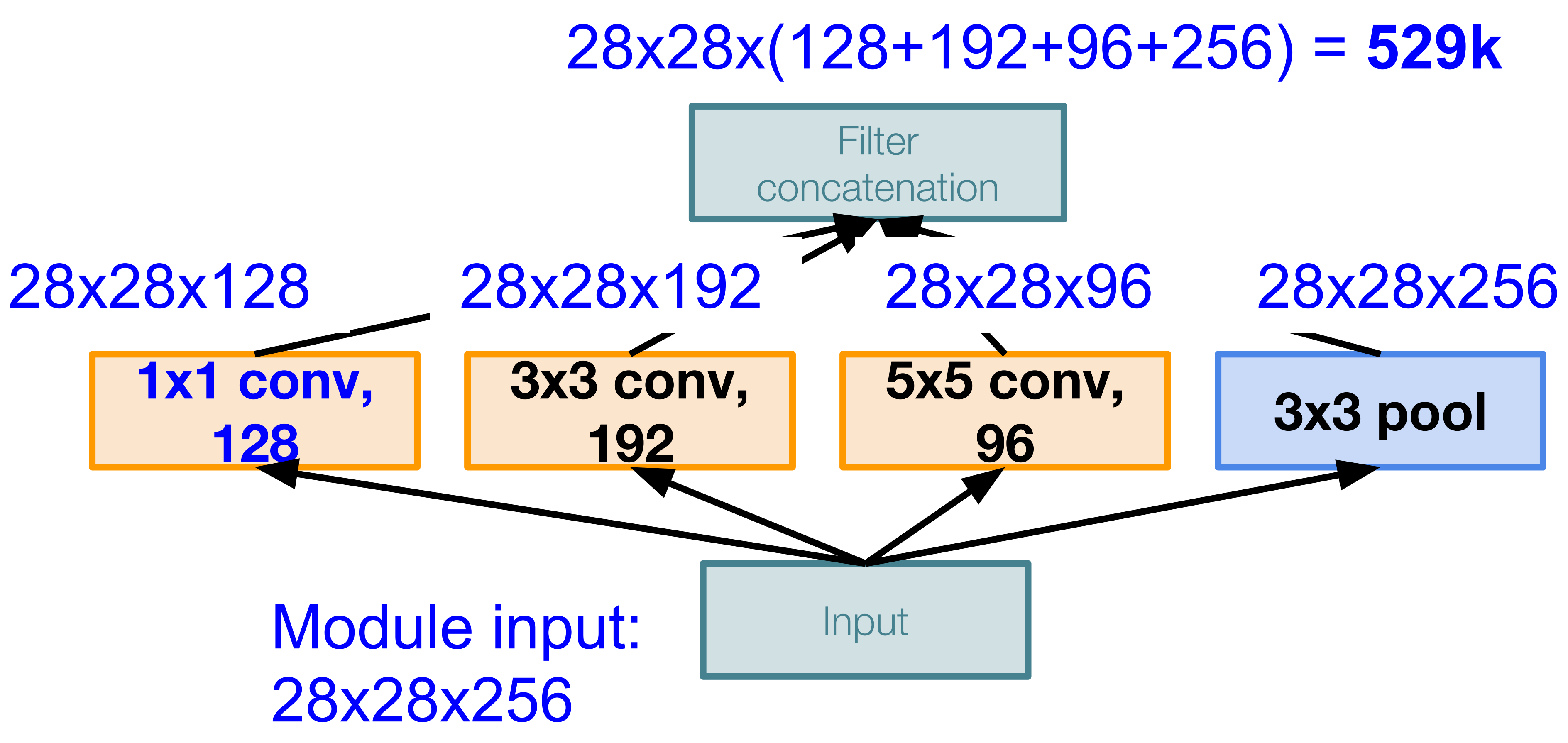
slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Case Study: GoogLeNet

[Szegedy et al., 2014]

Example:

Q3:What is output size after filter concatenation?



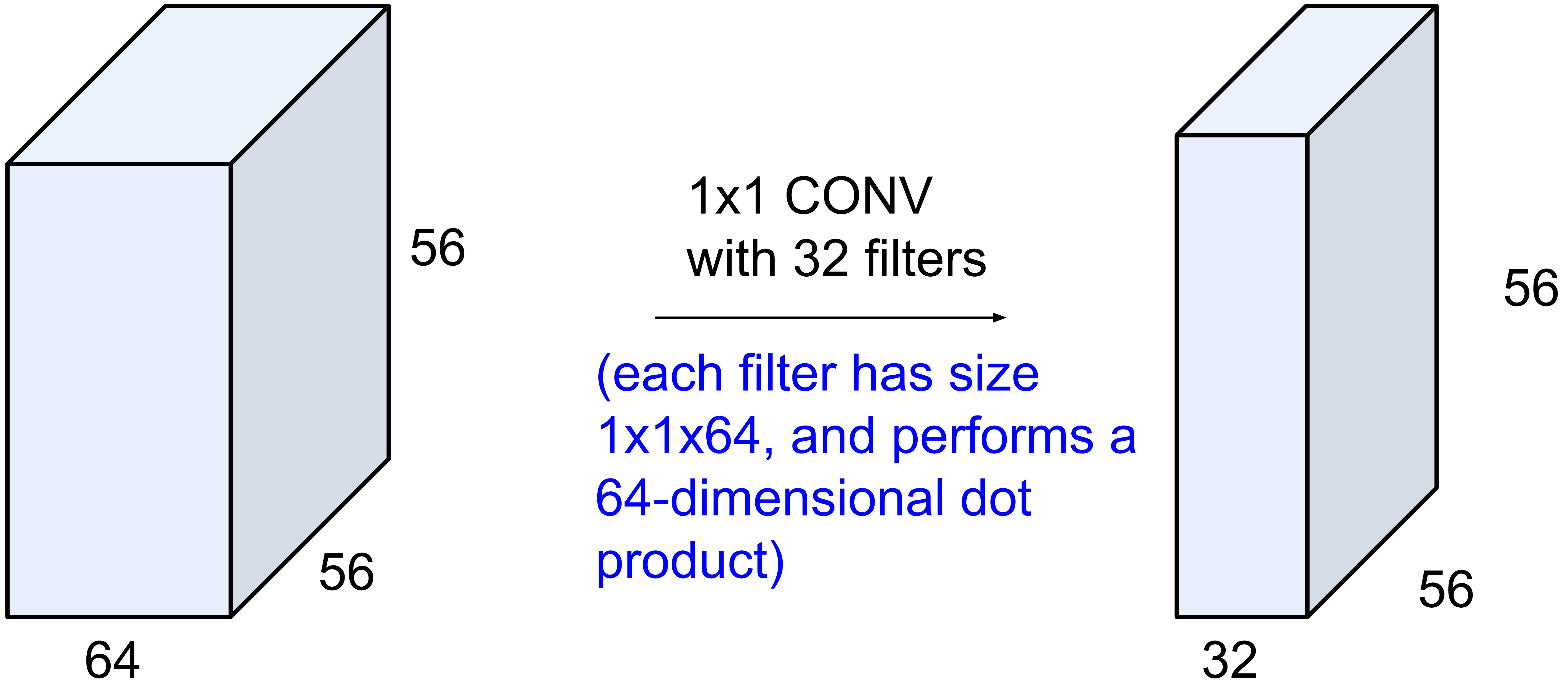
Naive Inception module

Q: What is the problem with this?
[Hint: Computational complexity]

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

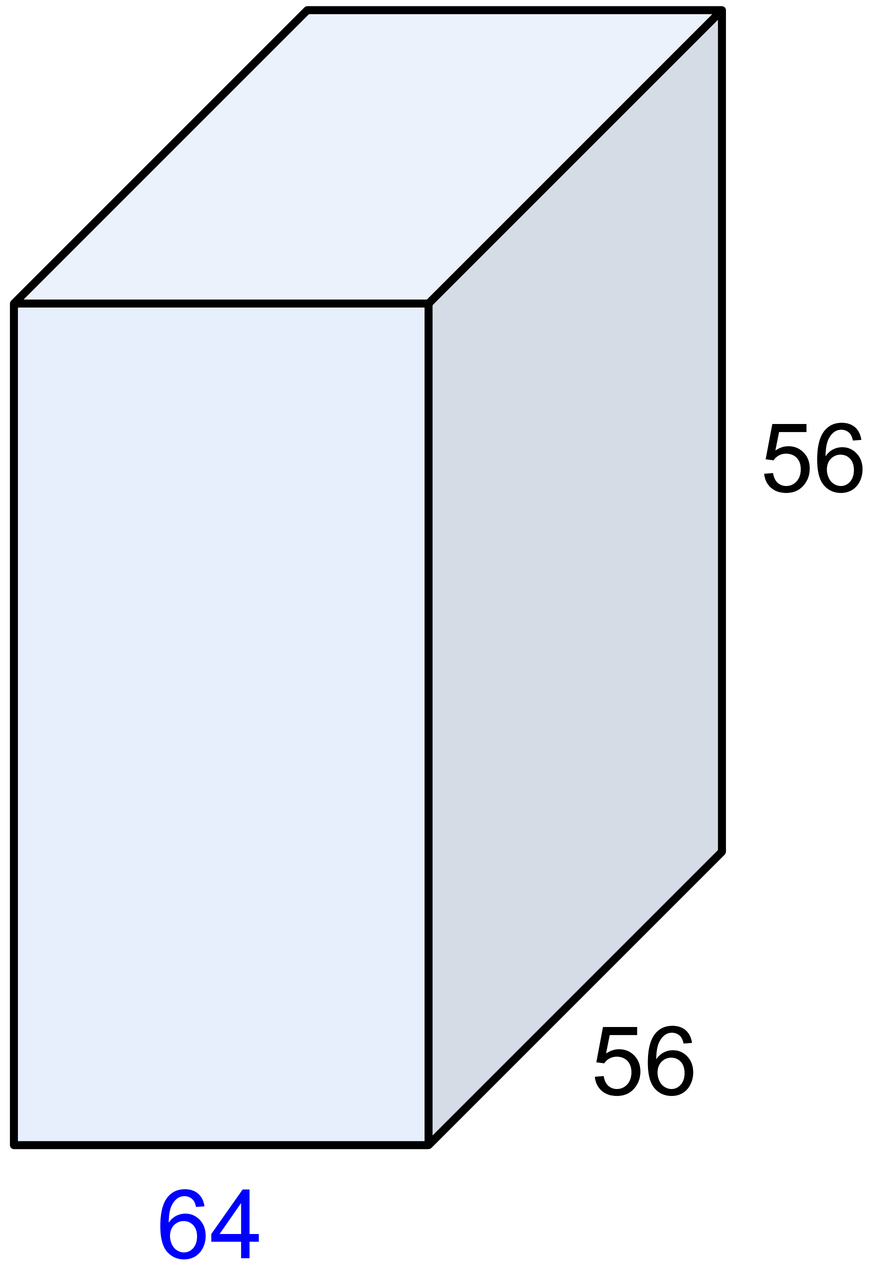
slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Reminder: 1x1 Convolutions



slide credit: Fei-Fei, Justin Johnson, Serena Yeung

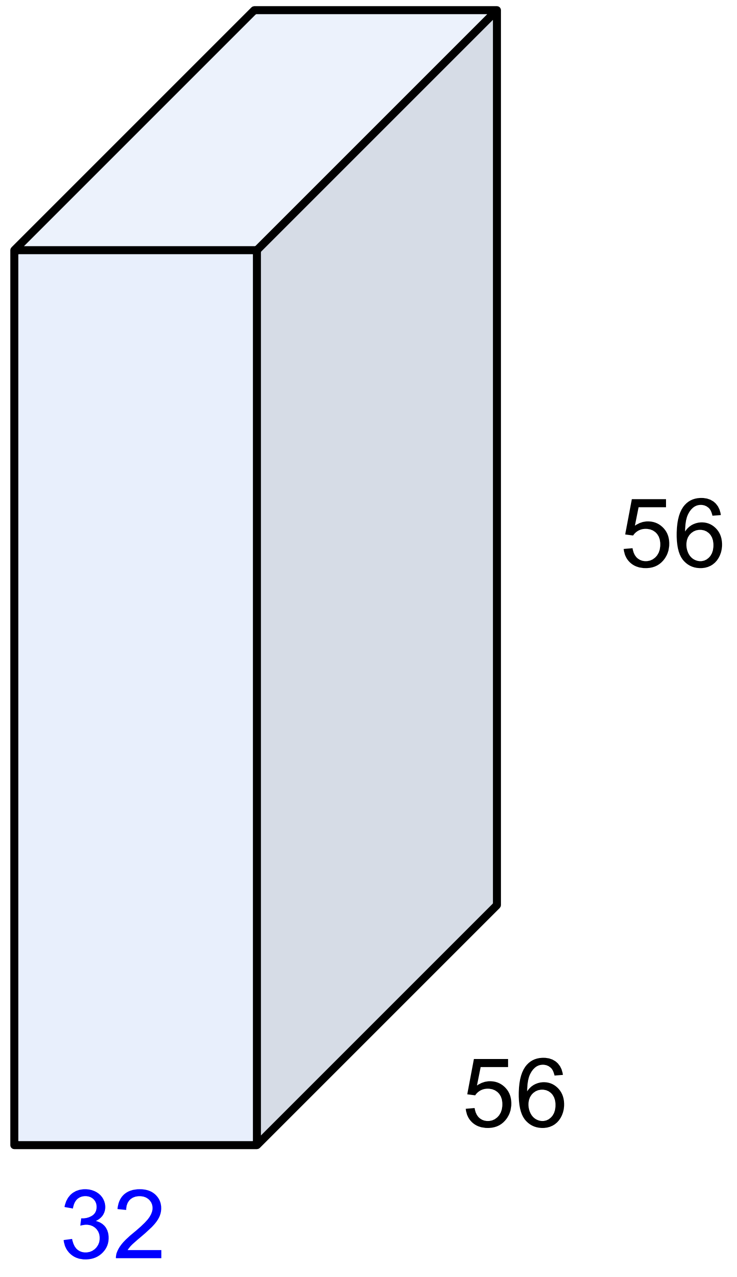
Reminder: 1x1 Convolutions



1x1 CONV
with 32 filters

preserves spatial dimensions, reduces depth!

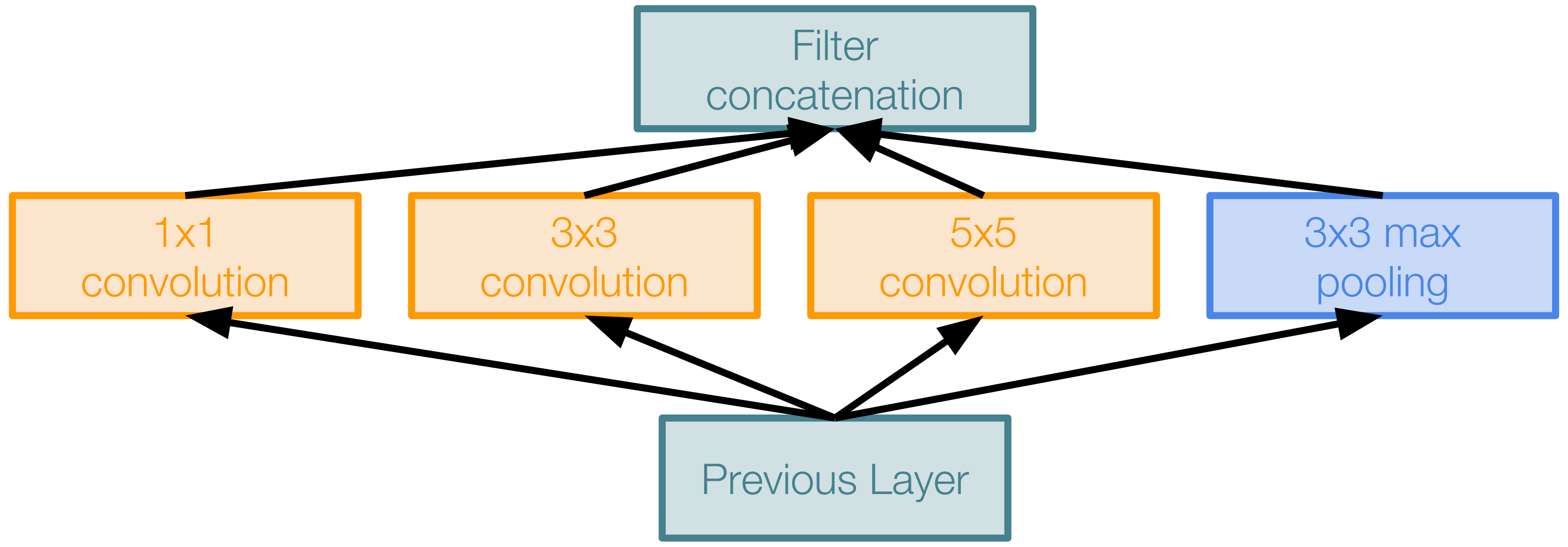
Projects depth to lower dimension (combination of feature maps)



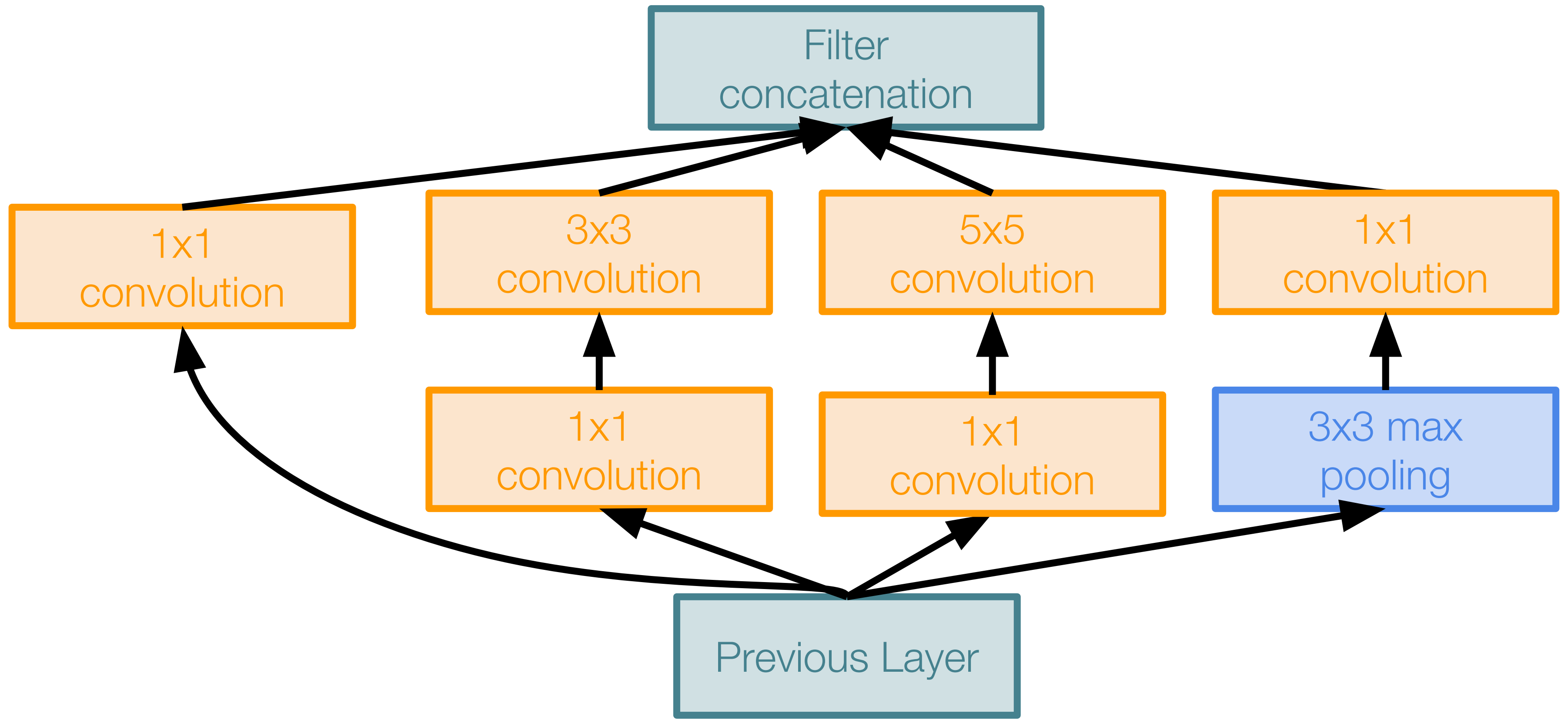
slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Case Study: GoogLeNet

[Szegedy et al., 2014]



Naive Inception module

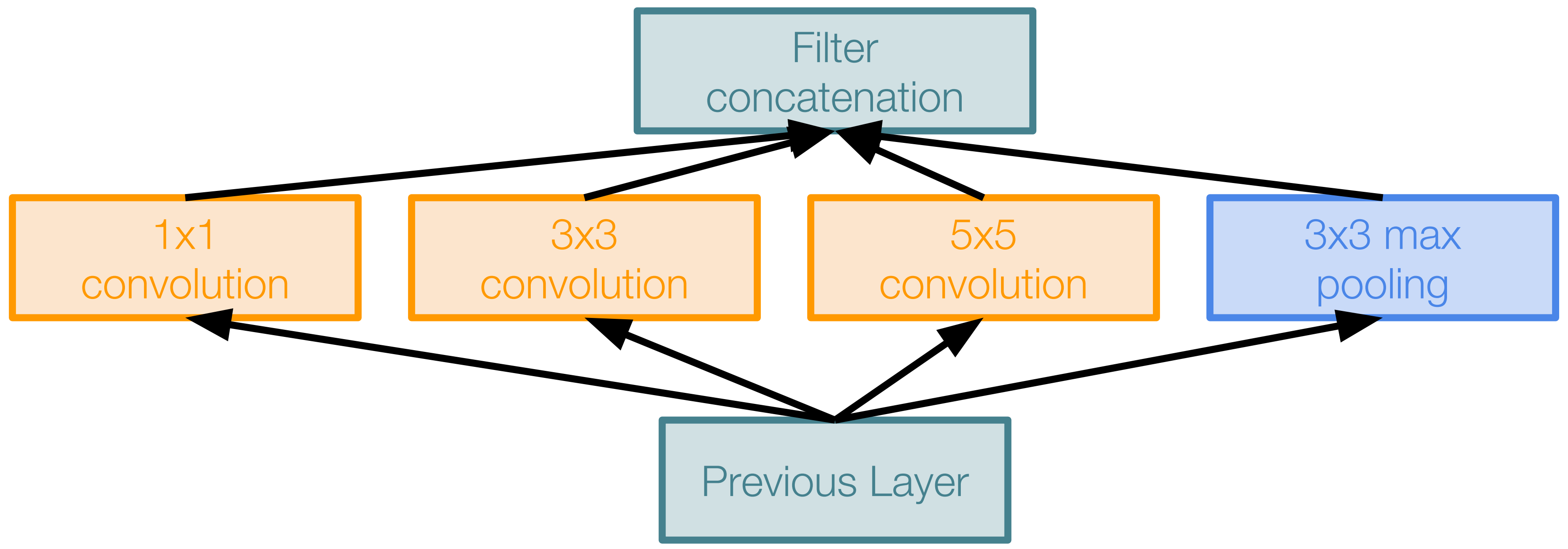


Inception module with dimension reduction

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

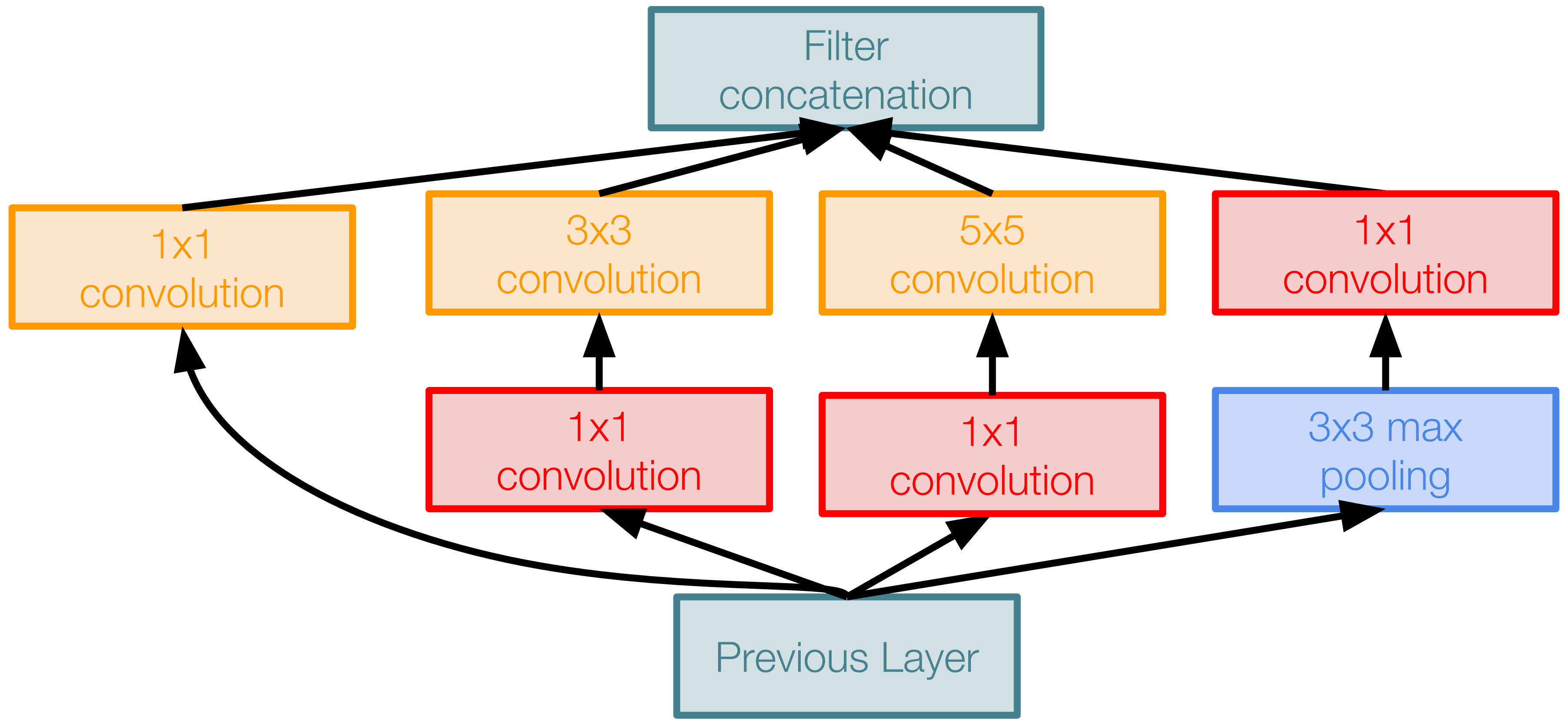
Case Study: GoogLeNet

[Szegedy et al., 2014]



