Indian Institute of Information Technology, Allahabad

CNN Architectures for Classification

By

Dr. Satish Kumar Singh & Dr. Shiv Ram Dubey
Computer Vision and Biometrics Lab
Department of Information Technology
Indian Institute of Information Technology, Allahabad

TEAM

Computer Vision and Biometrics Lab (CVBL)

Department of Information Technology

Indian Institute of Information Technology Allahabad

Course Instructors

Dr. Satish Kumar Singh, Associate Professor, IIIT Allahabad (Email: sk.singh@iiita.ac.in)

Dr. Shiv Ram Dubey, Assistant Professor, IIIT Allahabad (Email: srdubey@iiita.ac.in)

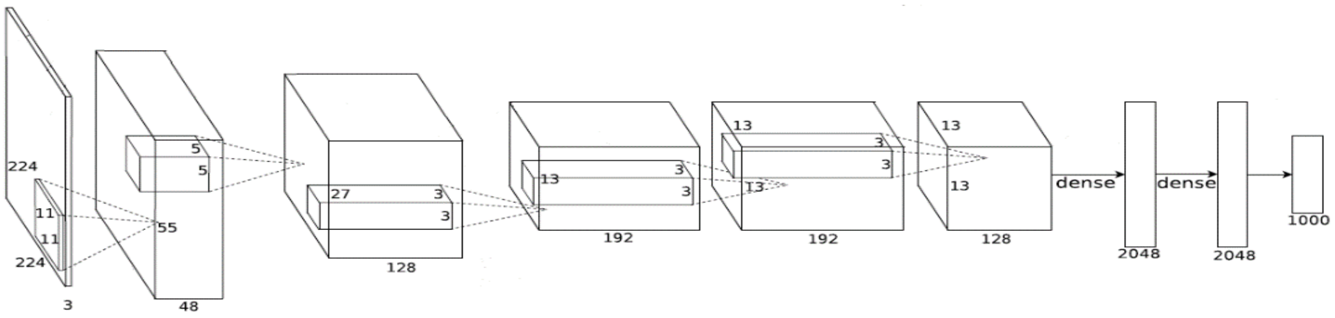
DISCLAIMER

The content (text, image, and graphics) used in this slide are adopted from many sources for academic purposes. Broadly, the sources have been given due credit appropriately. However, there is a chance of missing out some original primary sources. The authors of this material do not claim any copyright of such material.

CNN ARCHITECTURES FOR CLASSIFICATION

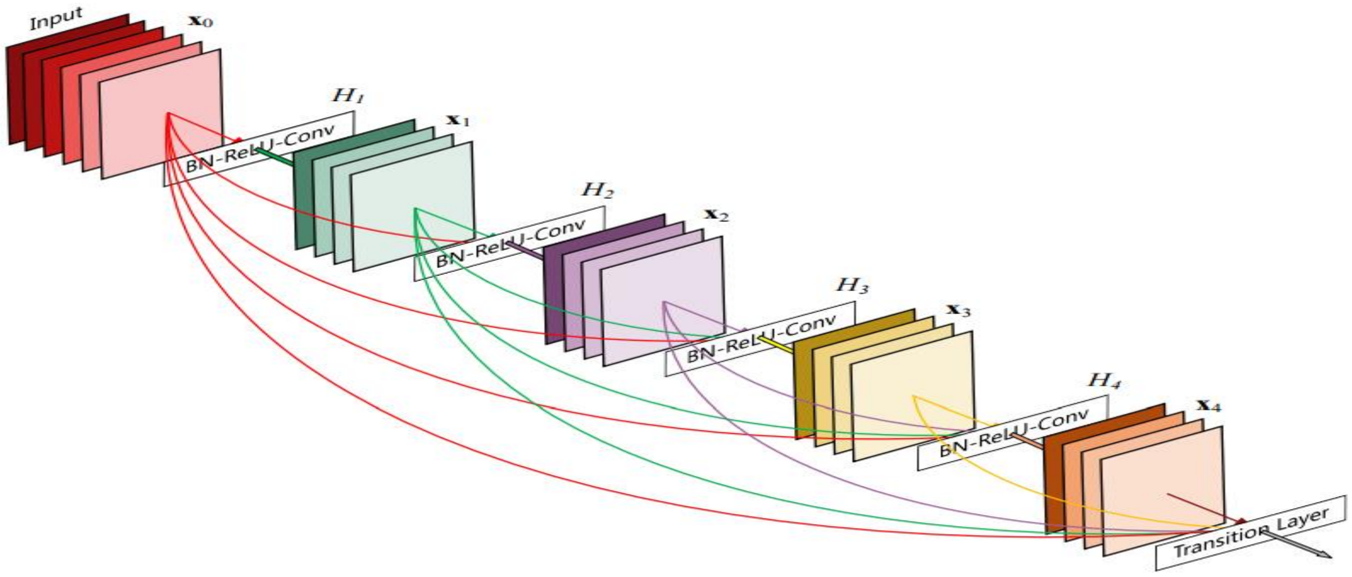
CNN Architectures: Plain Models

- LeNet
- AlexNet
- ZFNet
- VggNet
- Network in Network



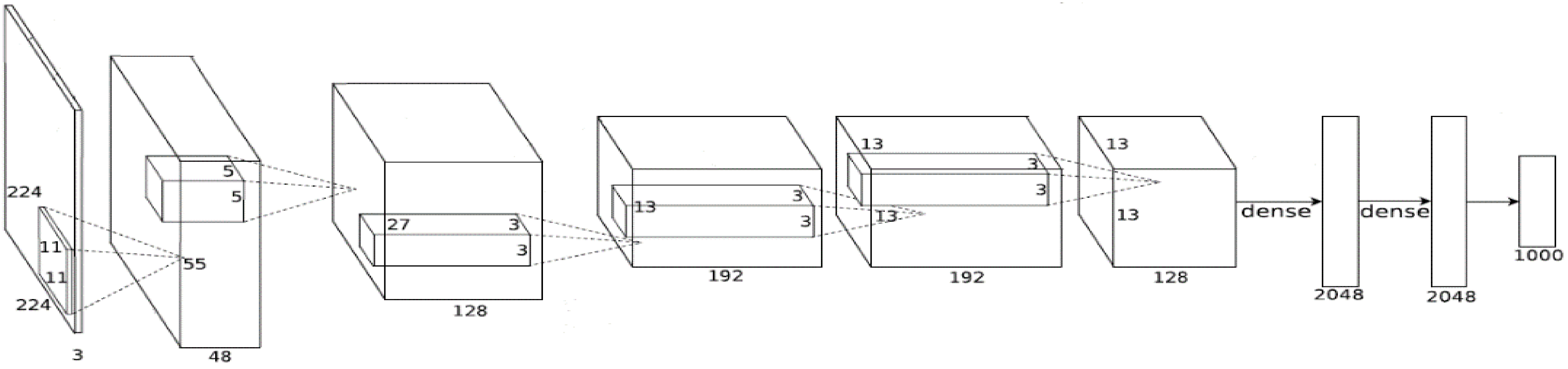
CNN Architectures: DAG Models

- GoogLeNet
- ResNet
- Pre-act ResNet
- SENet
- DenseNet
- ResNetXt
- Etc.

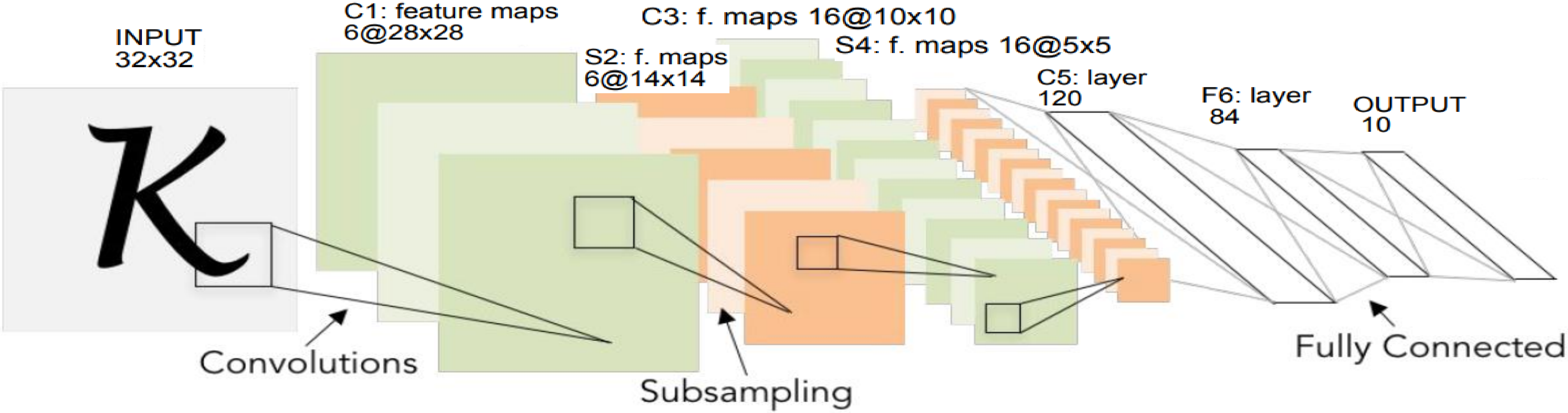


CNN Architectures: Plain Models

- LeNet
- AlexNet
- ZFNet
- VggNet
- Network in Network



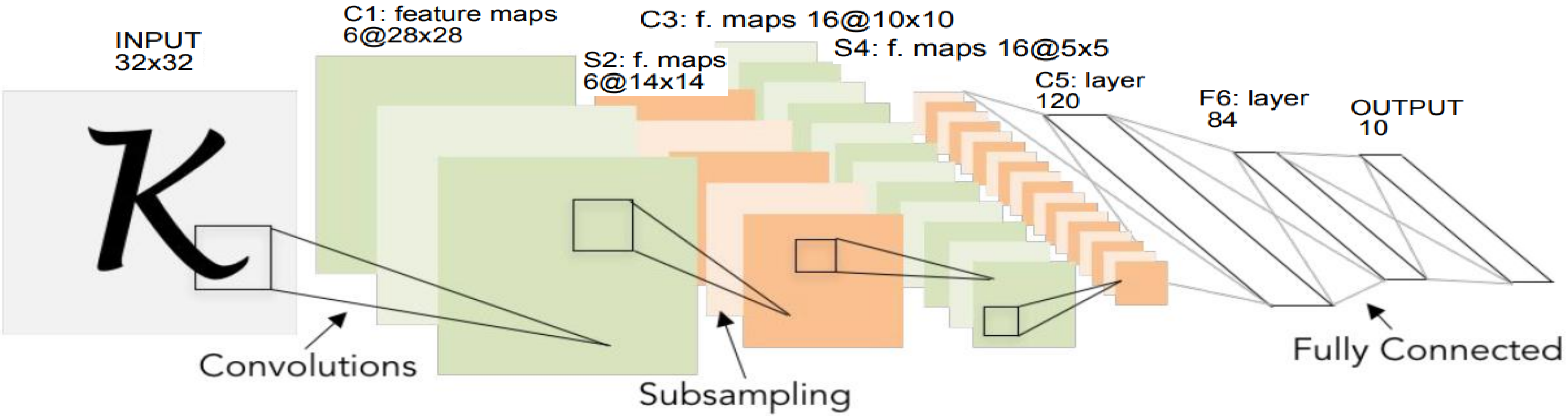
REVIEW: LENET-5



LeCun et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *1998*.

Source: cs231n

REVIEW: LENET-5

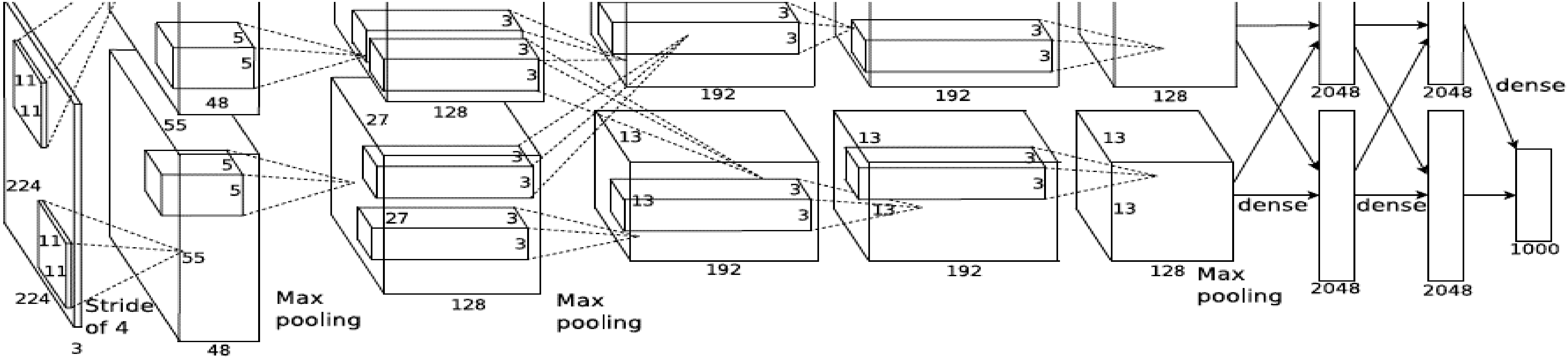


Conv filters are 5x5, applied at stride 1
Subsampling (Pooling) layers are 2x2 applied at stride 2
i.e. architecture is [CONV-POOL-CONV-POOL-CONV-FC-FC]

LeCun et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *1998*.

Source: cs231n

ALEXNET



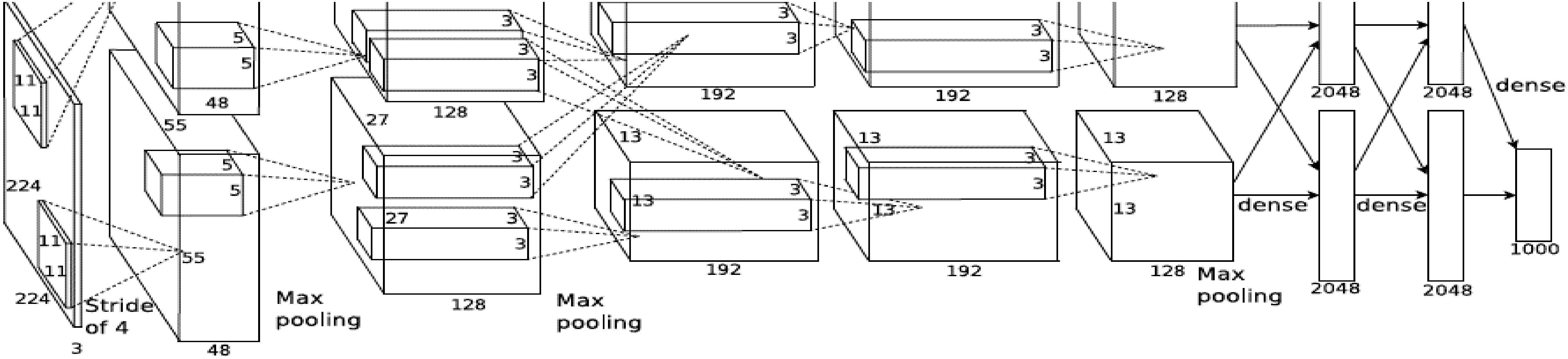
Architecture:

CONV1 MAX POOL1 NORM1(Local Response Normalization)
CONV2 MAX POOL2 NORM2(Local Response Normalization)
CONV3
CONV4
CONV5 Max POOL3
FC6
FC7
FC8

Krizhevsky, et al. Imagenet classification with deep convolutional neural networks. *NIPS 2012*.

Source: cs231n

ALEXNET



Input: 227x227x3 images

First layer (CONV1): 96 11x11 filters applied at stride 4

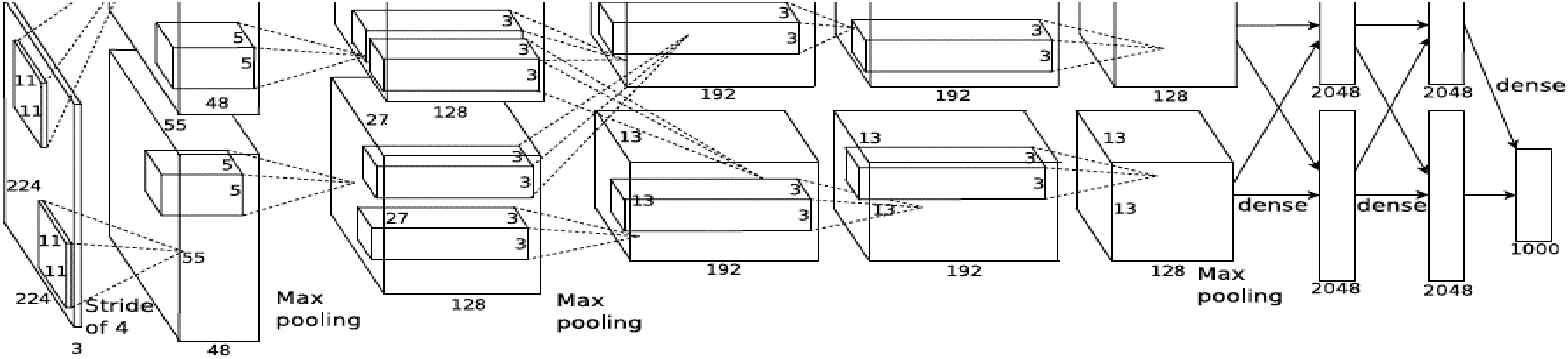
=>

Q: what is the output volume size? Hint: (227-11)/4+1 = 55

Krizhevsky, et al. Imagenet classification with deep convolutional neural networks. *NIPS 2012*.

Source: cs231n

ALEXNET



Input: 227x227x3 images

First layer (CONV1): 96 11x11 filters applied at stride 4

=>

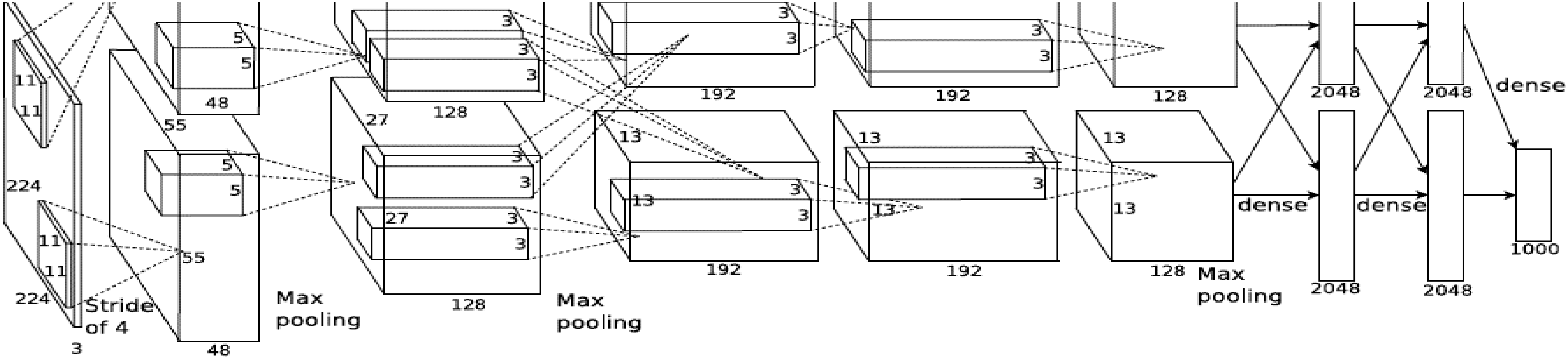
Output volume **[55x55x96]**

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

Krizhevsky, et al. Imagenet classification with deep convolutional neural networks. *NIPS 2012*.

Source: cs231n

ALEXNET



Input: 227x227x3 images

First layer (CONV1): 96 11x11 filters applied at stride 4

=>

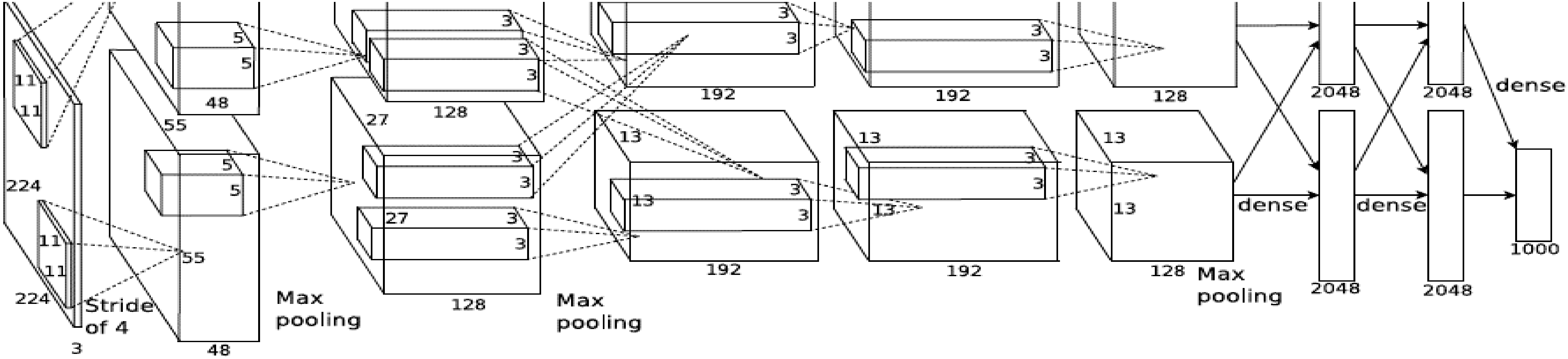
Output volume **[55x55x96]**

Parameters: (11*11*3)*96 = **35K**

Krizhevsky, et al. Imagenet classification with deep convolutional neural networks. *NIPS 2012*.

Source: cs231n

ALEXNET



Input: 227x227x3 images

After CONV1: 55x55x96

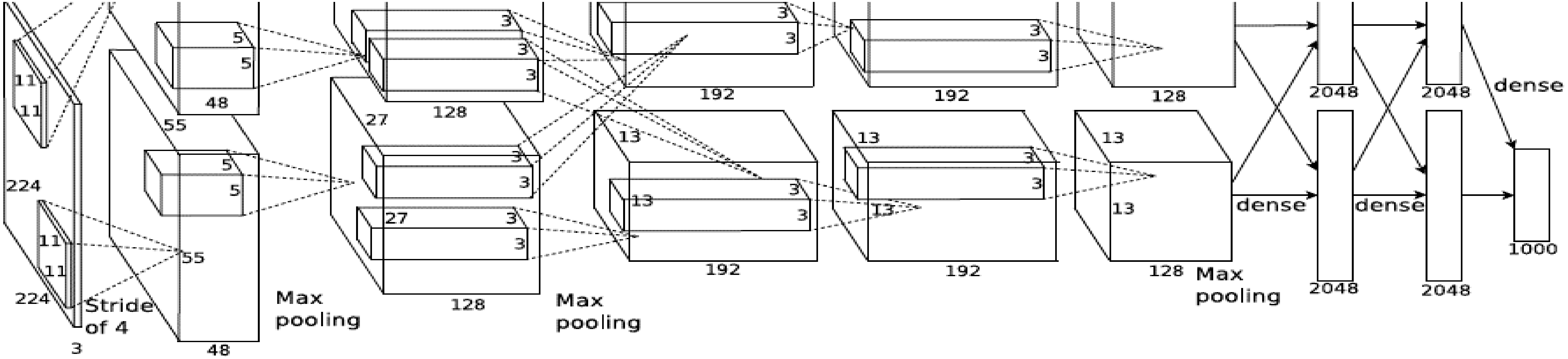
Second layer (POOL1): 3x3 filters applied at stride 2

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

Krizhevsky, et al. Imagenet classification with deep convolutional neural networks. *NIPS 2012*.

Source: cs231n

ALEXNET



Input: 227x227x3 images

After CONV1: 55x55x96

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

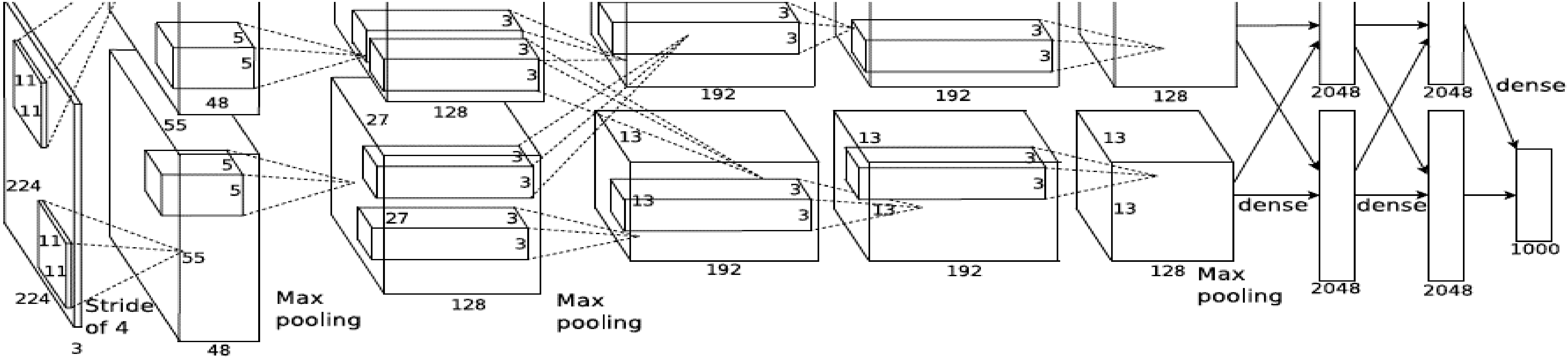
Output volume **[27x27x96]**

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

Krizhevsky, et al. Imagenet classification with deep convolutional neural networks. *NIPS 2012*.