Naive Inception module

1x1 conv "bottleneck" layers

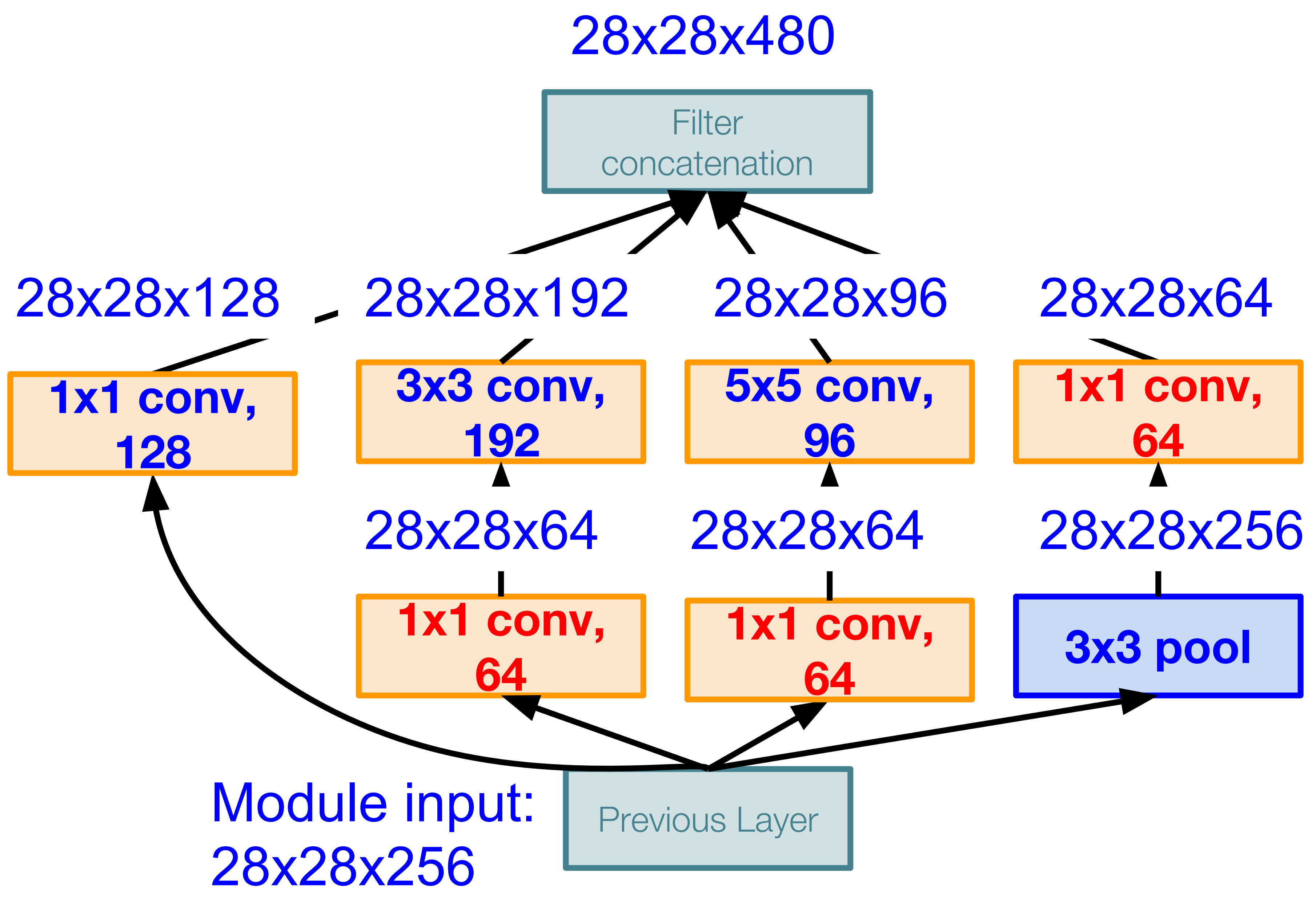


Inception module with dimension reduction

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Case Study: GoogLeNet

[Szegedy et al., 2014]



Inception module with dimension reduction

Using same parallel layers as naive example, and adding "1x1 conv, 64 filter" bottlenecks:

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

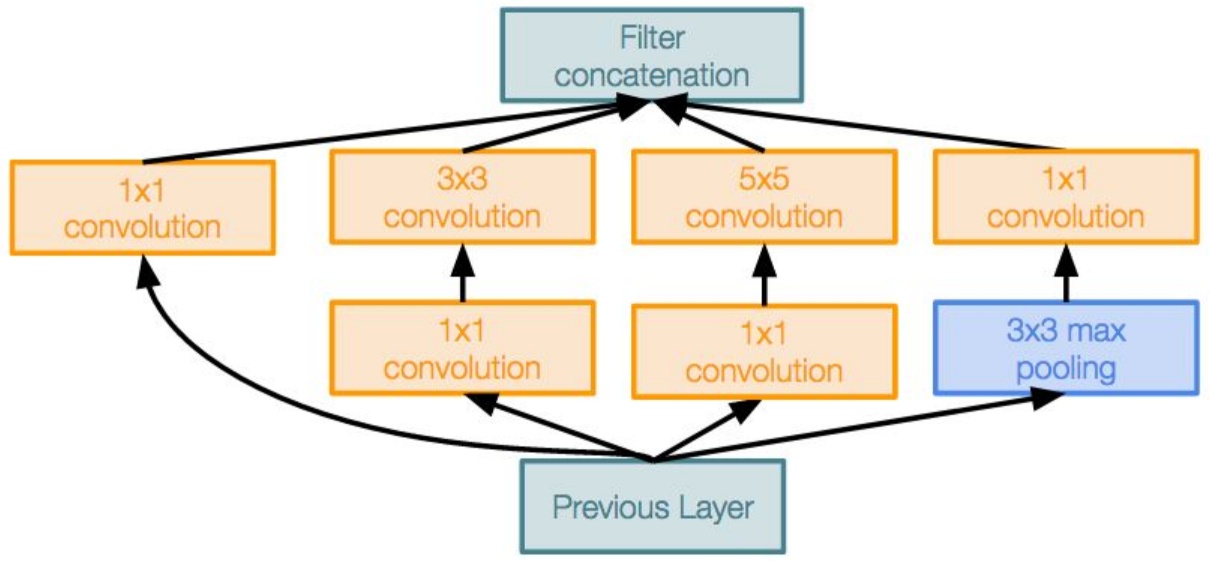
Compared to 854M ops for naive version
Bottleneck can also reduce depth after pooling layer

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

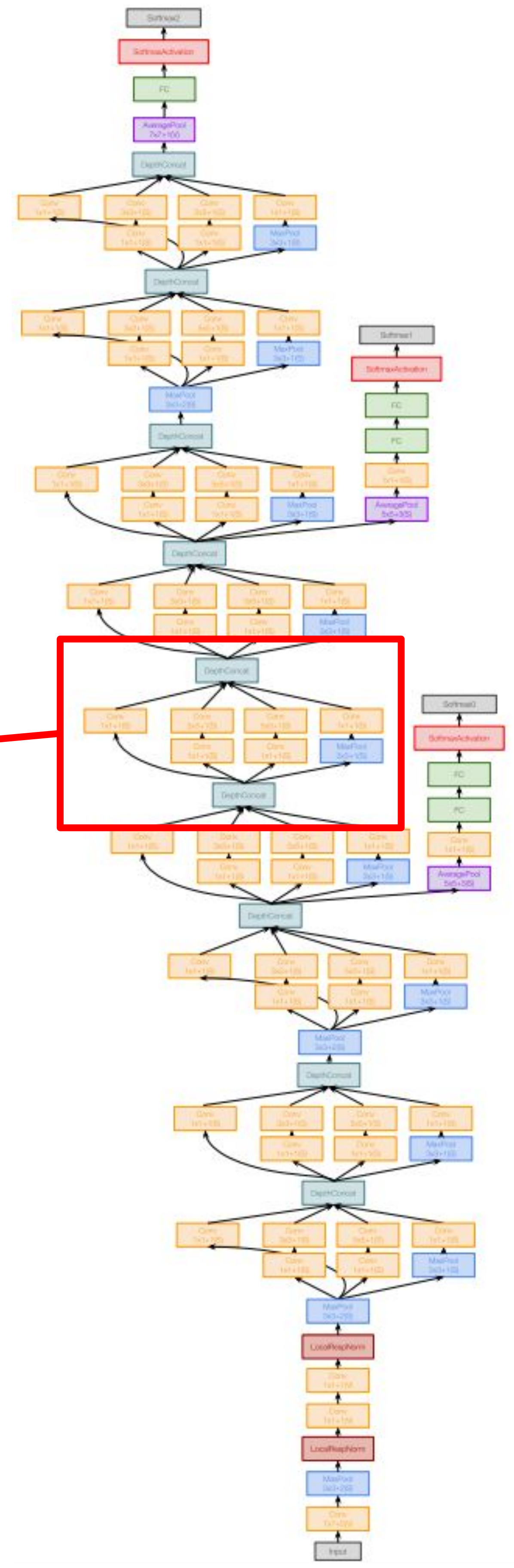
Case Study: GoogLeNet

[Szegedy et al., 2014]

Stack Inception modules
with dimension reduction
on top of each other



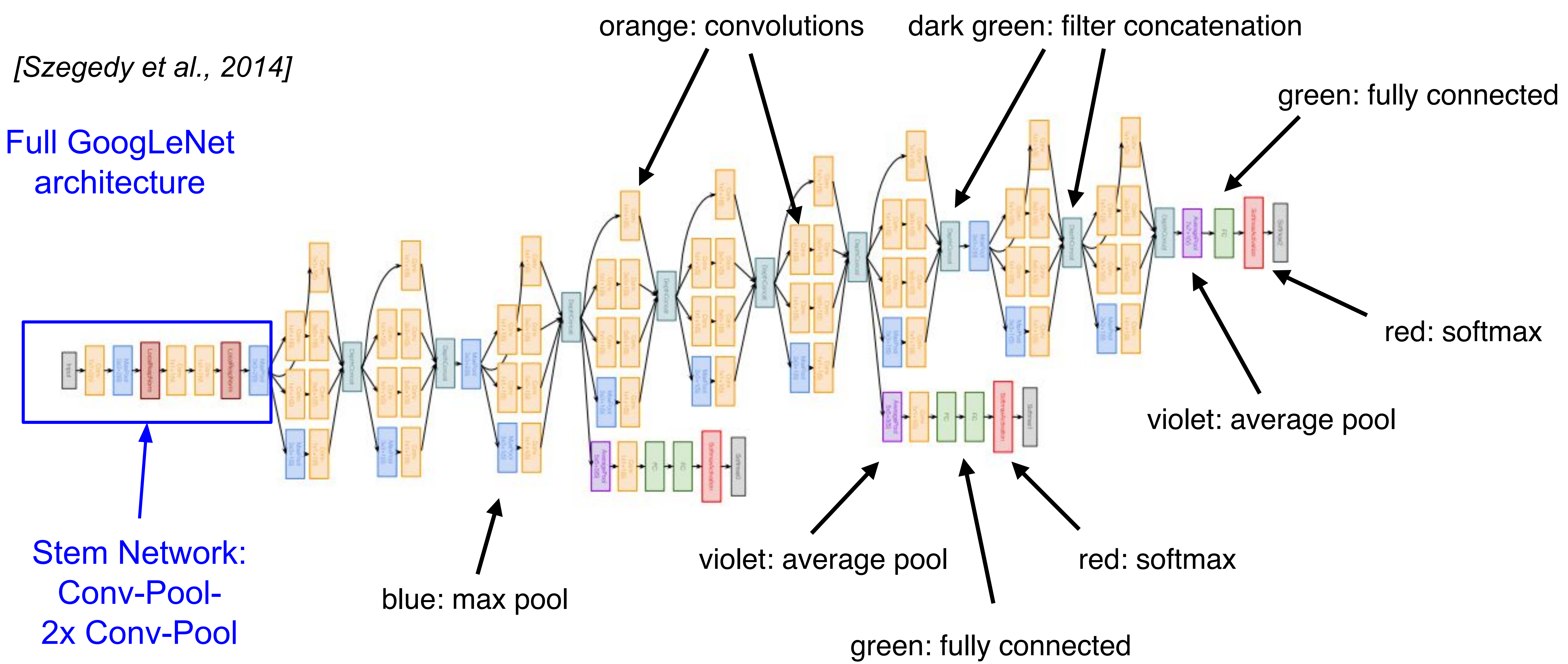
Inception module



slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Case Study: GoogLeNet

[Szegedy et al., 2014]

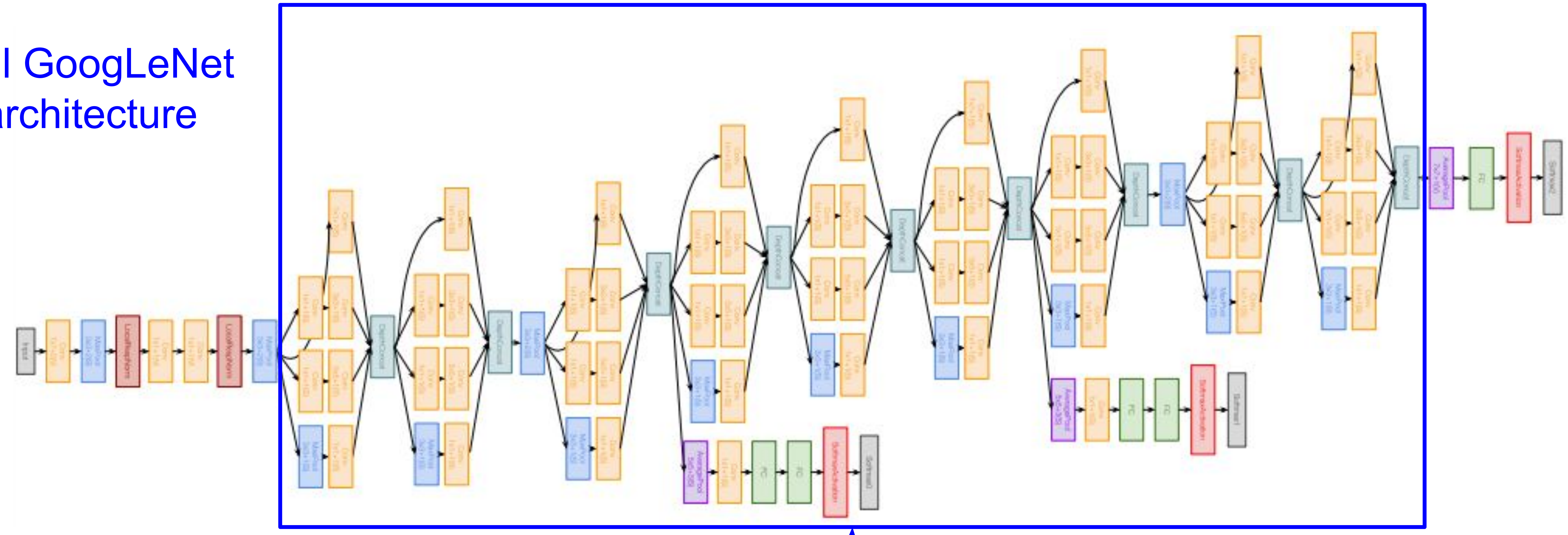


slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Case Study: GoogLeNet

[Szegedy et al., 2014]

Full GoogLeNet architecture



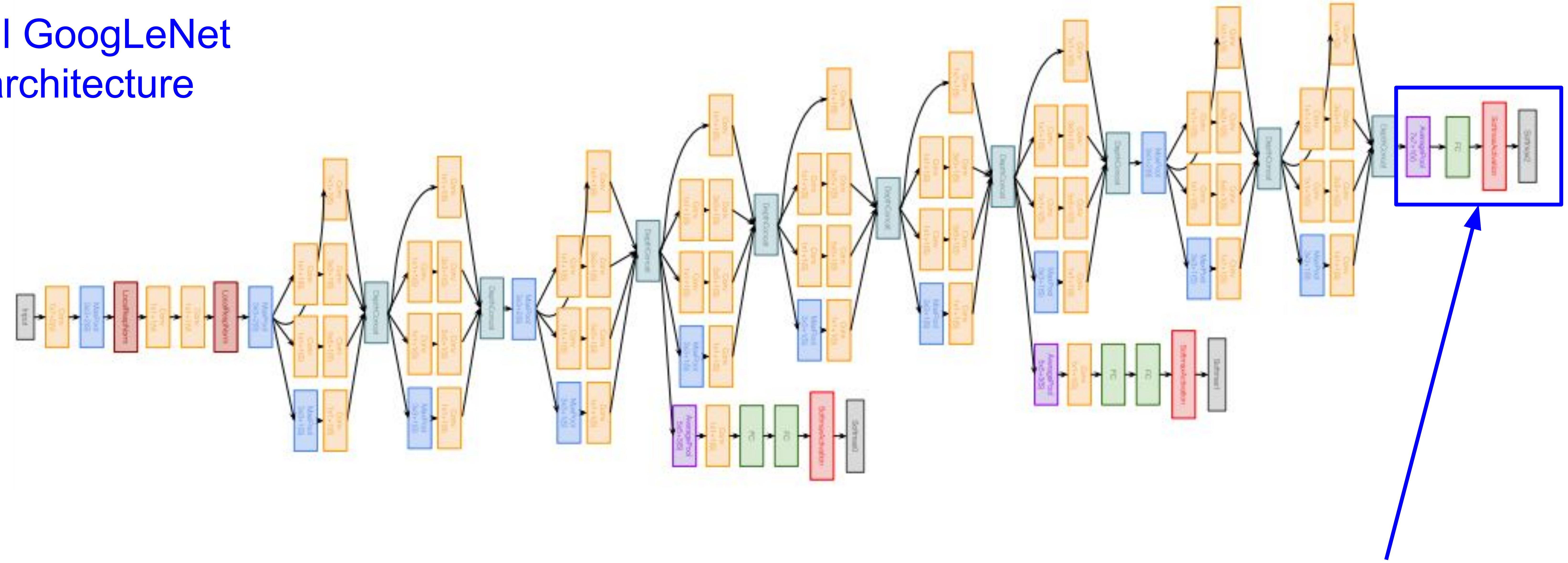
Stacked Inception Modules

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Case Study: GoogLeNet

[Szegedy et al., 2014]

Full GoogLeNet
architecture



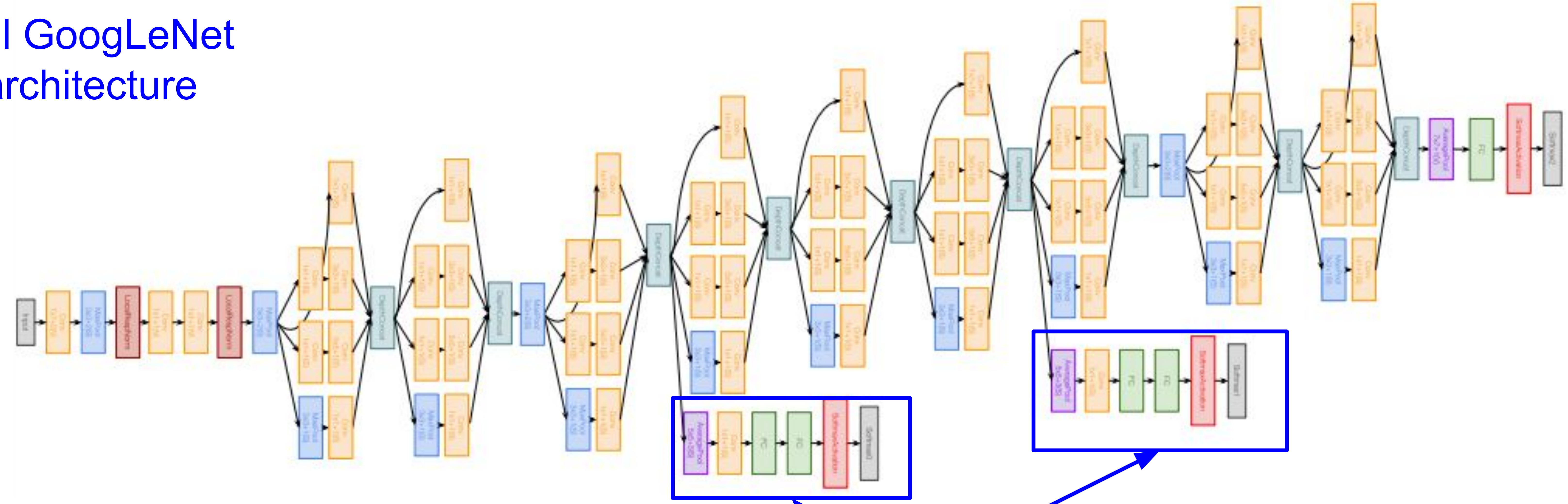
Classifier output
(removed expensive FC layers!)

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Case Study: GoogLeNet

[Szegedy et al., 2014]

Full GoogLeNet architecture



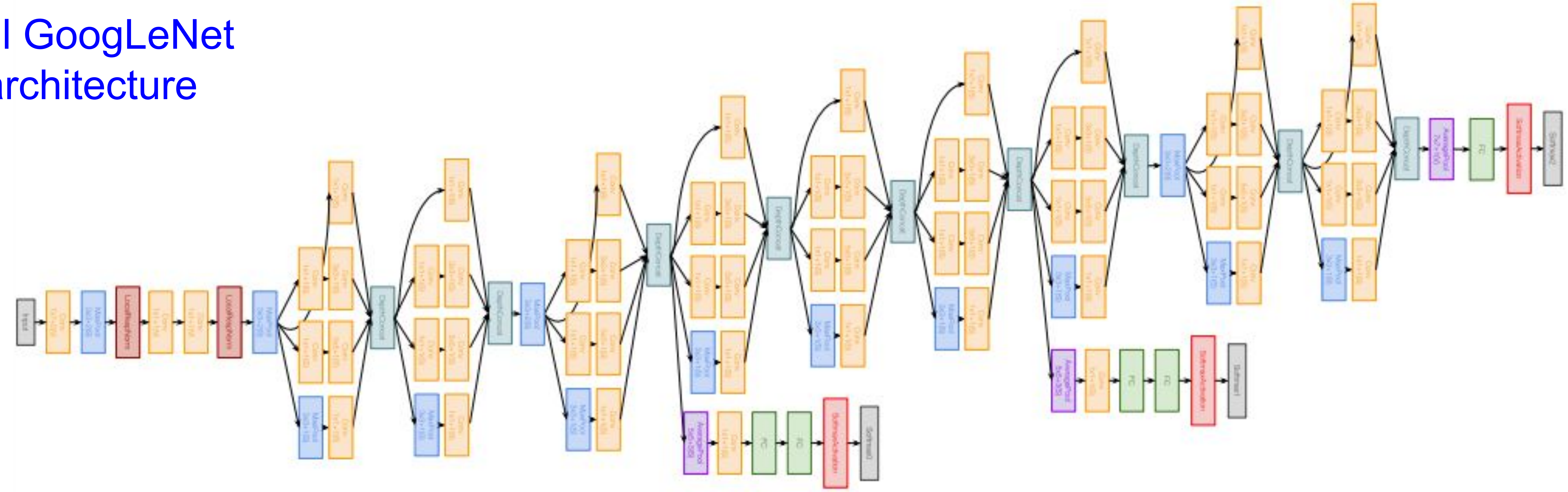
Auxiliary classification outputs to inject additional gradient at lower layers
(AvgPool-1x1Conv-FC-FC-Softmax)

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Case Study: GoogLeNet

[Szegedy et al., 2014]

Full GoogLeNet architecture



22 total layers with weights
(parallel layers count as 1 layer => 2 layers per Inception module. Don’t count auxiliary output layers)

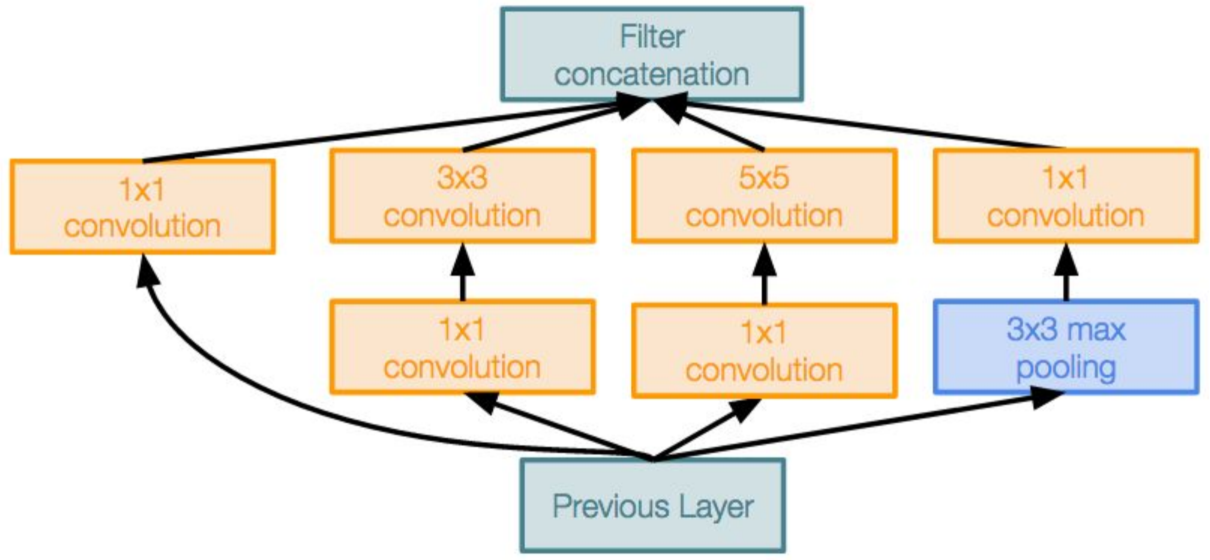
slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Case Study: GoogLeNet

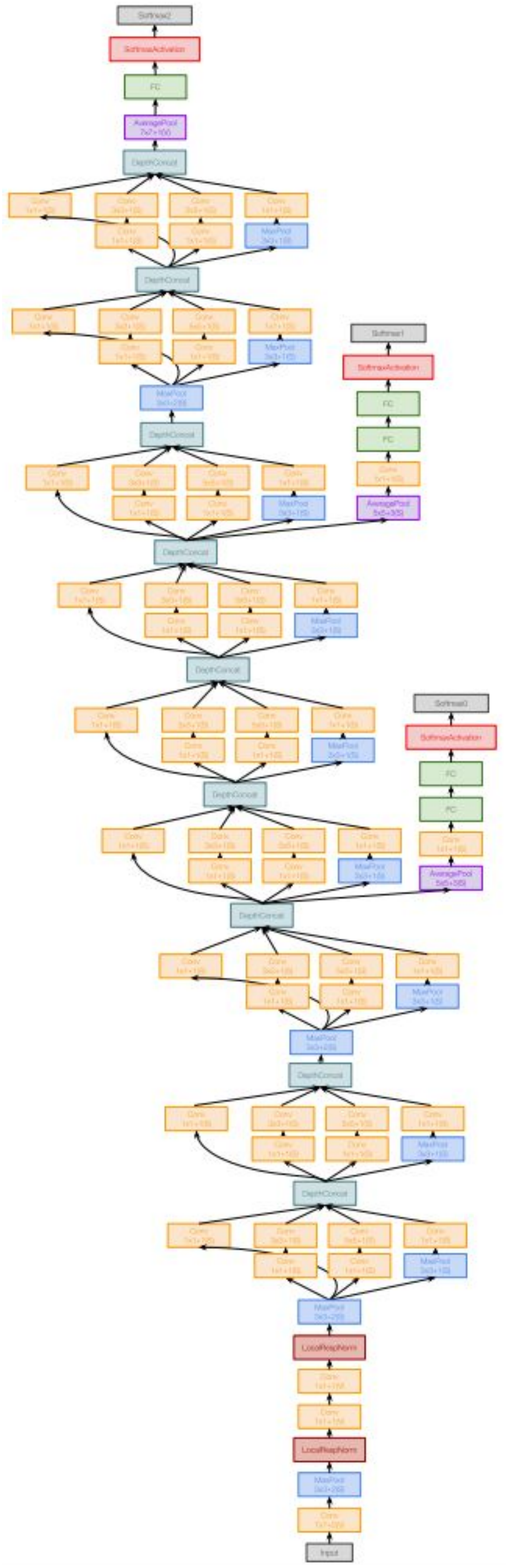
[Szegedy et al., 2014]

Deeper networks, with computational efficiency

- 22 layers
- Efficient “Inception” module
- No FC layers
- 12x less params than AlexNet
- ILSVRC’14 classification winner (6.7% top 5 error)



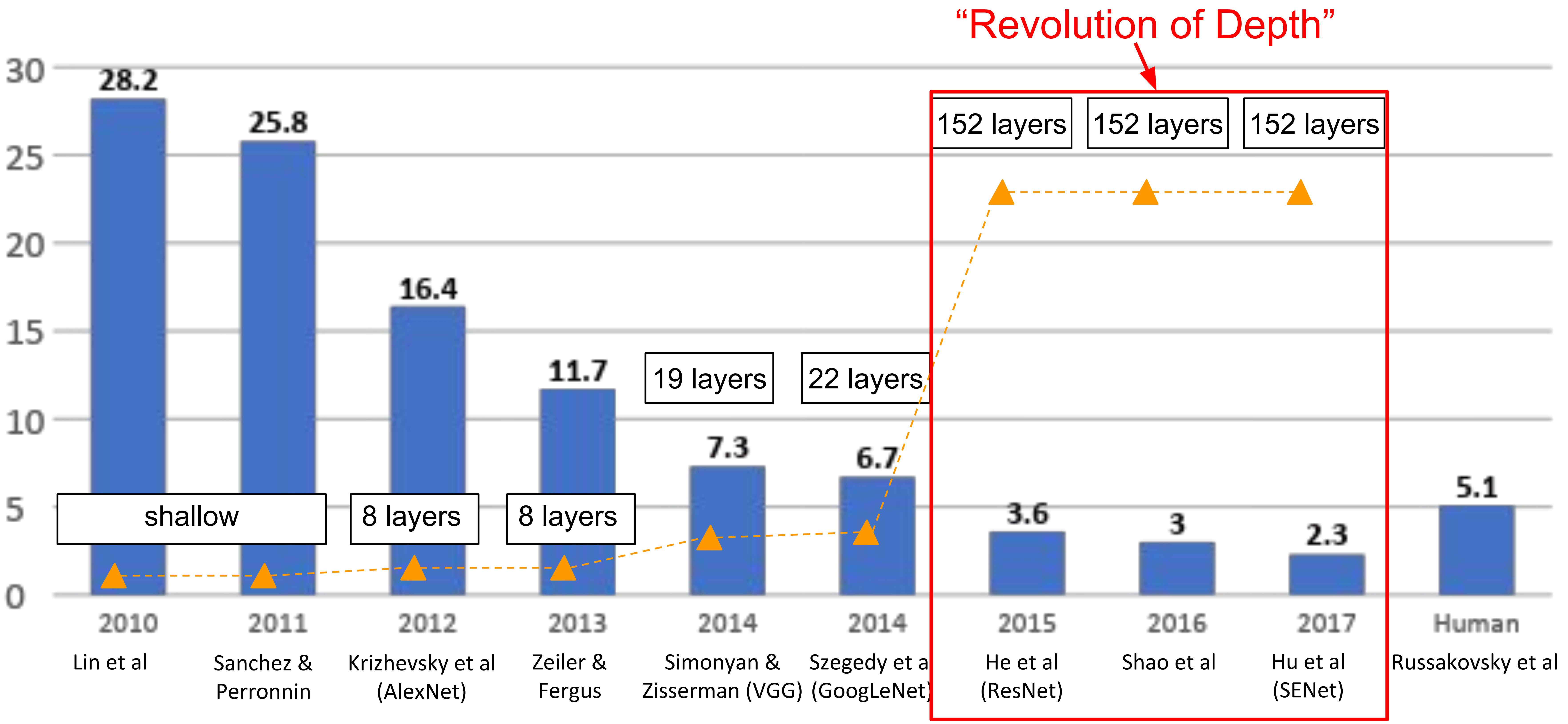
Inception module



slide credit: Fei-Fei, Justin Johnson, Serena Yeung

ImageNet Large Scale Visual Recognition Challenge (ILSVRC)

- Winners over the years (top-5 error):



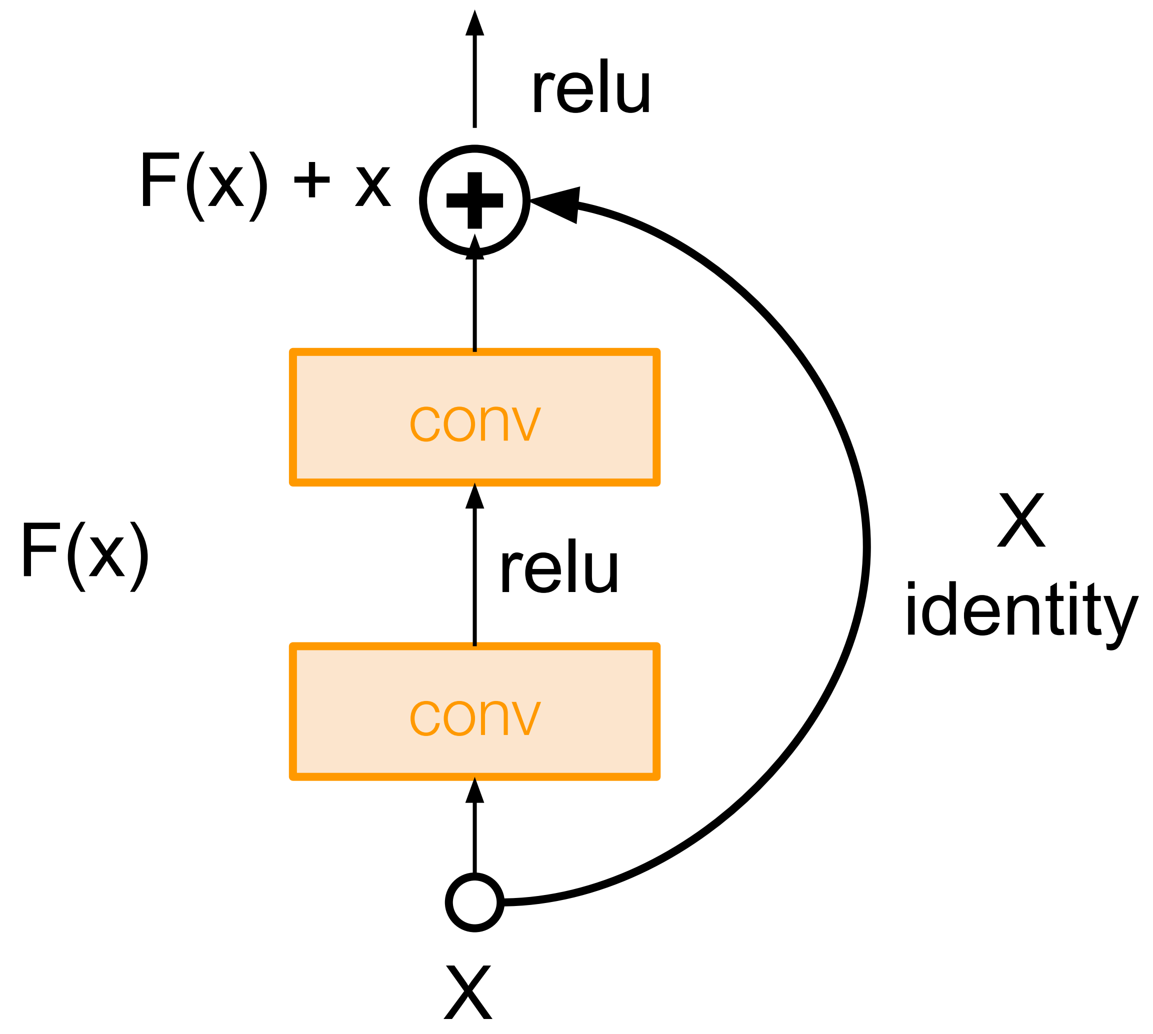
slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Case Study: ResNet

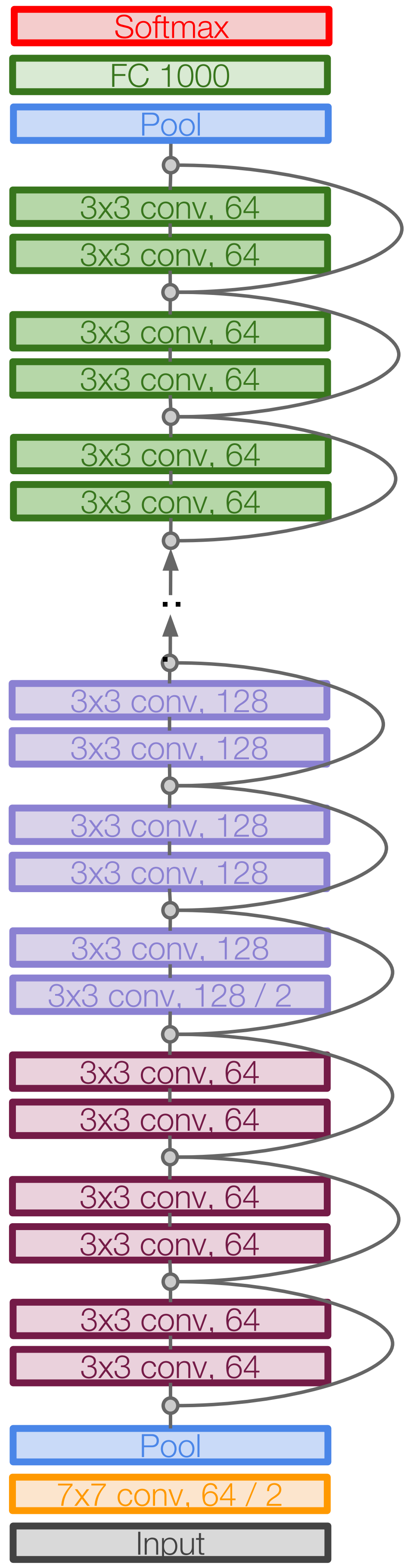
[He et al., 2015]

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!



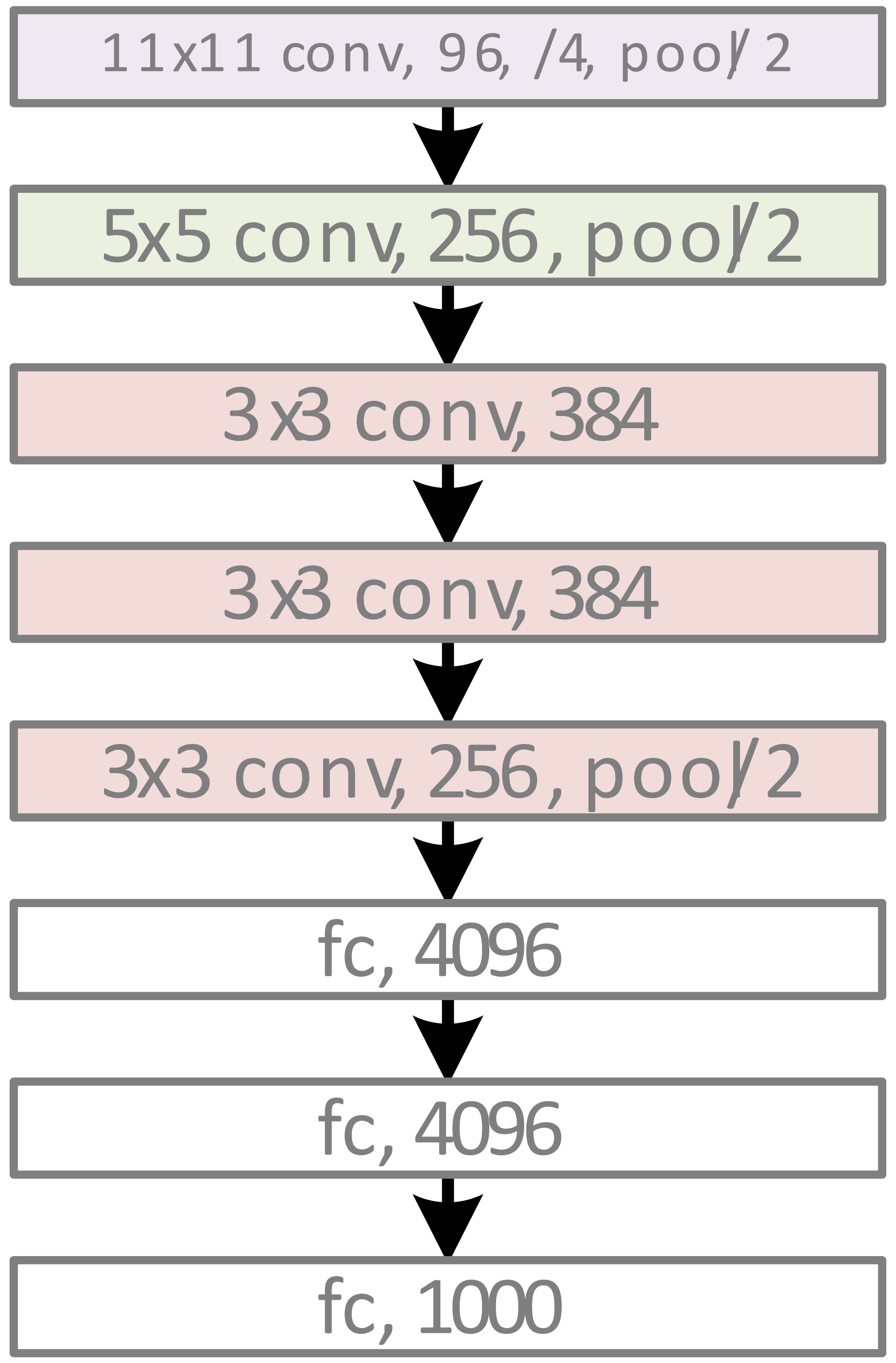
Residual block



slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Revolution of Depth

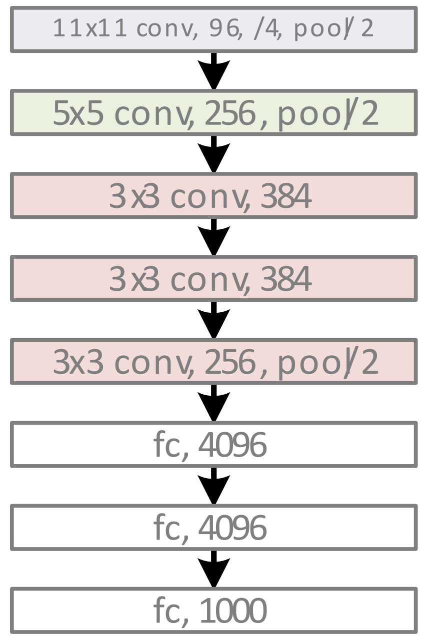
AlexNet, 8 layers (ILSVRC 2012)



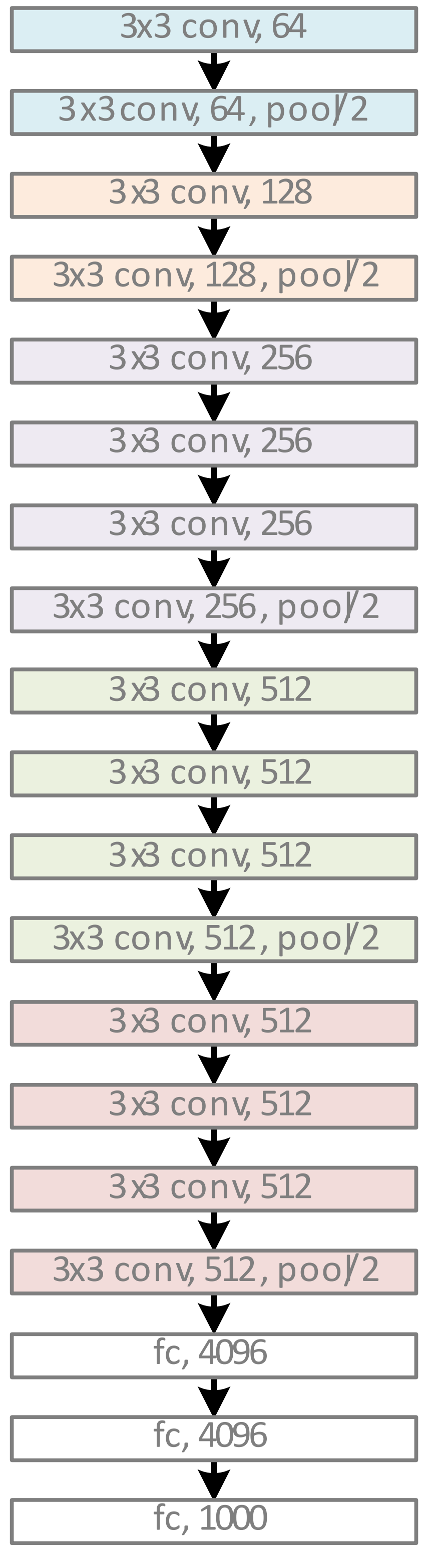
Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. “Deep Residual Learning for Image Recognition”. CVPR 2016.

Revolution of Depth

AlexNet, 8 layers (ILSVRC 2012)



VGG, 19 layers (ILSVRC 2014)

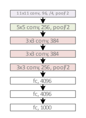


GoogleNet, 22 layers (ILSVRC 2014)

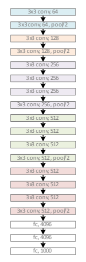
Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. “Deep Residual Learning for Image Recognition”. CVPR 2016.

Revolution of Depth

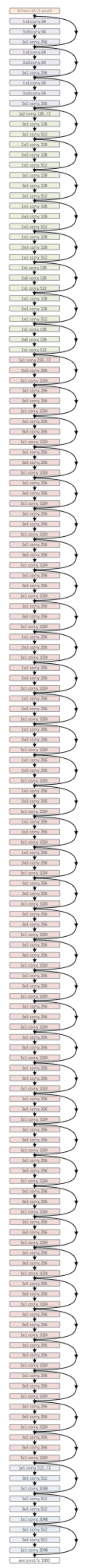
AlexNet, 8 layers (ILSVRC 2012)



VGG, 19 layers (ILSVRC 2014)



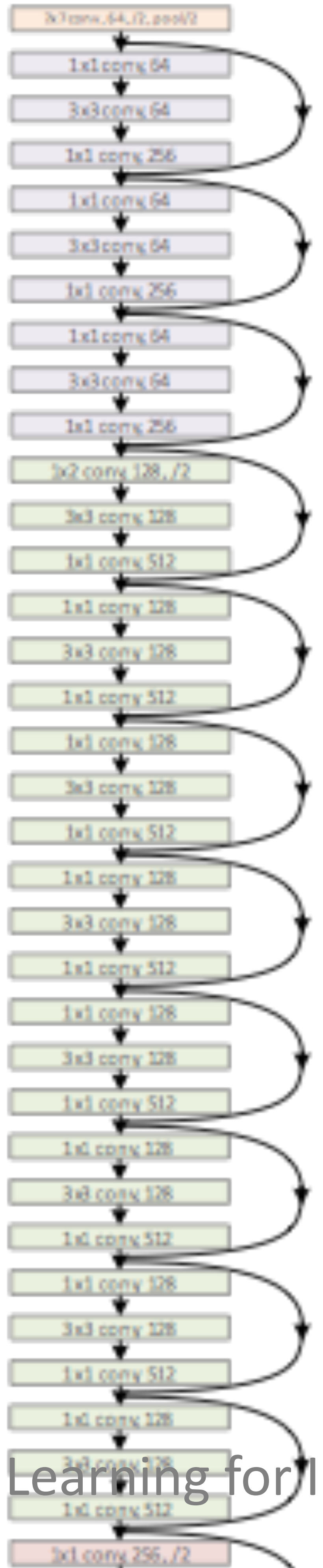
ResNet, 152 layers (ILSVRC 2015)



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

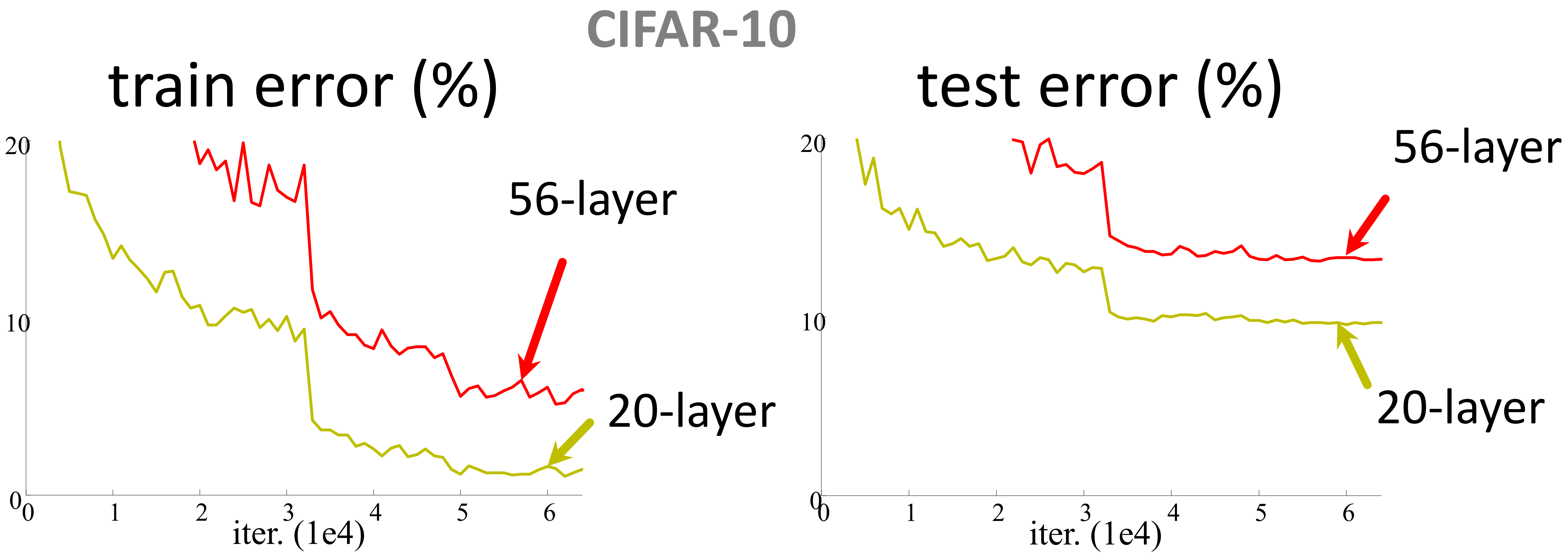
Revolution of Depth

ResNet, 152 layers



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

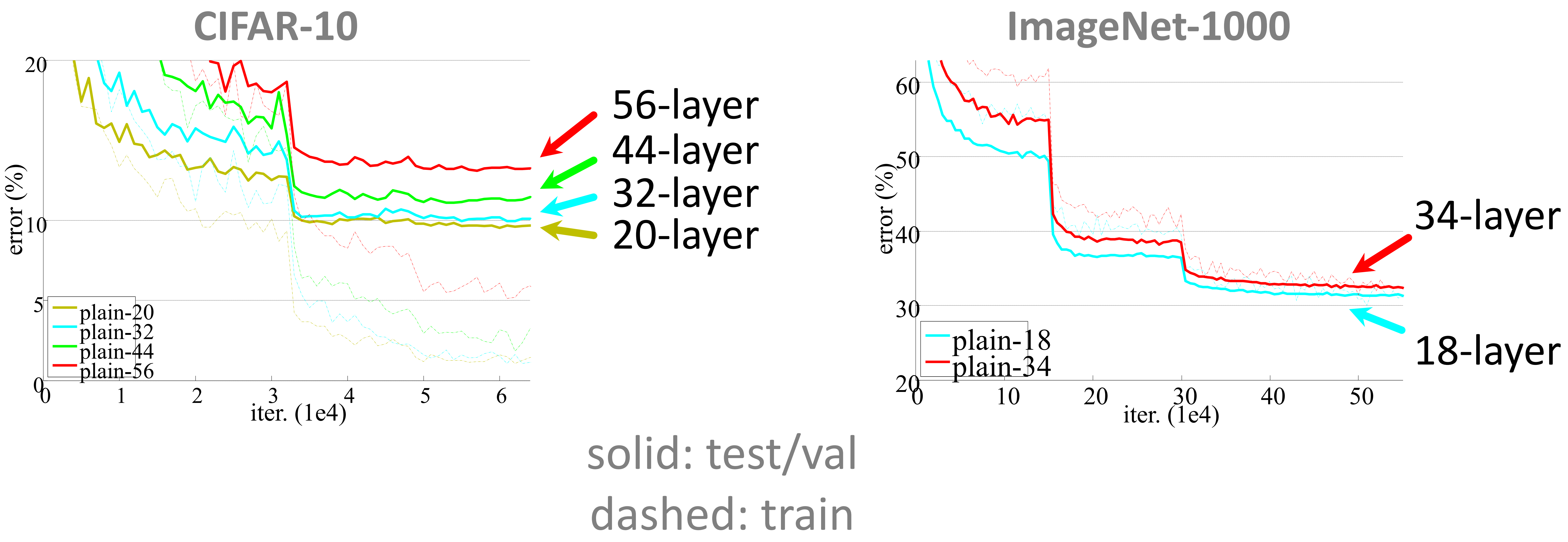
Simply stacking layers?



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

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

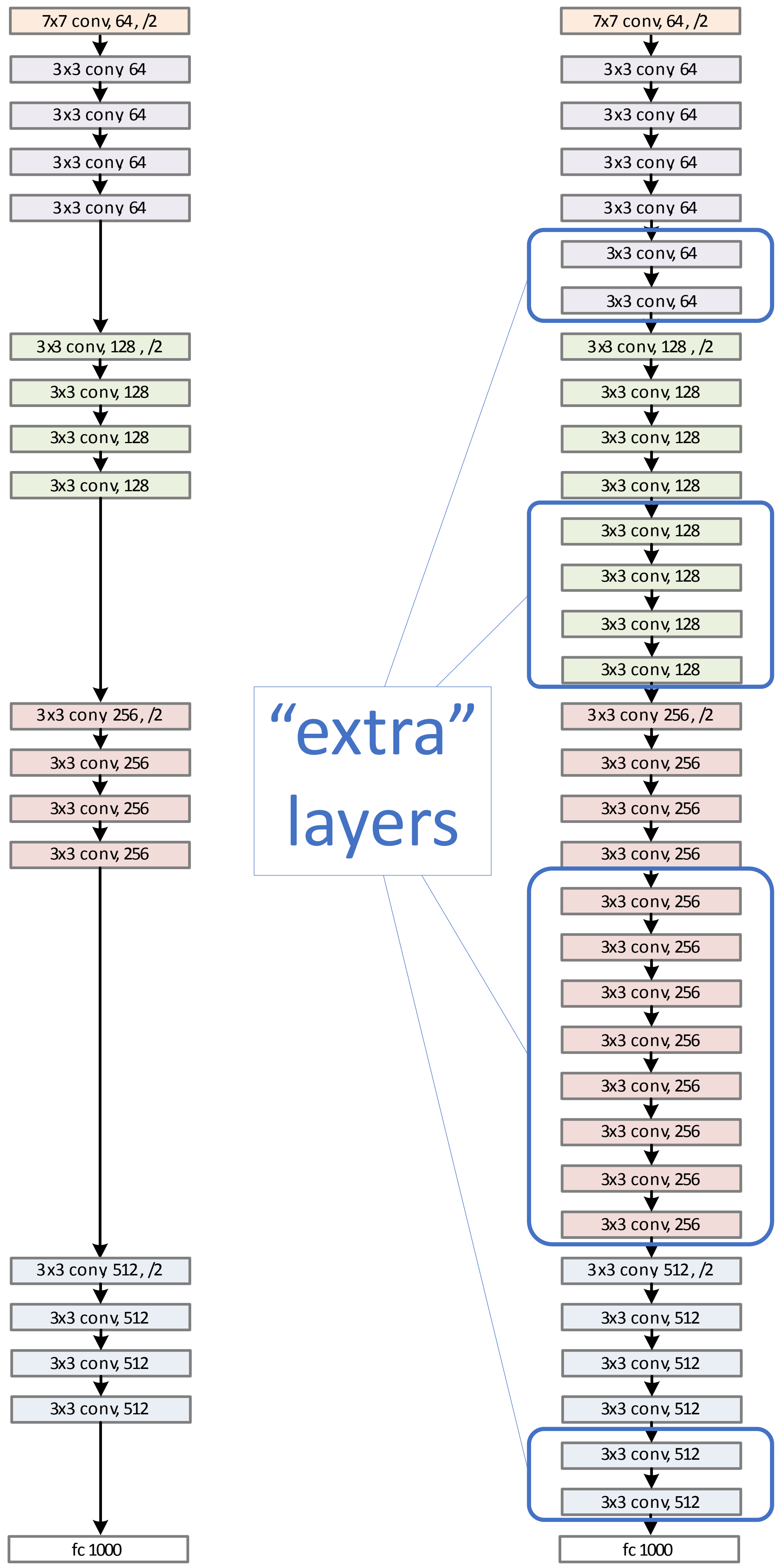
Simply stacking layers?