Source: cs231n

ALEXNET

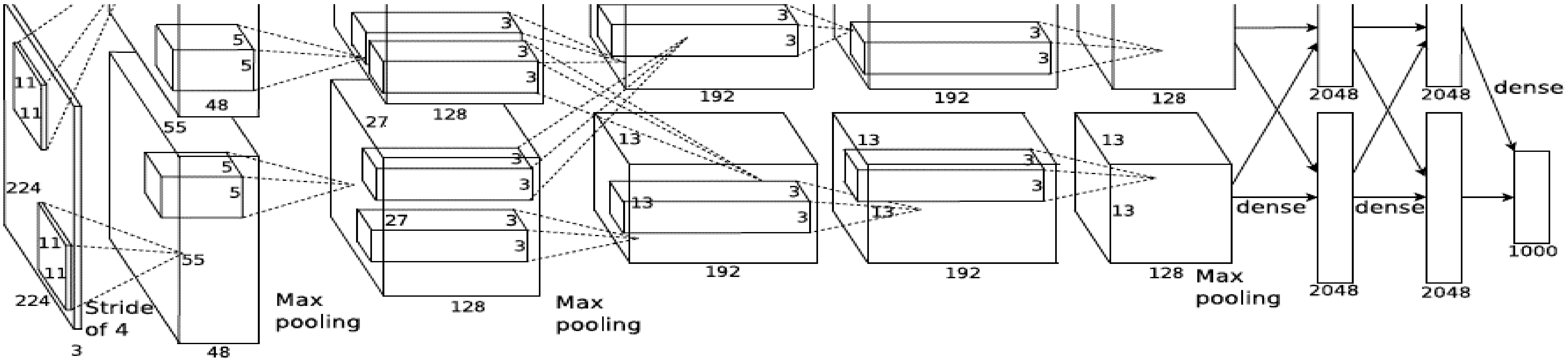


Input: 227x227x3 images
After CONV1: 55x55x96
Second layer (POOL1): 3x3 filters applied at stride 2
Output volume **[27x27x96]**
Parameters: 0!

Krizhevsky, et al. Imagenet classification with deep convolutional neural networks. *NIPS 2012*.

Source: cs231n

ALEXNET



Input: 227x227x3 images

After CONV1: 55x55x96

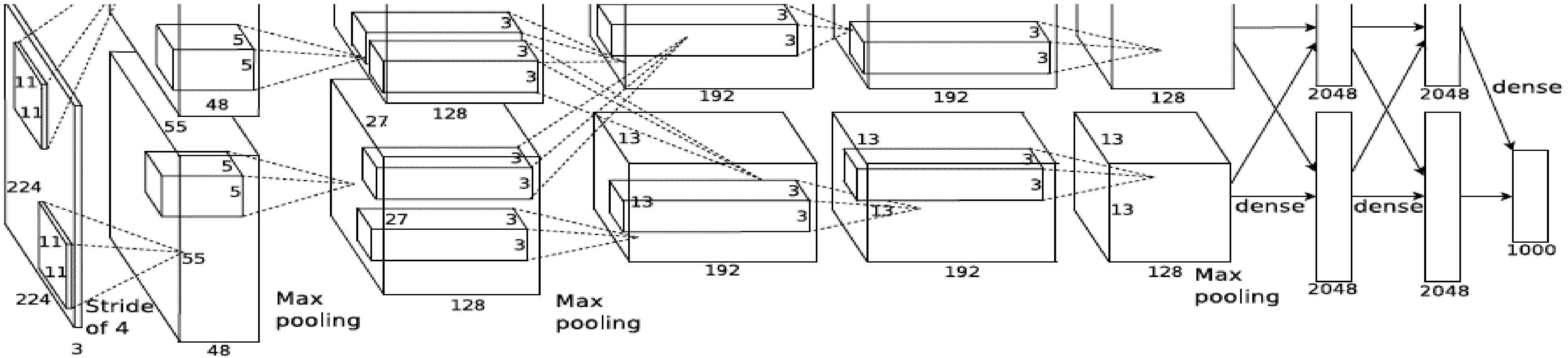
After POOL1: 27x27x96

...

Krizhevsky, et al. Imagenet classification with deep convolutional neural networks. *NIPS 2012*.

Source: cs231n

ALEXNET

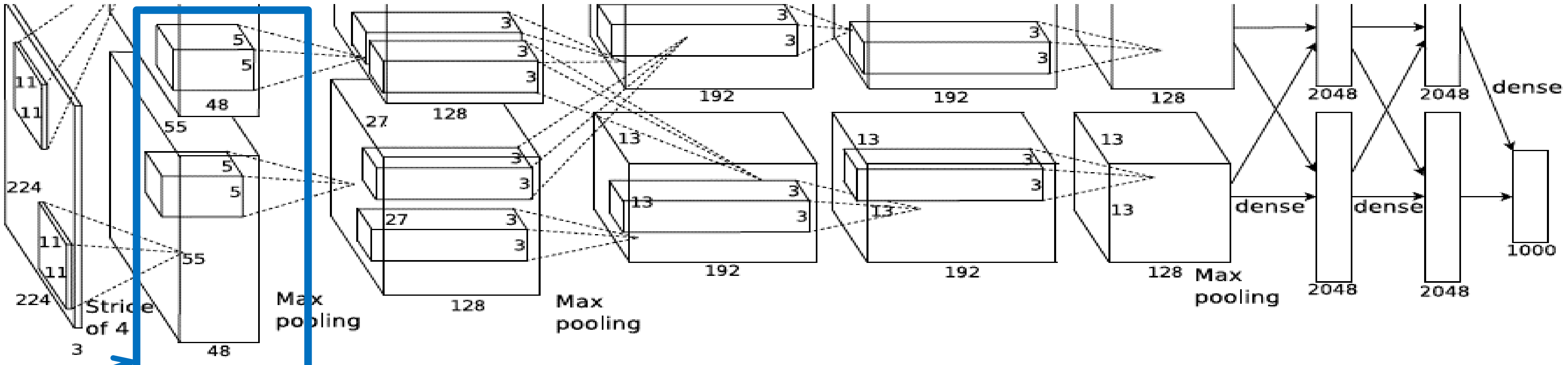


Full (simplified) AlexNet architecture:
[227x227x3] INPUT
[55x55x96] CONV1: 96 11x11 filters at stride 4, pad 0
[27x27x96] MAX POOL1: 3x3 filters at stride 2
[27x27x96] NORM1: Normalization layer
[27x27x256] CONV2: 256 5x5 filters at stride 1, pad 2
[13x13x256] MAX POOL2: 3x3 filters at stride 2
[13x13x256]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

Krizhevsky, et al. Imagenet classification with deep convolutional neural networks. *NIPS 2012*.

Source: cs231n

ALEXNET



[55x55x48] x 2

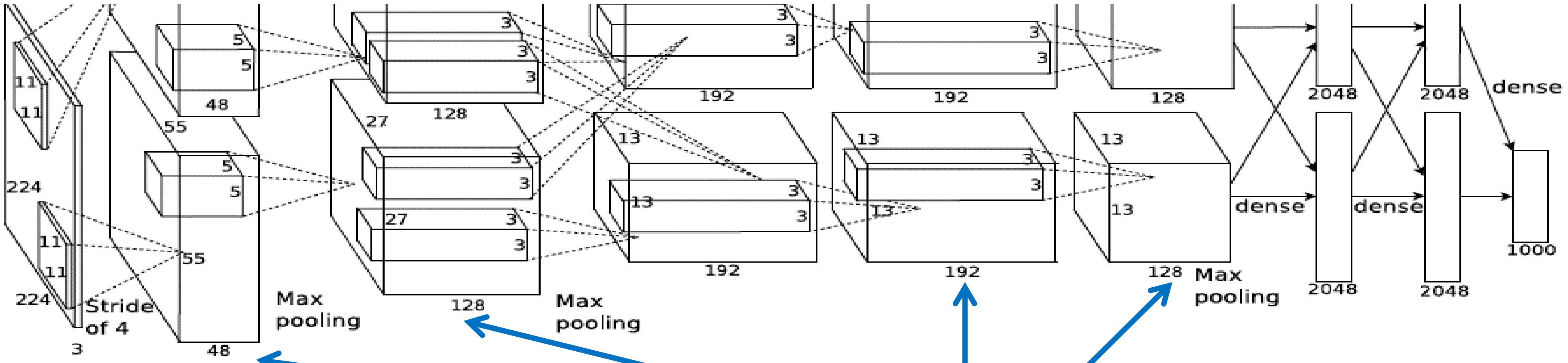
Full (simplified) AlexNet architecture:
[227x227x3] INPUT
[55x55x96] CONV1: 96 11x11 filters at stride 4, pad 0
[27x27x96] MAX POOL1: 3x3 filters at stride 2
[27x27x96] NORM1: Normalization layer
[27x27x256] CONV2: 256 5x5 filters at stride 1, pad 2
[13x13x256] MAX POOL2: 3x3 filters at stride 2
[13x13x256]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

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.

Krizhevsky, et al. Imagenet classification with deep convolutional neural networks. *NIPS 2012*.

Source: cs231n

ALEXNET



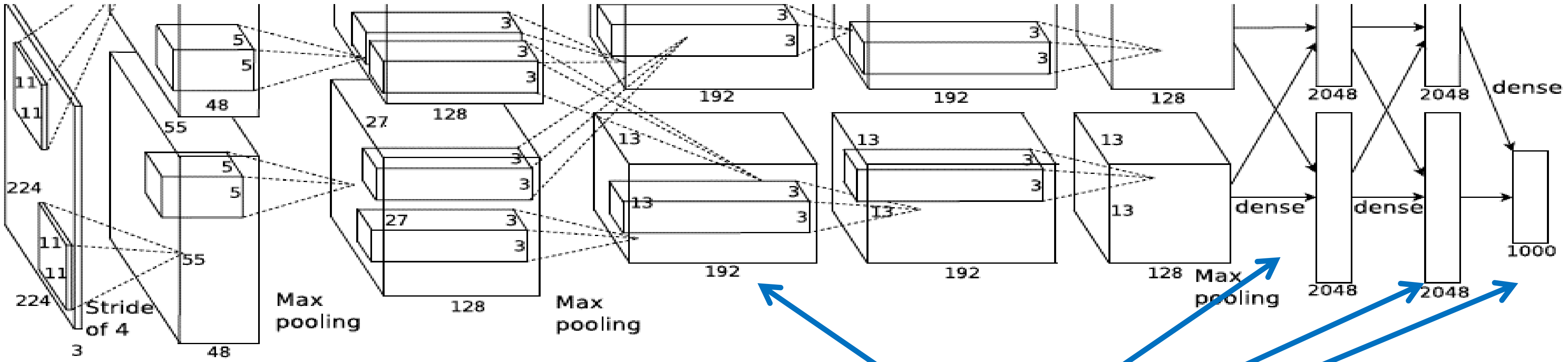
Full (simplified) AlexNet architecture:
[227x227x3] INPUT
[55x55x96] CONV1: 96 11x11 filters at stride 4, pad 0
[27x27x96] MAX POOL1: 3x3 filters at stride 2
[27x27x96] NORM1: Normalization layer
[27x27x256] CONV2: 256 5x5 filters at stride 1, pad 2
[13x13x256] MAX POOL2: 3x3 filters at stride 2
[13x13x256]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

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

Krizhevsky, et al. Imagenet classification with deep convolutional neural networks. *NIPS 2012*.

Source: cs231n

ALEXNET



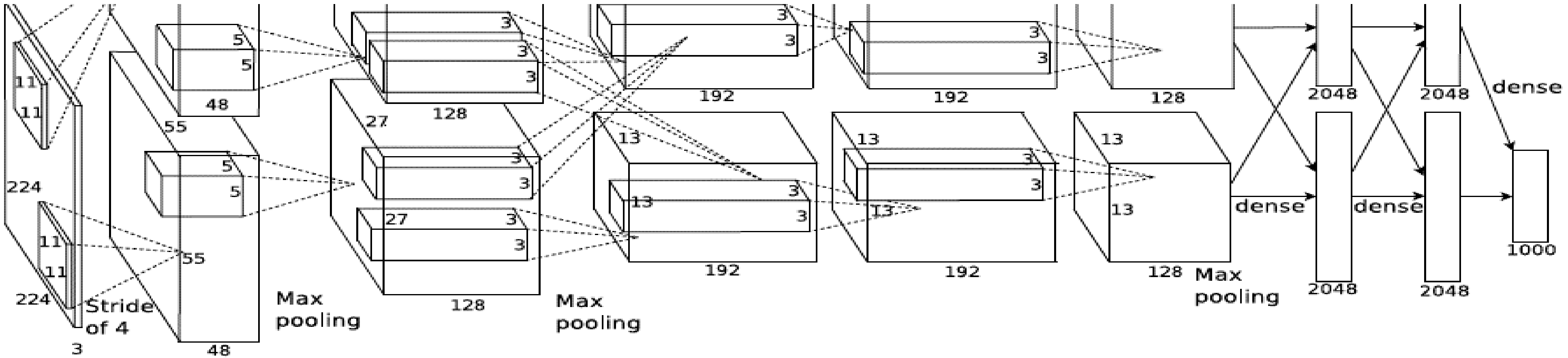
Full (simplified) AlexNet architecture:
[227x227x3] INPUT
[55x55x96] CONV1: 96 11x11 filters at stride 4, pad 0
[27x27x96] MAX POOL1: 3x3 filters at stride 2
[27x27x96] NORM1: Normalization layer
[27x27x256] CONV2: 256 5x5 filters at stride 1, pad 2
[13x13x256] MAX POOL2: 3x3 filters at stride 2
[13x13x256]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

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

Krizhevsky, et al. Imagenet classification with deep convolutional neural networks. *NIPS 2012*.

Source: cs231n

ALEXNET



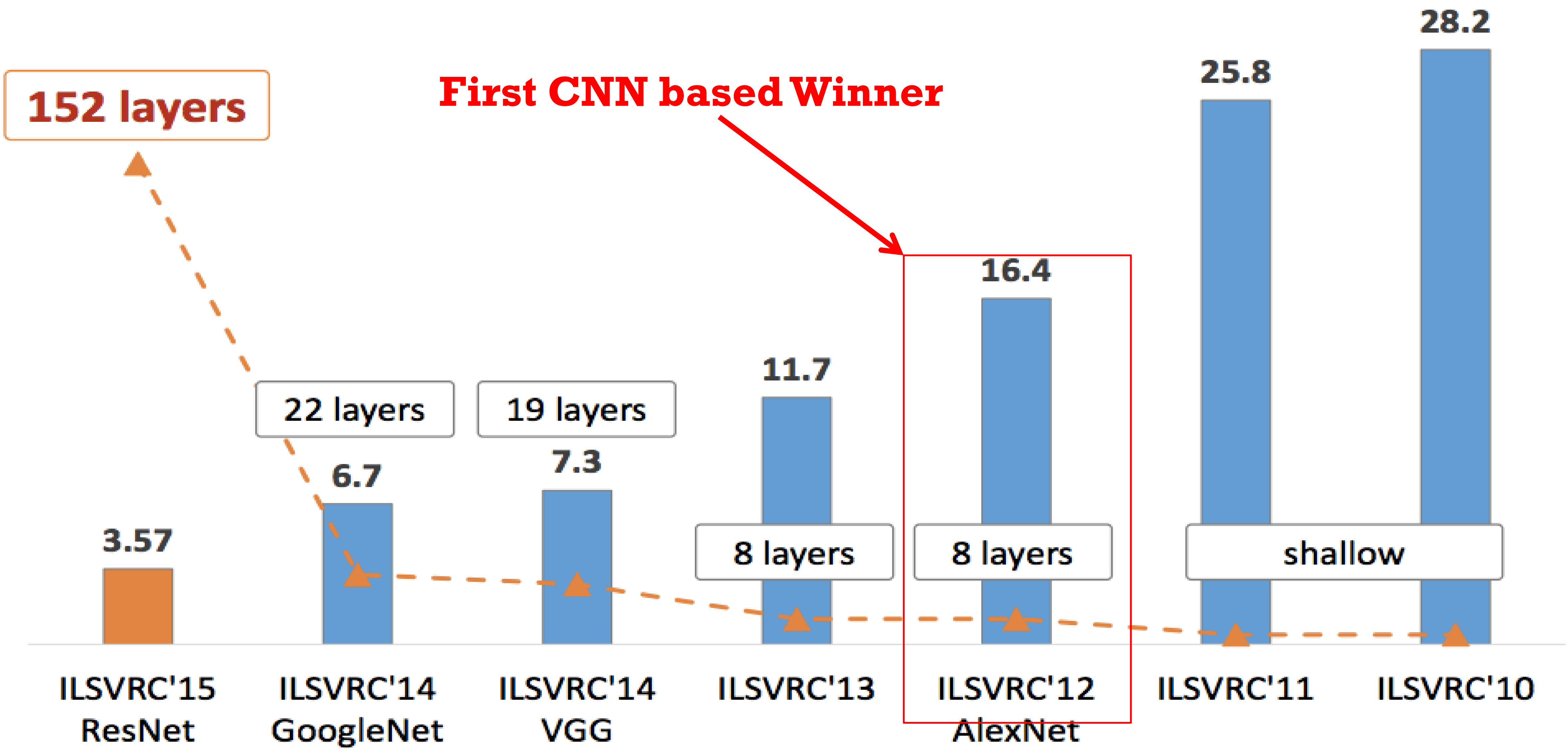
Full (simplified) AlexNet architecture:
[227x227x3] INPUT
[55x55x96] CONV1: 96 11x11 filters at stride 4, pad 0
[27x27x96] MAX POOL1: 3x3 filters at stride 2
[27x27x96] NORM1: Normalization layer
[27x27x256] CONV2: 256 5x5 filters at stride 1, pad 2
[13x13x256] MAX POOL2: 3x3 filters at stride 2
[13x13x256]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

Details/Retrospectives:
- first use of ReLU
- used Norm layers (not common anymore)
- heavy data augmentation
- batch size 128
- SGD Momentum 0.9
- Learning rate 0.01, reduced manually when val accuracy saturates

Krizhevsky, et al. Imagenet classification with deep convolutional neural networks. *NIPS 2012*.

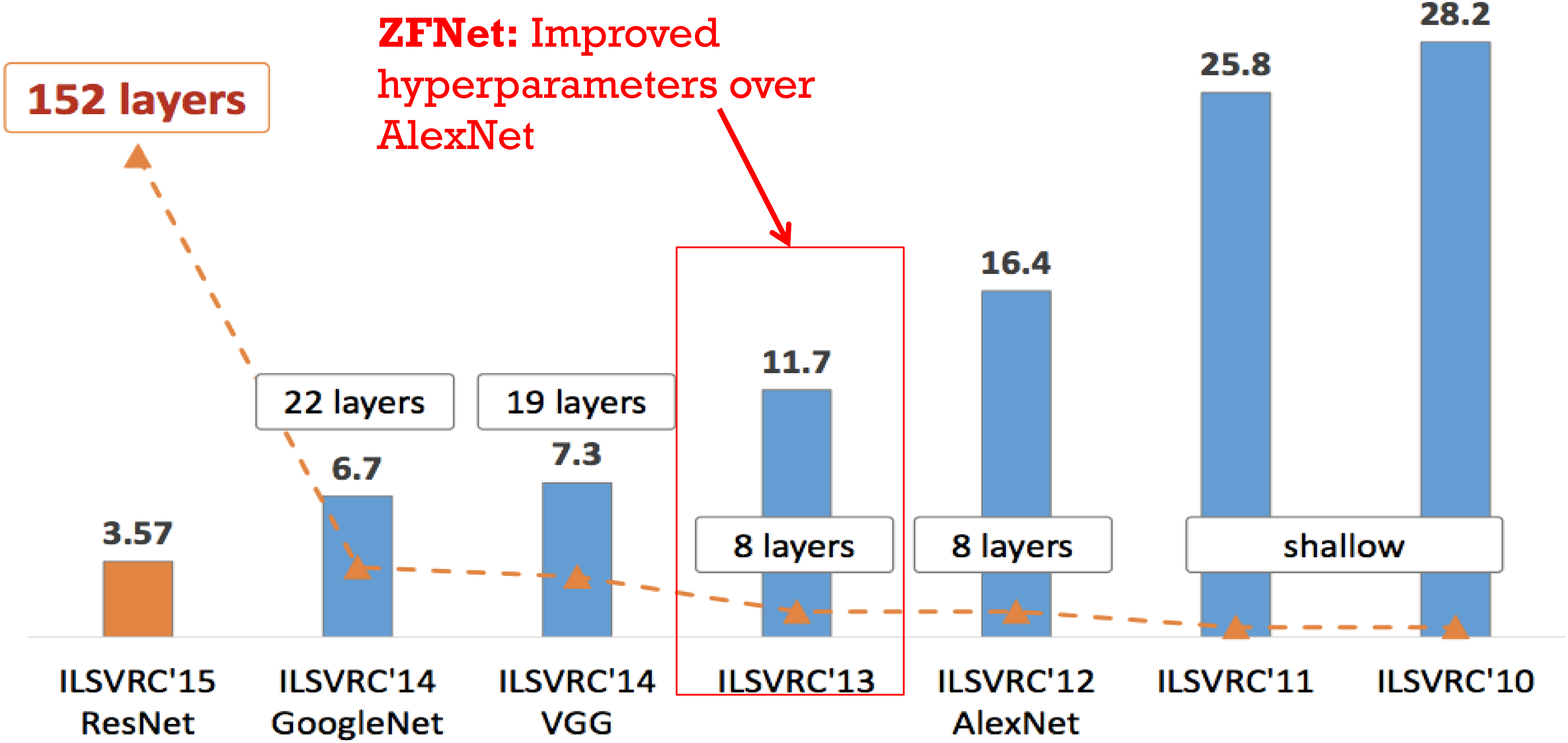
Source: cs231n

IMAGENET LARGE SCALE VISUAL RECOGNITION CHALLENGE (ILSVRC) WINNERS



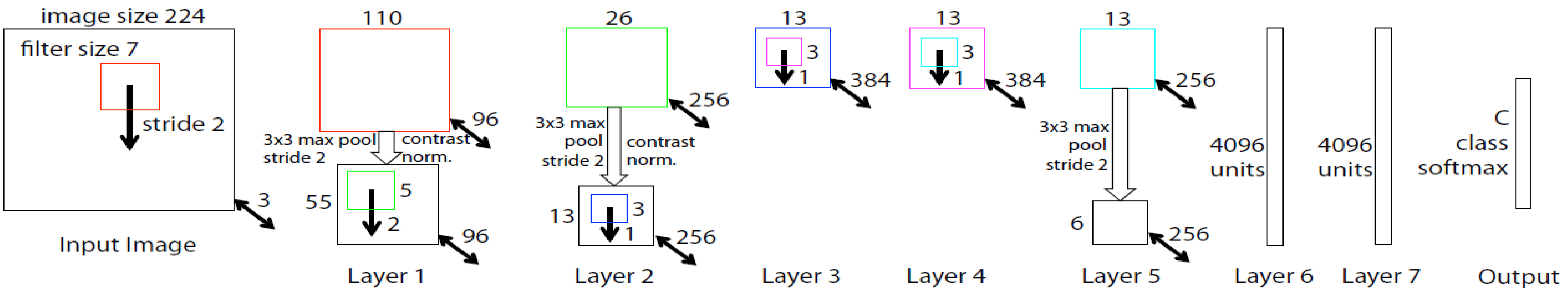
K. He, X. Zhang, S. Ren, J. Sun, [Deep Residual Learning for Image Recognition](), IEEE CVPR 2016.