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

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

a shallower model (18 layers)



a deeper counterpart (34 layers)

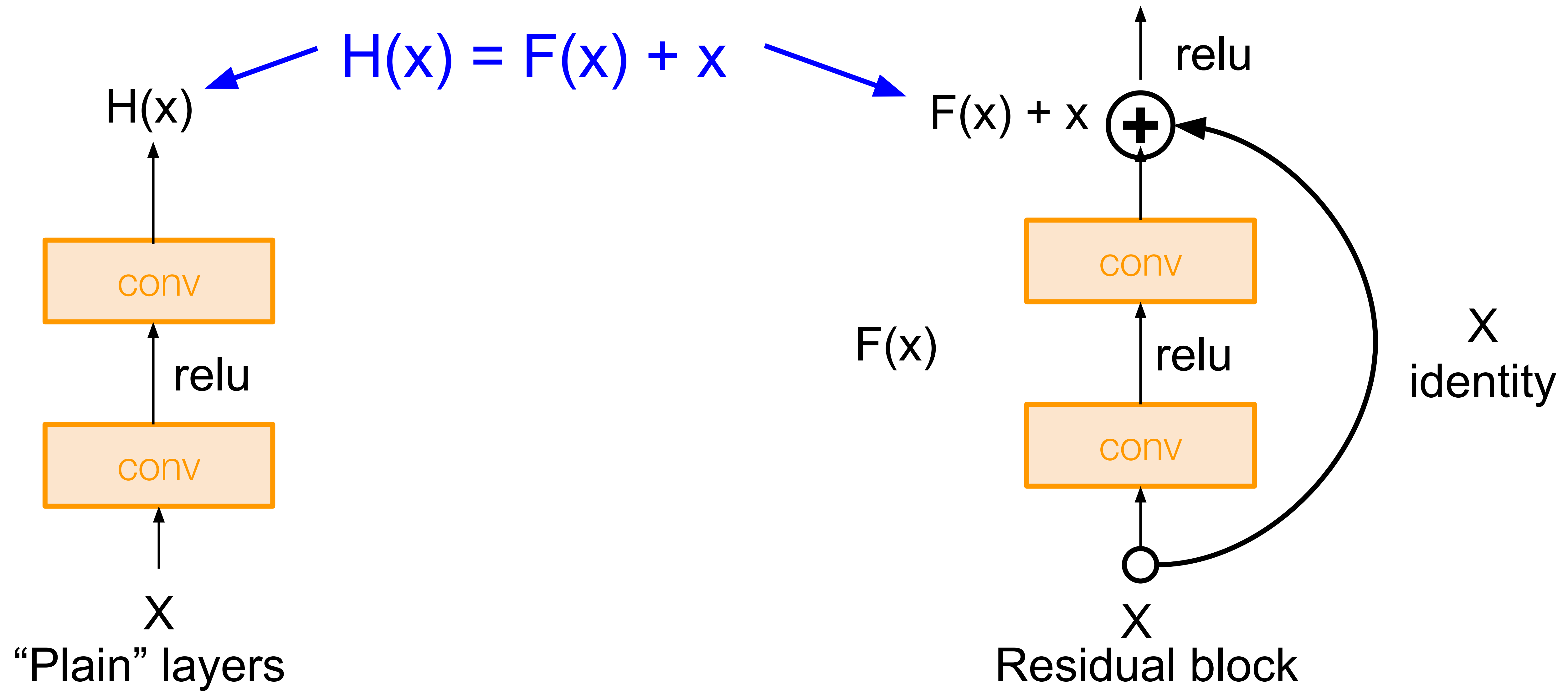
- Richer solution space
- A deeper model should not have **higher training error**
- A solution *by construction*:
  - original layers: copied from a learned shallower model
  - extra layers: set as identity
  - at least the same training error
- Optimization difficulties: solvers cannot find the solution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

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

Case Study: ResNet

[He et al., 2015]

Solution: Use network layers to fit a residual mapping instead of directly trying to fit a desired underlying mapping



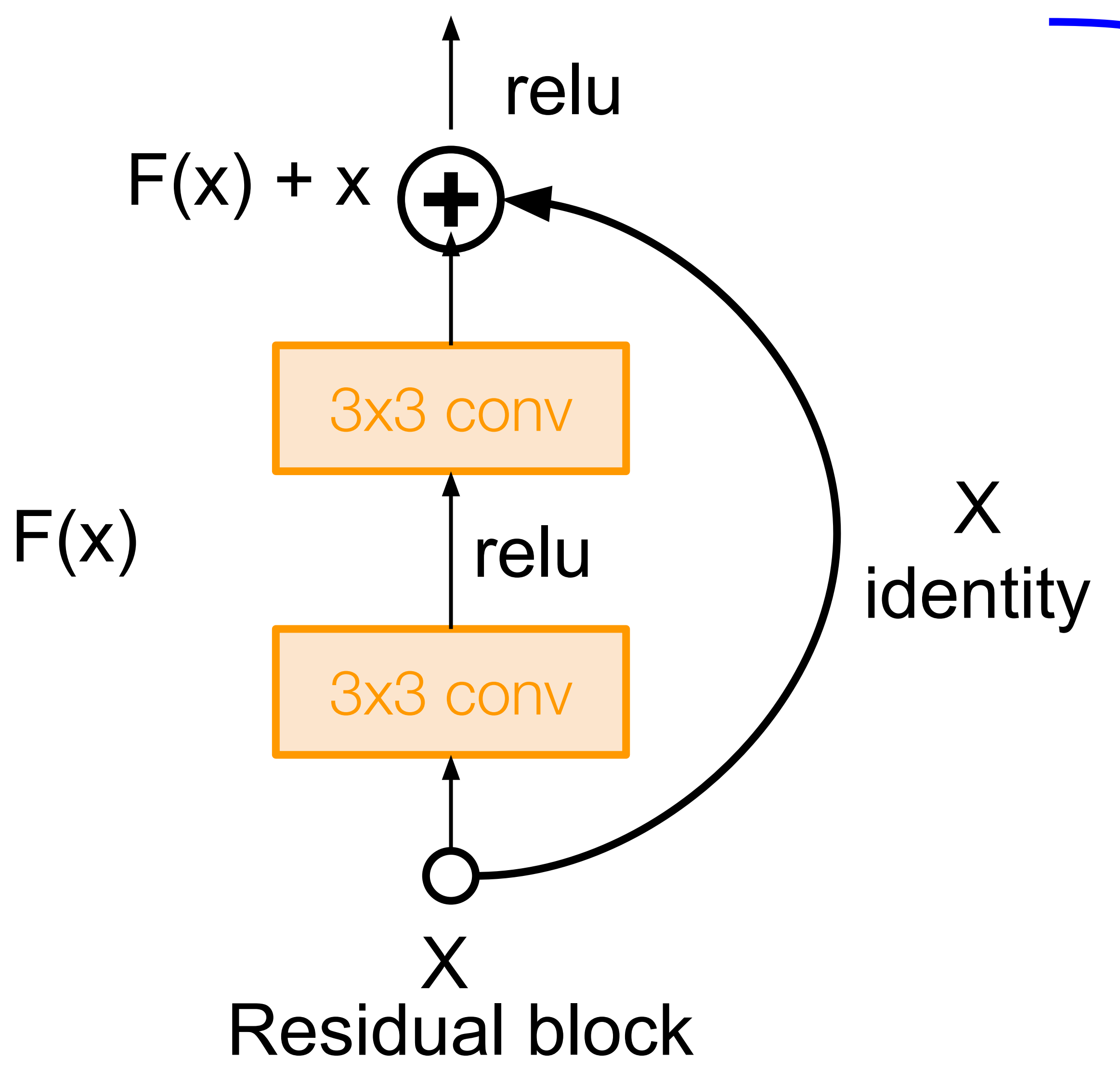
Use layers to fit residual F(x) = H(x) - x instead of H(x) directly

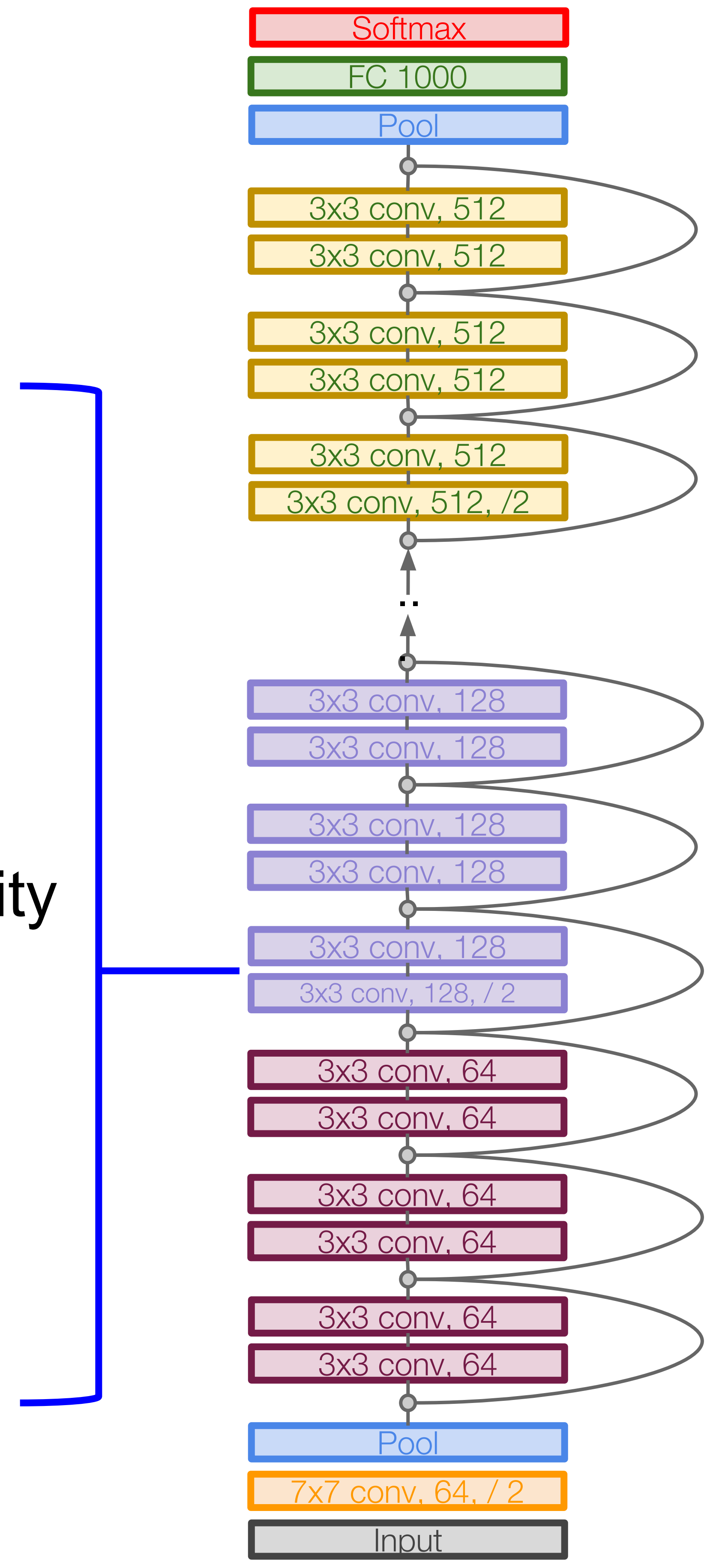
slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Case Study: ResNet

[He et al., 2015]

Full ResNet architecture:
- Stack residual blocks
- Every residual block has two 3x3 conv layers





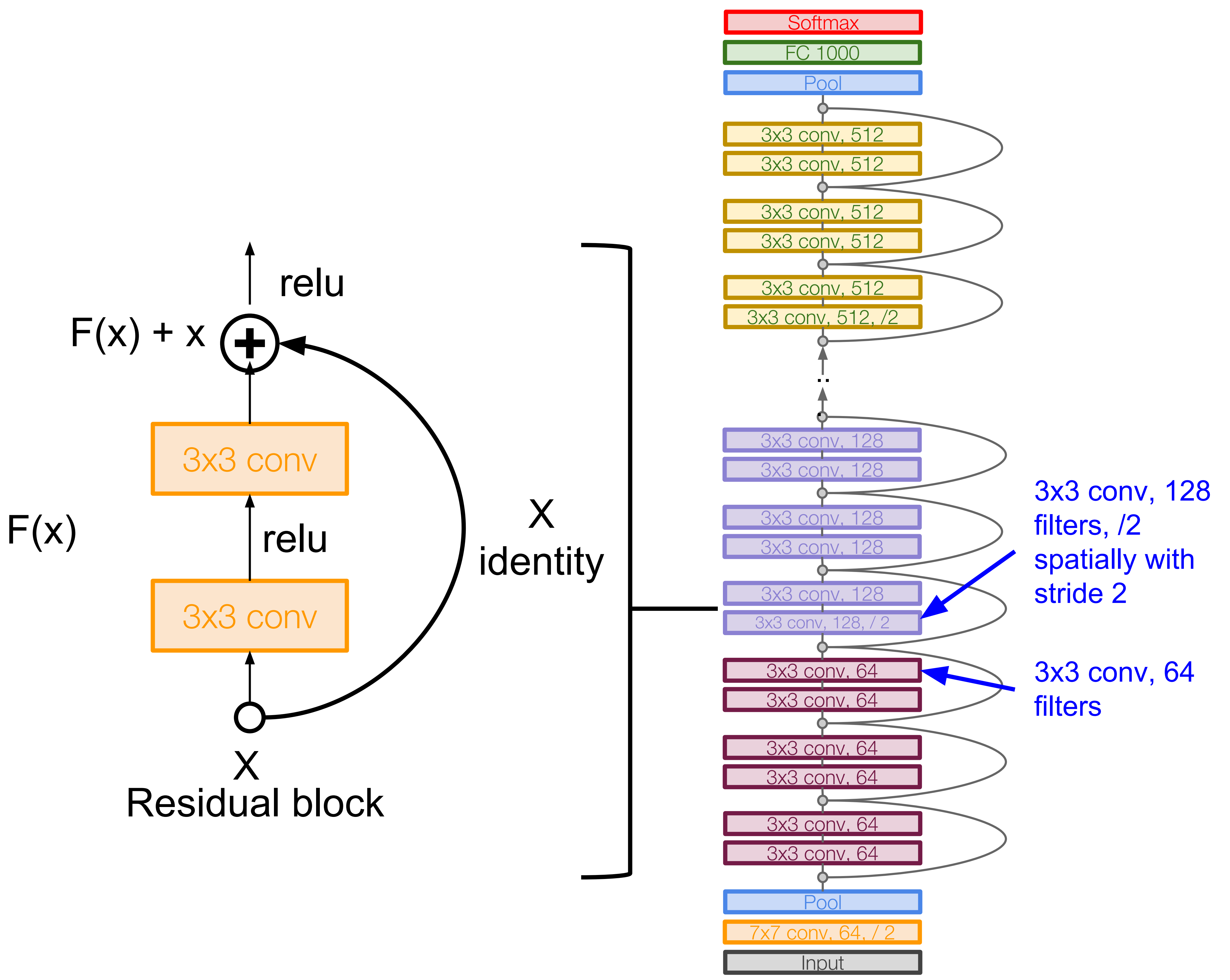
slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Case Study: ResNet

•

[He et al., 2015]

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)

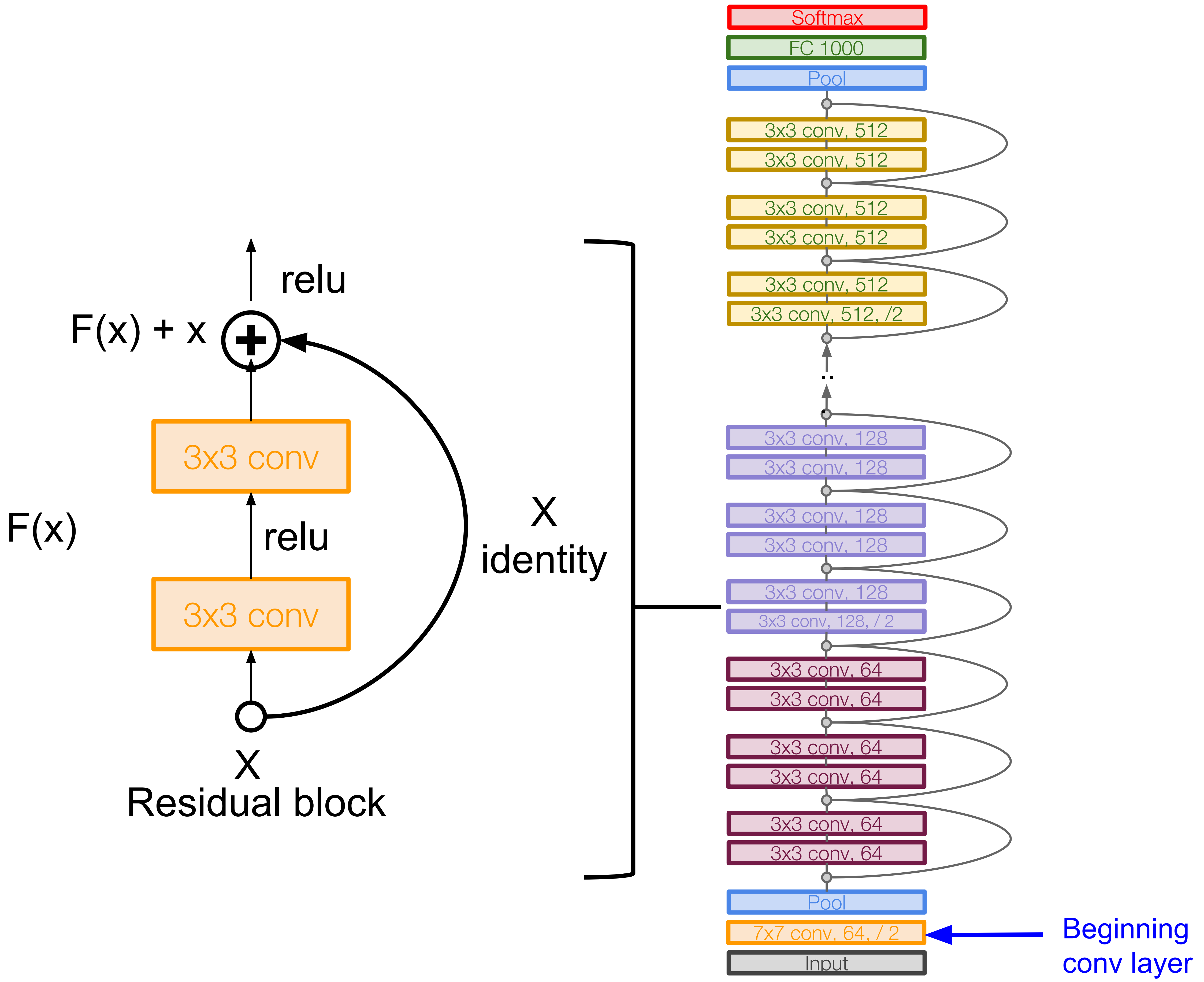


slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Case Study: ResNet

[He et al., 2015]

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

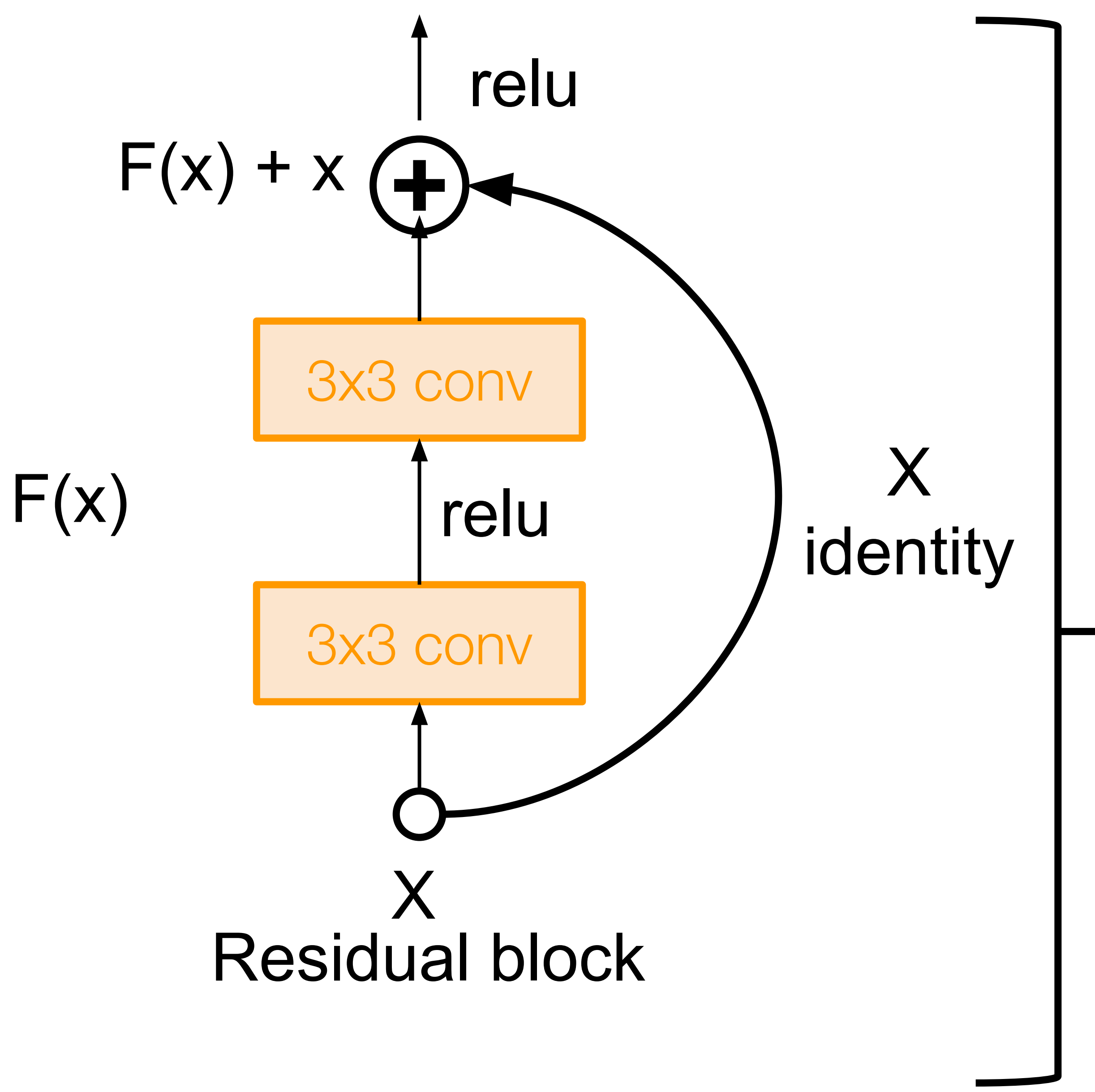


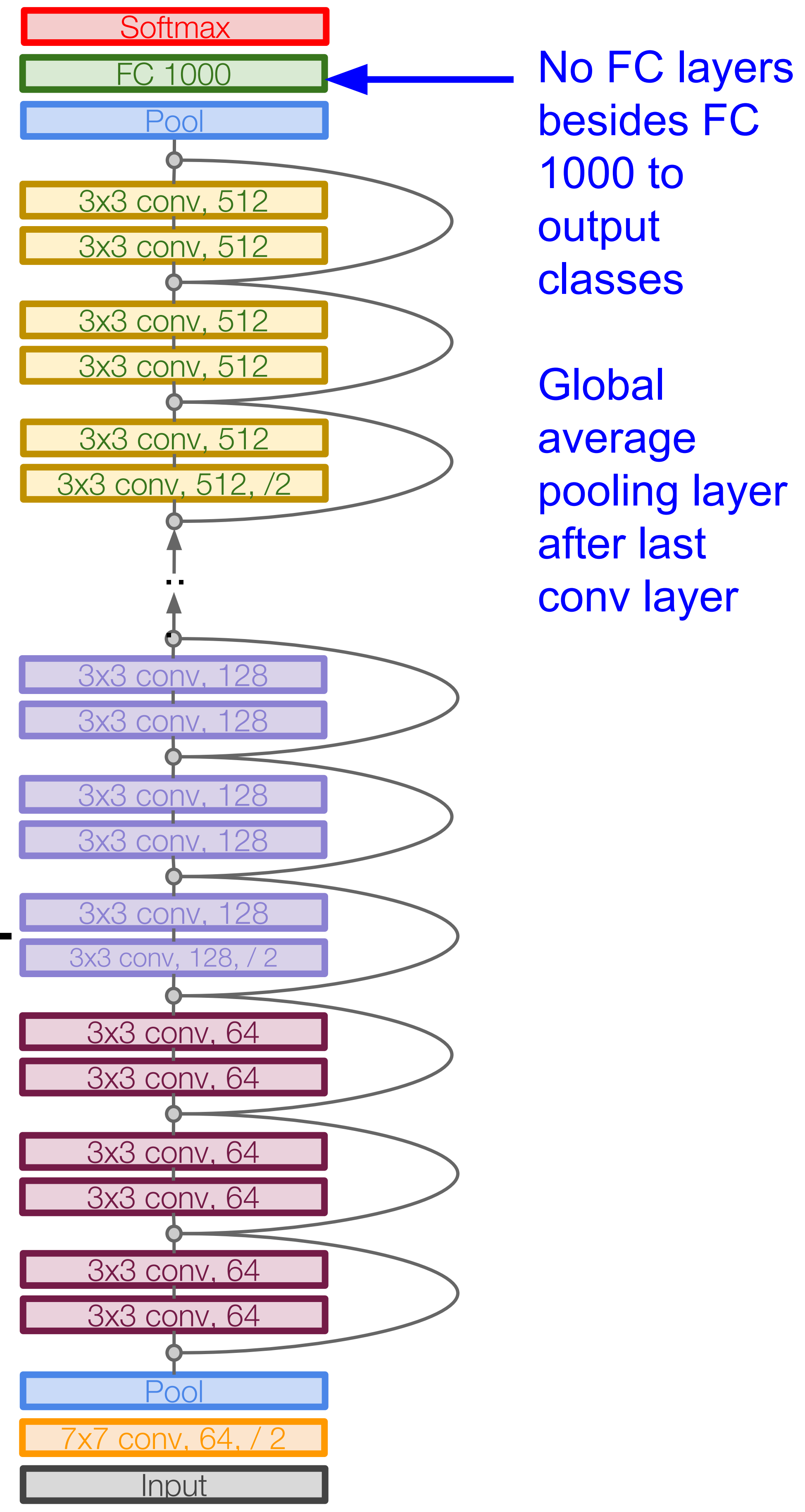
slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Case Study: ResNet

[He et al., 2015]

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)



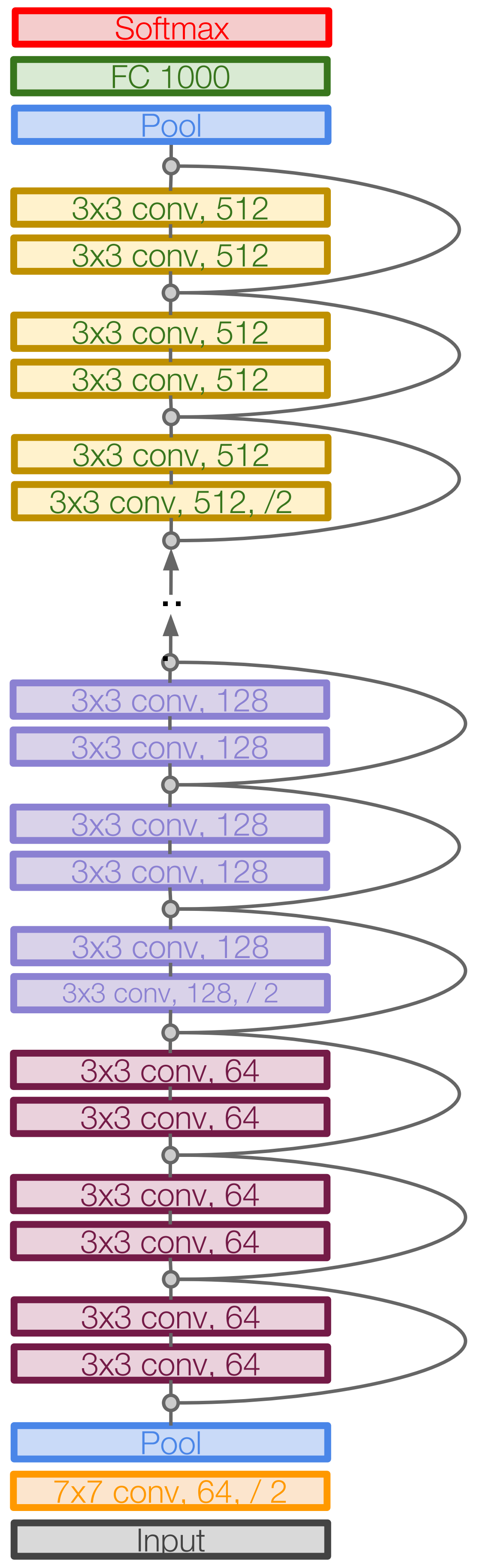


slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Case Study: ResNet

[He et al., 2015]