IMAGENET LARGE SCALE VISUAL RECOGNITION CHALLENGE (ILSVRC) WINNERS



K. He, X. Zhang, S. Ren, J. Sun, Deep Residual Learning for Image Recognition, IEEE CVPR 2016.

ZFNET



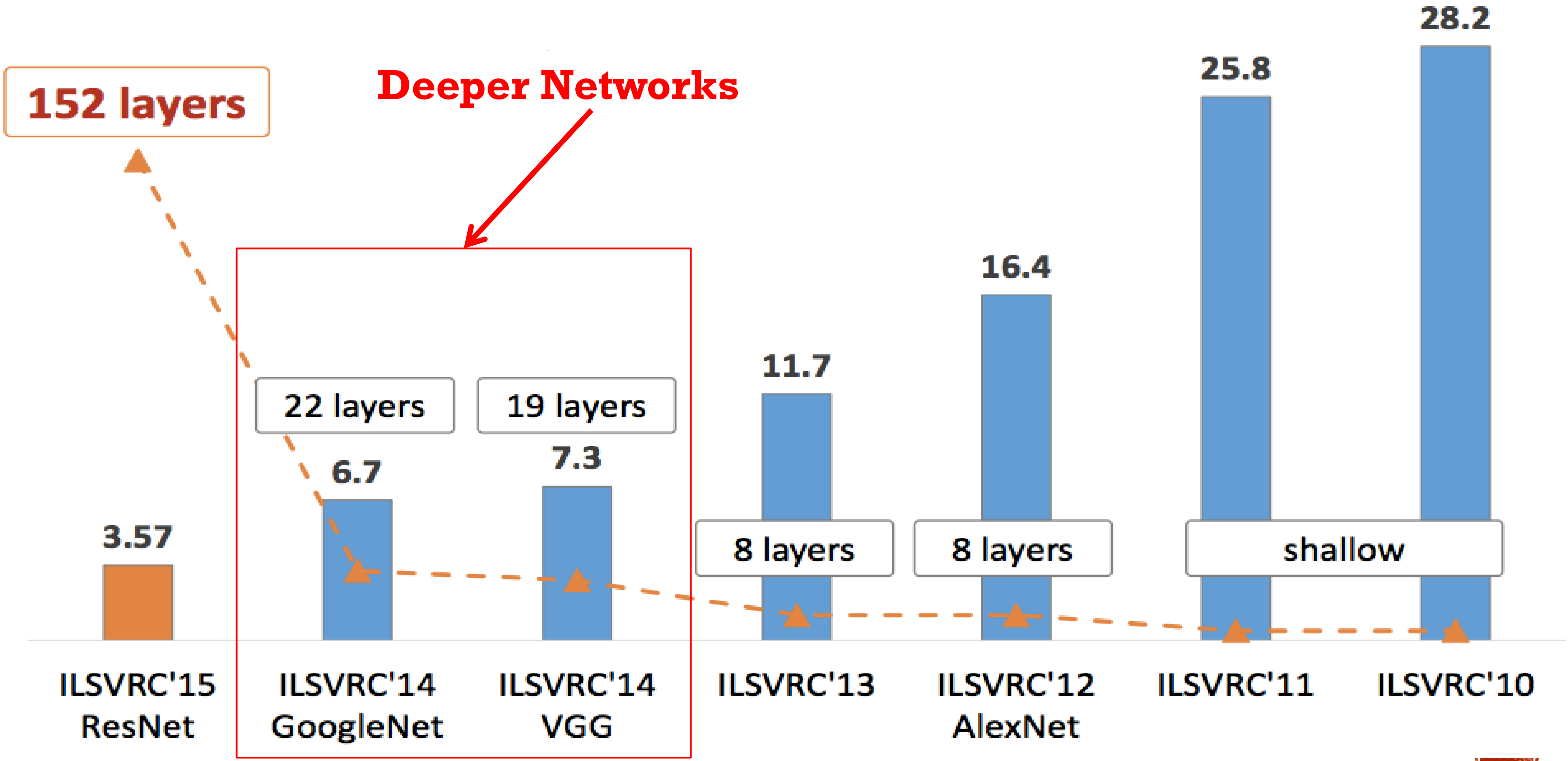
AlexNet but:
CONV1: change from (11x11 stride 4) to (7x7 stride 2)
CONV3,4,5: instead of 384, 384, 256 filters use 512, 1024, 512

ImageNet top 5 error: 16.4% -> 11.7%

M. Zeiler, R. Fergus. Visualizing and understanding convolutional networks. *ECCV* 2014.

Source: cs231n

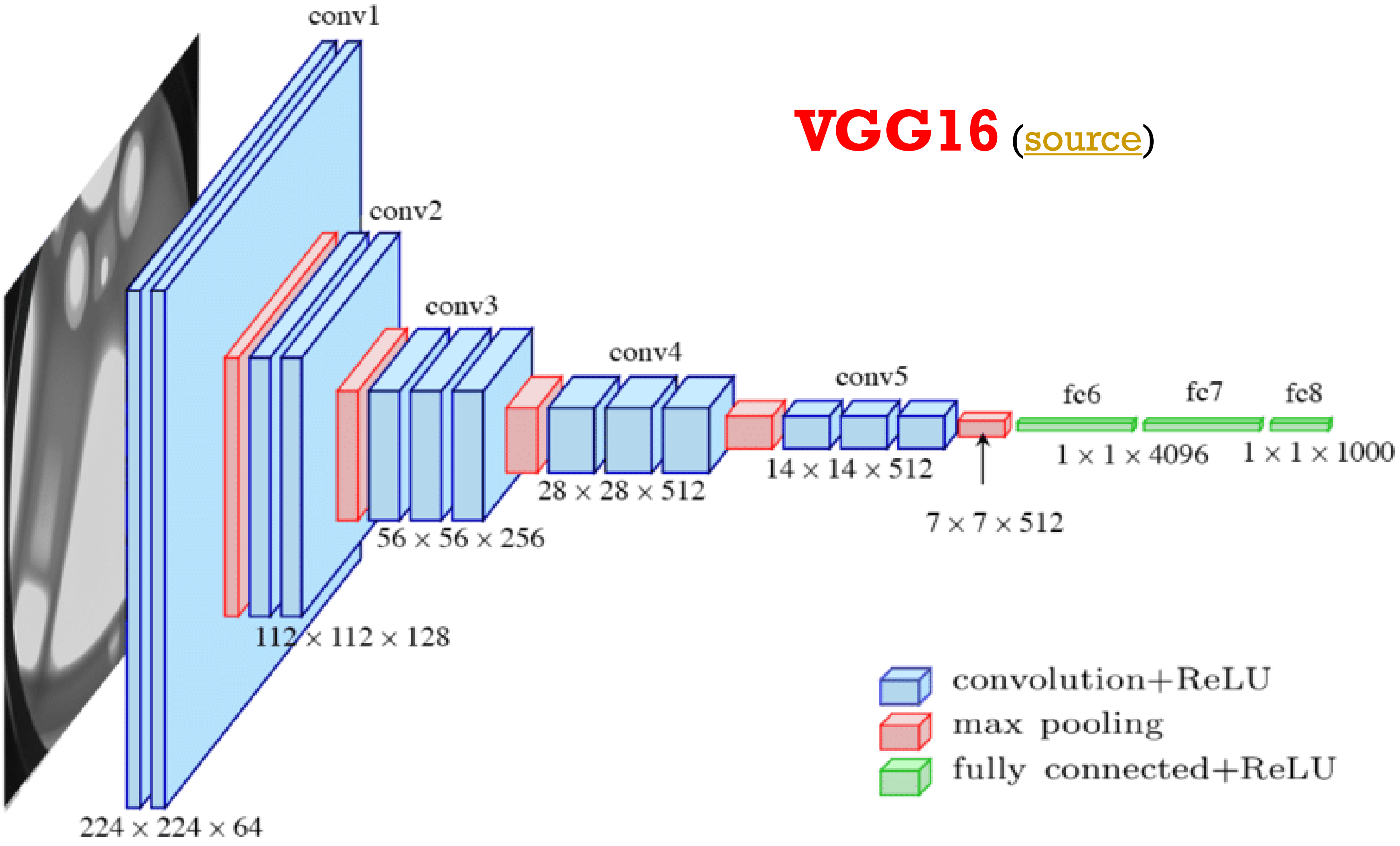
IMAGENET LARGE SCALE VISUAL RECOGNITION CHALLENGE (ILSVRC) WINNERS



K. He, X. Zhang, S. Ren, J. Sun, Deep Residual Learning for Image Recognition, IEEE CVPR 2016.

VGGNET

VGG16 (source)



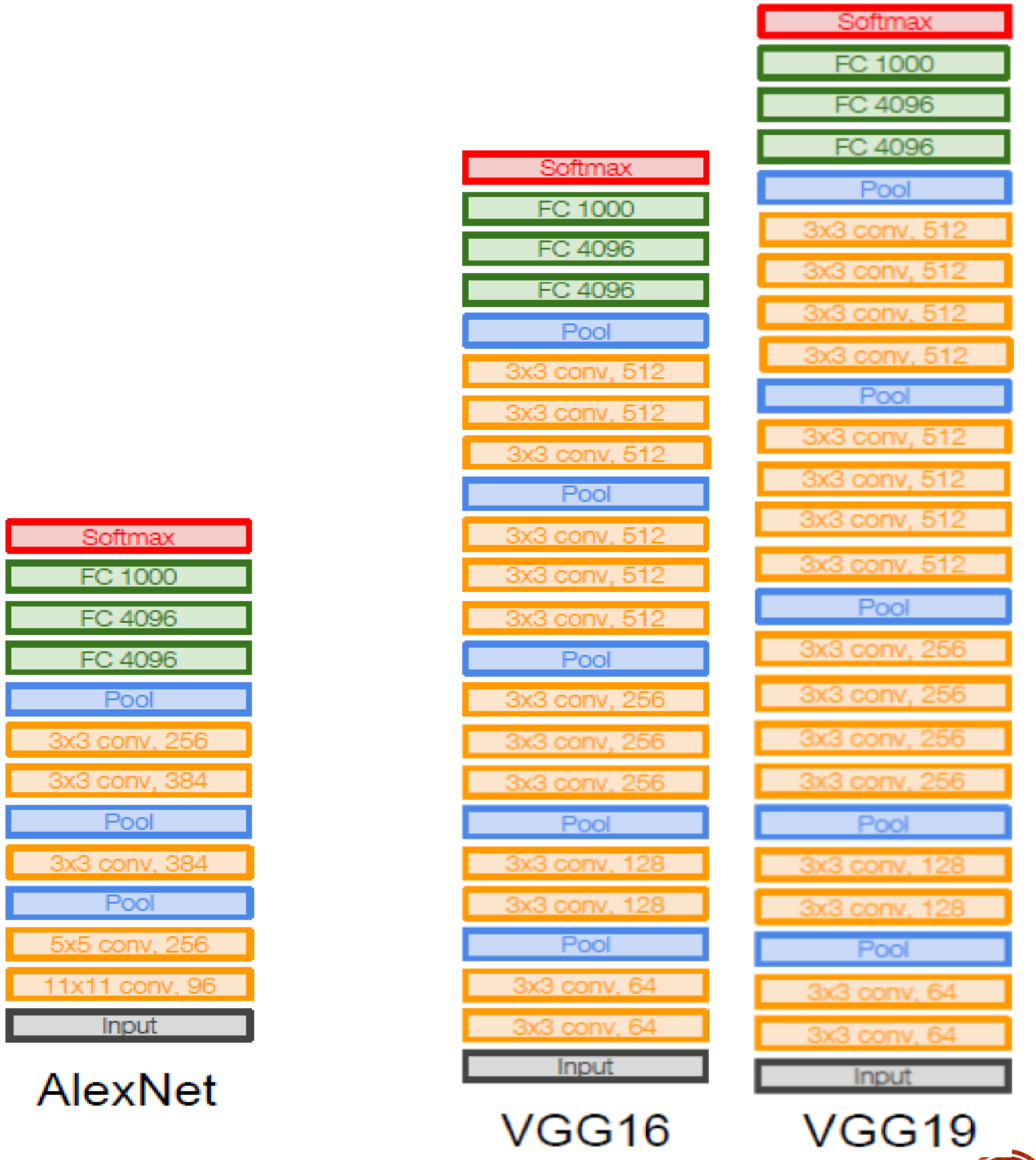
Simonyan et al. Very deep convolutional networks for large-scale image recognition. ICLR2015.

VGGNET

Small filters, Deeper networks

8 layers (AlexNet)
-> 16 - 19 layers (VGGNet)

Only 3x3 CONV stride 1, pad 1
and 2x2 MAX POOL stride 2



Simonyan et al. Very deep convolutional networks for large-scale image recognition. ICLR2015.

Source: cs231n

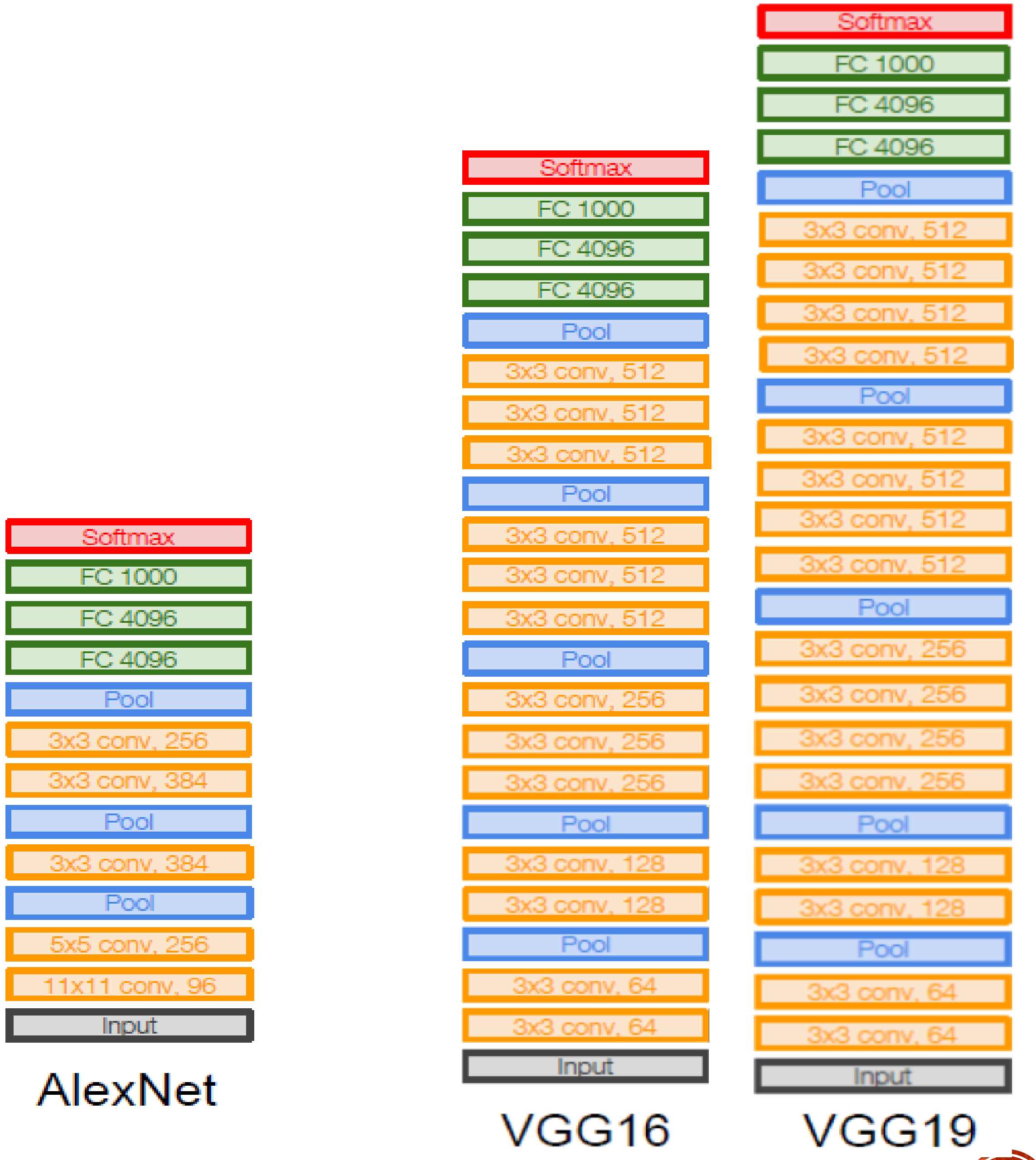
VGGNET

Small filters, Deeper networks

8 layers (AlexNet)
-> 16 - 19 layers (VGGNet)

Only 3x3 CONV stride 1, pad 1
and 2x2 MAX POOL stride 2

ImageNet top 5 error:
11.4% (ZFNet, 2013)
->
7.3% (VGGNet, 2014)



Simonyan et al. Very deep convolutional networks for large-scale image recognition. ICLR2015.

Source: cs231n

VGGNET

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

Simonyan et al. Very deep convolutional networks for large-scale image recognition. ICLR2015.

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

Simonyan et al. Very deep convolutional networks for large-scale image recognition. ICLR2015.

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

Q: What is the effective receptive field of three 3x3 conv (stride 1) layers?

Simonyan et al. Very deep convolutional networks for large-scale image recognition. ICLR2015.

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

Q: What is the effective receptive field of three 3x3 conv (stride 1) layers?

[7x7]

But deeper, more non-linearities

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

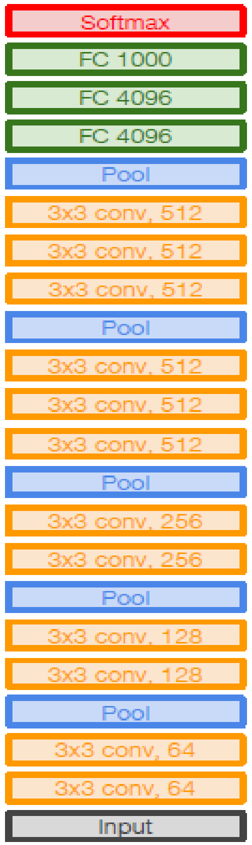
Simonyan et al. Very deep convolutional networks for large-scale image recognition. ICLR2015.

Source: cs231n

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



VGG16

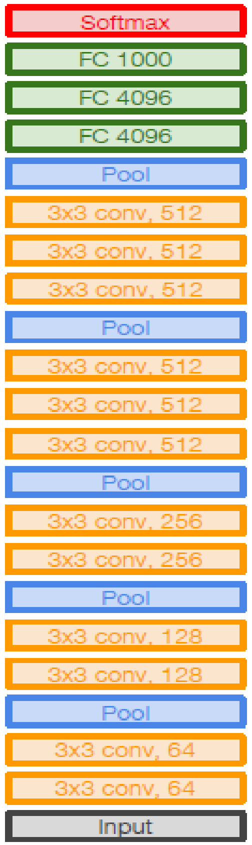
Simonyan et al. Very deep convolutional networks for large-scale image recognition. ICLR2015.

Source: cs231n

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



VGG16

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

Simonyan et al. Very deep convolutional networks for large-scale image recognition. ICLR2015.

Source: cs231n

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

Note:

Most memory is in early CONV

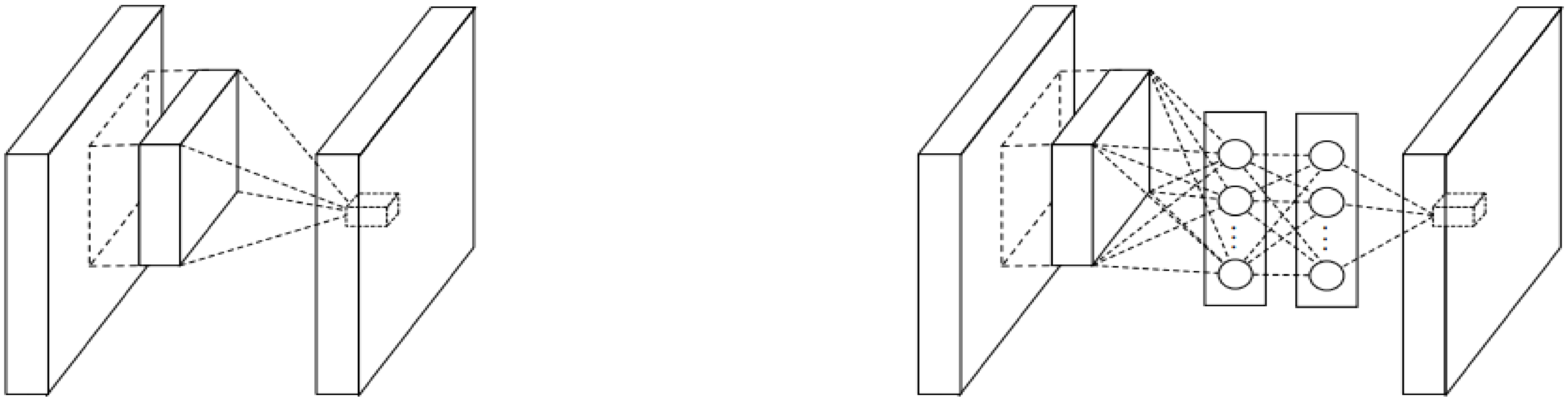
Most params are in late FC

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

Simonyan et al. Very deep convolutional networks for large-scale image recognition. ICLR2015.

Source: cs231n

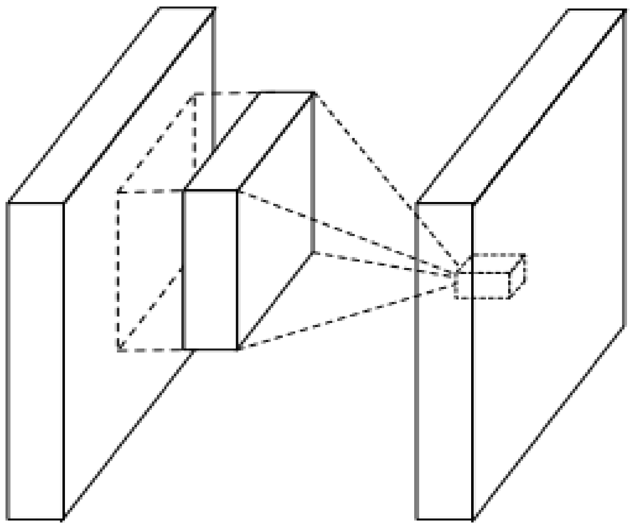
NETWORK IN NETWORK (NIN)



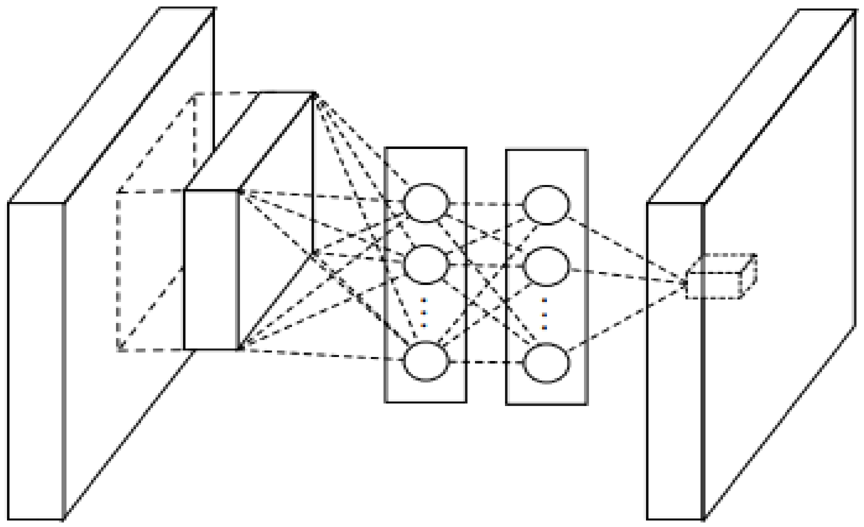
(a) Linear convolution layer (b) Mlpconv layer

Lin et al. Network in Network. ICLR 2014.

NETWORK IN NETWORK (NIN)



(a) Linear convolution layer



(b) Mlpconv layer

- Mlpconv layer with "micronetwork" within each conv layer to compute more abstract features for local patches

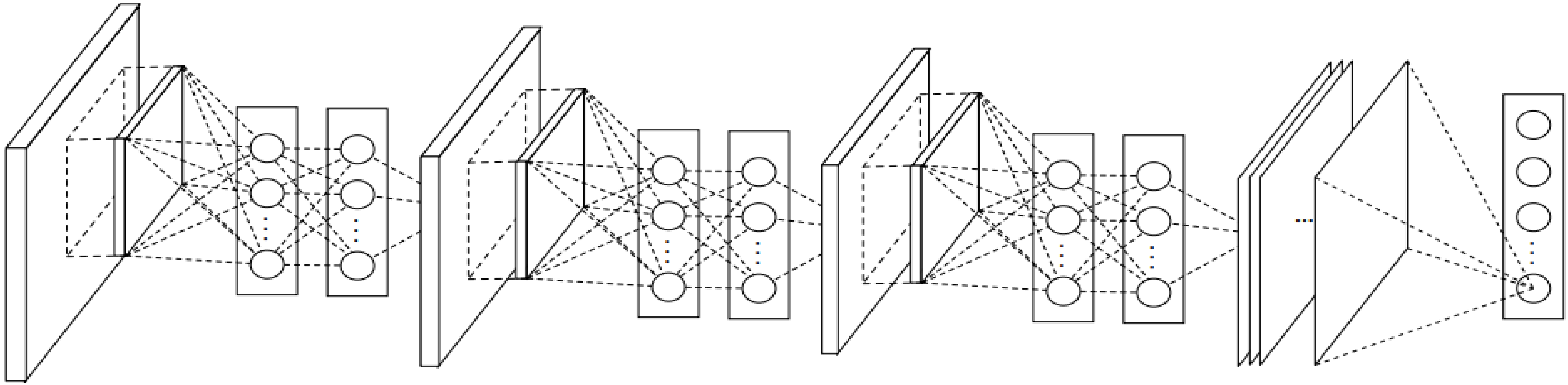
- Micronetwork uses multilayer perceptron (FC, i.e. 1x1 conv layers)

Lin et al. Network in Network. ICLR 2014.

Source: cs231n

NETWORK IN NETWORK (NIN)

The overall structure of NiN: stacking of three mlpconv layers and one global average pooling layer



Lin et al. Network in Network. 2014.