Total depths of 34, 50, 101, or 152 layers for ImageNet

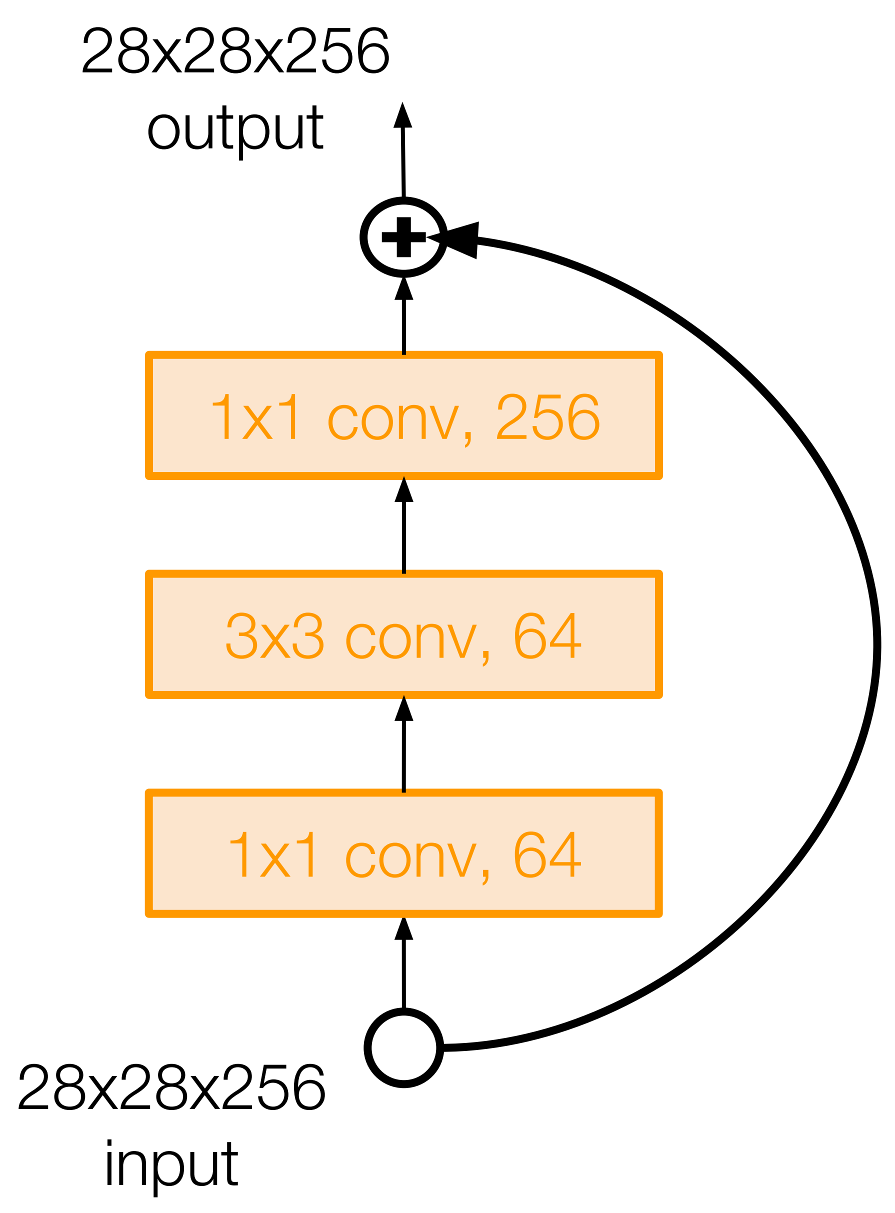


slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Case Study: ResNet

[He et al., 2015]

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



slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Case Study: ResNet

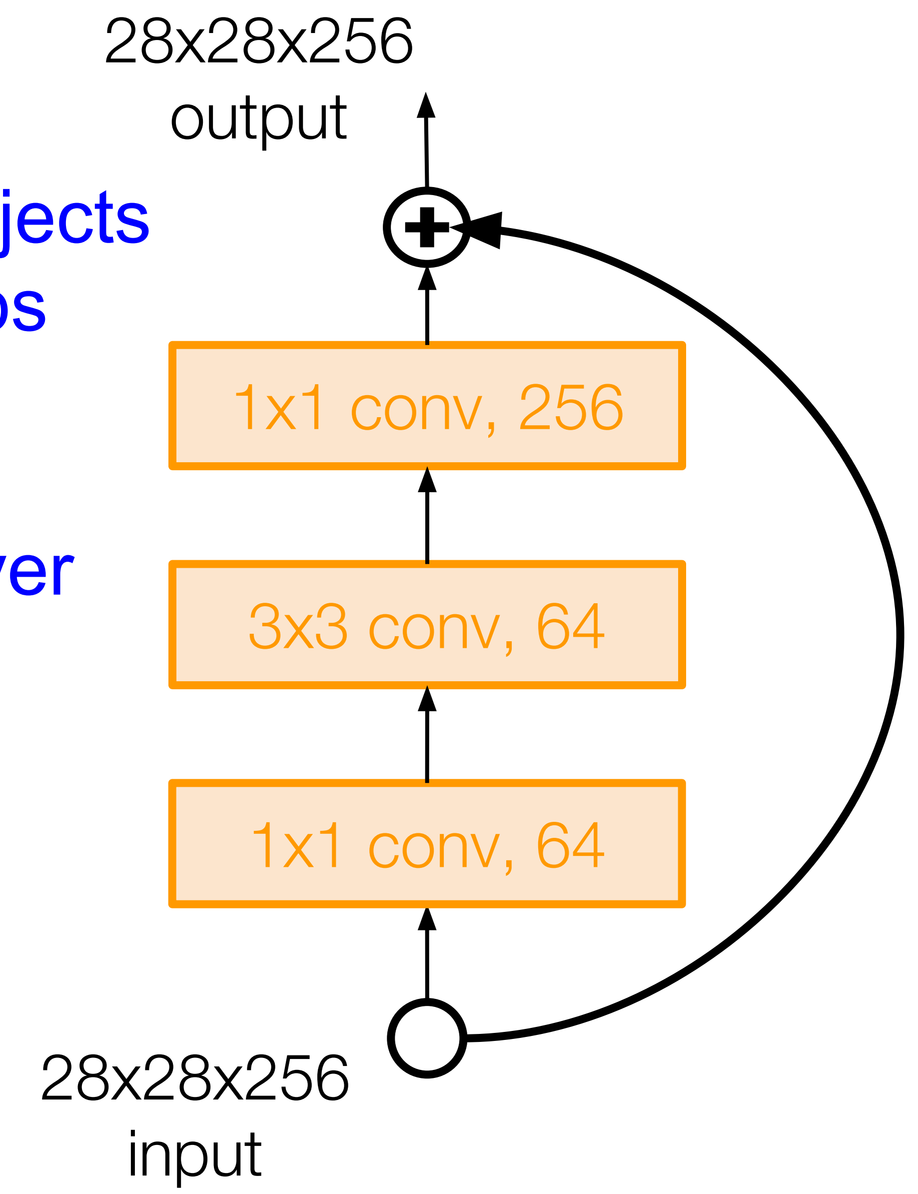
[He et al., 2015]

For deeper networks (ResNet-50+), use "bottleneck" layer to improve efficiency (similar to GoogLeNet)

1x1 conv, 256 filters projects back to 256 feature maps (28x28x256)

3x3 conv operates over only 64 feature maps

1x1 conv, 64 filters to project to 28x28x64



slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Case Study: ResNet

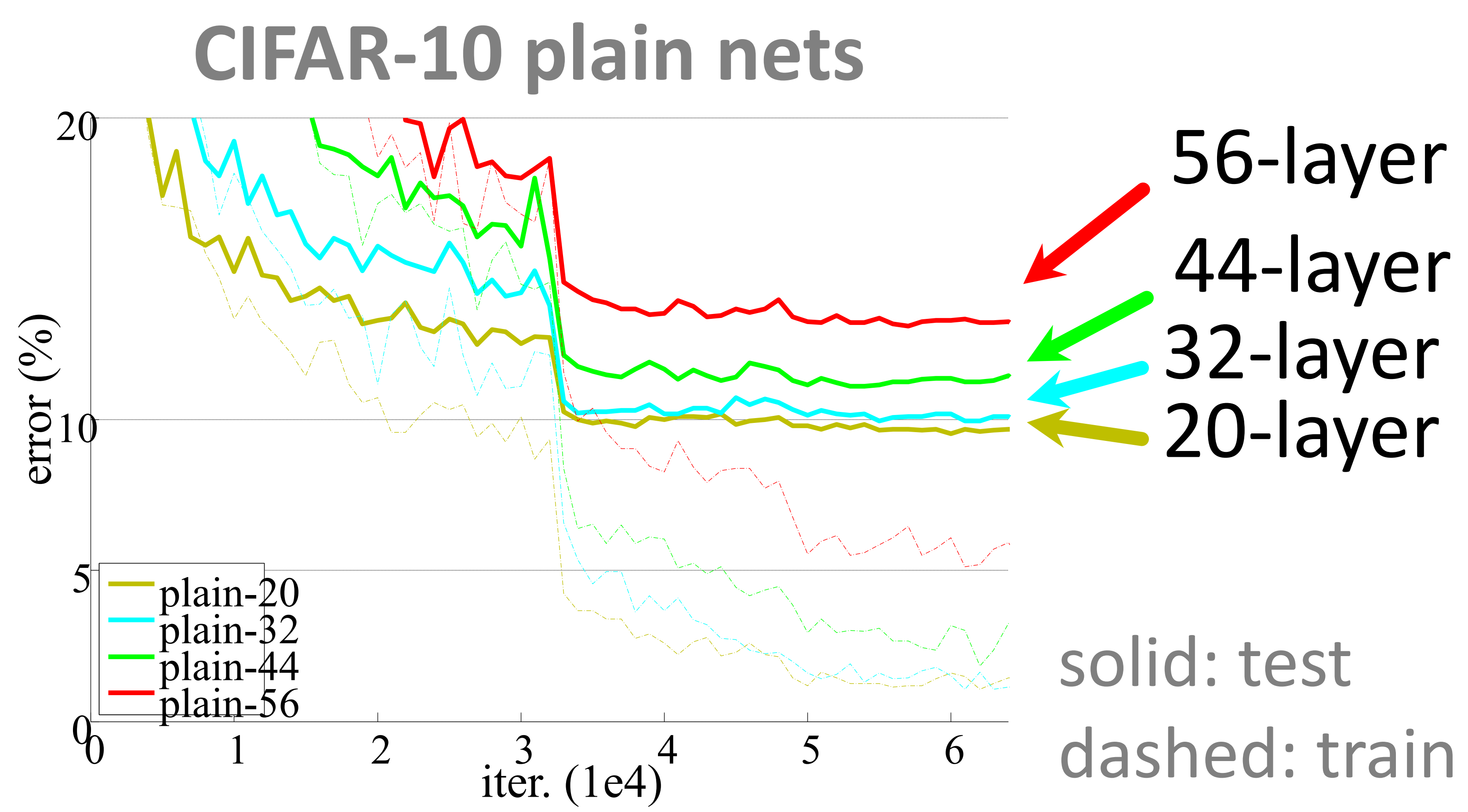
[He et al., 2015]

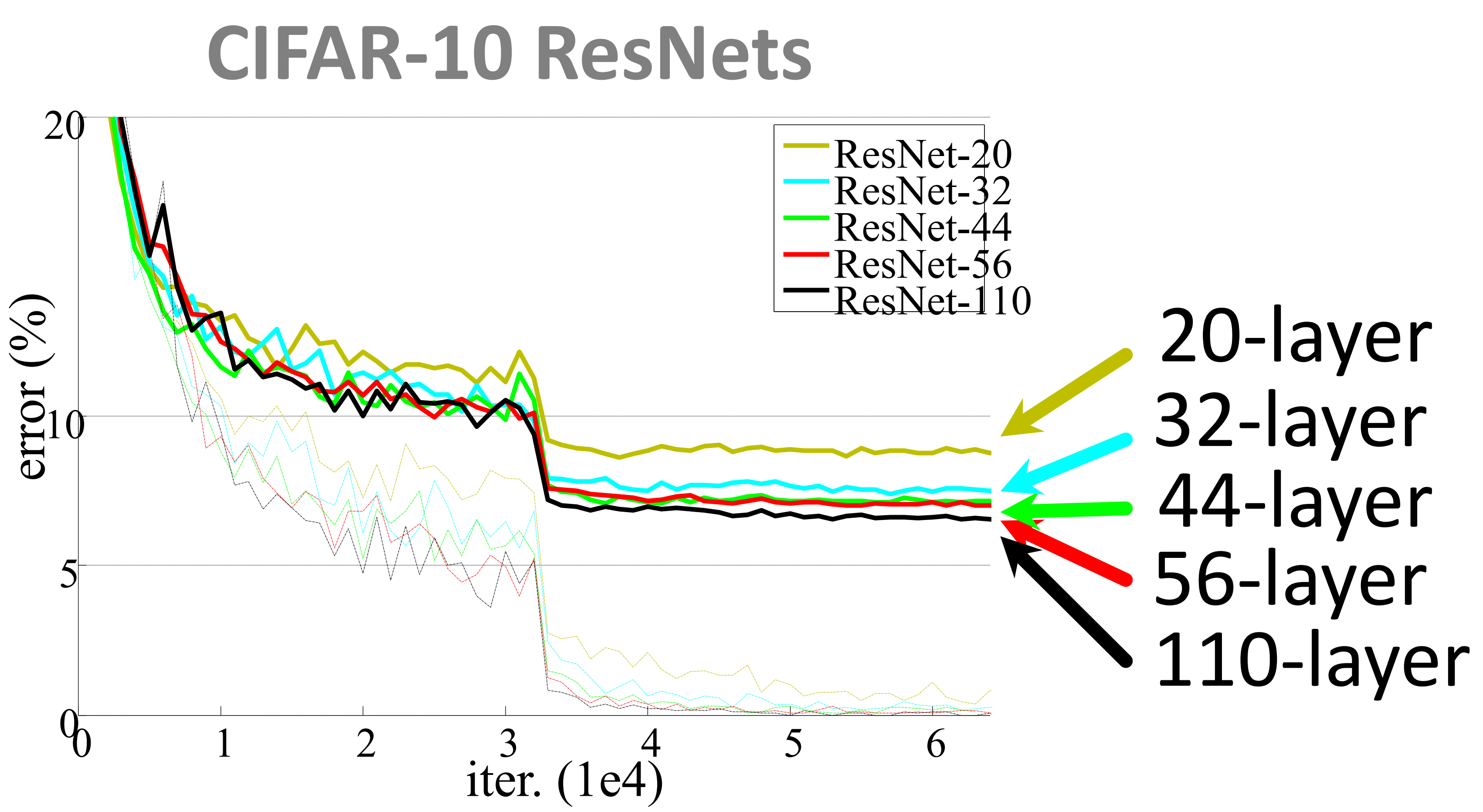
Training ResNet in practice:

- 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

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

CIFAR-10 experiments

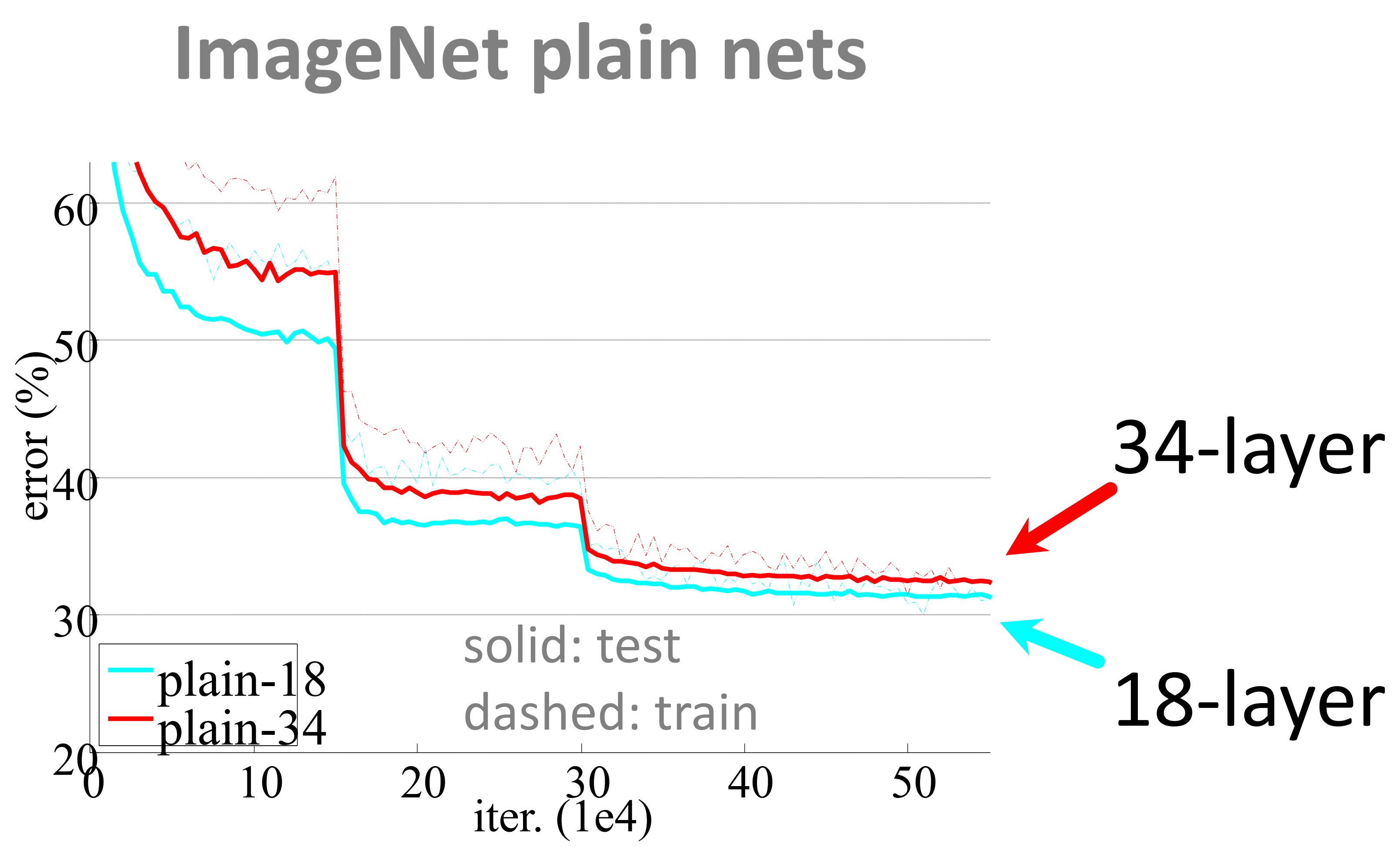


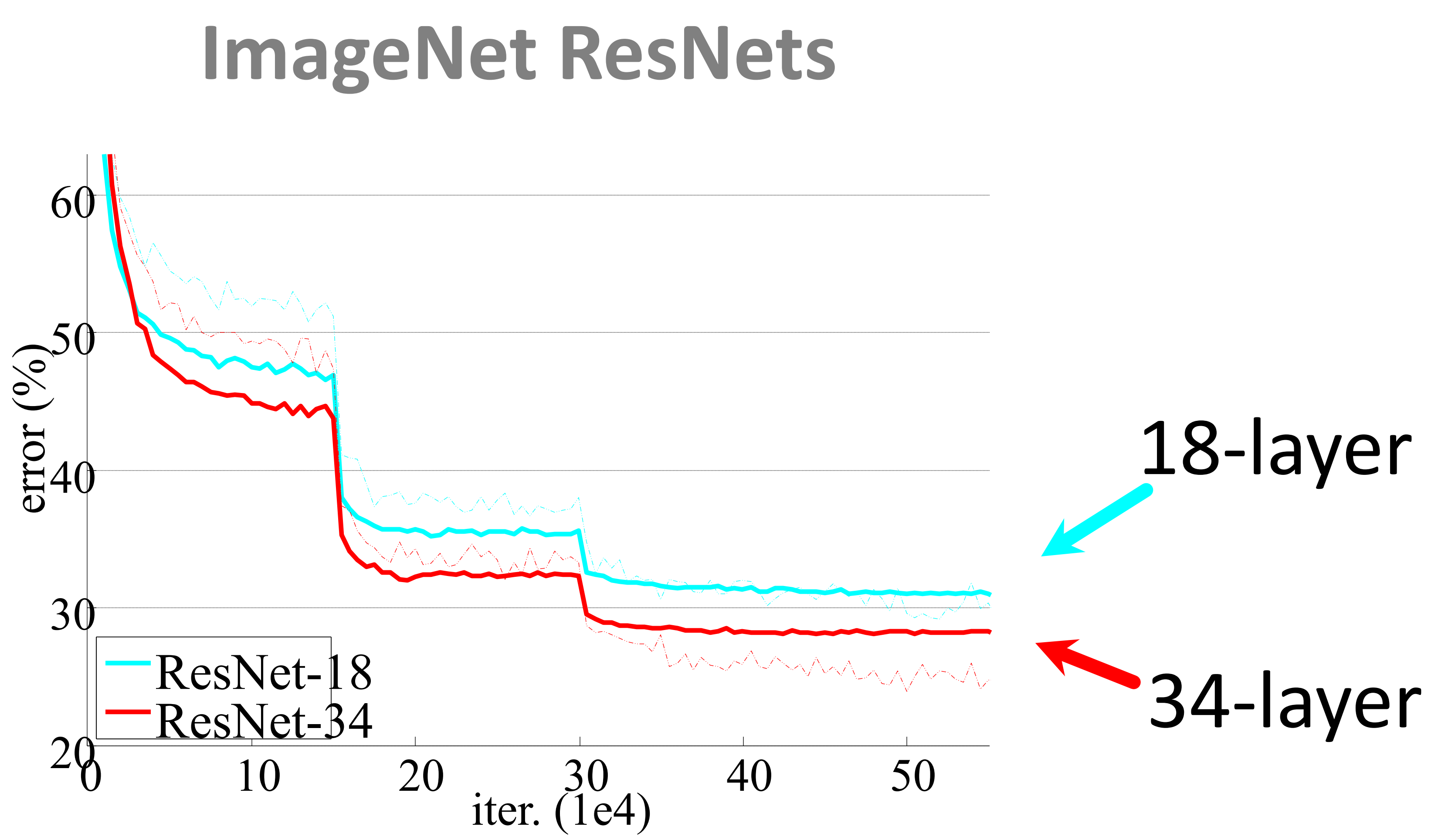


- Deep ResNets can be trained without difficulties
- Deeper ResNets have **lower training error**, and also lower test error

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

ImageNet experiments





- Deep ResNets can be trained without difficulties
- Deeper ResNets have **lower training error**, and also lower test error

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

Case Study: ResNet

[He et al., 2015]

Experimental Results
- Able to train very deep networks without degrading (152 layers on ImageNet, 1202 on Cifar)
- Deeper networks now achieve lowing training error as expected
- Swept 1st place in all ILSVRC and COCO 2015 competitions

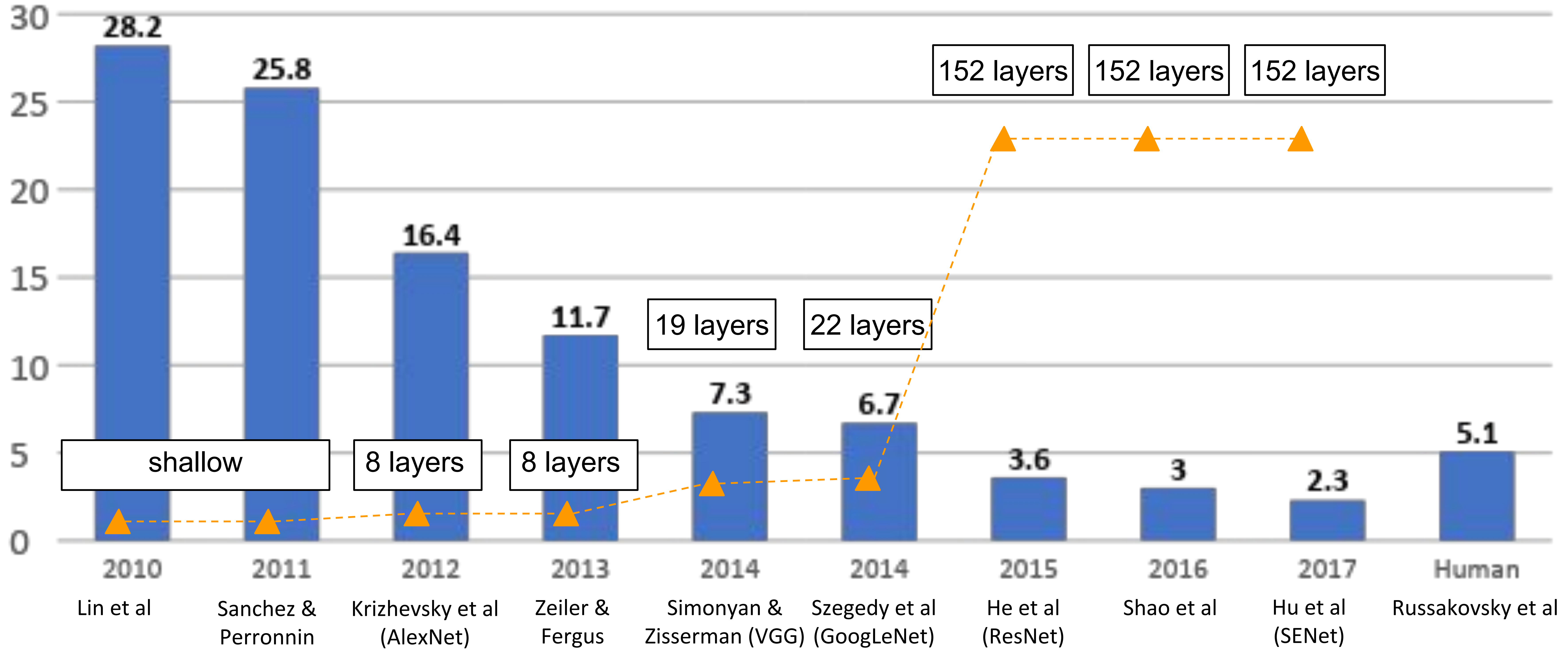
> MSRA @ ILSVRC & COCO 2015 Competitions
>
> - **1st places in all five main tracks**
>   - ImageNet Classification: "*Ultra-deep*" (quote Yann) 152-layer nets
>   - ImageNet Detection: 16% better than 2nd
>   - ImageNet Localization: 27% better than 2nd
>   - COCO Detection: 11% better than 2nd
>   - COCO Segmentation: 12% better than 2nd

ILSVRC 2015 classification winner (3.6% top 5 error) -- better than "human performance"! (Russakovsky 2014)

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

ImageNet Large Scale Visual Recognition Challenge (ILSVRC)

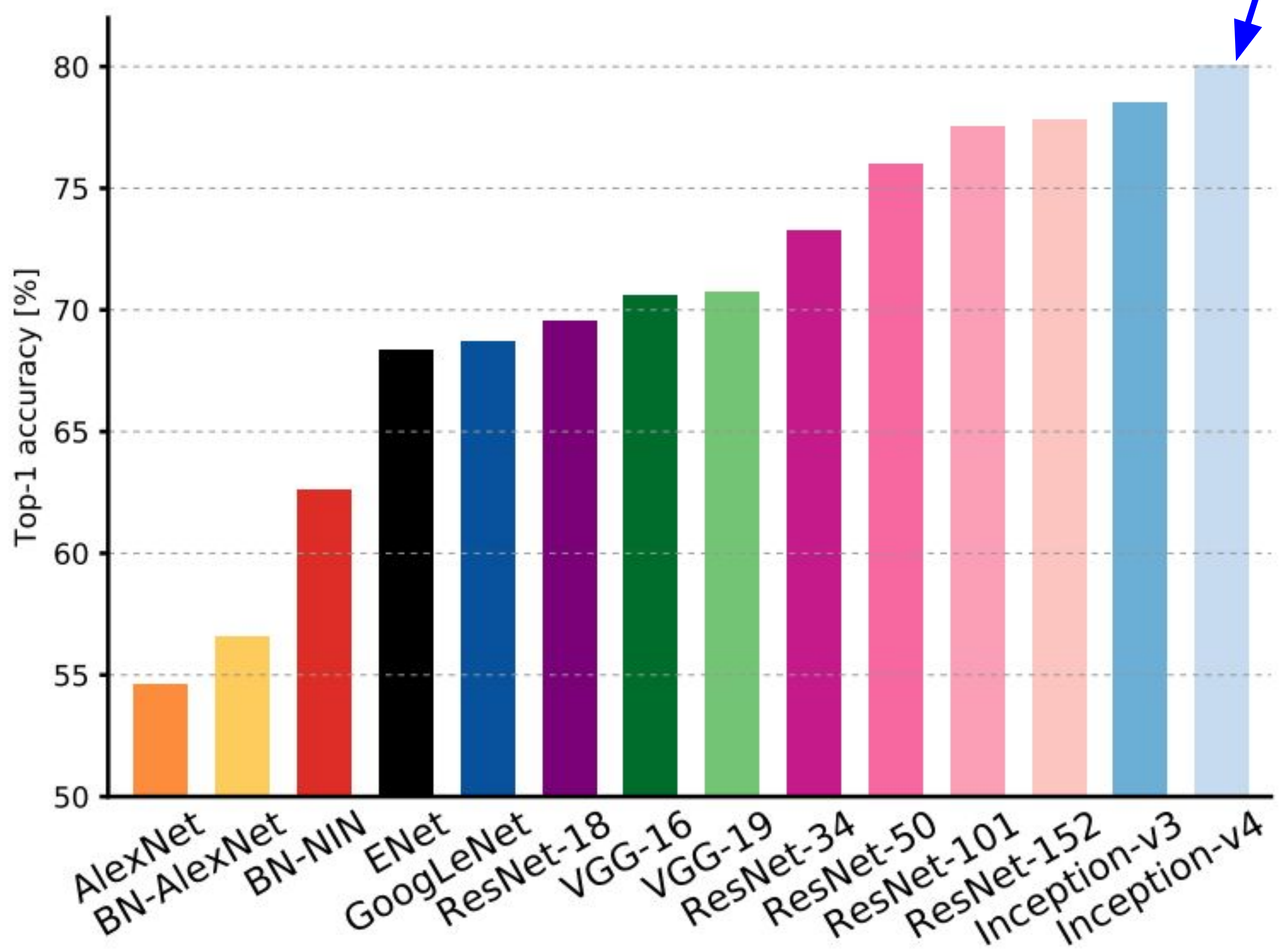
- Winners over the years (top-5 error):