NETWORK IN NETWORK (NIN)

The overall structure of NiN: stacking of three mlpconv layers and one global average pooling layer

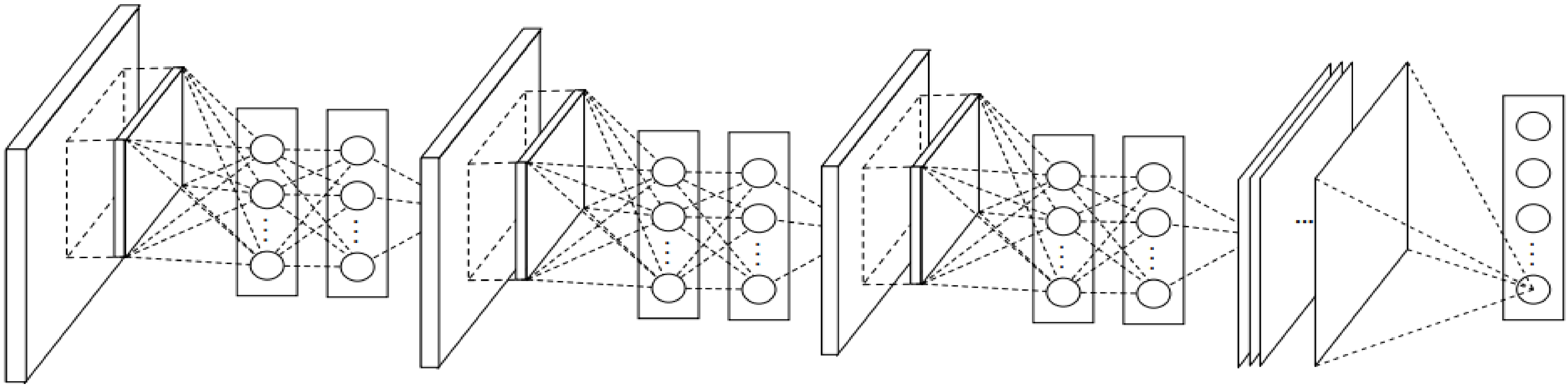


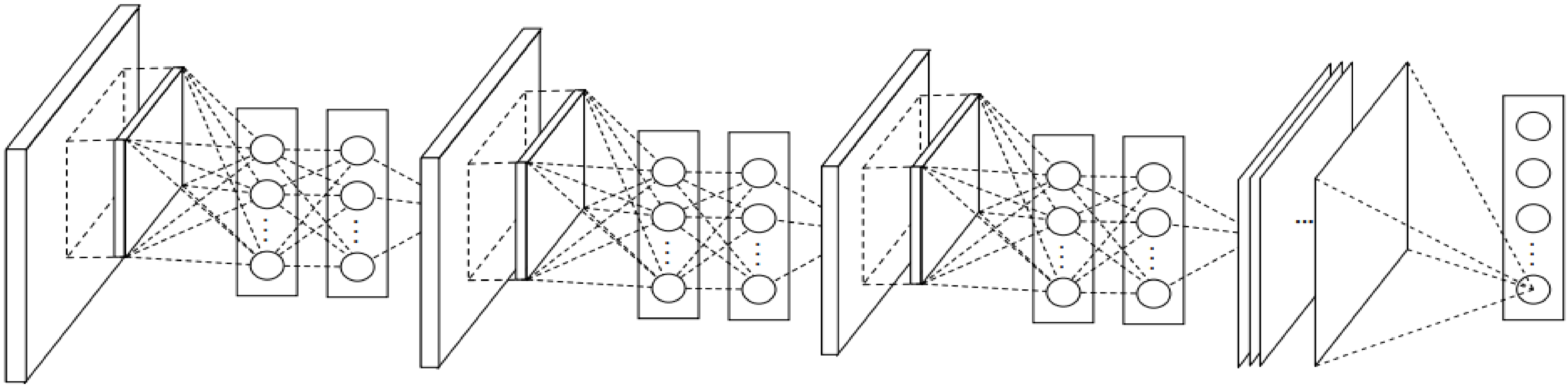
Table 1: Test set error rates for CIFAR-10 of various methods.

Method	Test Error
Stochastic Pooling [11]	15.13%
CNN + Spearmint [14]	14.98%
Conv. maxout + Dropout [8]	11.68%
NIN + Dropout	**10.41%**
CNN + Spearmint + Data Augmentation [14]	9.50%
Conv. maxout + Dropout + Data Augmentation [8]	9.38%
DropConnect + 12 networks + Data Augmentation [15]	9.32%
NIN + Dropout + Data Augmentation	**8.81%**

Lin et al. Network in Network. 2014.

NETWORK IN NETWORK (NIN)

The overall structure of NiN: stacking of three mlpconv layers and one global average pooling layer



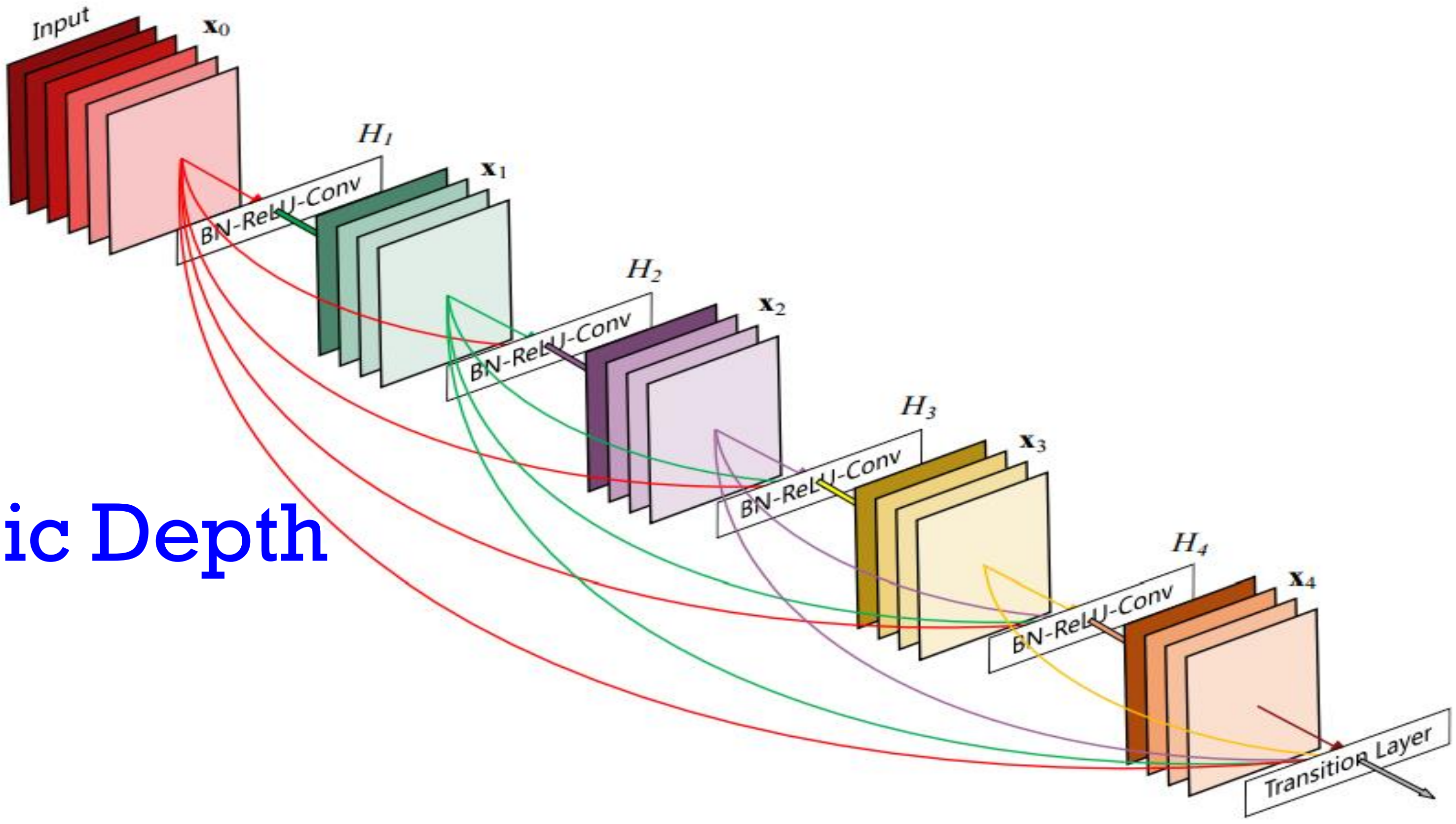
- Precursor to GoogLeNet and ResNet “bottleneck” layers
- Philosophical inspiration for GoogLeNet

Lin et al. Network in Network. 2014.

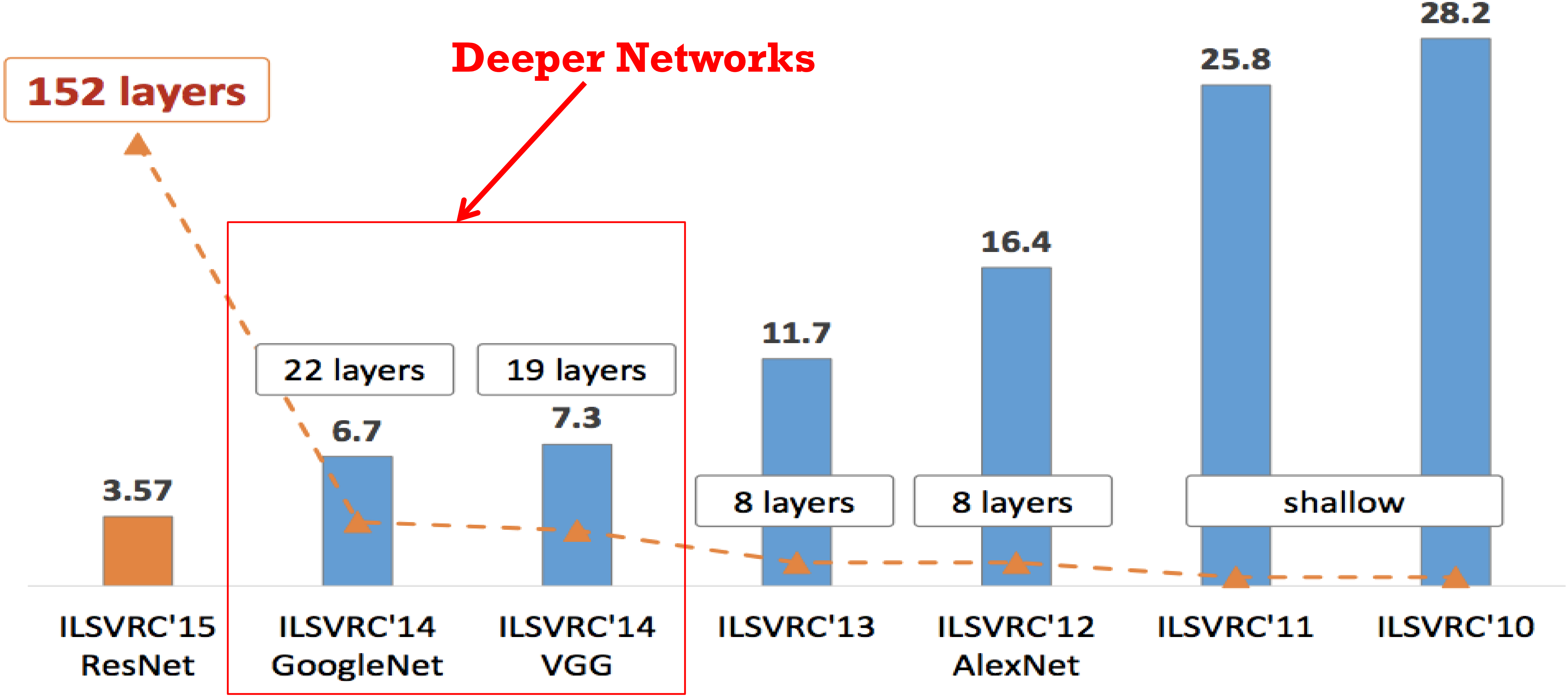
Source: cs231n

CNN Architectures: DAG Models

- GoogLeNet
- ResNet
- Pre-act ResNet
- SENet
- Network with Stochastic Depth
- DenseNet
- ResNetXt



IMAGENET LARGE SCALE VISUAL RECOGNITION CHALLENGE (ILSVRC) WINNERS



K. He, X. Zhang, S. Ren, J. Sun, Deep Residual Learning for Image Recognition, IEEE CVPR 2016.

GOOGLENET

Deeper networks, with computational efficiency

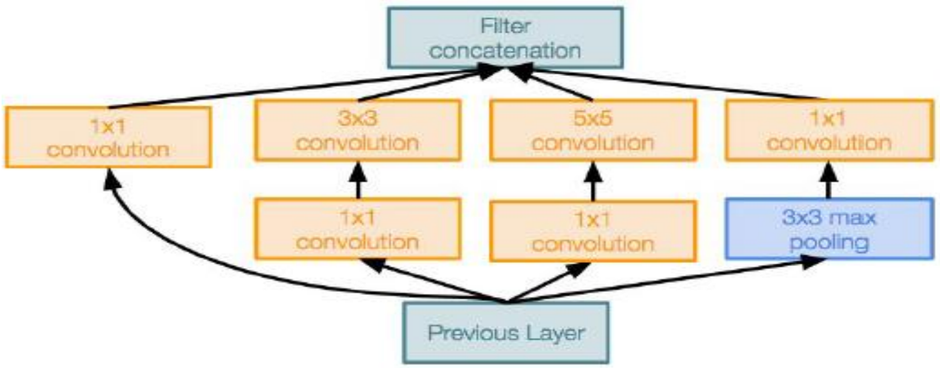
- 22 layers

- Efficient "Inception" module

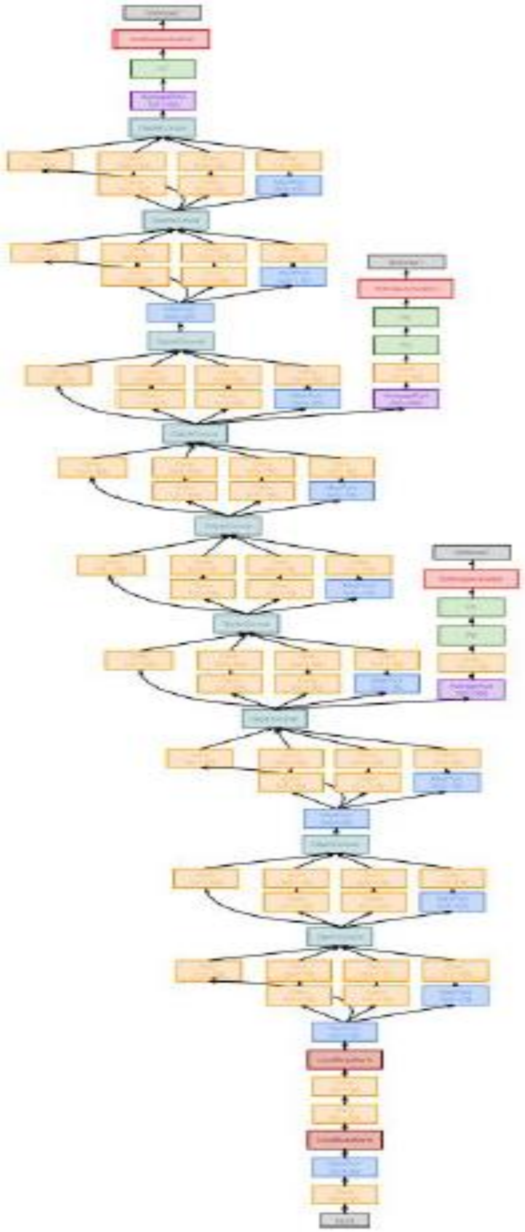
- No FC layers

- Only 5 million parameters!
  12x less than AlexNet

- Imagenet classification winner
  (6.7% top 5 error)



Inception module

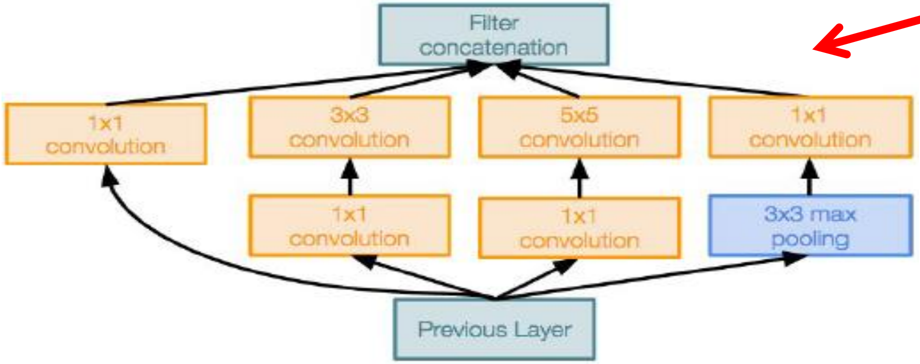


Szegedy, Christian, et al. "Going deeper with convolutions." CVPR 2015.

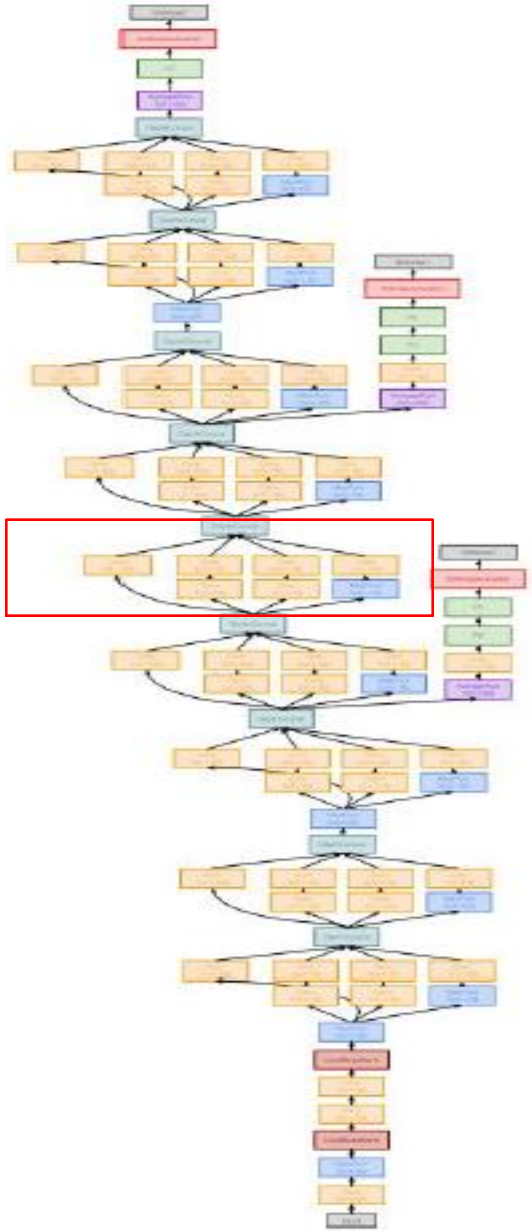
Source: cs231n

GOOGLENET

"Inception module":
design a good local network topology and then stack these modules on top of each other



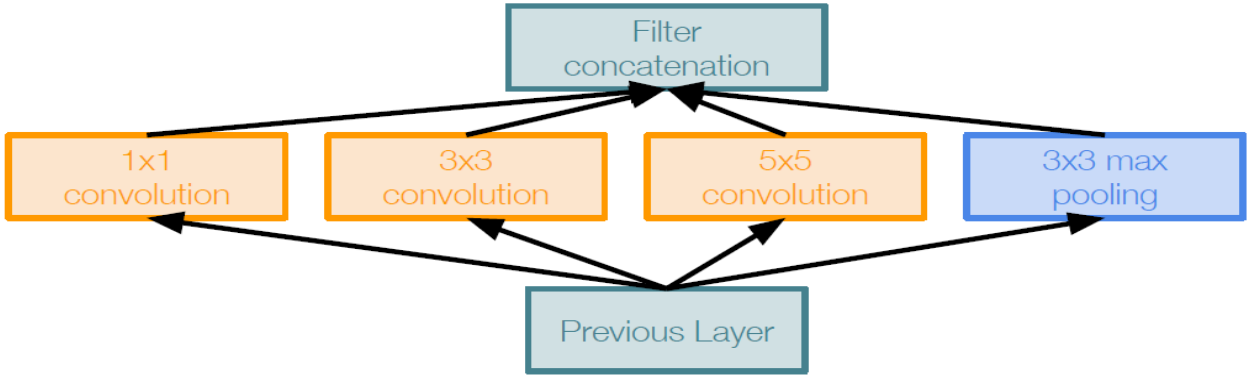
Inception module



Szegedy, Christian, et al. "Going deeper with convolutions." CVPR 2015.

Source: cs231n

GOOGLENET



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

Problem:
Computational Complexity

Szegedy, Christian, et al. "Going deeper with convolutions." CVPR 2015.

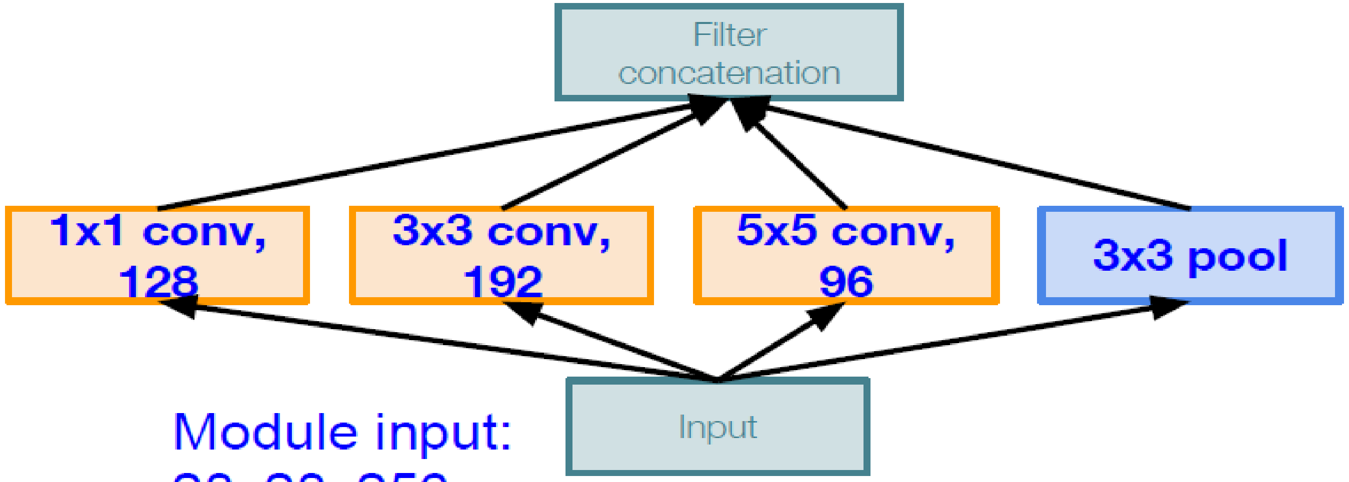
Source: cs231n

GOOGLENET

Problem:
Computational Complexity

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

Example:

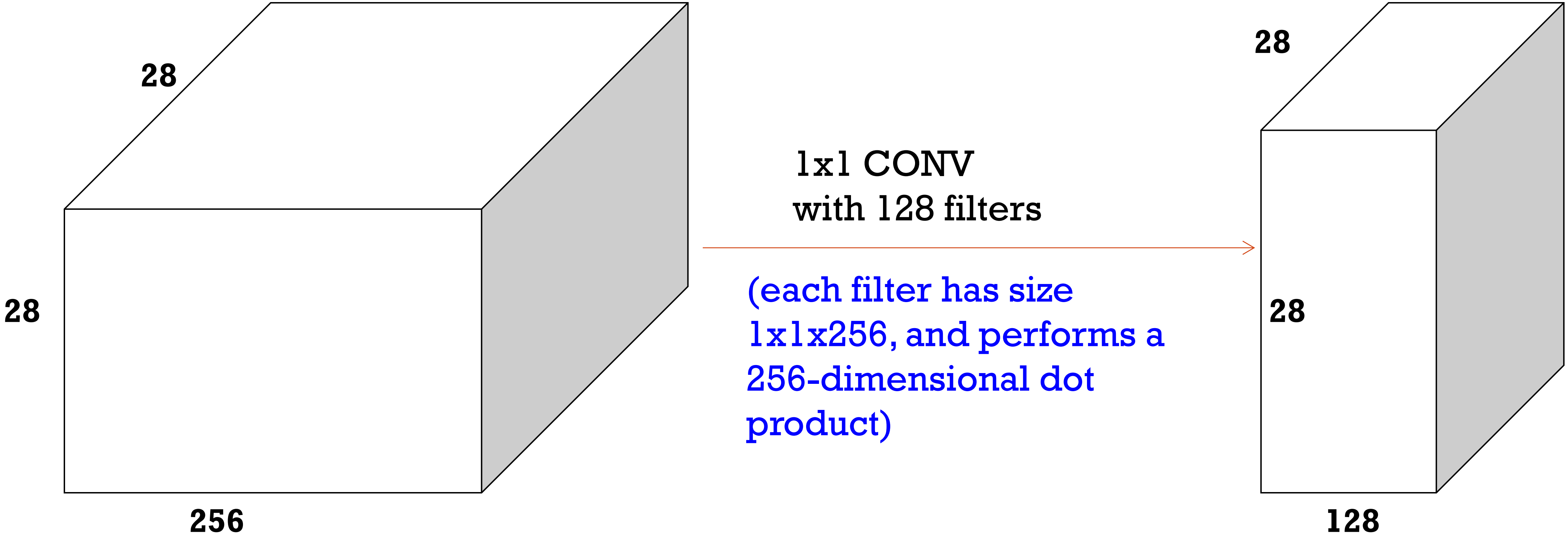


Naive Inception module

Szegedy, Christian, et al. "Going deeper with convolutions." CVPR 2015.

Source: cs231n

1×1 CONVOLUTIONS

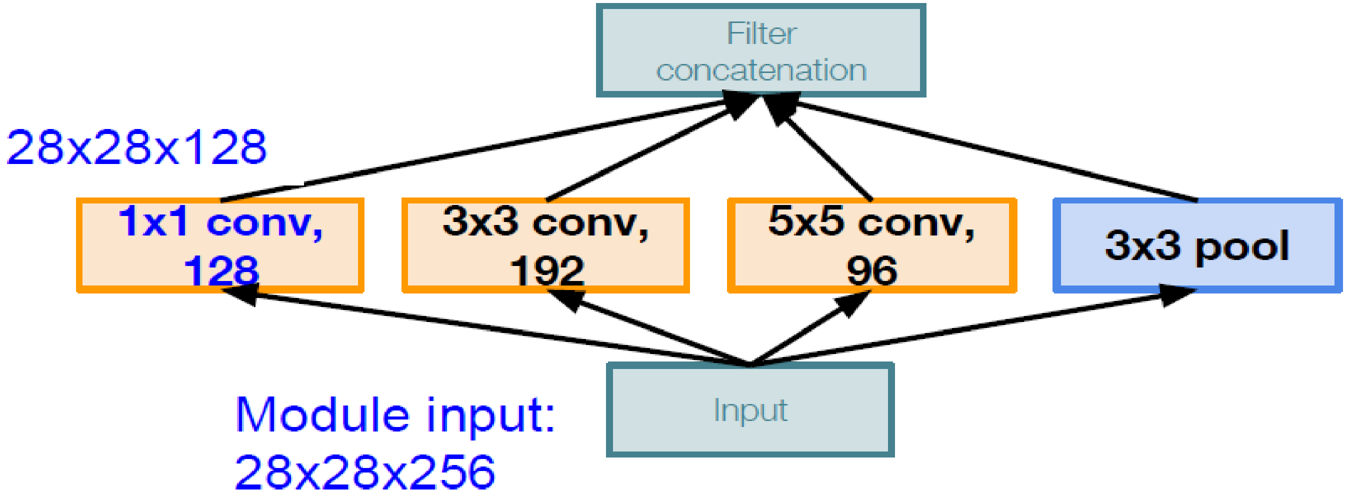


GOOGLENET

Problem:
Computational Complexity

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

Example:



Naive Inception module

Szegedy, Christian, et al. "Going deeper with convolutions." CVPR 2015.

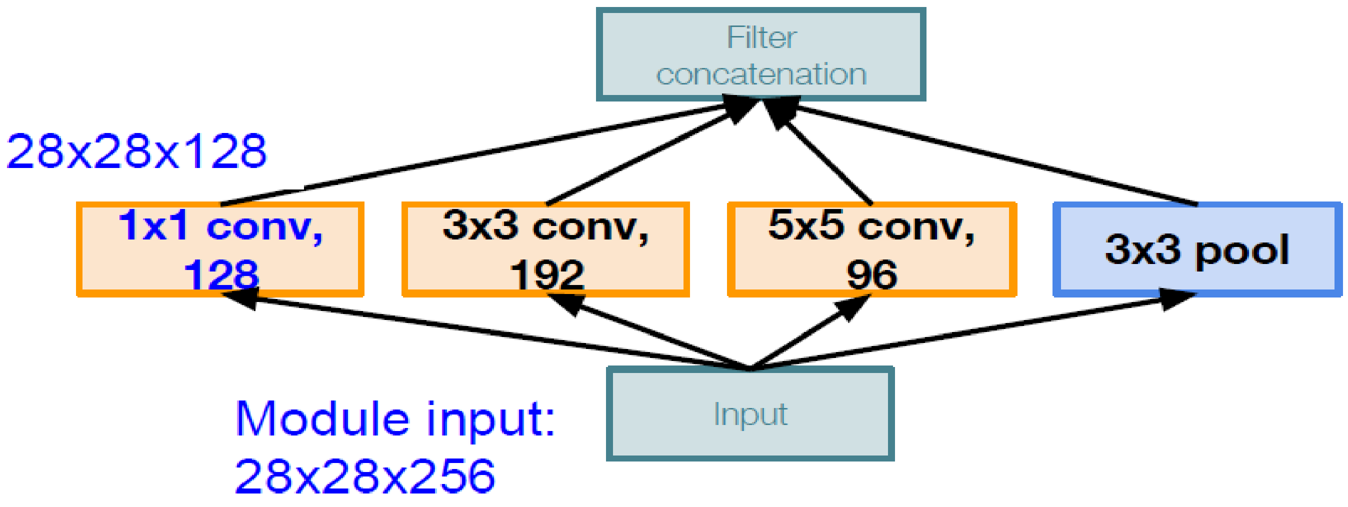
Source: cs231n

GOOGLENET

Problem:
Computational Complexity

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

Example:



Naive Inception module

Szegedy, Christian, et al. "Going deeper with convolutions." CVPR 2015.

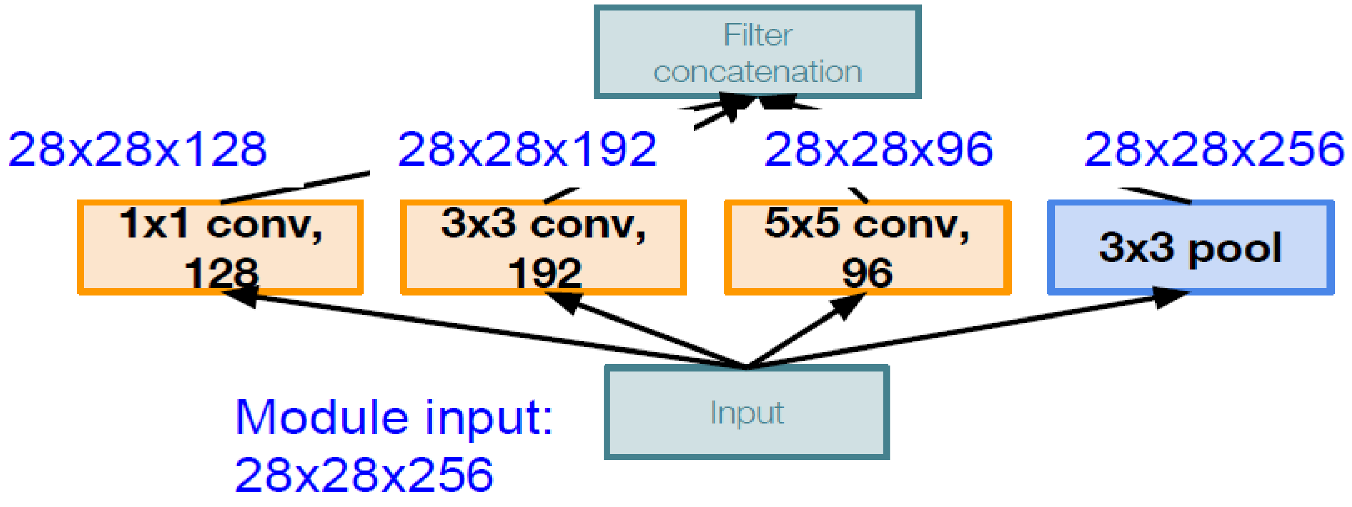
Source: cs231n

GOOGLENET

Problem:
Computational Complexity

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

Example:



Naive Inception module

Szegedy, Christian, et al. "Going deeper with convolutions." CVPR 2015.

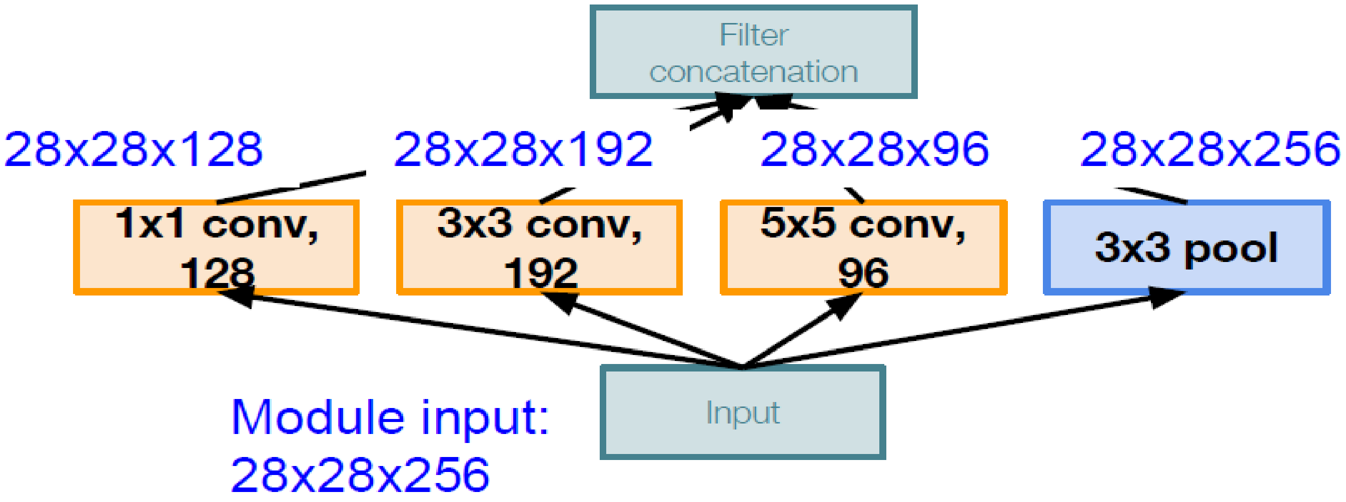
Source: cs231n

GOOGLENET

Problem:
Computational Complexity

Q3:What is output size after filter concatenation?

Example:



Naive Inception module

Szegedy, Christian, et al. "Going deeper with convolutions." CVPR 2015.

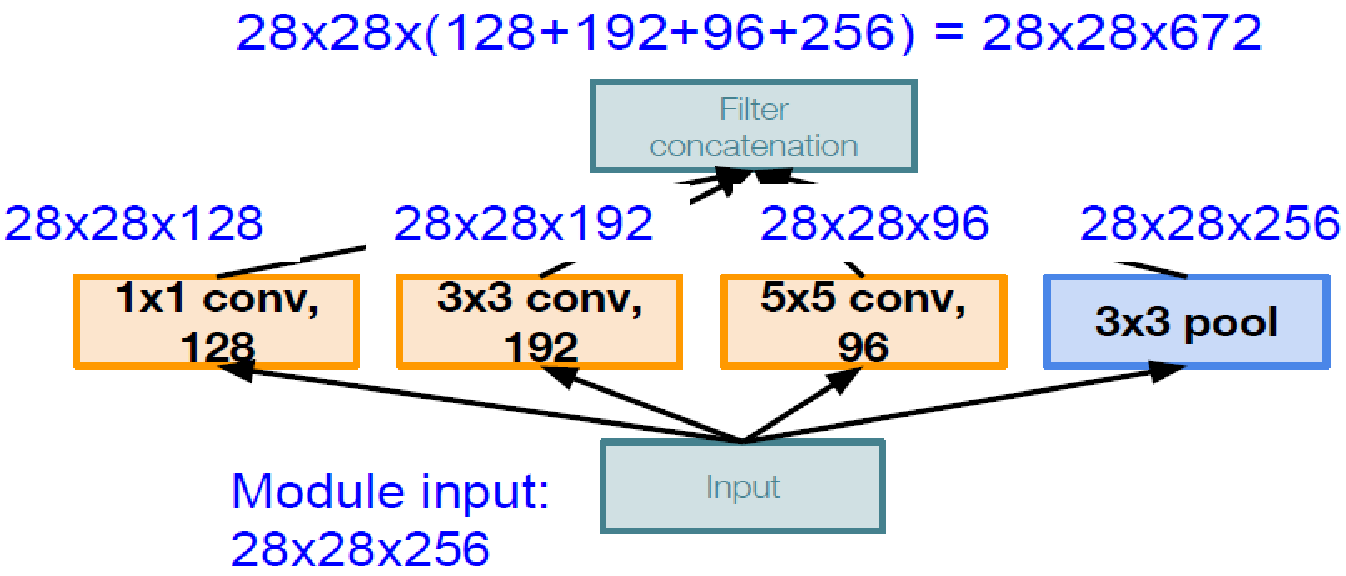
Source: cs231n

GOOGLENET

Problem:
Computational Complexity

Q3:What is output size after filter concatenation?

Example:



Naive Inception module

Szegedy, Christian, et al. "Going deeper with convolutions." CVPR 2015.

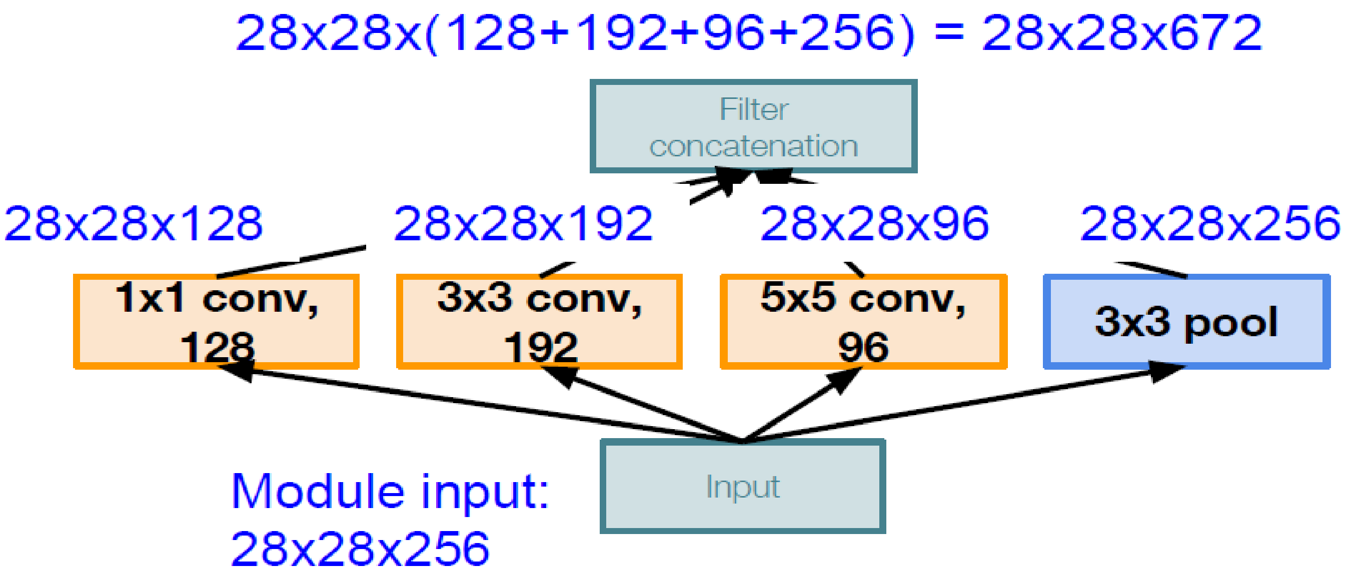
Source: cs231n

GOOGLENET

Problem:
Computational Complexity

Q3:What is output size after filter concatenation?

Example:



Naive Inception module

Conv Ops:

[1x1 conv, 128] 28x28x128x1x1x256

[3x3 conv, 192] 28x28x192x3x3x256

[5x5 conv, 96] 28x28x96x5x5x256

Total: 854M ops

Very expensive compute

Szegedy, Christian, et al. "Going deeper with convolutions." CVPR 2015.

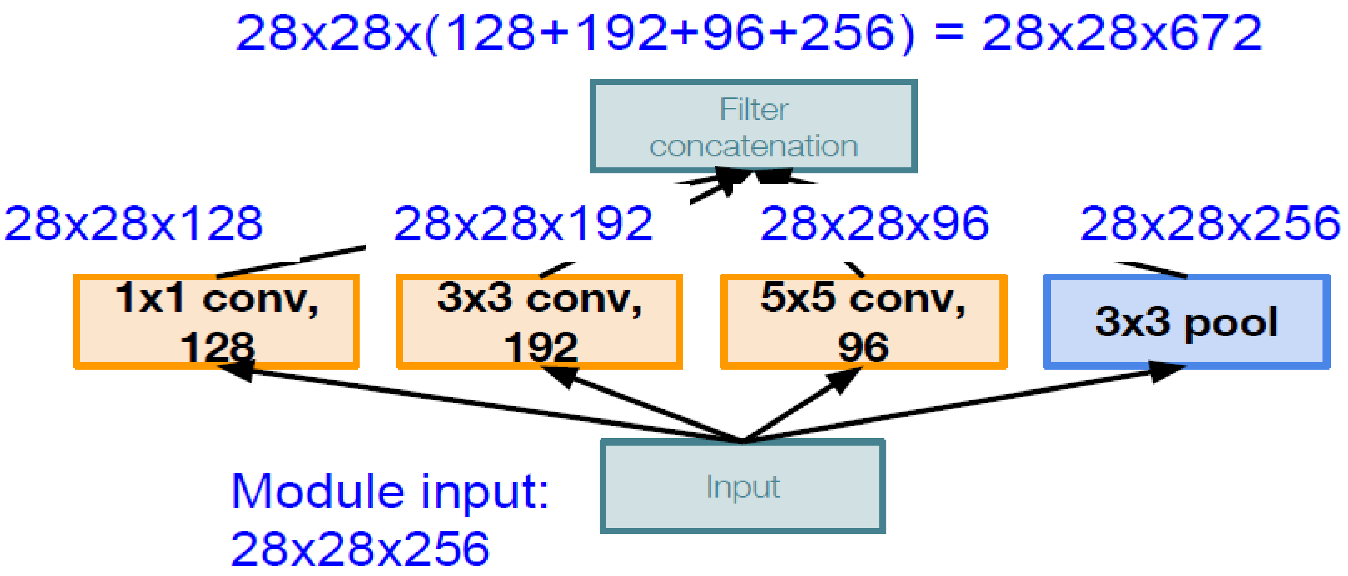
Source: cs231n

GOOGLENET

Problem:
Computational Complexity

Q3:What is output size after filter concatenation?

Example:



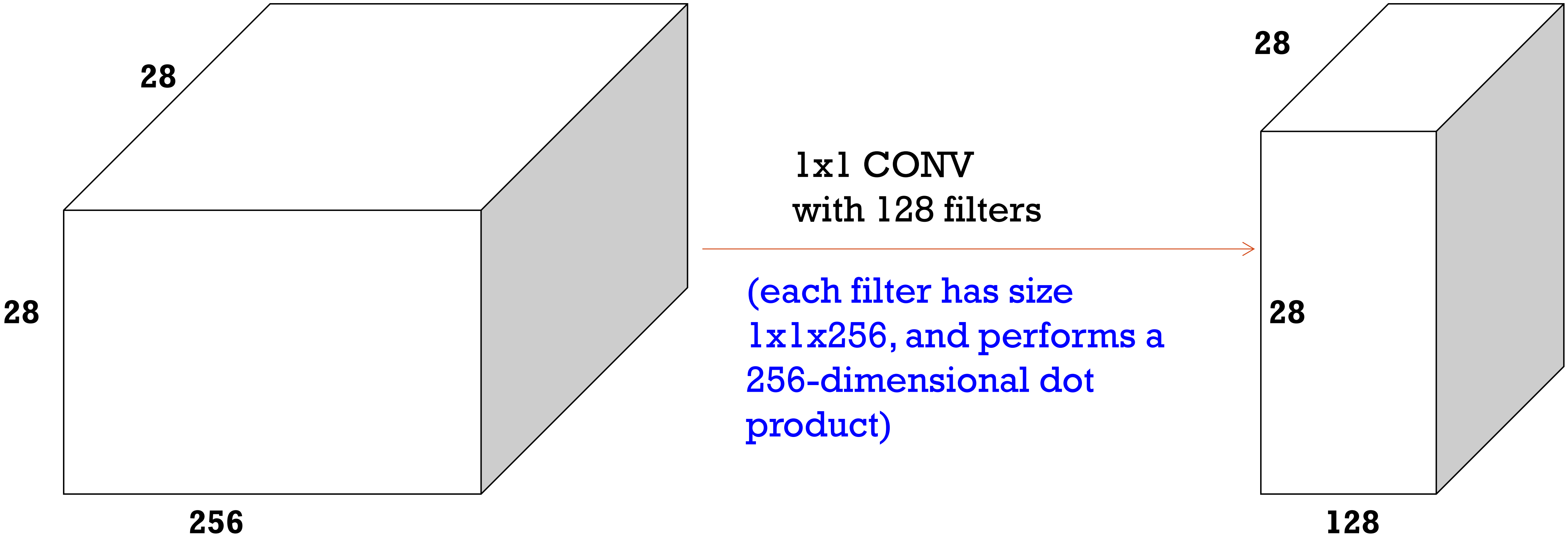
Naive Inception module

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

Szegedy, Christian, et al. "Going deeper with convolutions." CVPR 2015.

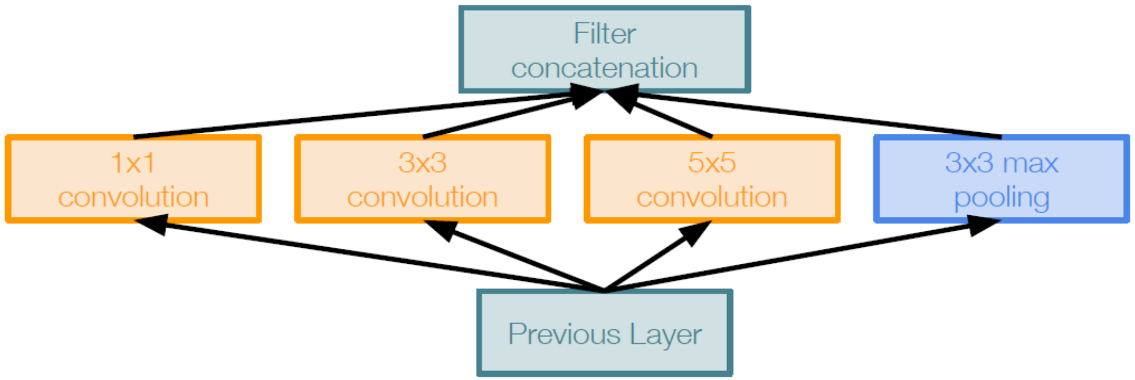
Source: cs231n

1×1 CONVOLUTIONS



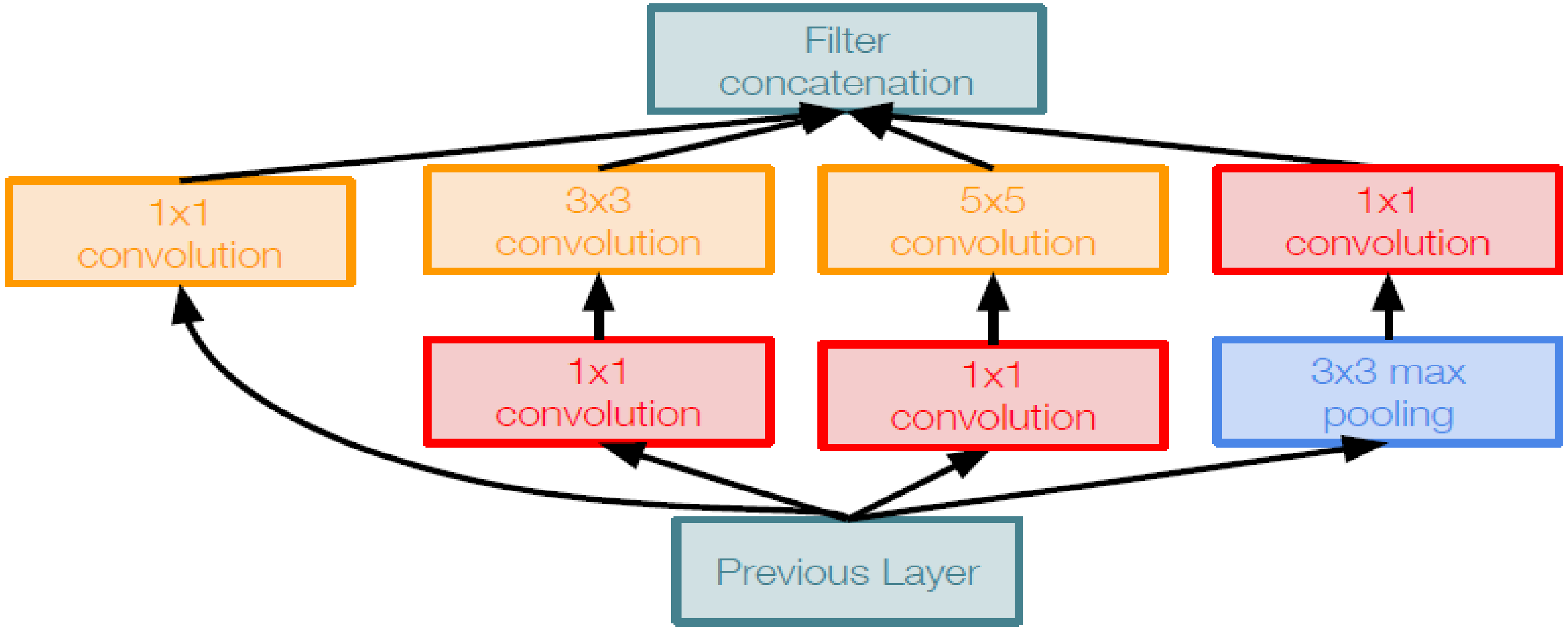
preserves spatial dimensions, reduces depth!
Projects depth to lower dimension (combination of feature maps)

GOOGLENET



Naive Inception module

1x1 conv "bottleneck" layers

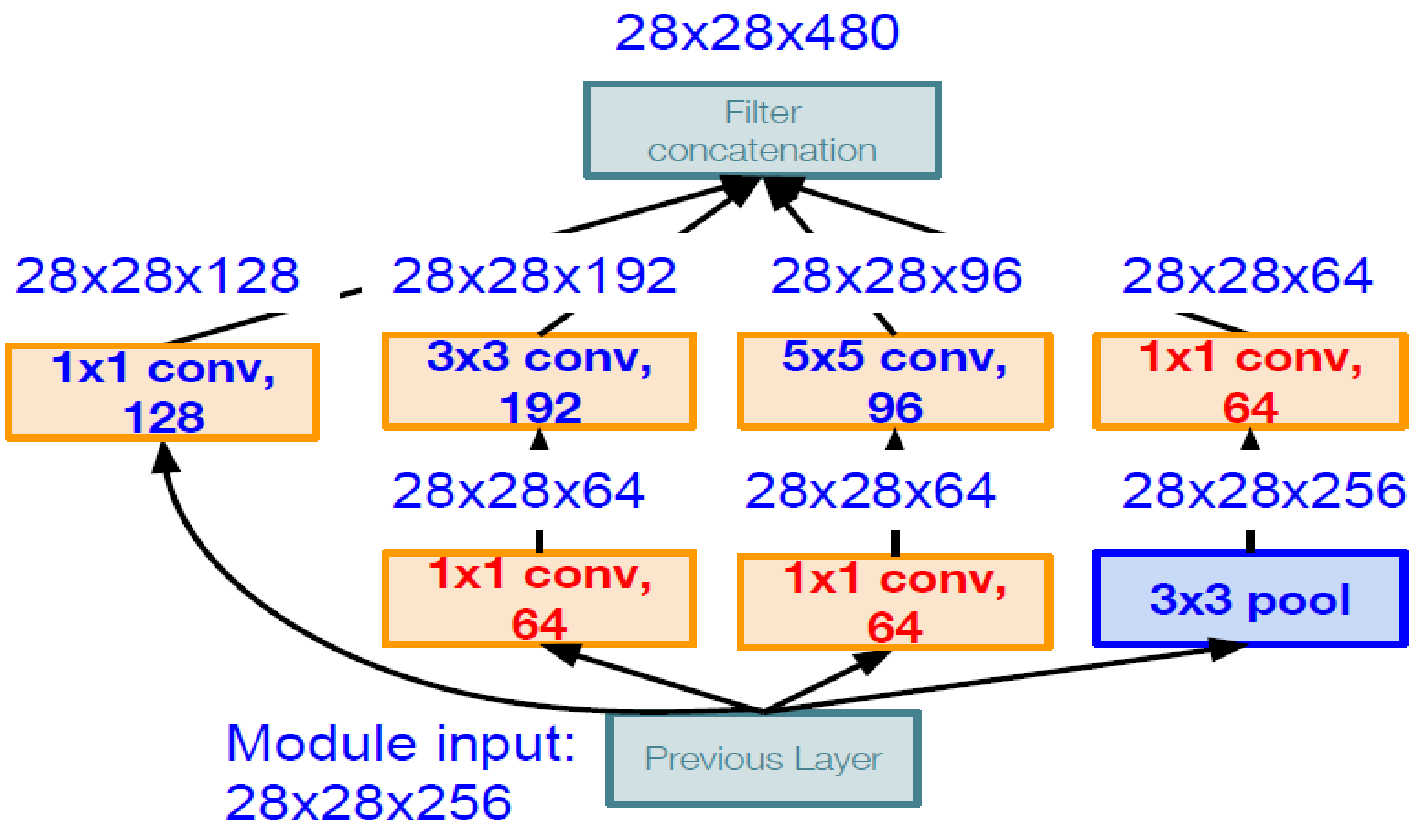


Inception module with dimension reduction

Szegedy, Christian, et al. "Going deeper with convolutions." CVPR 2015.