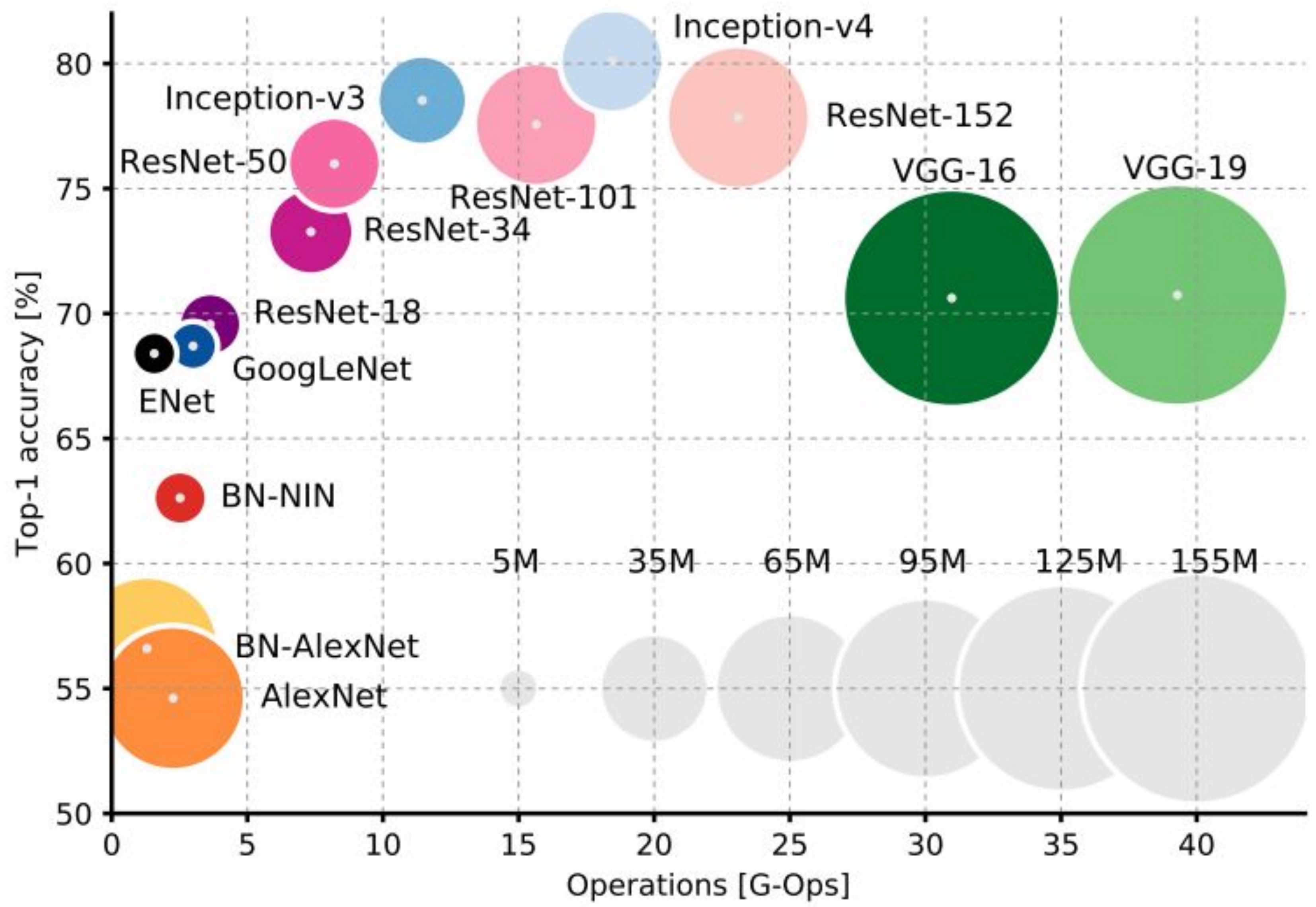


slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Comparing Complexity… (again on ImageNet Challenge)

Inception-v4: Resnet + Inception!



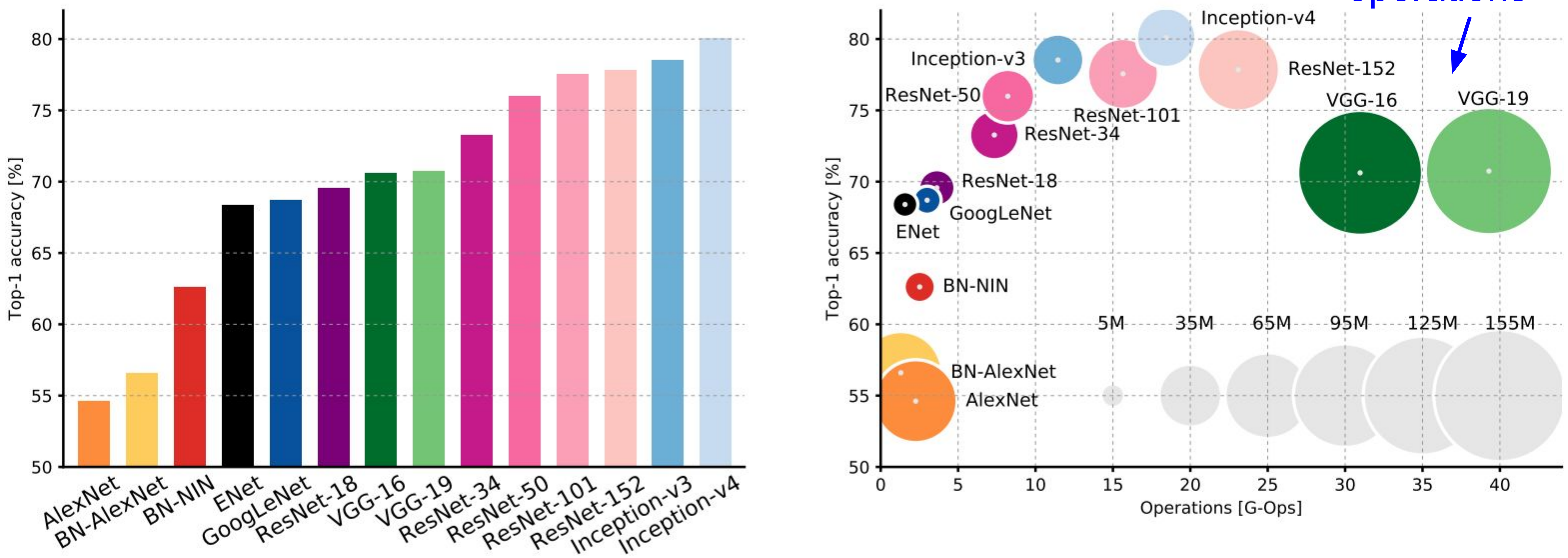


An Analysis of Deep Neural Network Models for Practical Applications, 2017.

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Comparing Complexity… (again on ImageNet Challenge)

VGG: Highest memory, most operations

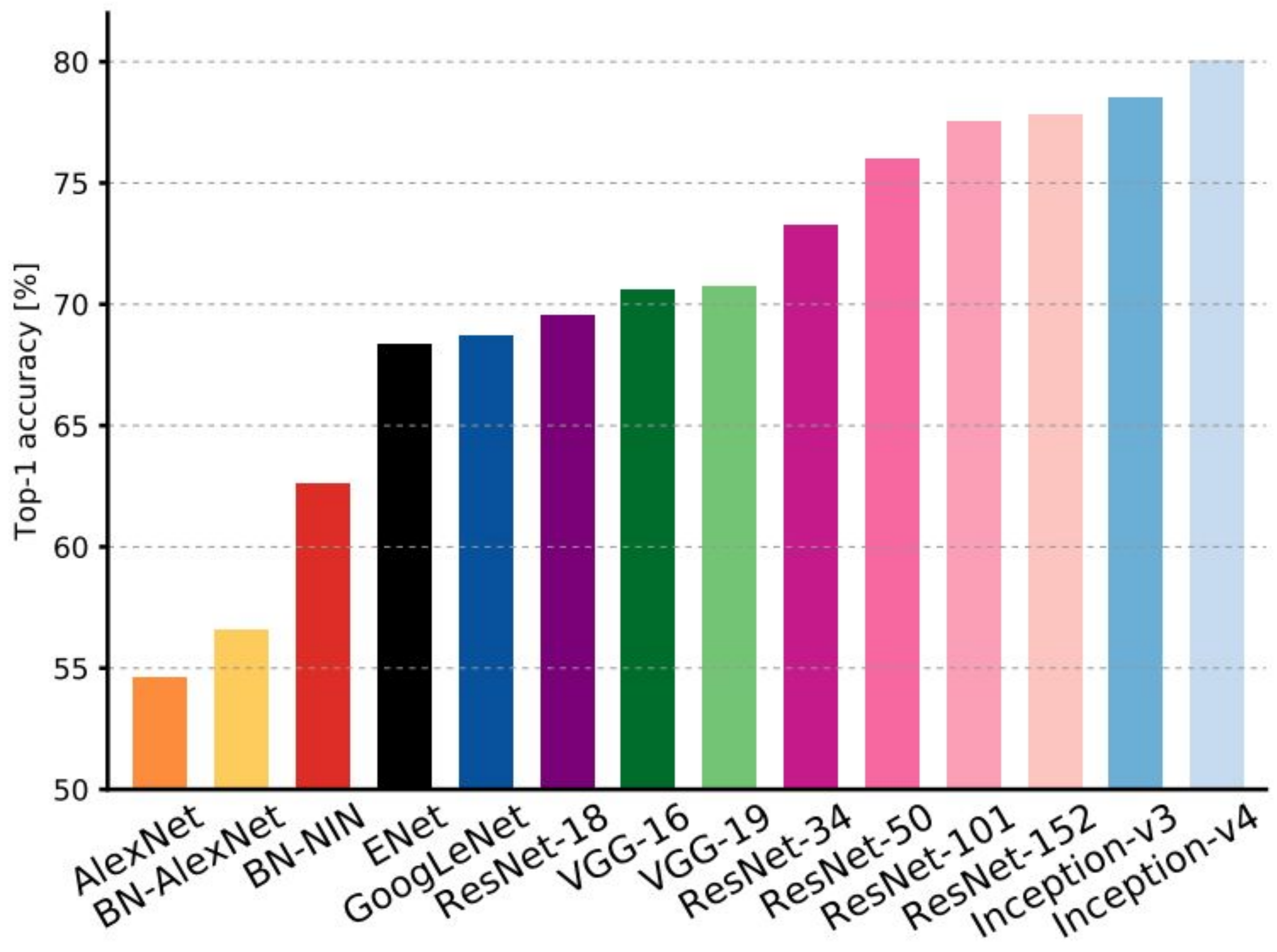


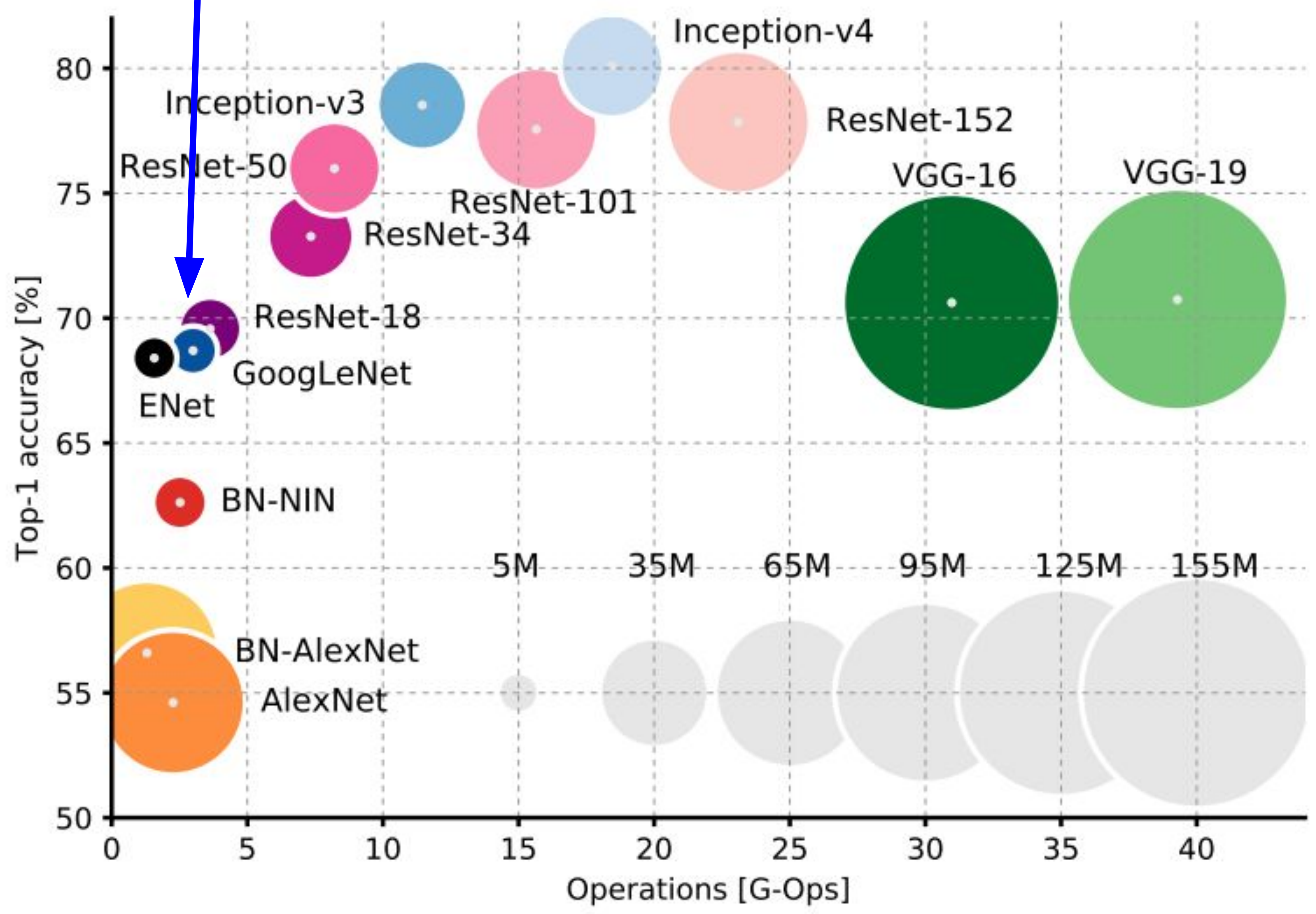
An Analysis of Deep Neural Network Models for Practical Applications, 2017.

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Comparing Complexity… (again on ImageNet Challenge)

GoogLeNet:
most efficient



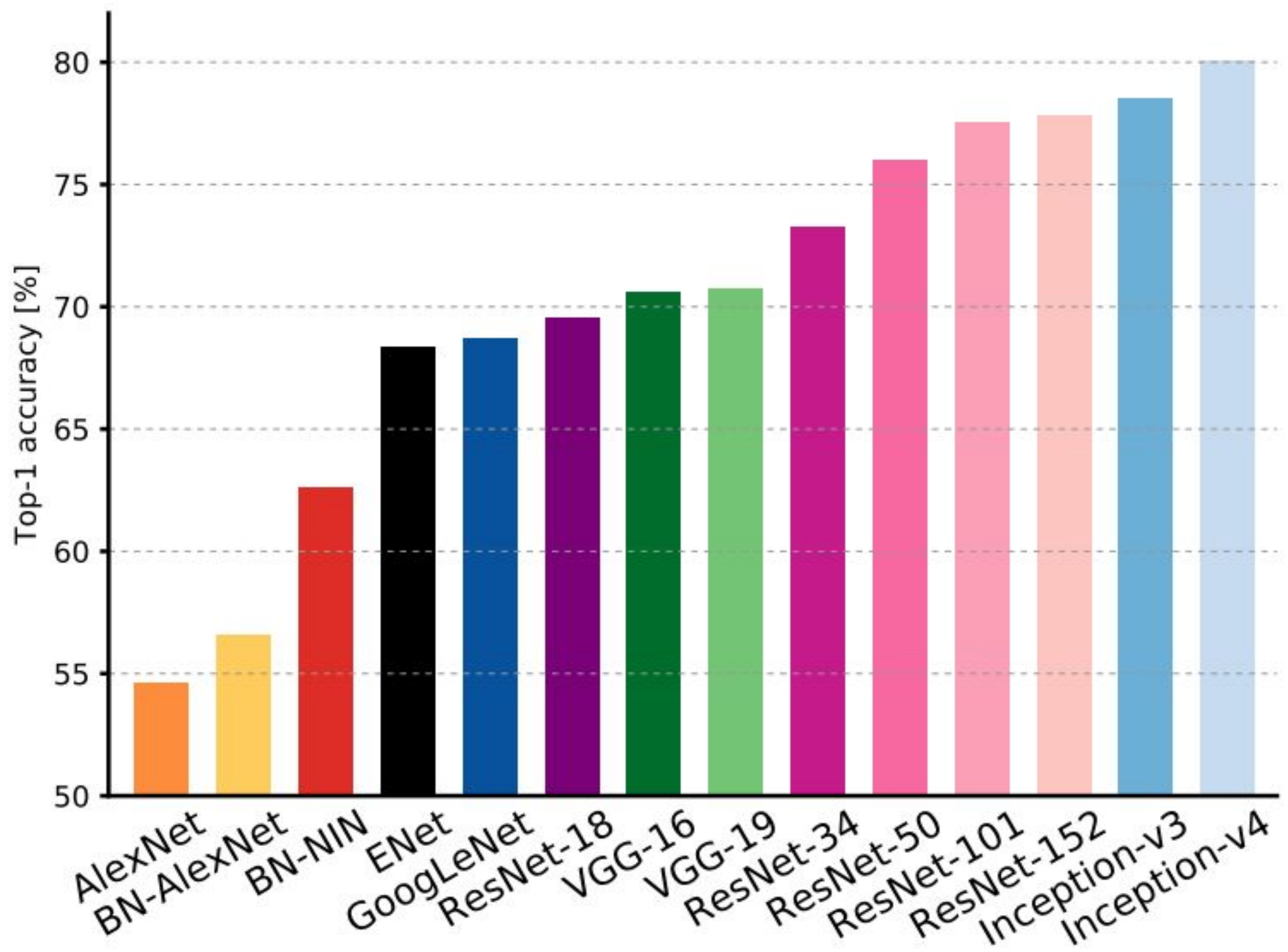


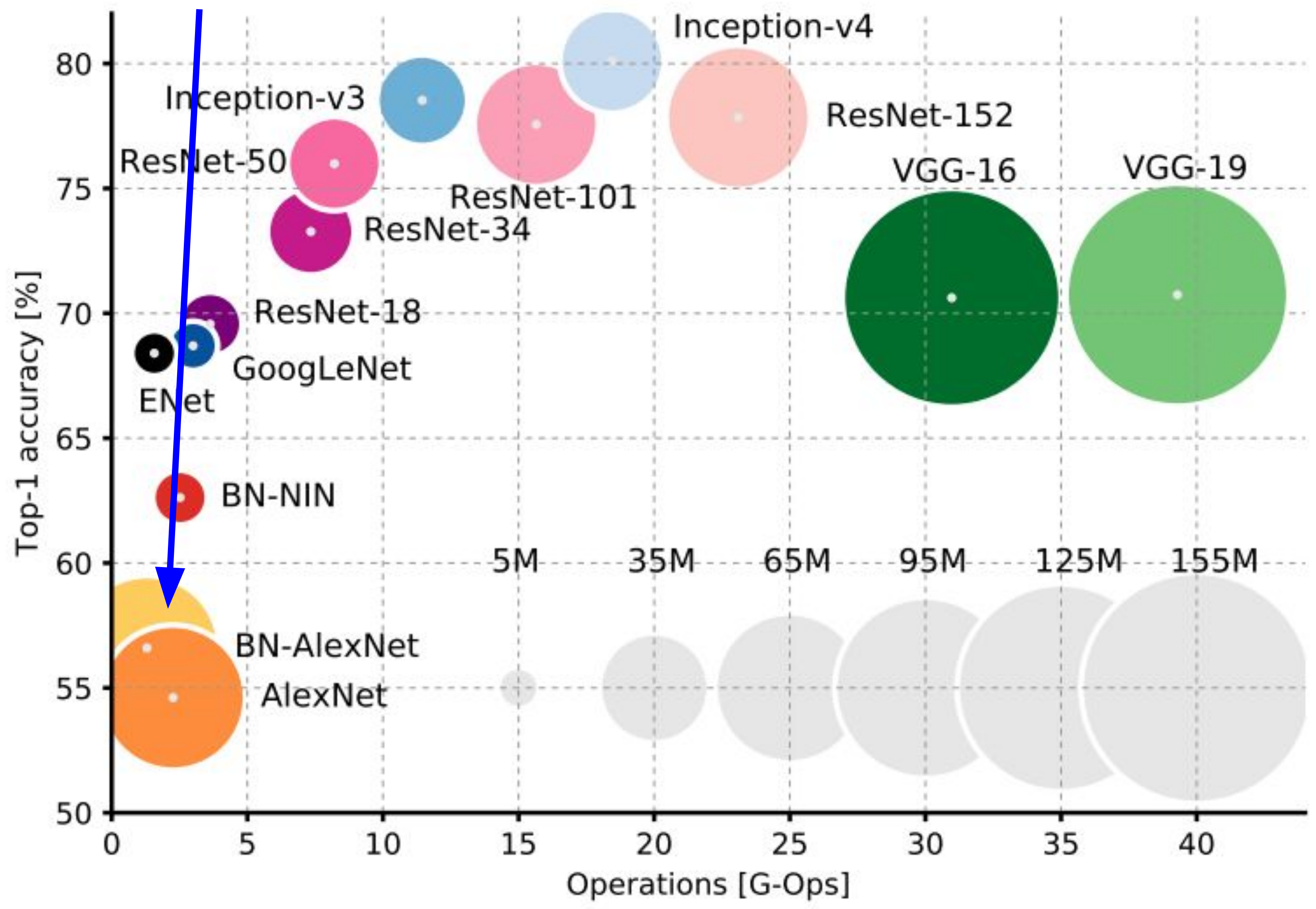
An Analysis of Deep Neural Network Models for Practical Applications, 2017.

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Comparing Complexity… (again on ImageNet Challenge)

AlexNet:
Smaller compute, still memory heavy, lower accuracy



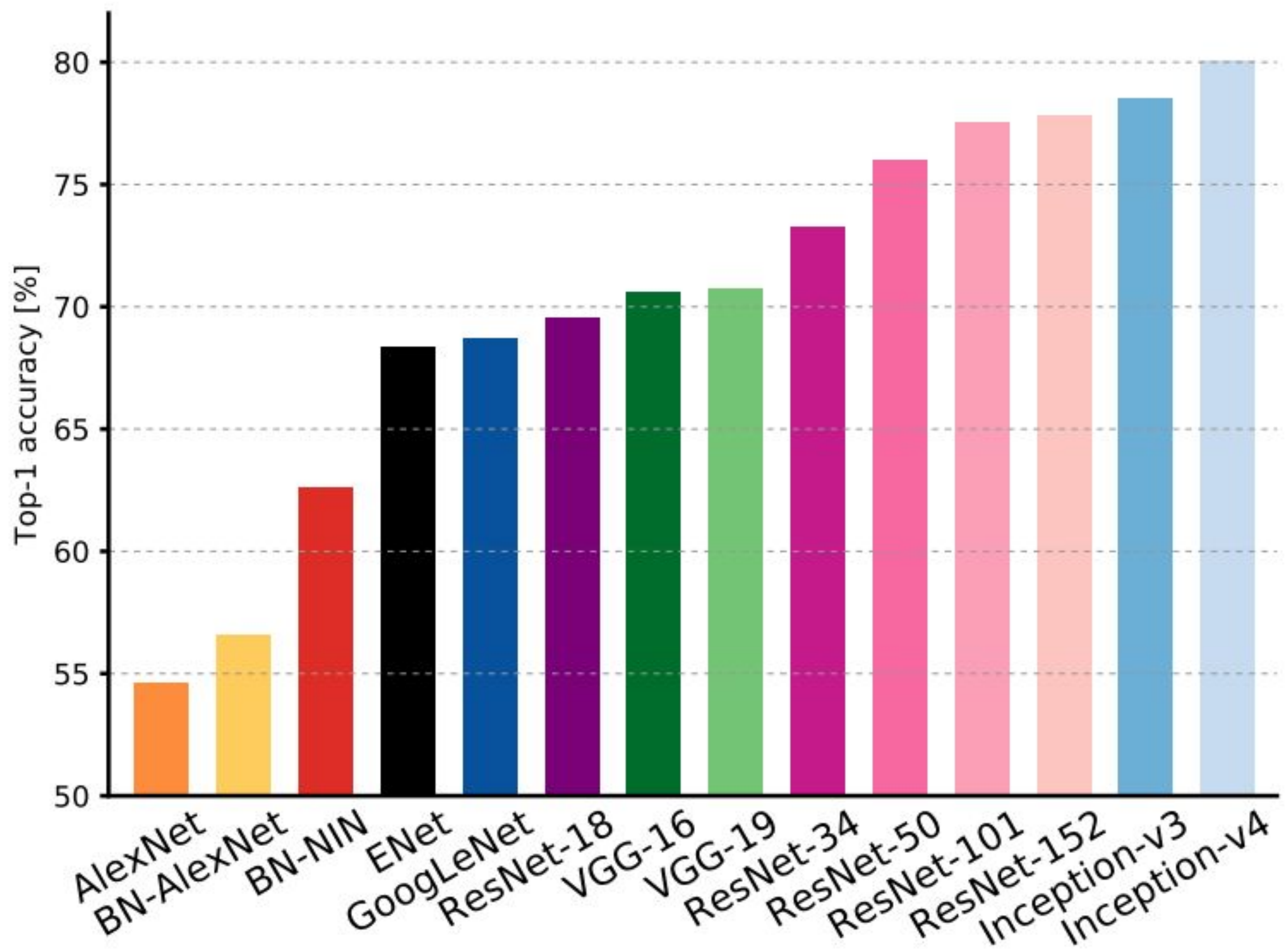


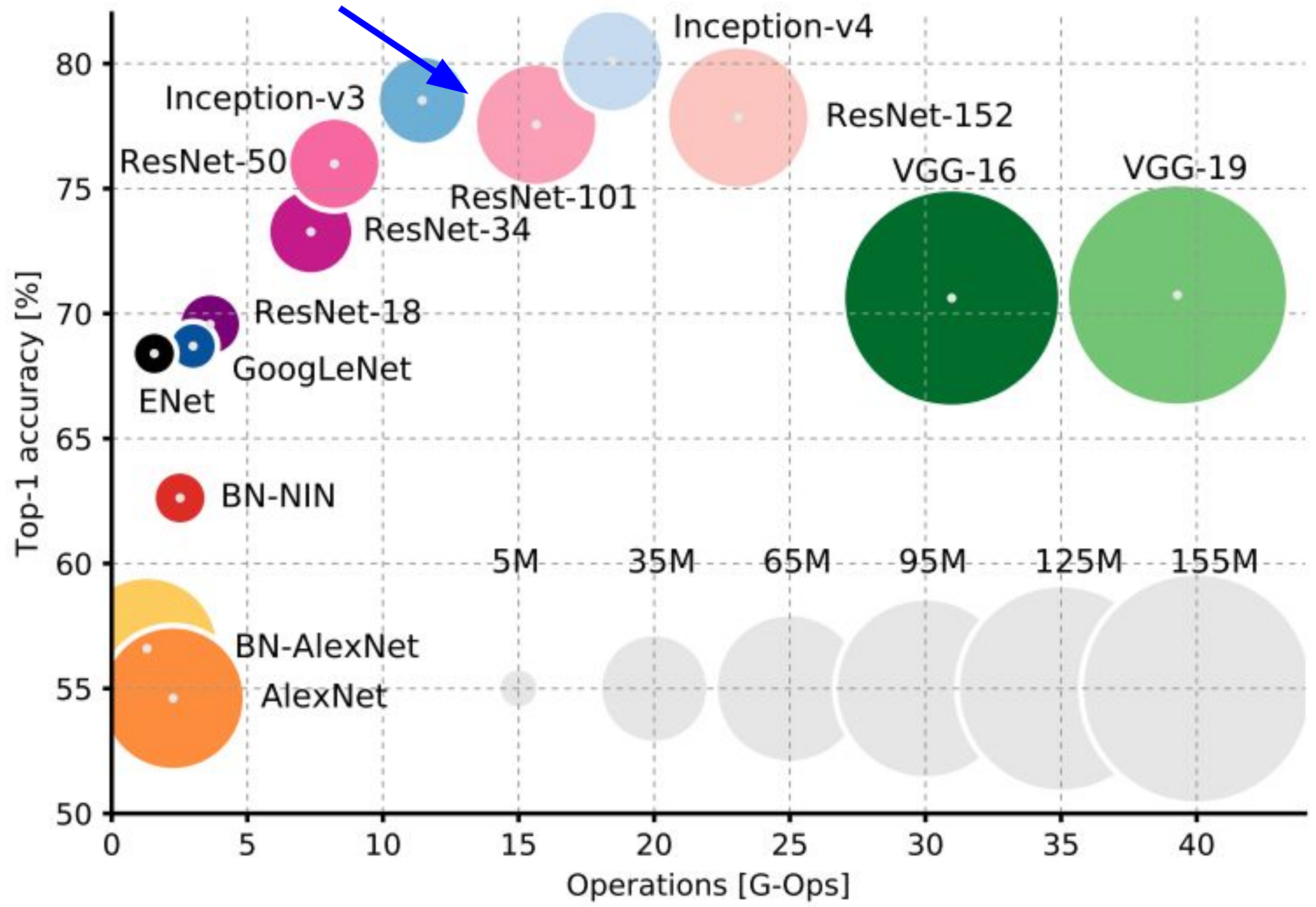
An Analysis of Deep Neural Network Models for Practical Applications, 2017.