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

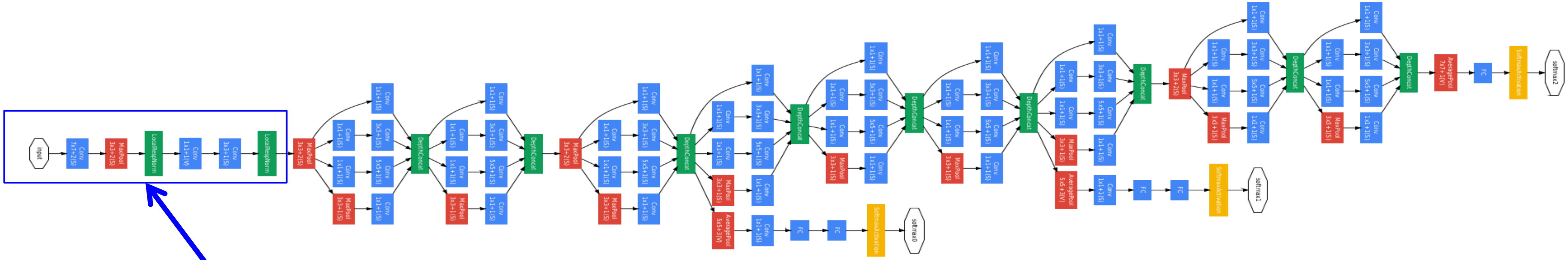
Compared to 854M ops for naive version, Bottleneck can also reduce depth after pooling layer

Szegedy, Christian, et al. "Going deeper with convolutions." CVPR 2015.

Source: cs231n

GOOGLENET

Full GoogLeNet Architecture



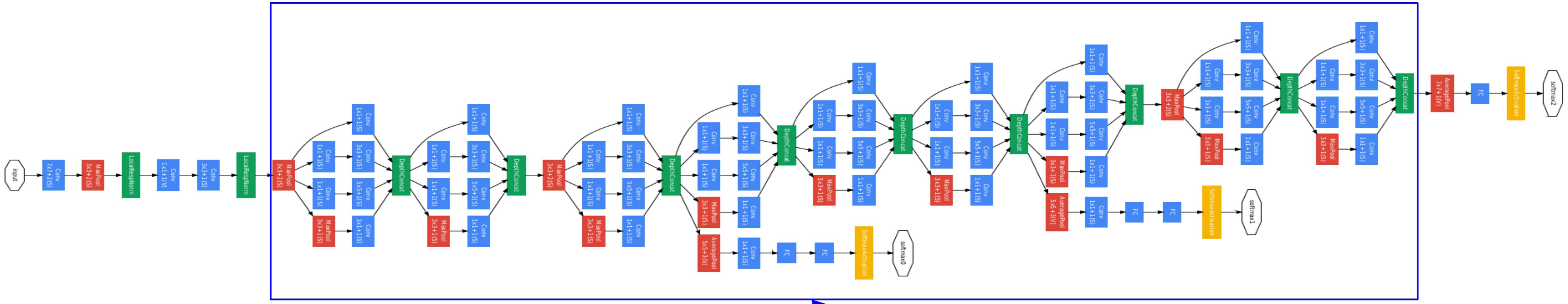
Stem Network:
Conv-Pool-
2x Conv-Pool

Szegedy, Christian, et al. "Going deeper with convolutions." CVPR 2015.

Source: cs231n

GOOGLENET

Full GoogLeNet Architecture



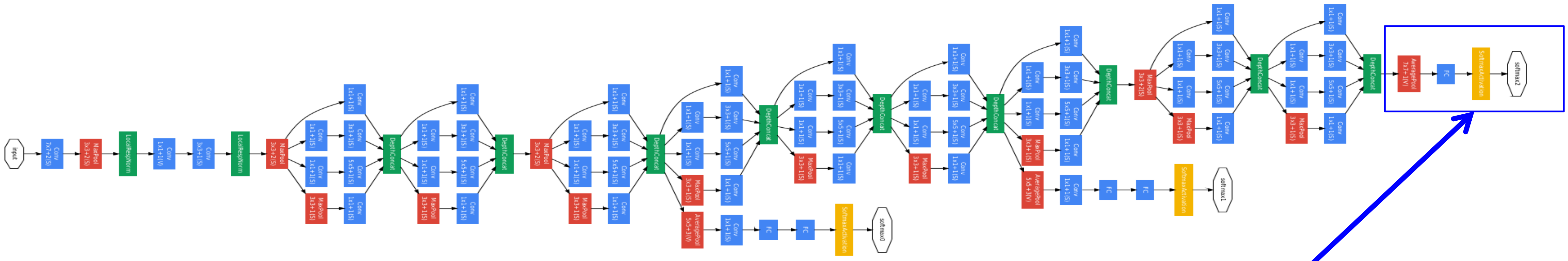
Stacked Inception Modules

Szegedy, Christian, et al. "Going deeper with convolutions." CVPR 2015.

Source: cs231n

GOOGLENET

Full GoogLeNet Architecture



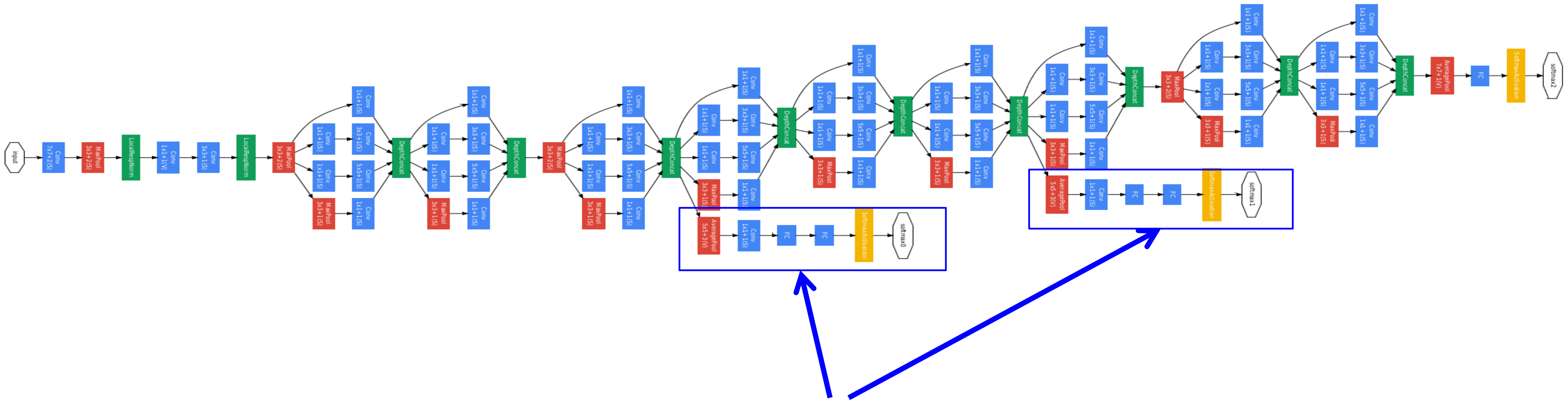
Classifier output
(removed expensive FC layers!)

Szegedy, Christian, et al. "Going deeper with convolutions." CVPR 2015.

Source: cs231n

GOOGLENET

Full GoogLeNet Architecture



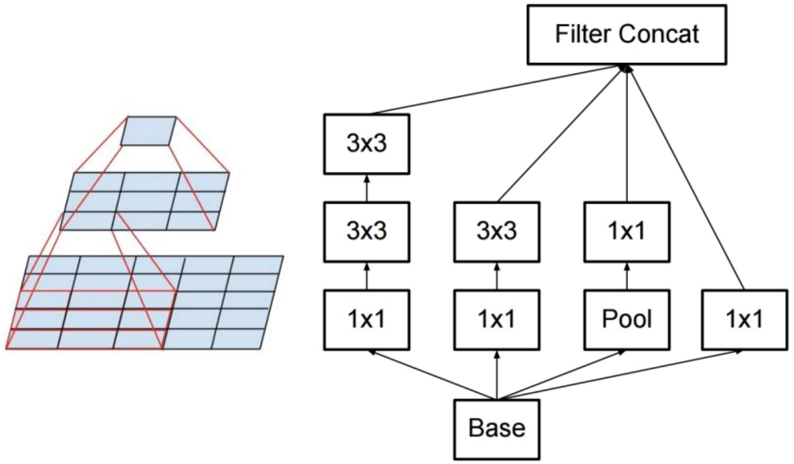
Auxiliary classification outputs to inject additional gradient at lower layers
(AvgPool-1x1Conv-FC-FC-Softmax)

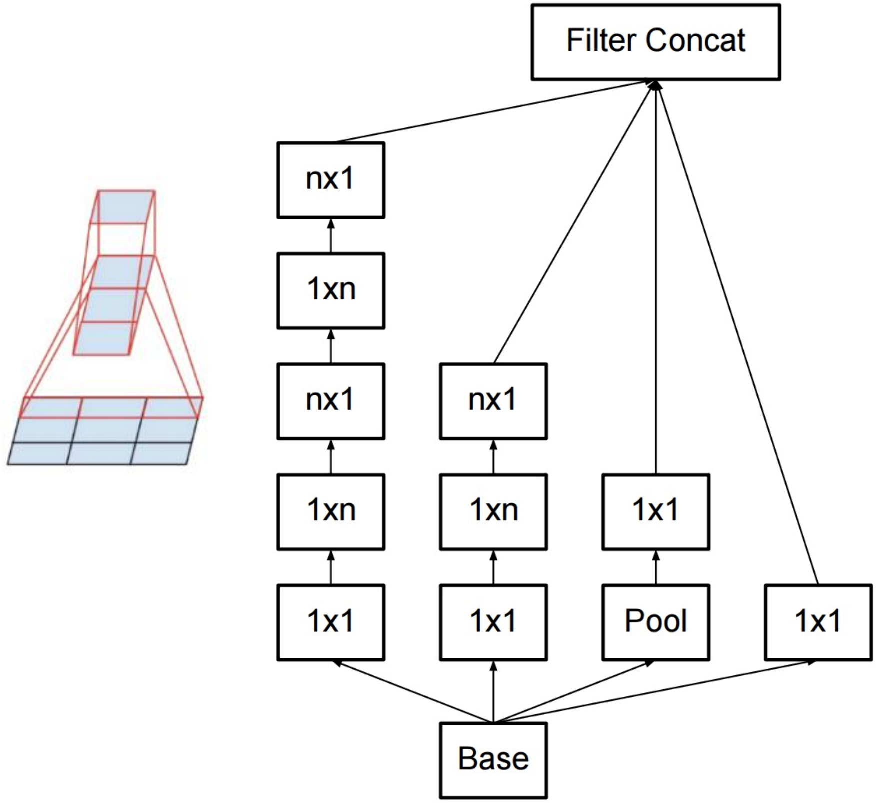
Szegedy, Christian, et al. "Going deeper with convolutions." CVPR 2015.

Source: cs231n

INCEPTION V2, V3

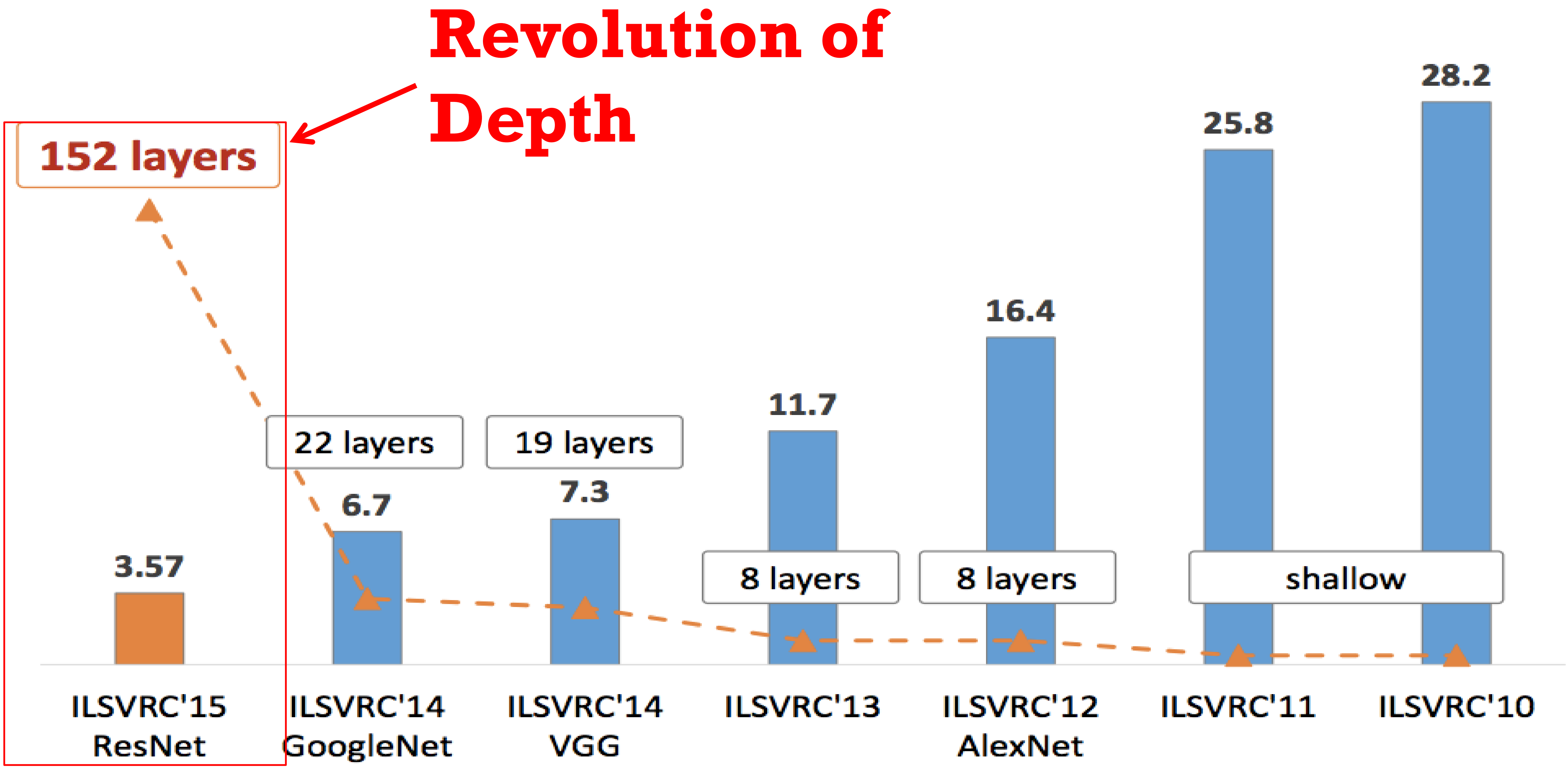
- Improve training with batch normalization, reducing importance of auxiliary classifiers
- More variants of inception modules with aggressive factorization of filters





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

IMAGENET LARGE SCALE VISUAL RECOGNITION CHALLENGE (ILSVRC) WINNERS



K. He, X. Zhang, S. Ren, J. Sun, Deep Residual Learning for Image Recognition, IEEE CVPR 2016.

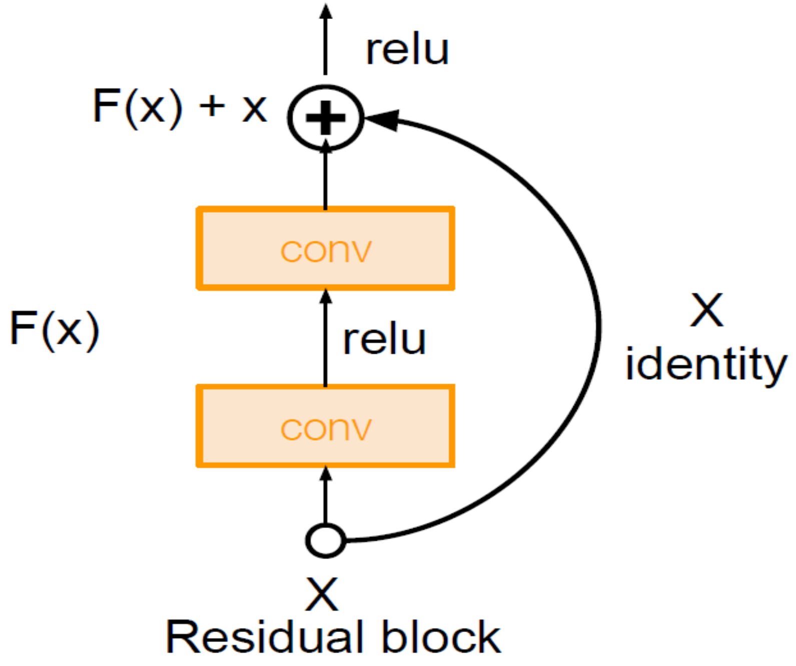
RESNET

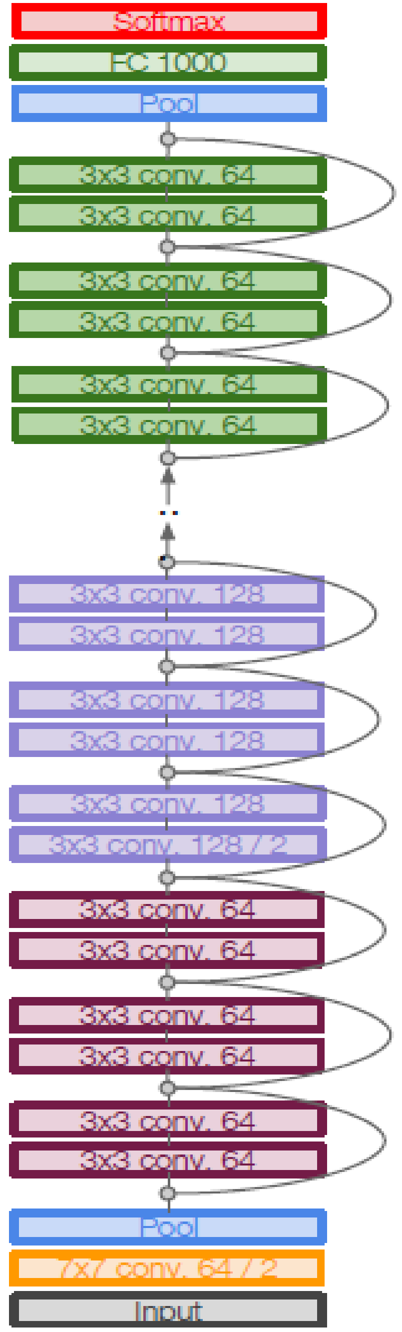
Very deep networks using residual connections

- 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!





He et al. Deep Residual Learning for Image Recognition, IEEE CVPR 2016.

Source: cs231n

RESNET

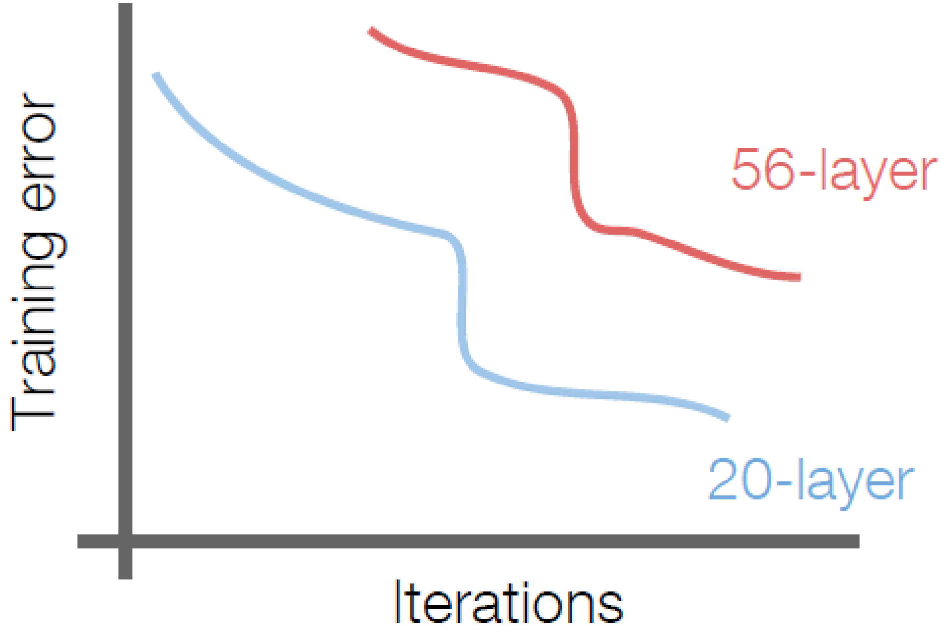
What happens when we continue stacking deeper layers on a “plain” convolutional neural network?

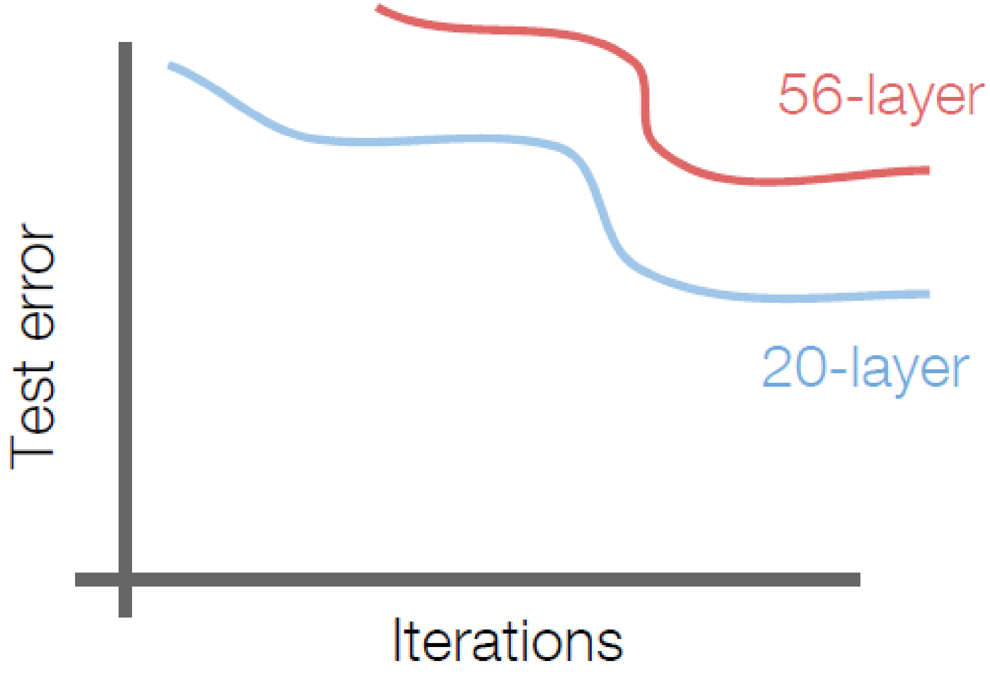
He et al. Deep Residual Learning for Image Recognition, IEEE CVPR 2016.

Source: cs231n

RESNET

What happens when we continue stacking deeper layers on a "plain" convolutional neural network?





56-layer model performs worse on both training and test error
-> The deeper model performs worse, but it's not caused by overfitting!

He et al. [Deep Residual Learning for Image Recognition](), IEEE CVPR 2016.

Source: cs231n

RESNET

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

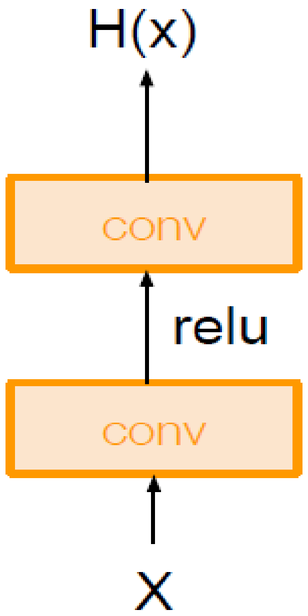
The deeper model should be able to perform at least as well as the shallower model.