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Comparing Complexity… (again on ImageNet Challenge)

ResNet:
Moderate efficiency depending on model, highest accuracy





An Analysis of Deep Neural Network Models for Practical Applications, 2017.

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Overview Today's Lecture

- Convolutional Neural Network (CNN) Architectures
  - case studies
    - AlexNet,
    - VGG
    - GoogLeNet
    - ResNet

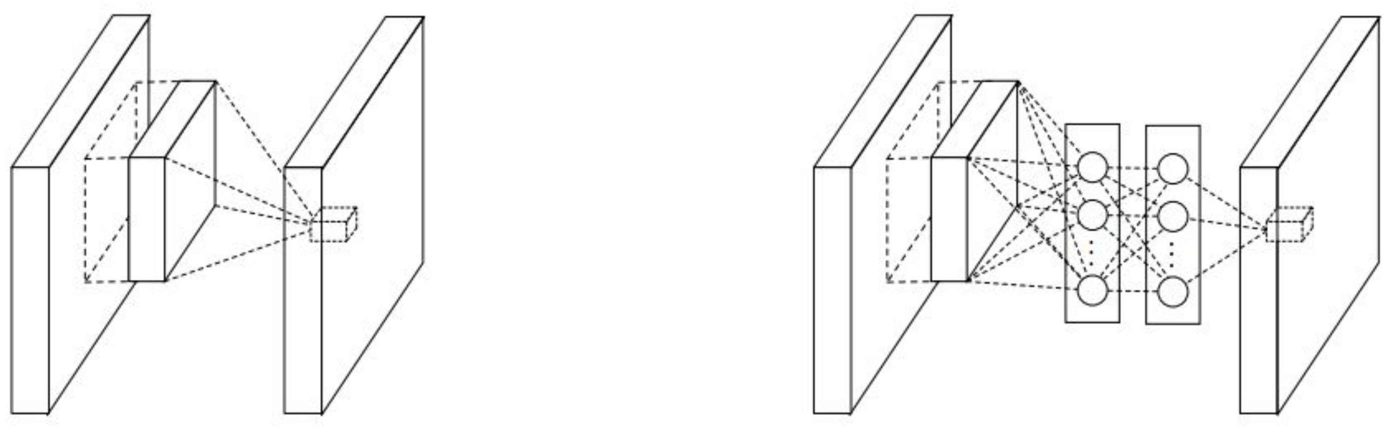
  - **other architectures to know (briefly discussed today)**
    - **NiN (Network in Network), Wide ResNet, ResNeXT, Stochastic Depth, Squeeze-and-Excitation Network, DenseNet, NASNet**

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Network in Network (NiN)

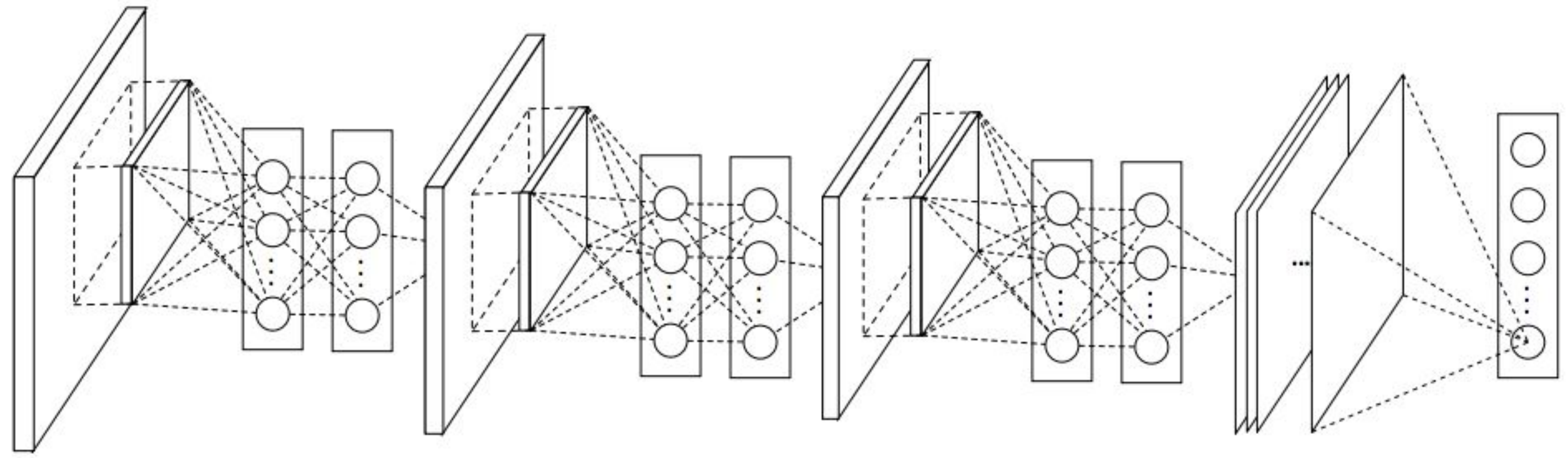
[Lin et al. 2014]

- Mlpconv layer with “micronetwork” within each conv layer to compute more abstract features for local patches
- Micronetwork uses multilayer perceptron (FC, i.e. 1x1 conv layers)
- Precursor to GoogLeNet and ResNet “bottleneck” layers
- Philosophical inspiration for GoogLeNet



(a) Linear convolution layer

(b) Mlpconv layer



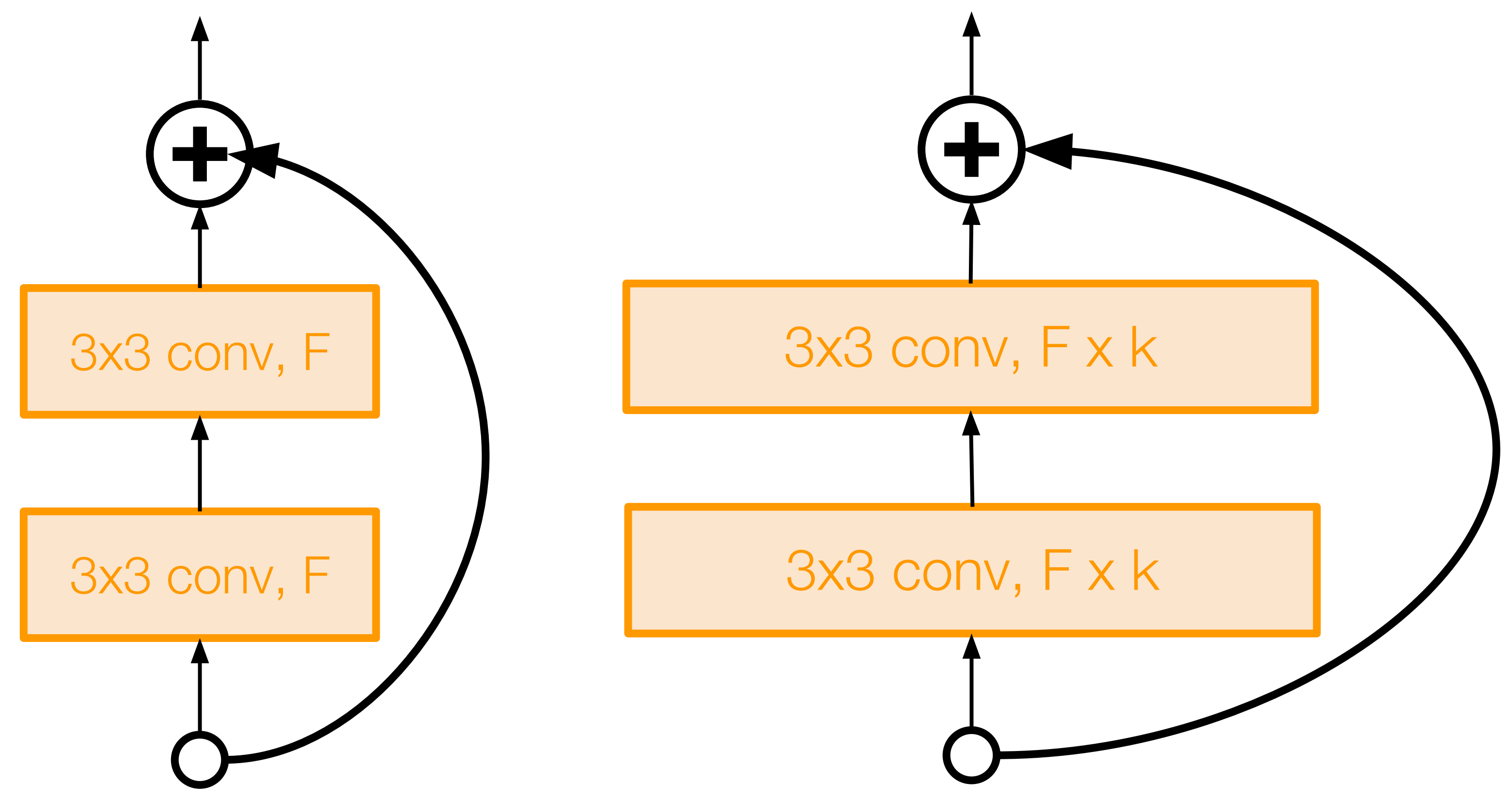
slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Improving ResNets...

- Wide Residual Networks

[Zagoruyko et al. 2016]

- Argues that residuals are the important factor, not depth
- User wider residual blocks (F x k filters instead of F filters in each layer)
- 50-layer wide ResNet outperforms 152-layer original ResNet
- Increasing width instead of depth more computationally efficient (parallelizable)



Basic residual block   Wide residual block

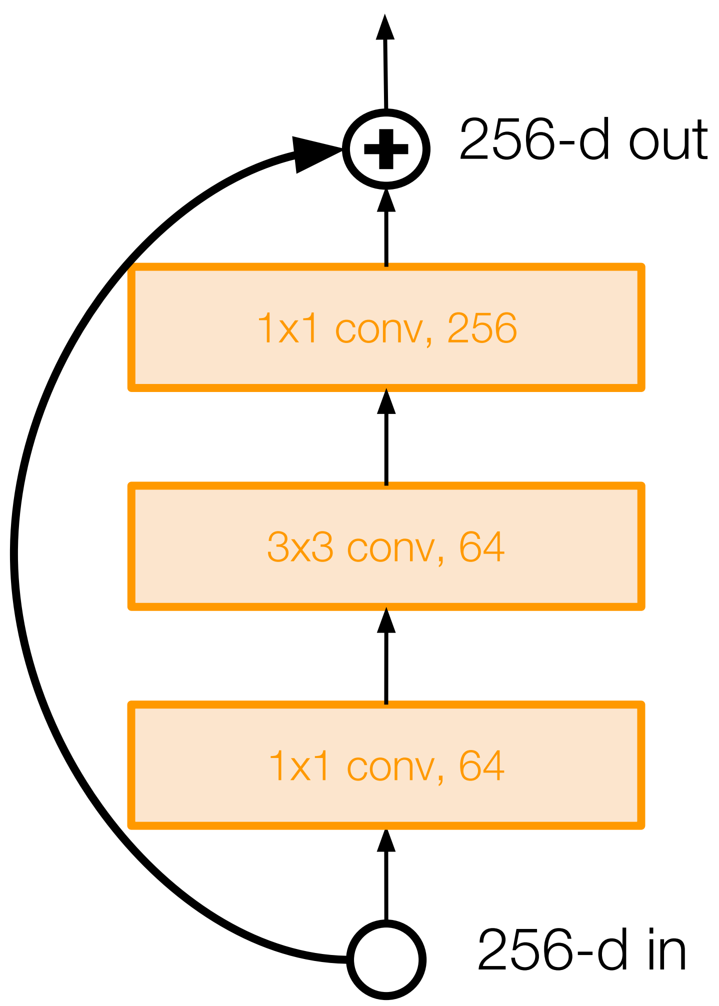
slide credit: Fei-Fei, Justin Johnson, Serena Yeung

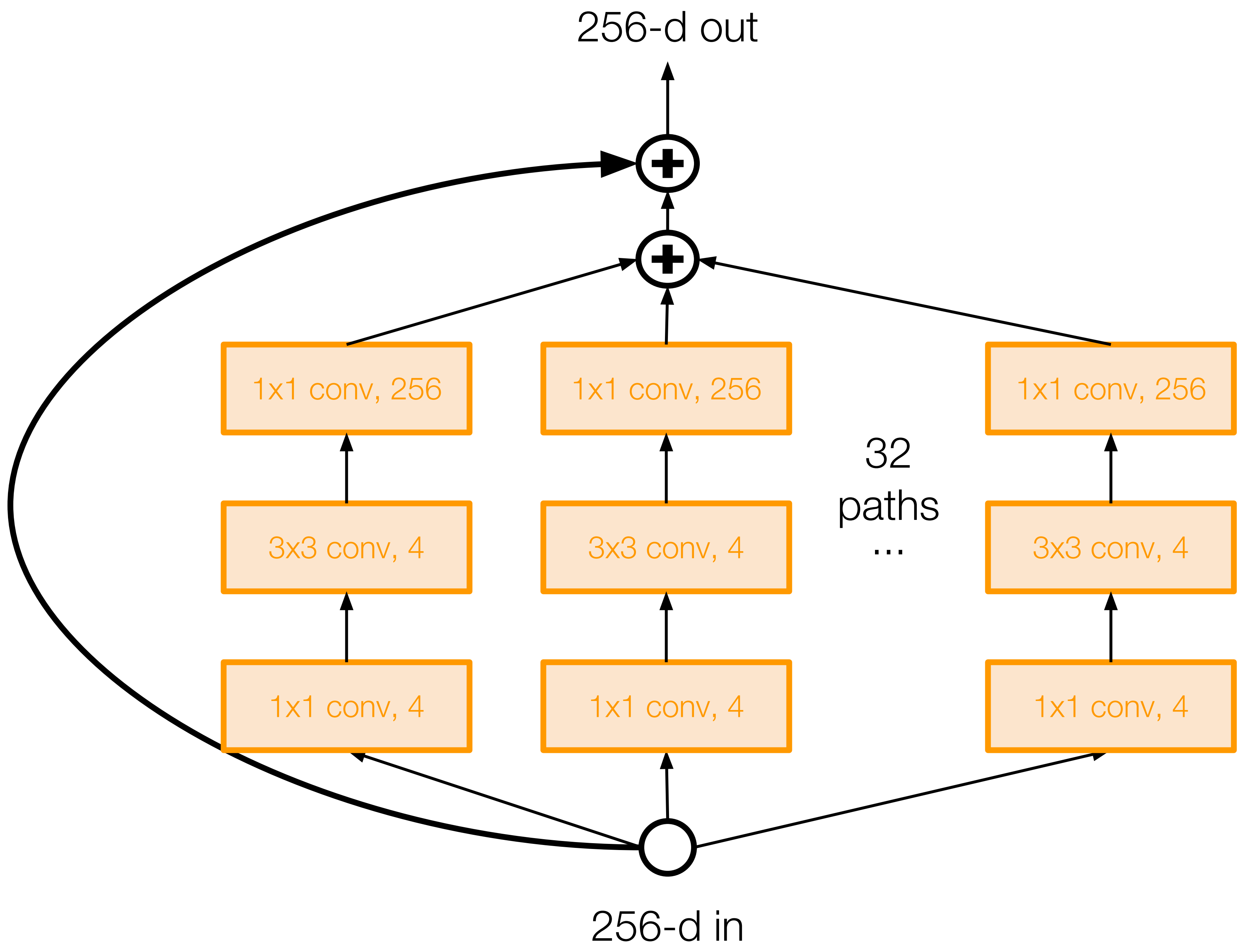
Improving ResNets...

- Aggregated Residual Transformations for Deep Neural Networks (ResNeXt)

[Xie et al. 2016]

- Also from creators of ResNet
- Increases width of residual block through multiple parallel pathways (“cardinality”)
- Parallel pathways similar in spirit to Inception module





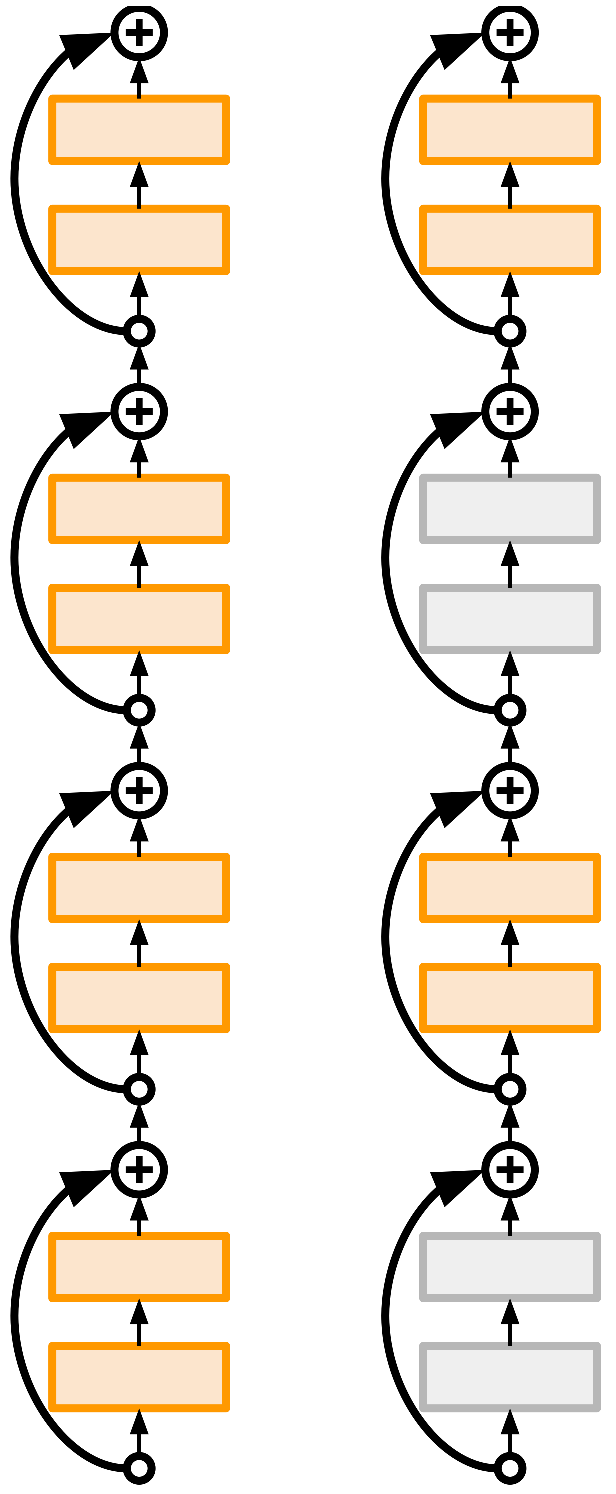
slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Improving ResNets...

- Deep Networks with Stochastic Depth

[Huang et al. 2016]

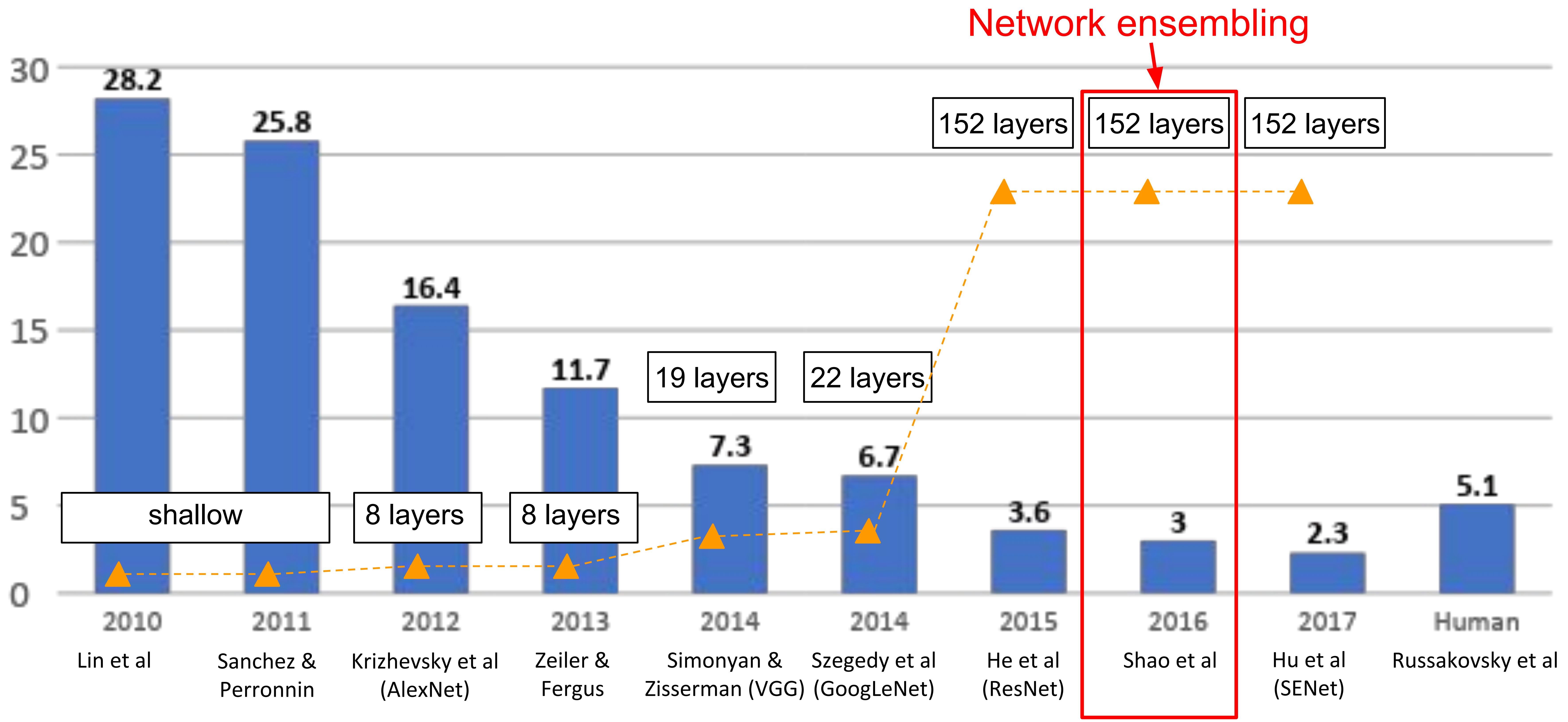
- Motivation: reduce vanishing gradients and training time through short networks during training
- Randomly drop a subset of layers during each training pass
- Bypass with identity function
- Use full deep network at test time



slide credit: Fei-Fei, Justin Johnson, Serena Yeung

ImageNet Large Scale Visual Recognition Challenge (ILSVRC)

- Winners over the years (top-5 error):



slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Improving ResNets...