He et al. Deep Residual Learning for Image Recognition, IEEE CVPR 2016.

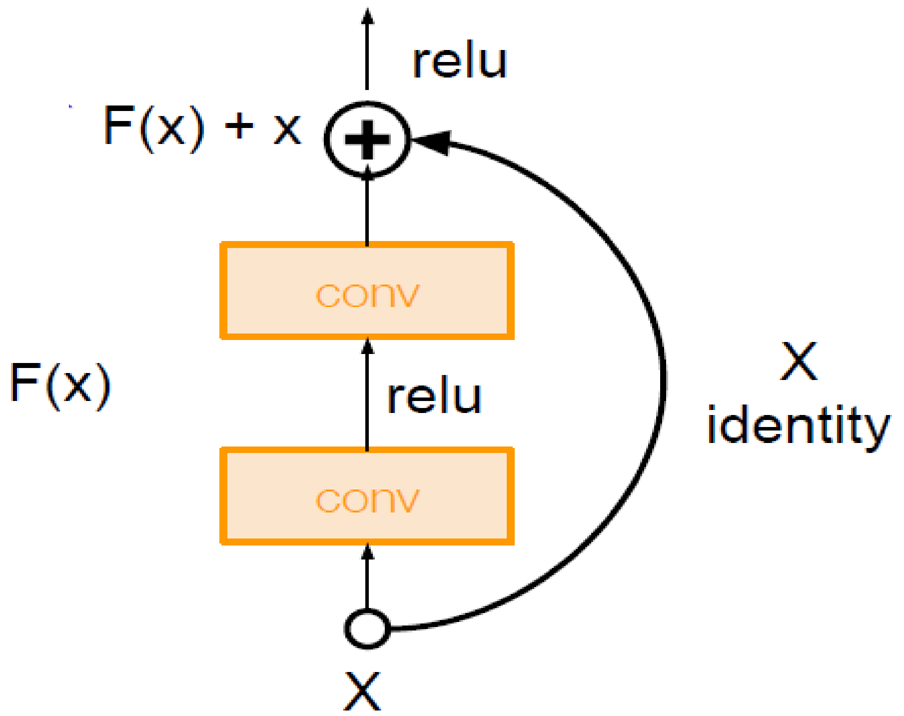
Source: cs231n

RESNET

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



"Plain" layers



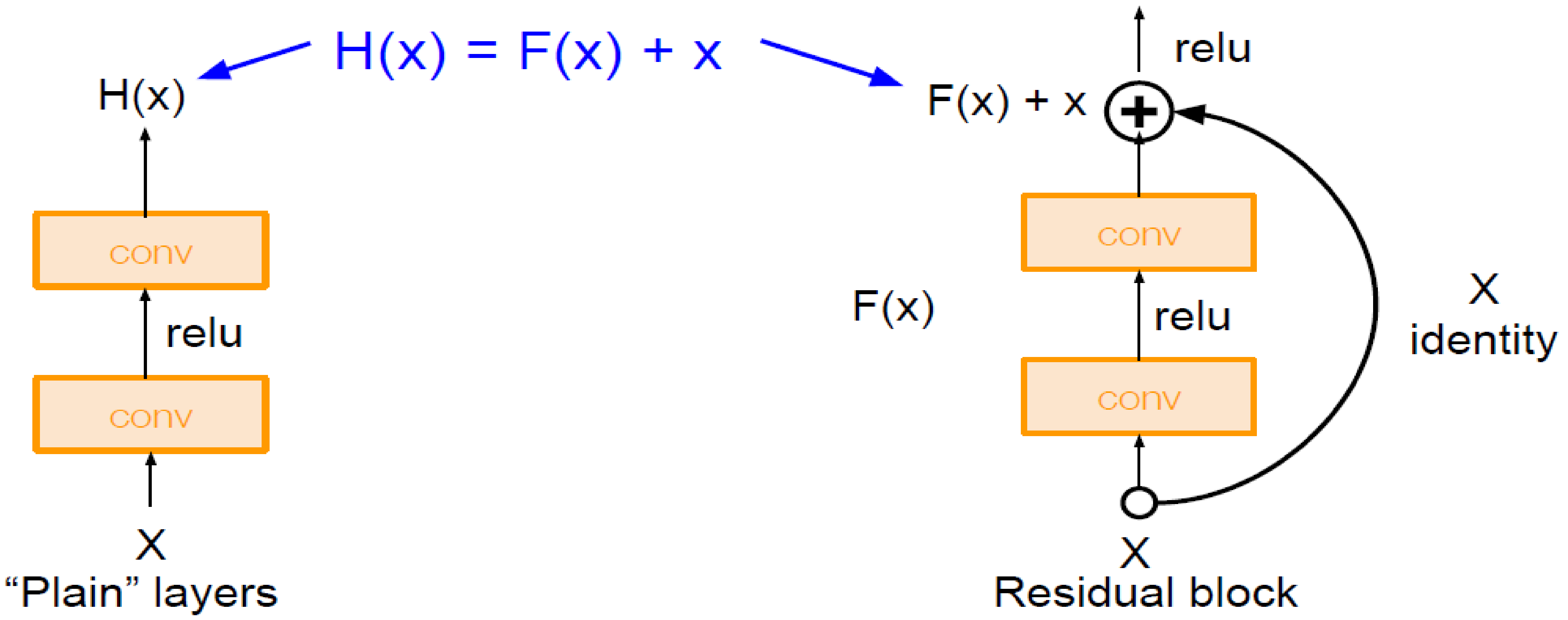
Residual block

He et al. Deep Residual Learning for Image Recognition, IEEE CVPR 2016.

Source: cs231n

RESNET

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

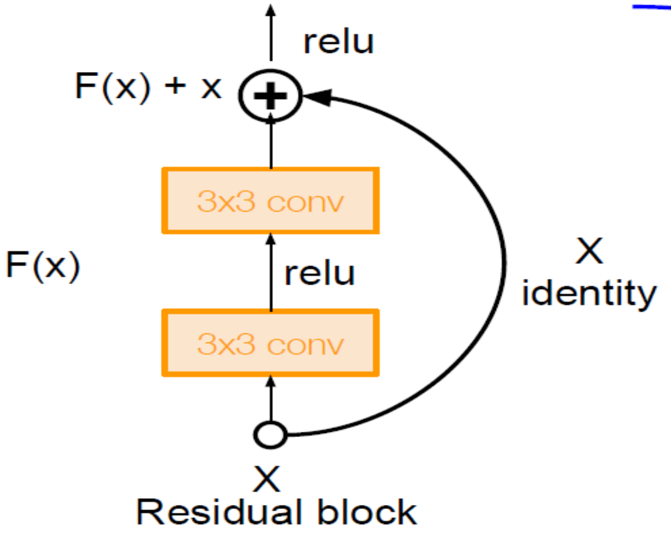
He et al. Deep Residual Learning for Image Recognition, IEEE CVPR 2016.

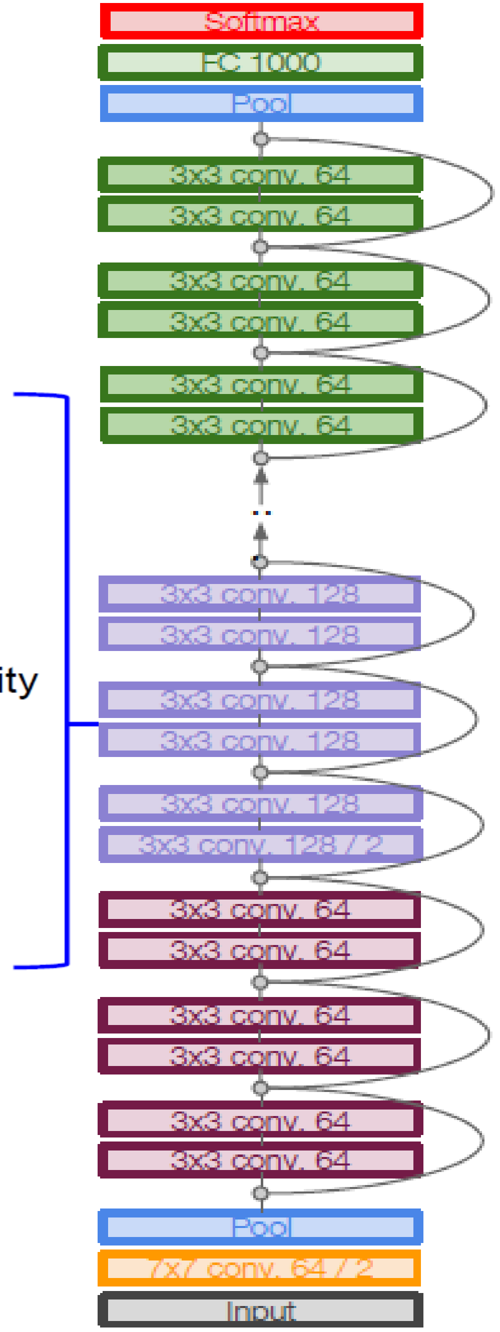
Source: cs231n

RESNET

Full ResNet architecture:

- Stack residual blocks
- Residual block has two 3x3 conv layers





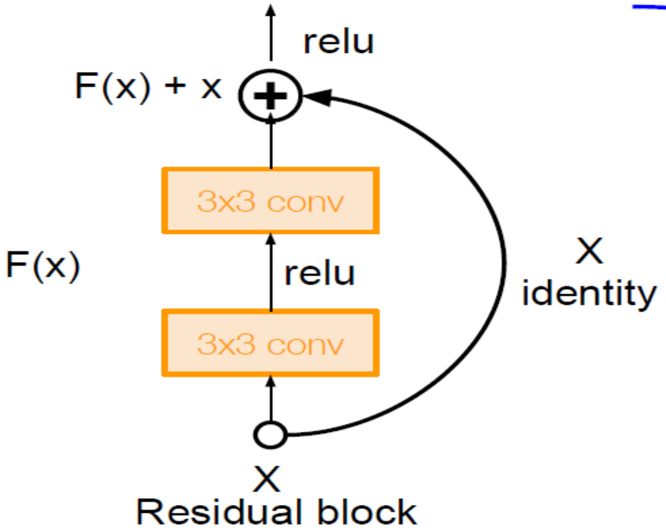
He et al. Deep Residual Learning for Image Recognition, IEEE CVPR 2016.

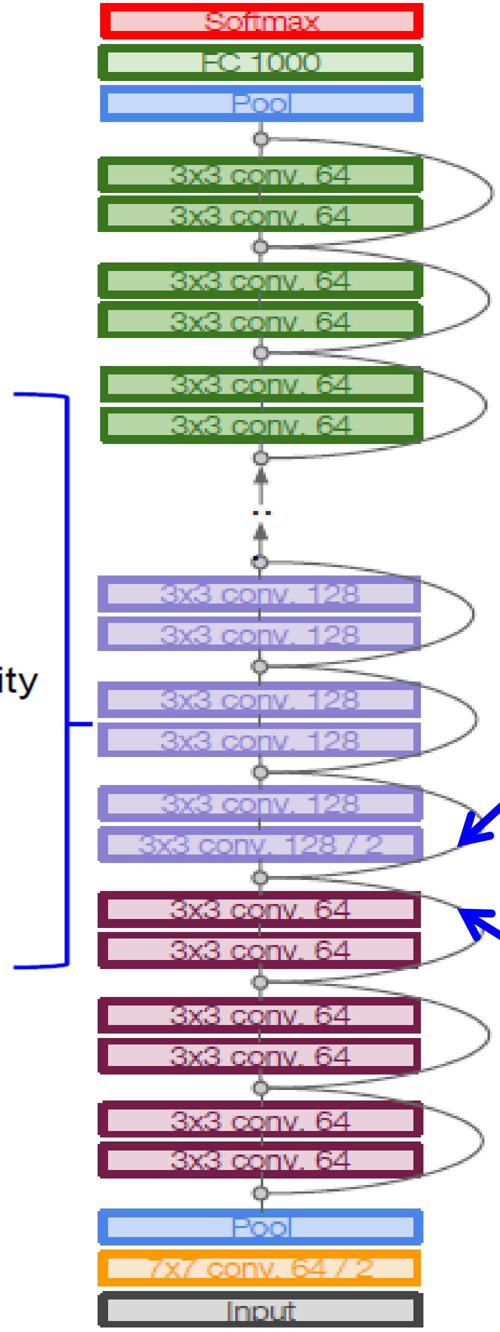
Source: cs231n

RESNET

Full ResNet architecture:

- Stack residual blocks
- Residual block has two 3x3 conv layers
- Periodically, double # of filters and downsample spatially using stride 2 (/2 in each dimension)





3x3 conv, 128 filters, /2 spatially with stride 2

3x3 conv, 64 filters

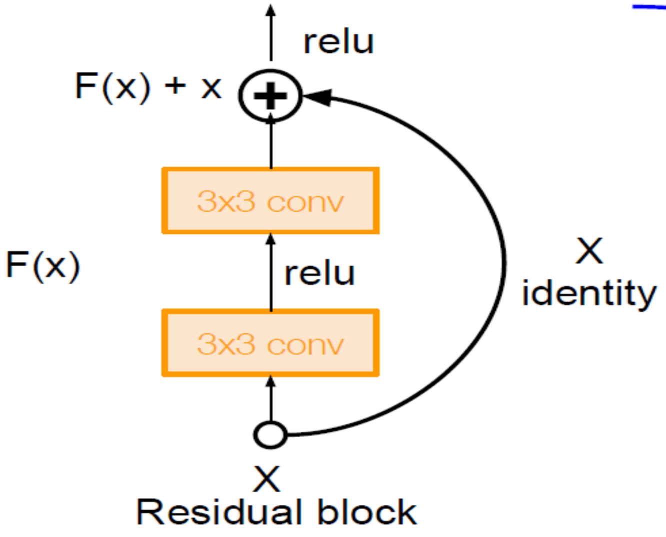
He et al. Deep Residual Learning for Image Recognition, IEEE CVPR 2016.

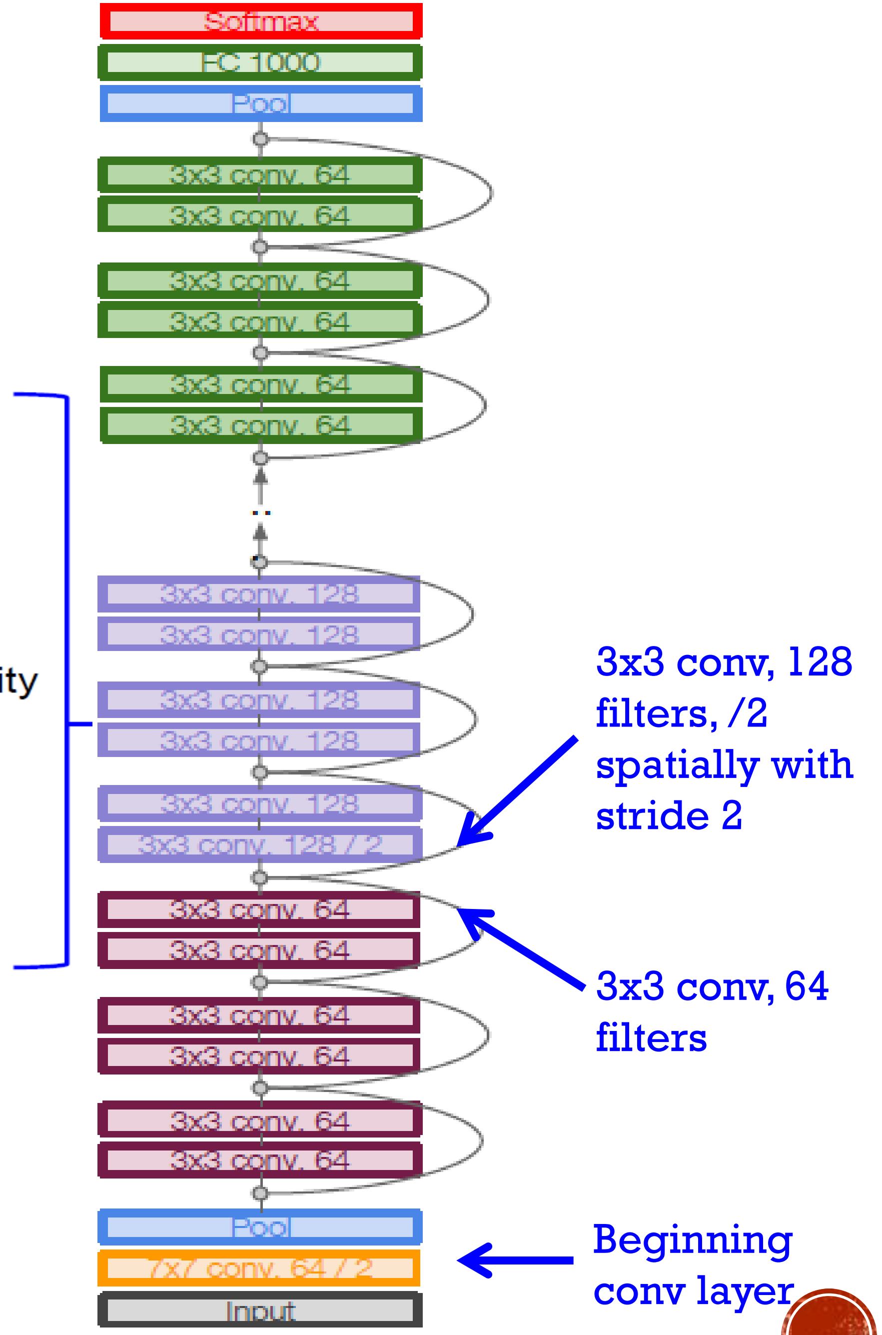
Source: cs231n

RESNET

Full ResNet architecture:

- Stack residual blocks
- 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





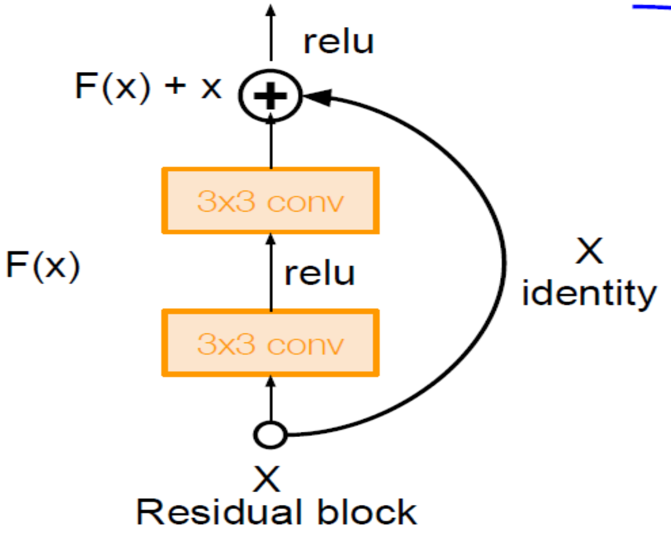
He et al. [Deep Residual Learning for Image Recognition](), IEEE CVPR 2016.

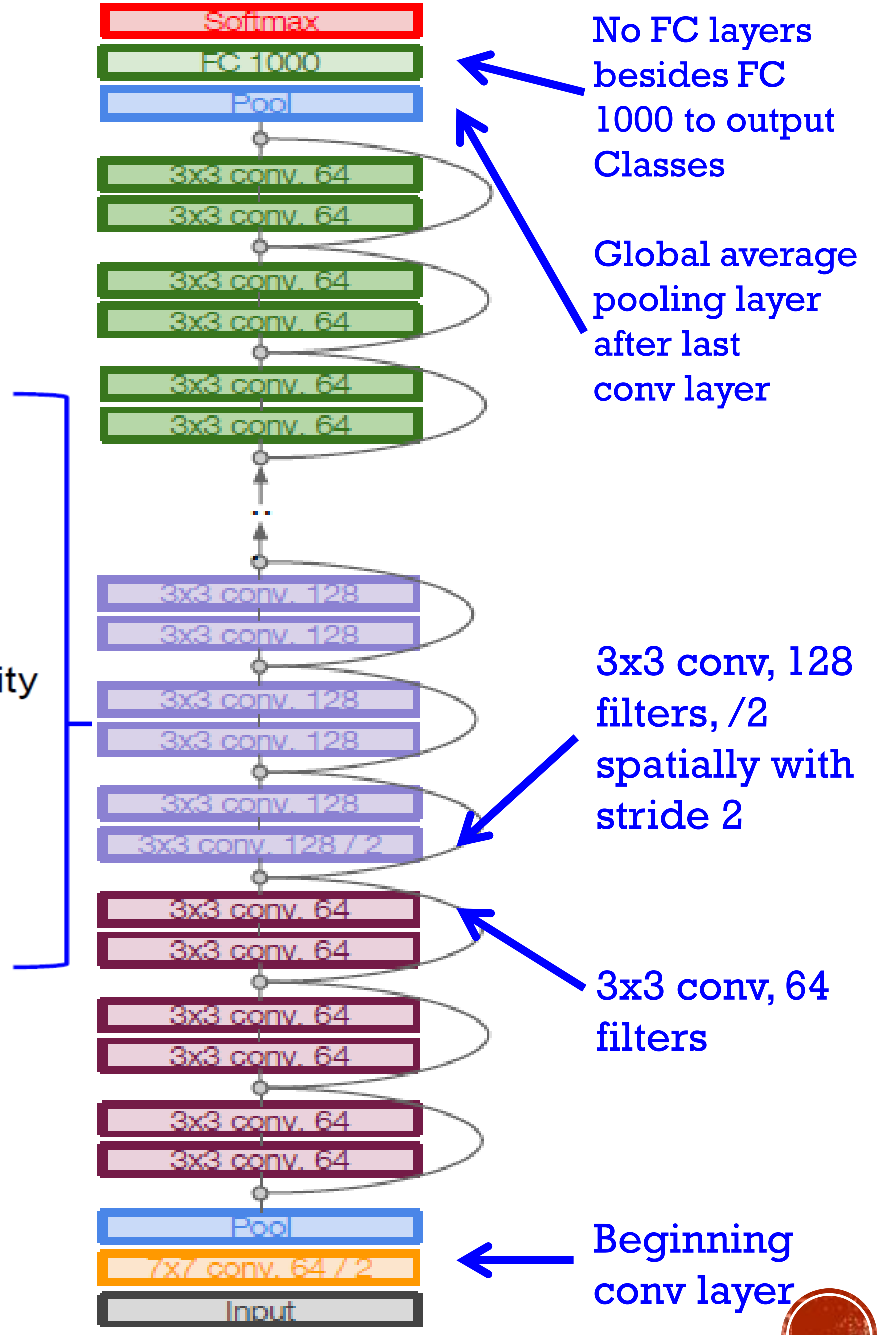
Source: cs231n

RESNET

Full ResNet architecture:

- Stack residual blocks
- 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)



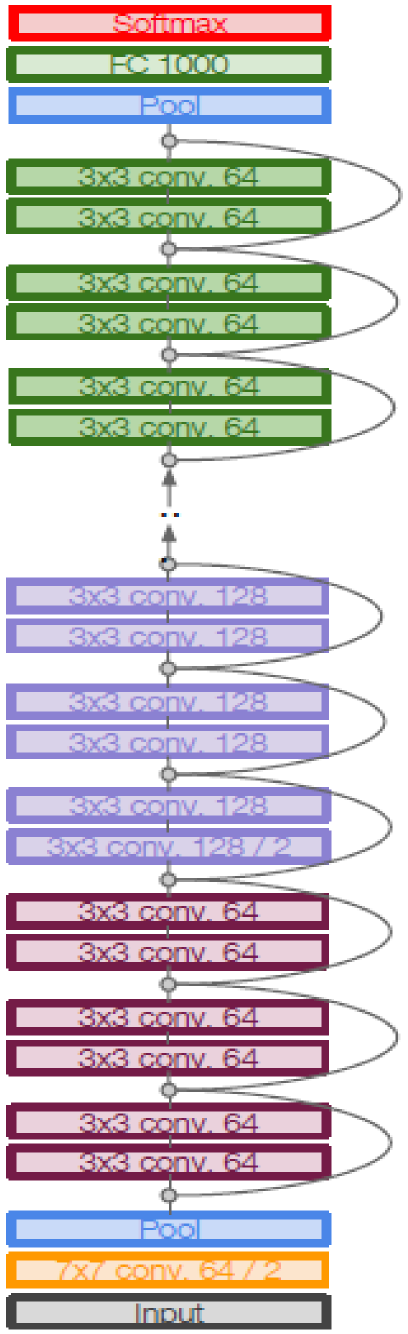


He et al. Deep Residual Learning for Image Recognition, IEEE CVPR 2016.