- Good Practices for Deep Feature Fusion

[Shao et al. 2016]

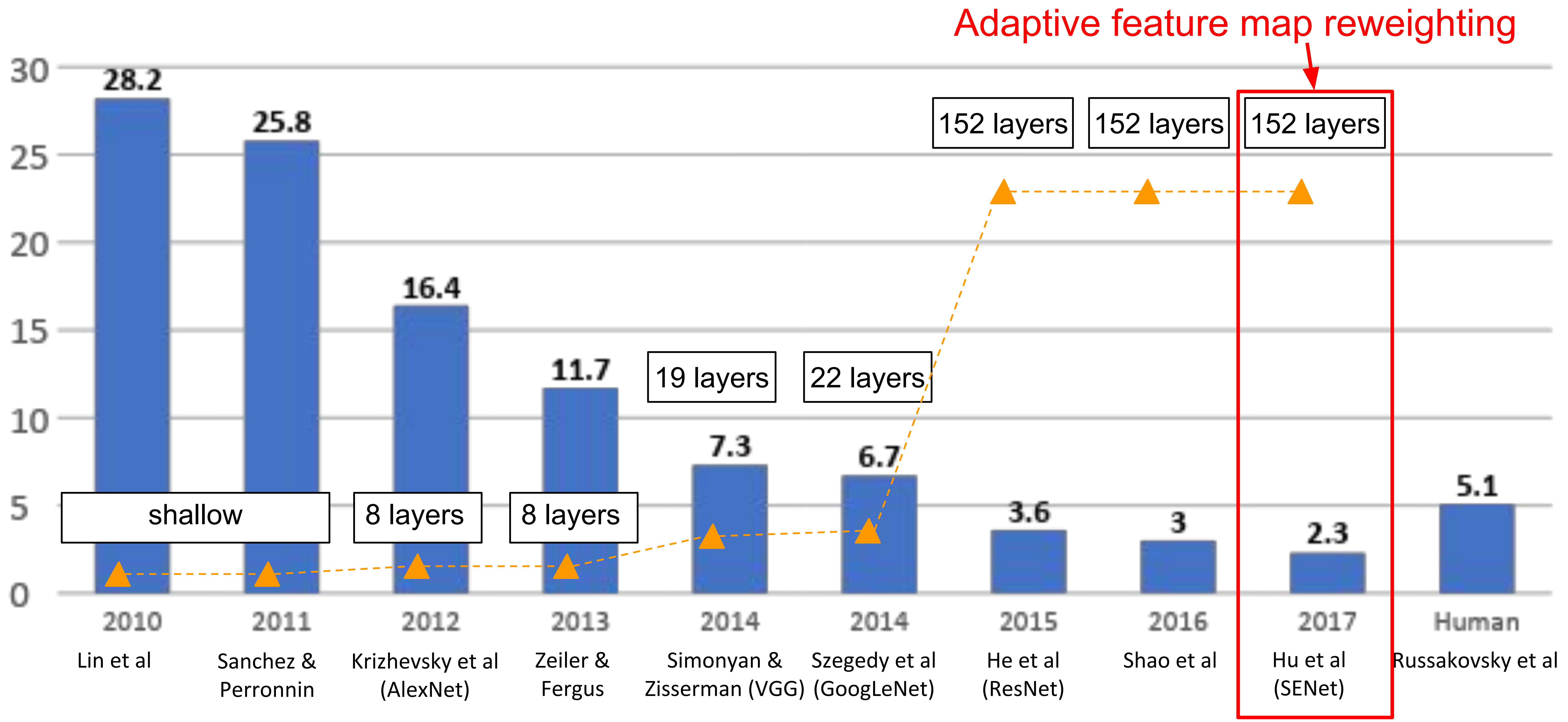
- Multi-scale ensembling of Inception, Inception-Resnet, Resnet, Wide Resnet models
- ILSVRC'16 classification winner

	Inception-v3	Inception-v4	Inception-Resnet-v2	Resnet-200	Wrn-68-3	Fusion（Val.）	Fusion（Test）
Err. (%)	4.20	4.01	3.52	4.26	4.65	2.92 (-0.6)	2.99

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

ImageNet Large Scale Visual Recognition Challenge (ILSVRC)

- Winners over the years (top-5 error):



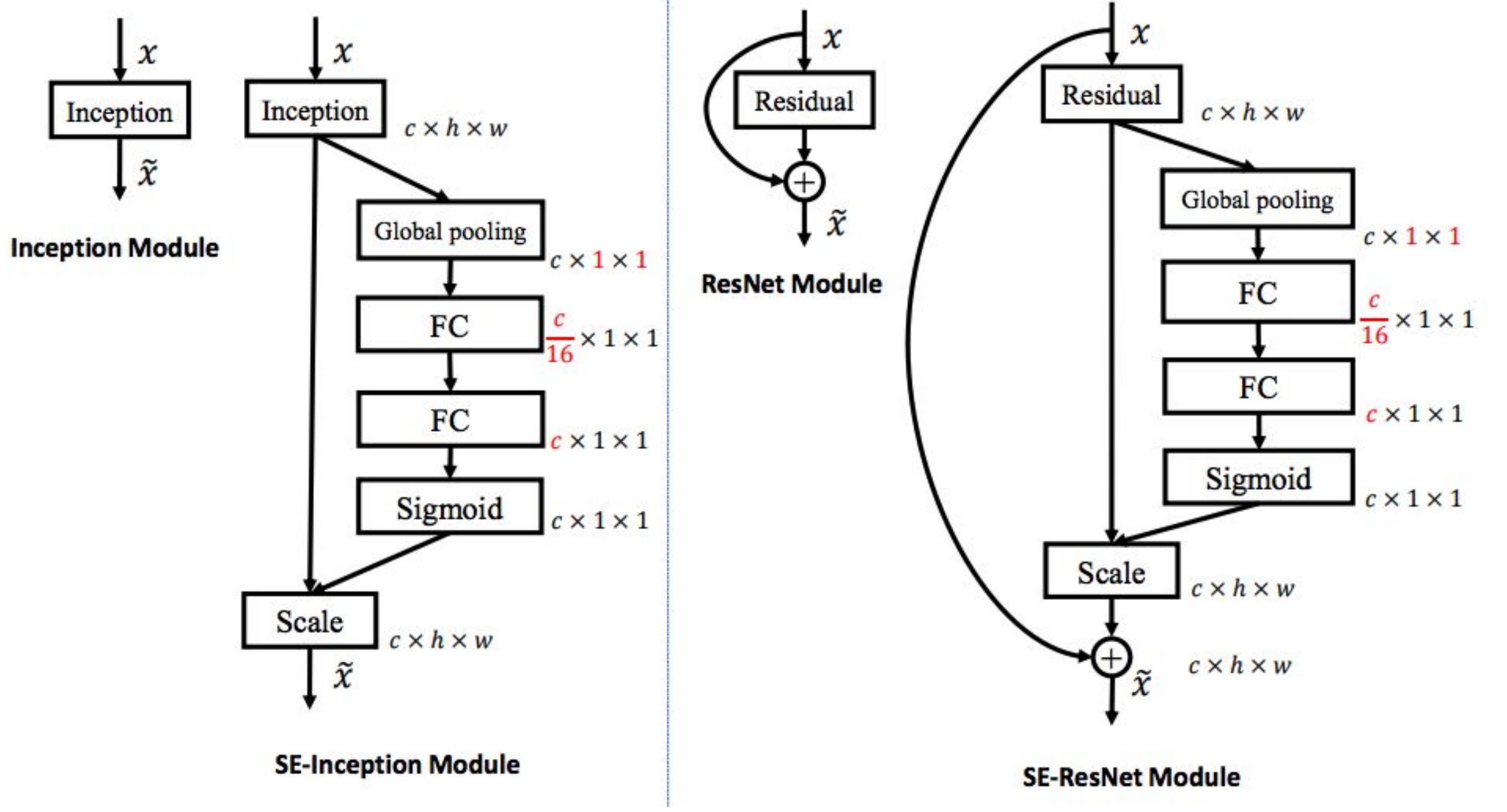
slide credit: Fei-Fei, Justin Johnson, Serena Yeung

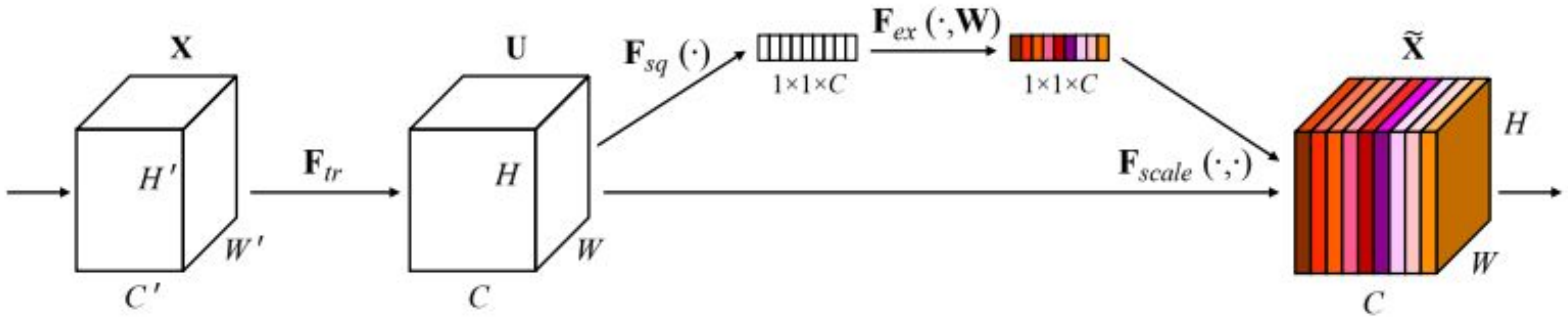
Improving ResNets...

- Squeeze-and-Excitation Networks (SENet)

[Hu et al. 2017]

- Add a “feature recalibration” module that learns to adaptively reweight feature maps
- Global information (global avg. pooling layer) + 2 FC layers used to determine feature map weights
- ILSVRC’17 classification winner (using ResNeXt-152 as a base architecture)





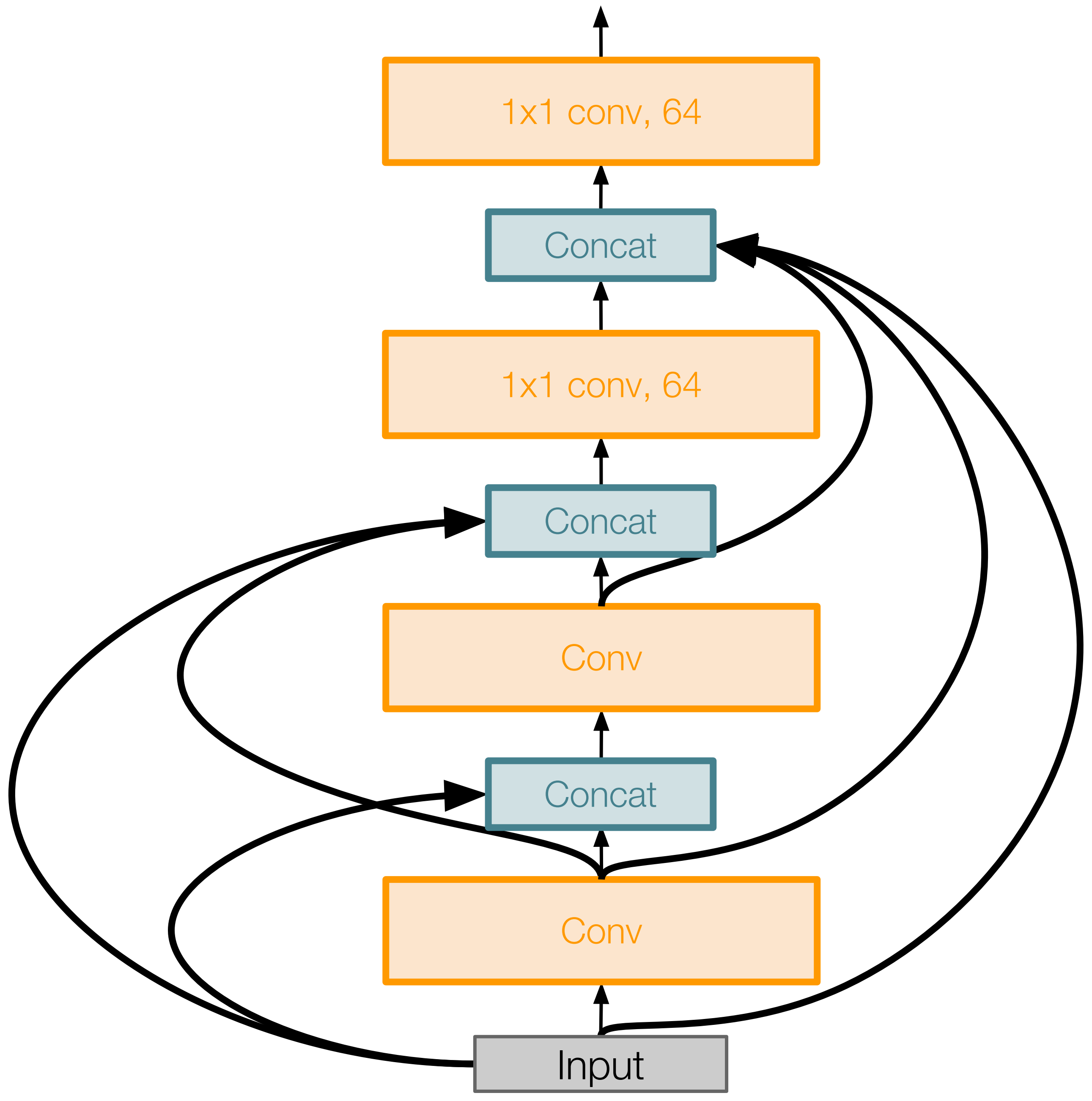
slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Beyond ResNets...

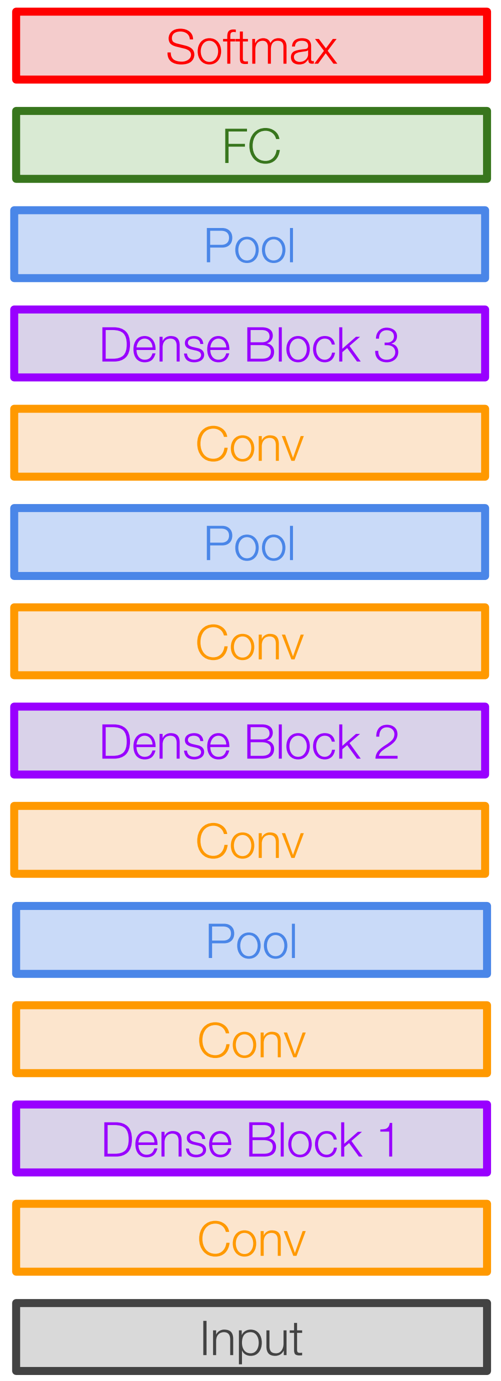
- Densely Connected Convolutional Networks

[Huang et al. 2017]

- Dense blocks where each layer is connected to every other layer in feedforward fashion
- Alleviates vanishing gradient, strengthens feature propagation, encourages feature reuse



Dense Block



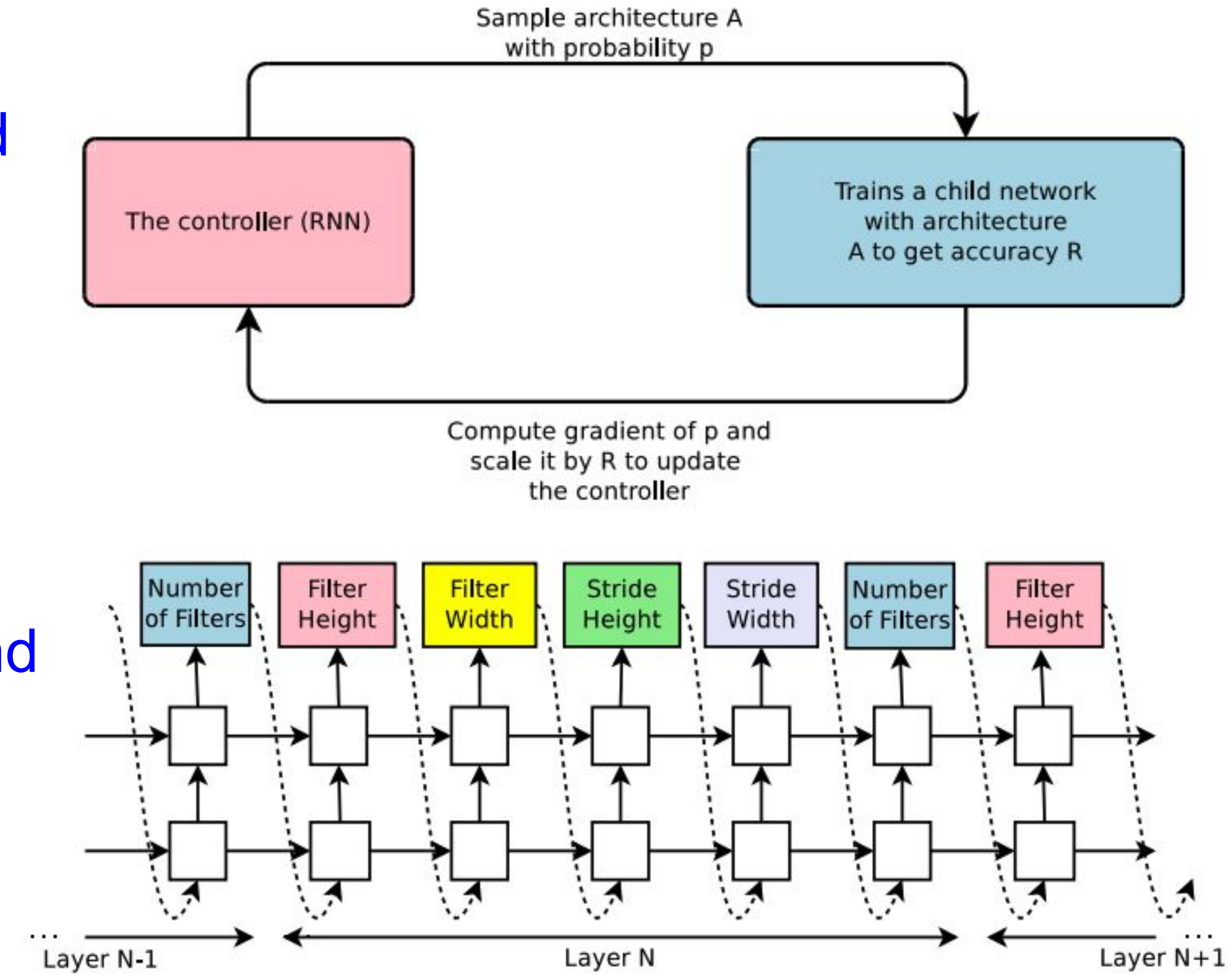
slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Meta-Learning: Learning to learn network architectures

Neural Architecture Search with Reinforcement Learning (NAS)

[Zoph et al. 2016]

- “Controller” network that learns to design a good network architecture (output a string corresponding to network design)
- Iterate:
  1) Sample an architecture from search space
  2) Train the architecture to get a “reward” R corresponding to accuracy
  3) Compute gradient of sample probability, and scale by R to perform controller parameter update (i.e. increase likelihood of good architecture being sampled, decrease likelihood of bad architecture)



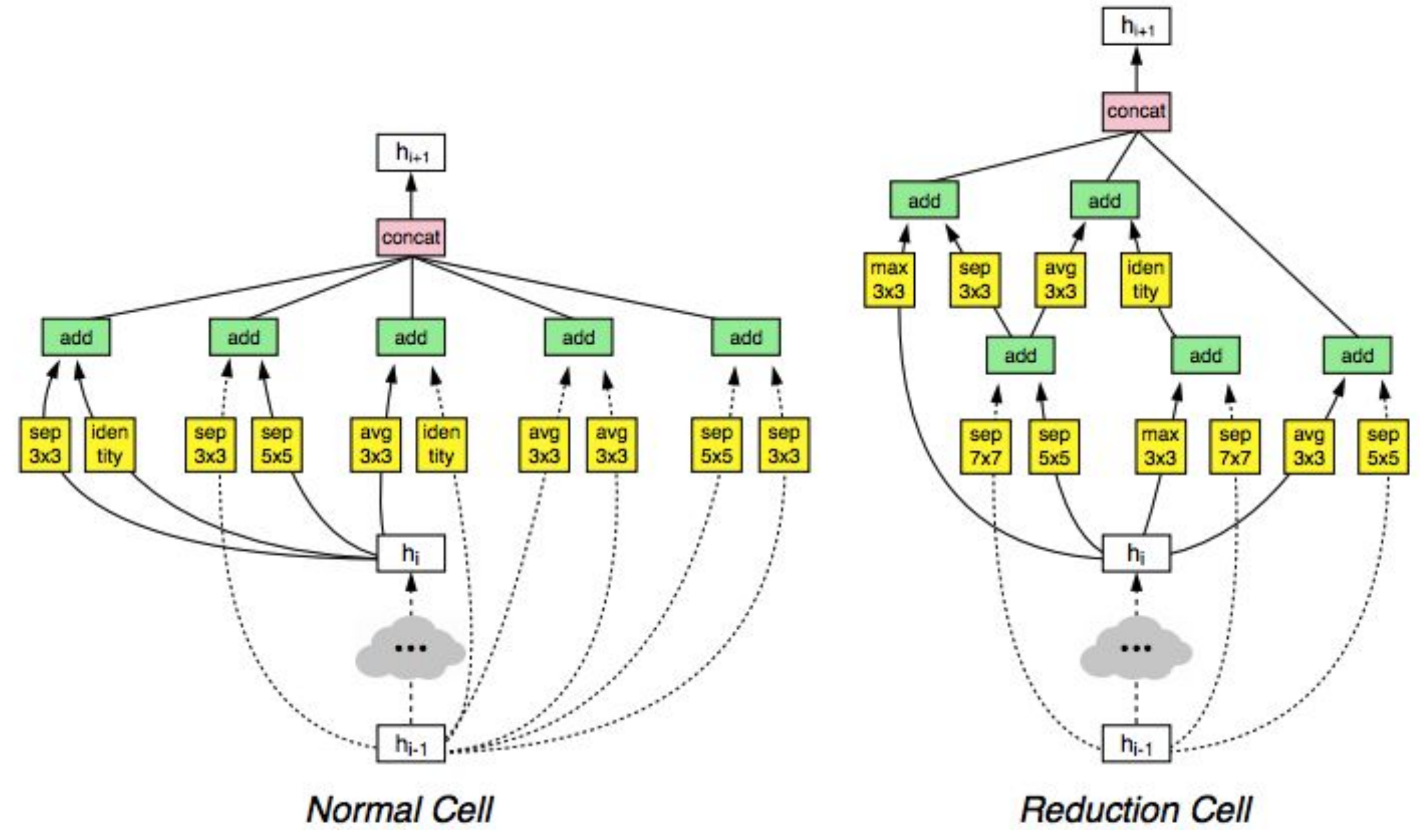
slide credit: Fei-Fei, Justin Johnson, Serena Yeung

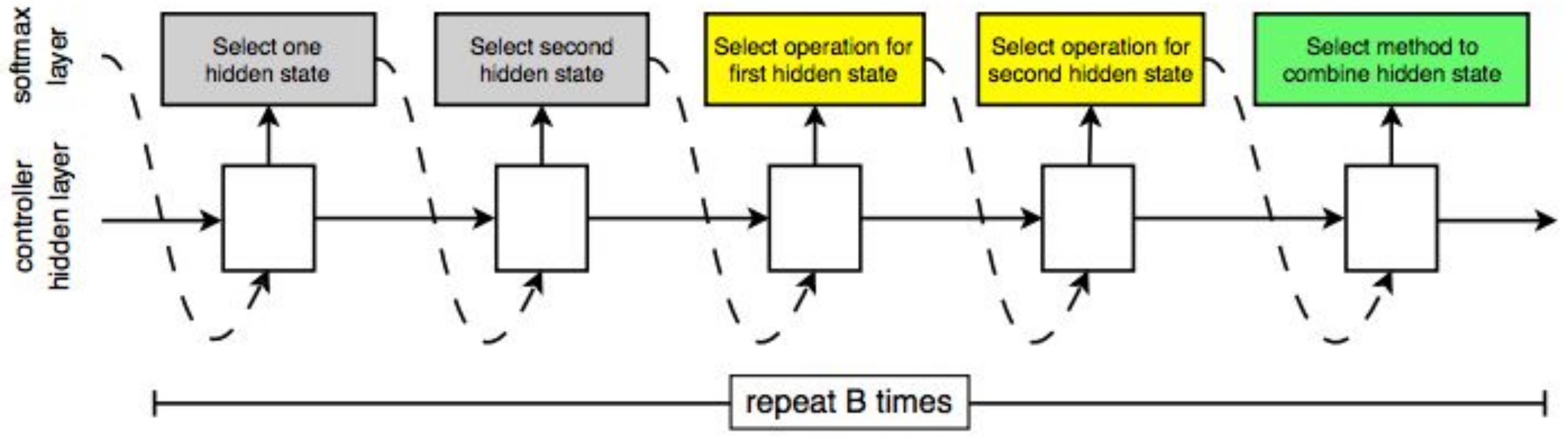
Meta-Learning: Learning to learn network architectures

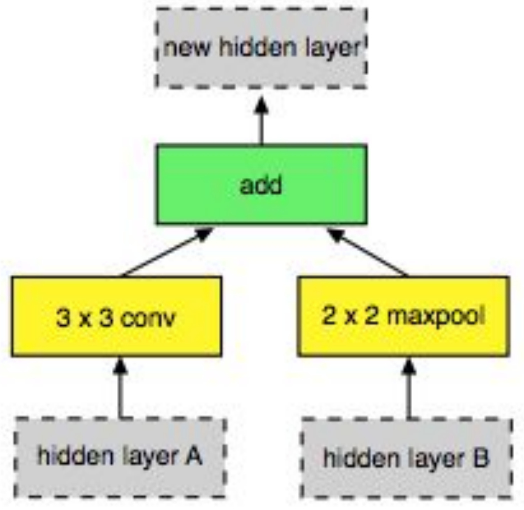
Learning Transferable Architectures for Scalable Image Recognition

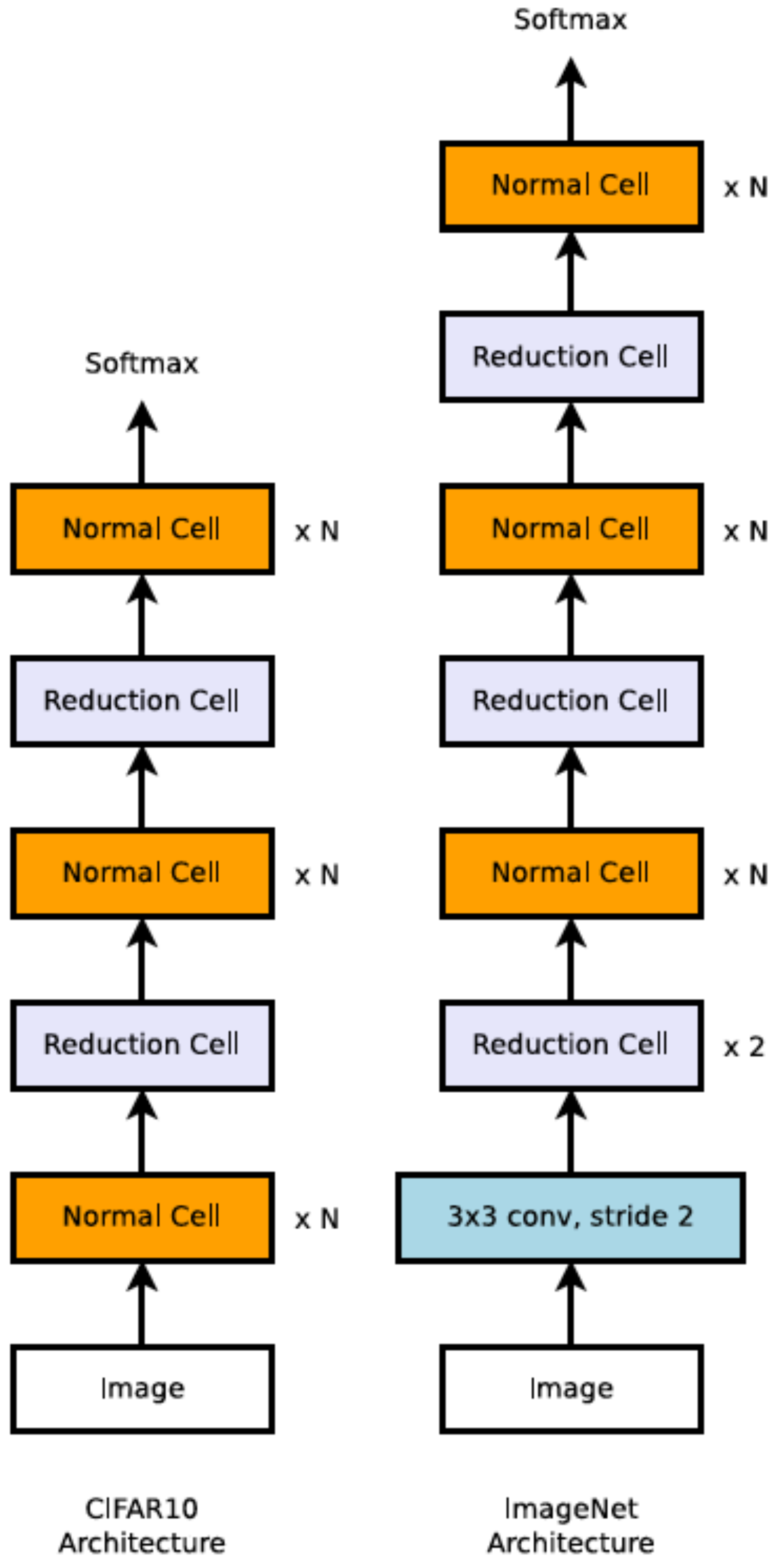
[Zoph et al. 2017]

- Applying neural architecture search (NAS) to a large dataset like ImageNet is expensive
- Design a search space of building blocks ("cells") that can be flexibly stacked
- NASNet: Use NAS to find best cell structure on smaller CIFAR-10 dataset, then transfer architecture to ImageNet









slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Overview Today's Lecture

- Convolutional Neural Network (CNN) Architectures
  - case studies
    - AlexNet,
    - VGG
    - GoogLeNet
    - ResNet
  - other architectures to know (briefly discussed today)
    - NiN (Network in Network), Wide ResNet, ResNeXT, Stochastic Depth, Squeeze-and-Excitation Network, DenseNet, NASNet

slide credit: Fei-Fei, Justin Johnson, Serena Yeung