Source: cs231n

RESNET

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



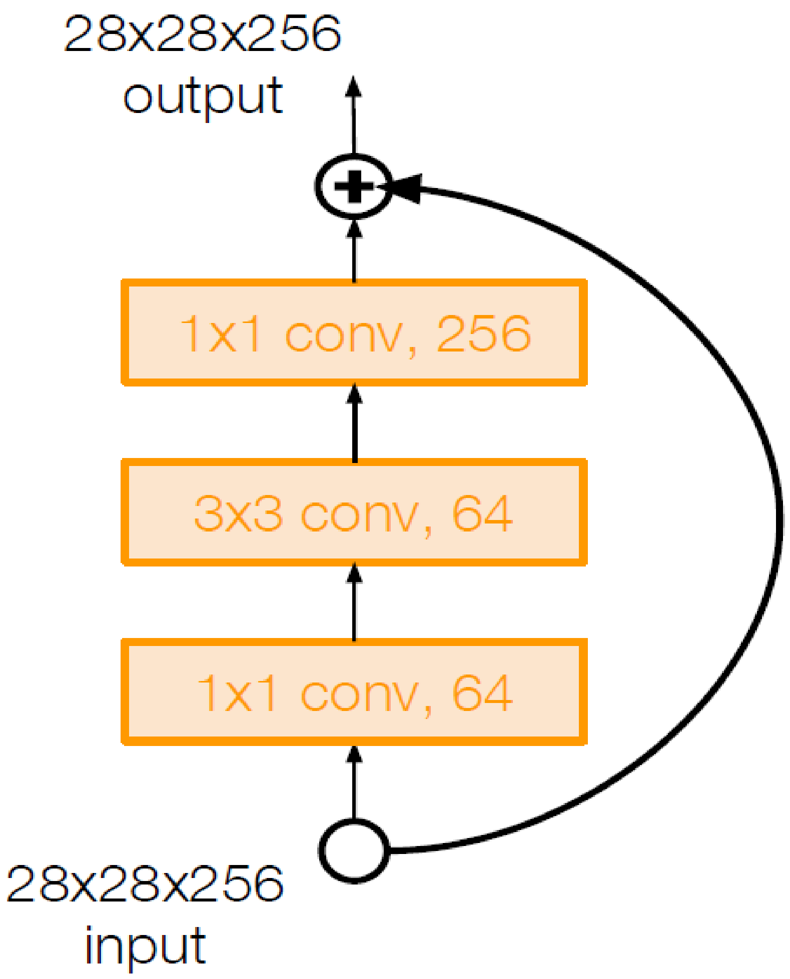
He et al. [Deep Residual Learning for Image Recognition](), IEEE CVPR 2016.

Source: cs231n

RESNET

For deeper networks (ResNet-50+):

use "bottleneck" layer to improve efficiency (similar to GoogLeNet)



He et al. [Deep Residual Learning for Image Recognition](), IEEE CVPR 2016.

Source: cs231n

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 saturates
- Mini-batch size 256
- Weight decay of 1e-5 for penalizing regularization term
- No dropout used

He et al. Deep Residual Learning for Image Recognition, IEEE CVPR 2016.

Source: cs231n

RESNET

Experimental Results:

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

RESNET

Experimental Results:

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

He et al. Deep Residual Learning for Image Recognition, IEEE CVPR 2016.

Source: cs231n

RESNET

Experimental Results:

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

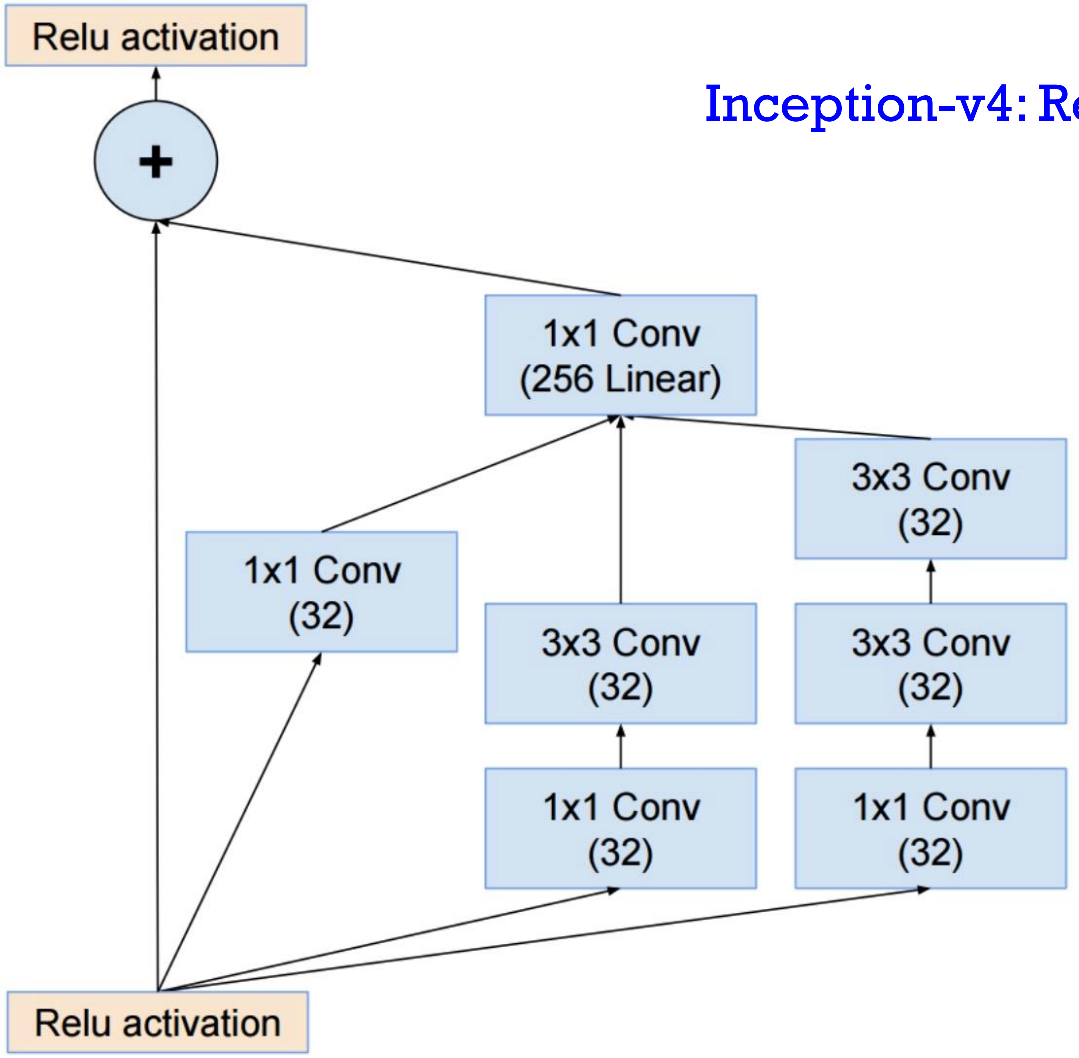
- **1st places in all five main tracks**
  - ImageNet Classification: "*Ultra-deep*" (quote Yann) 152-layer nets
  - ImageNet Detection: 16% better than 2nd
  - ImageNet Localization: 27% better than 2nd
  - COCO Detection: 11% better than 2nd
  - COCO Segmentation: 12% better than 2nd

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

He et al. Deep Residual Learning for Image Recognition, IEEE CVPR 2016.

Source: cs231n

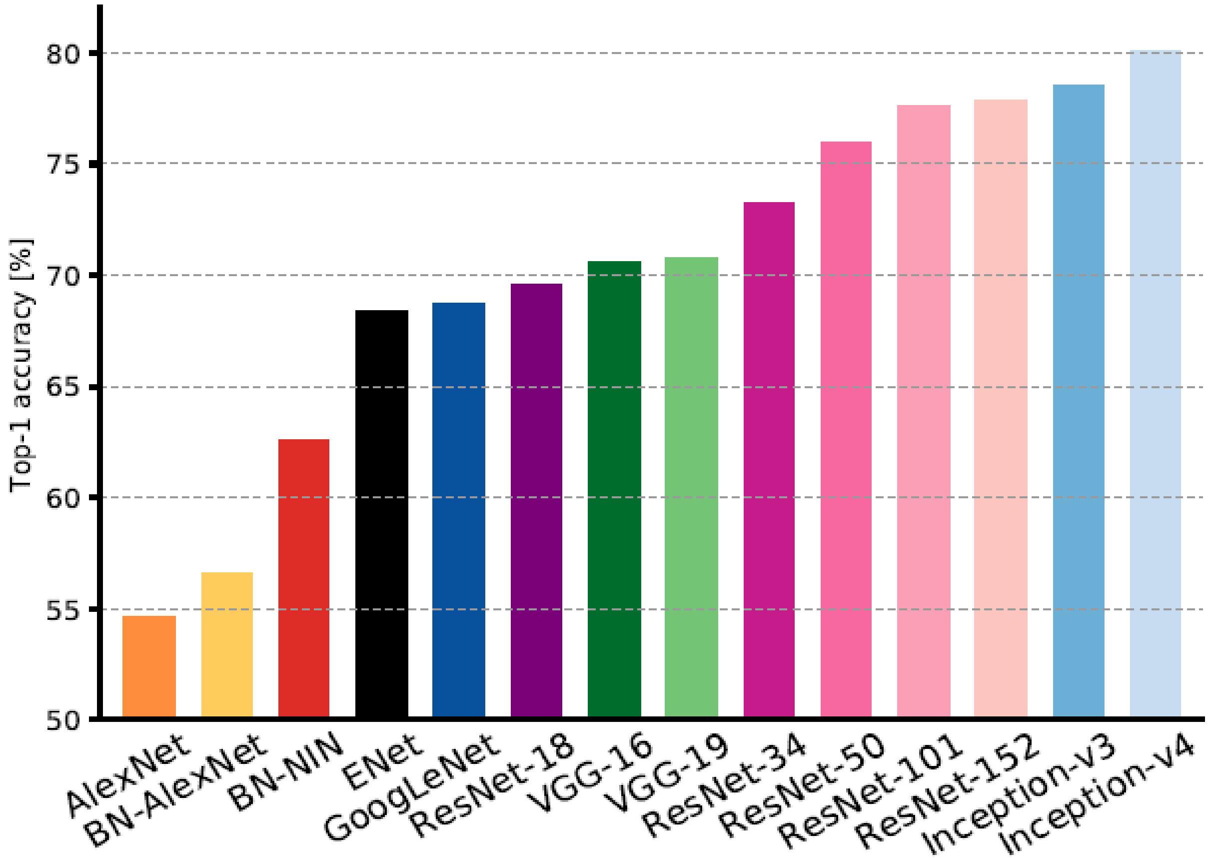
INCEPTION V4

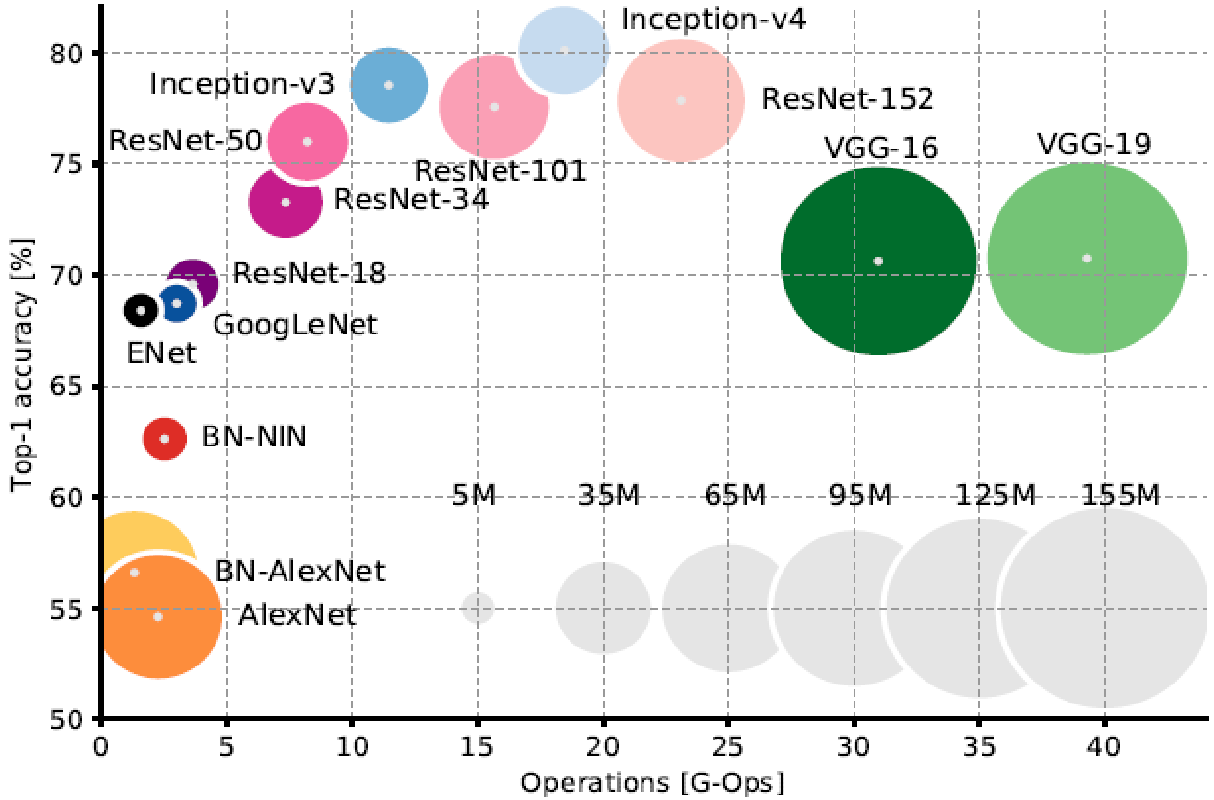


Inception-v4: Resnet + Inception!

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

COMPARING COMPLEXITY ...



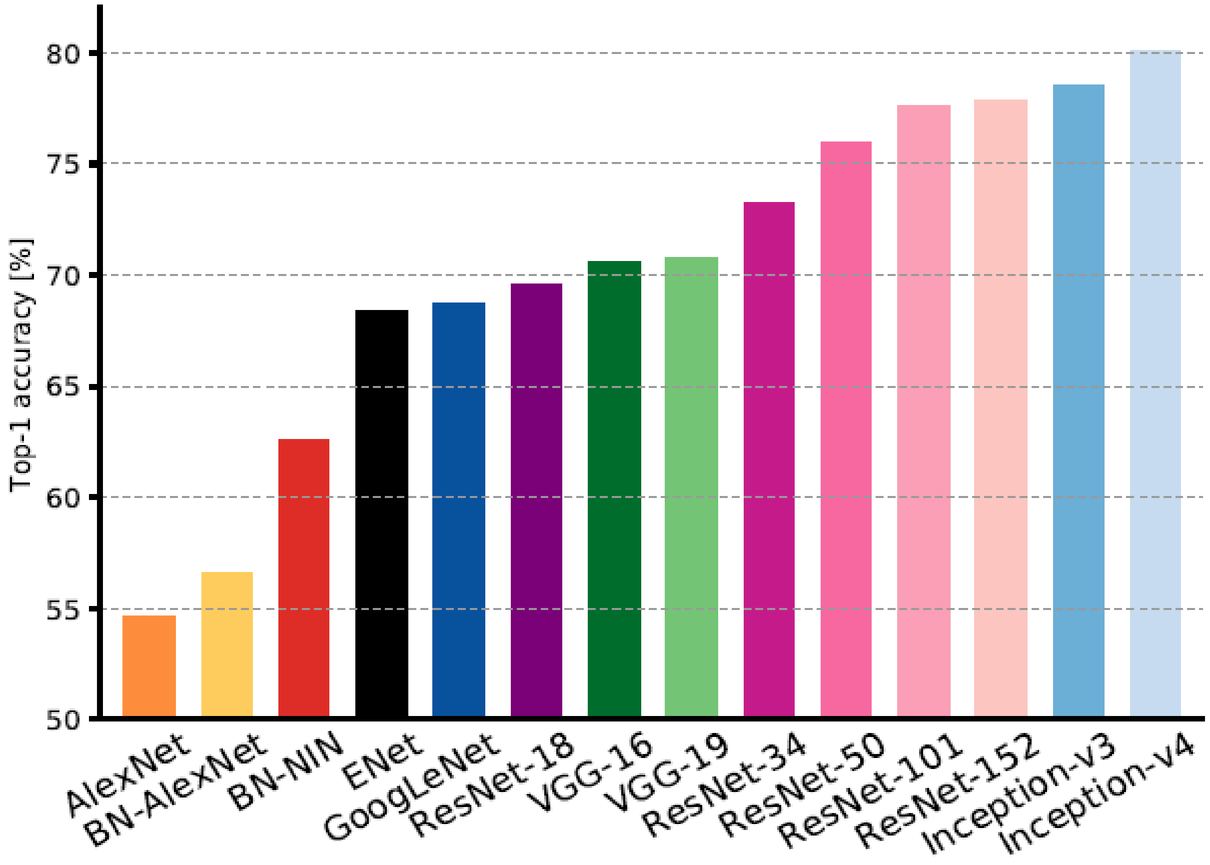


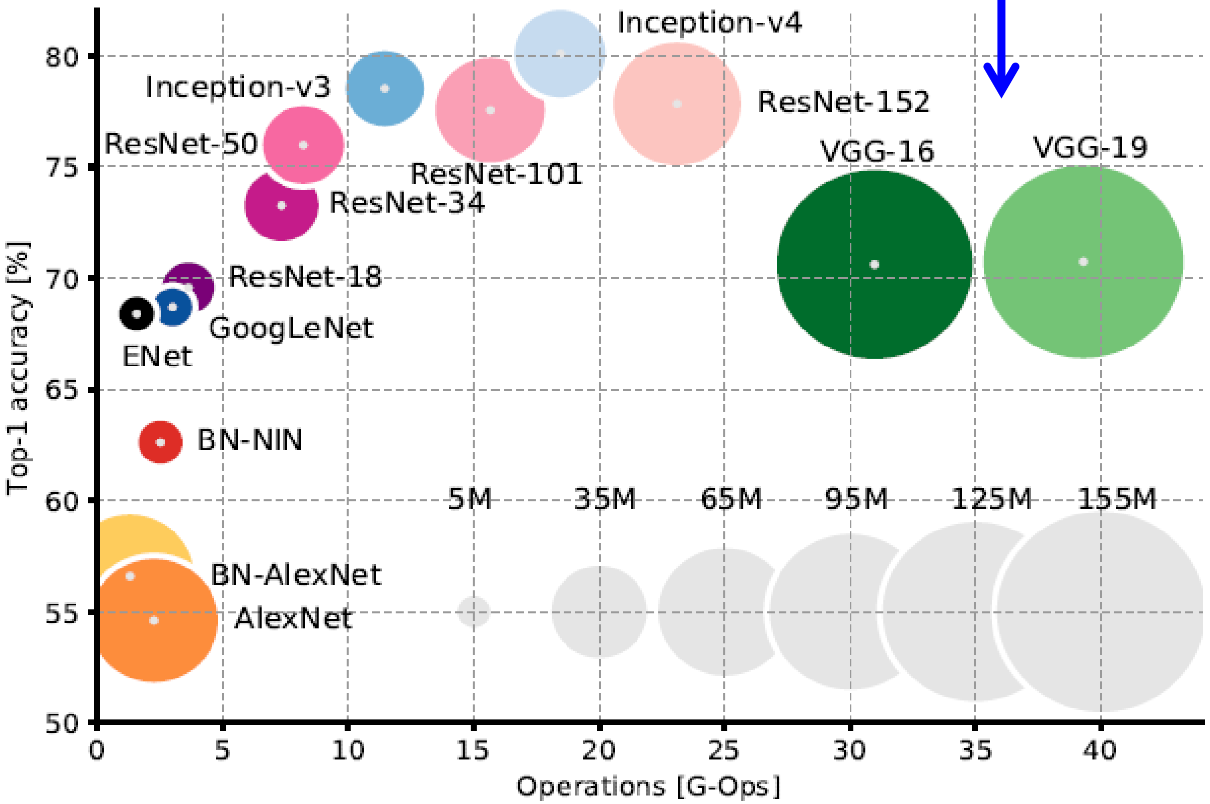
Canziani et al. An Analysis of Deep Neural Network Models for Practical Applications, 2017.

Source: cs231n

COMPARING COMPLEXITY ...

VGG: Highest memory, most operations



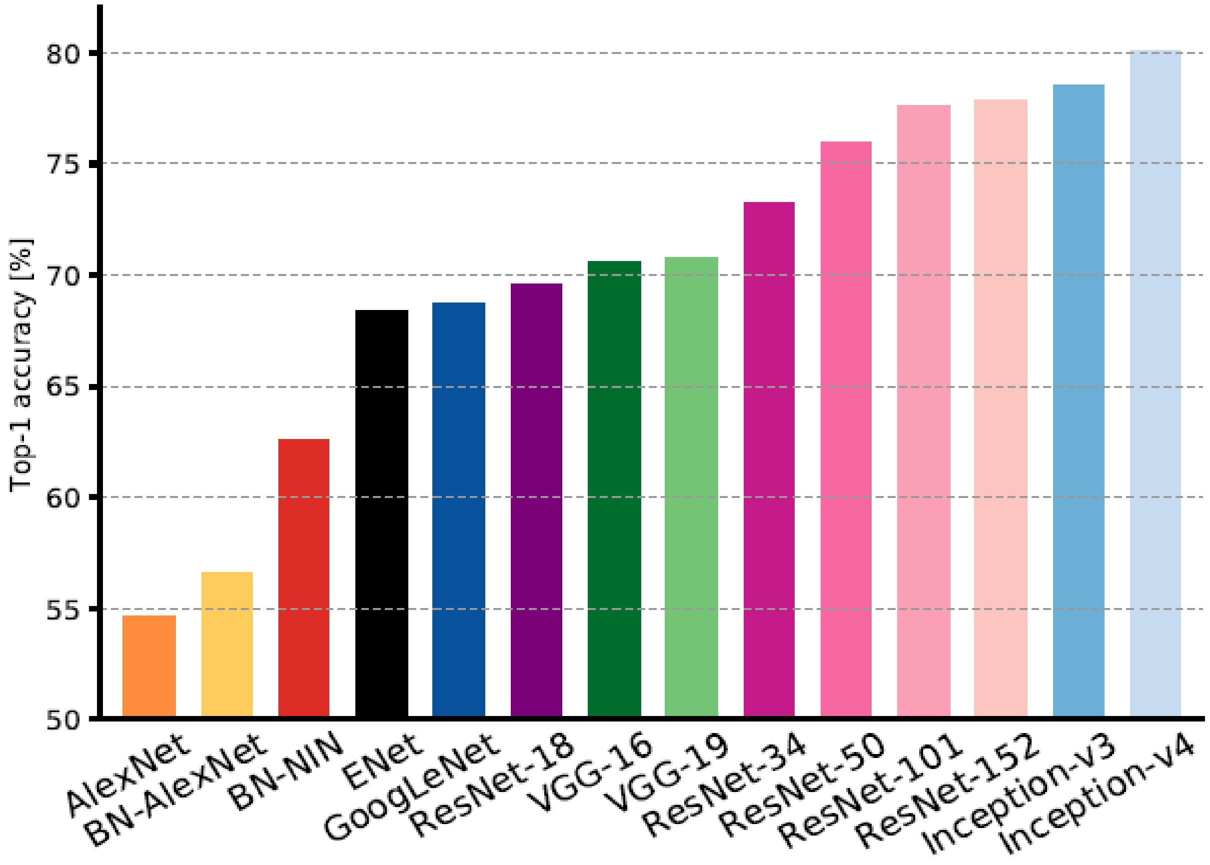


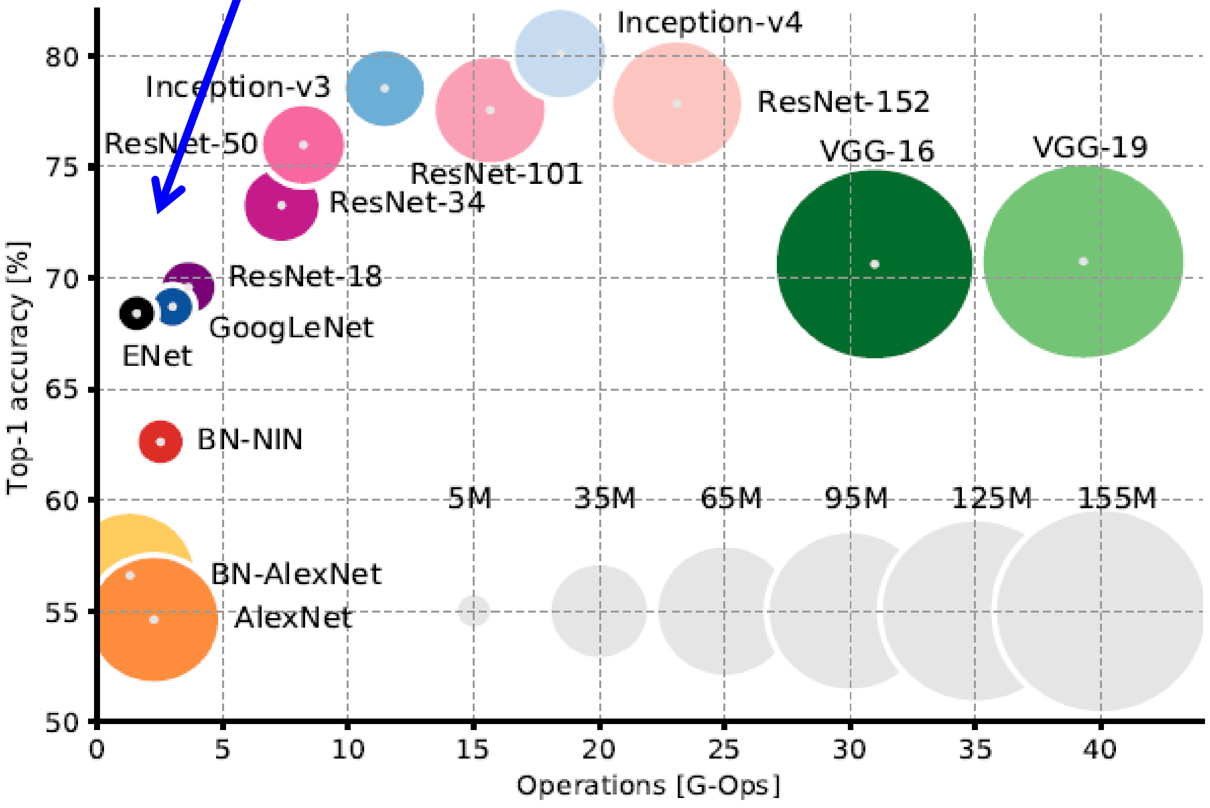
Canziani et al. An Analysis of Deep Neural Network Models for Practical Applications, 2017.

Source: cs231n

COMPARING COMPLEXITY ...

GoogLeNet:
most efficient



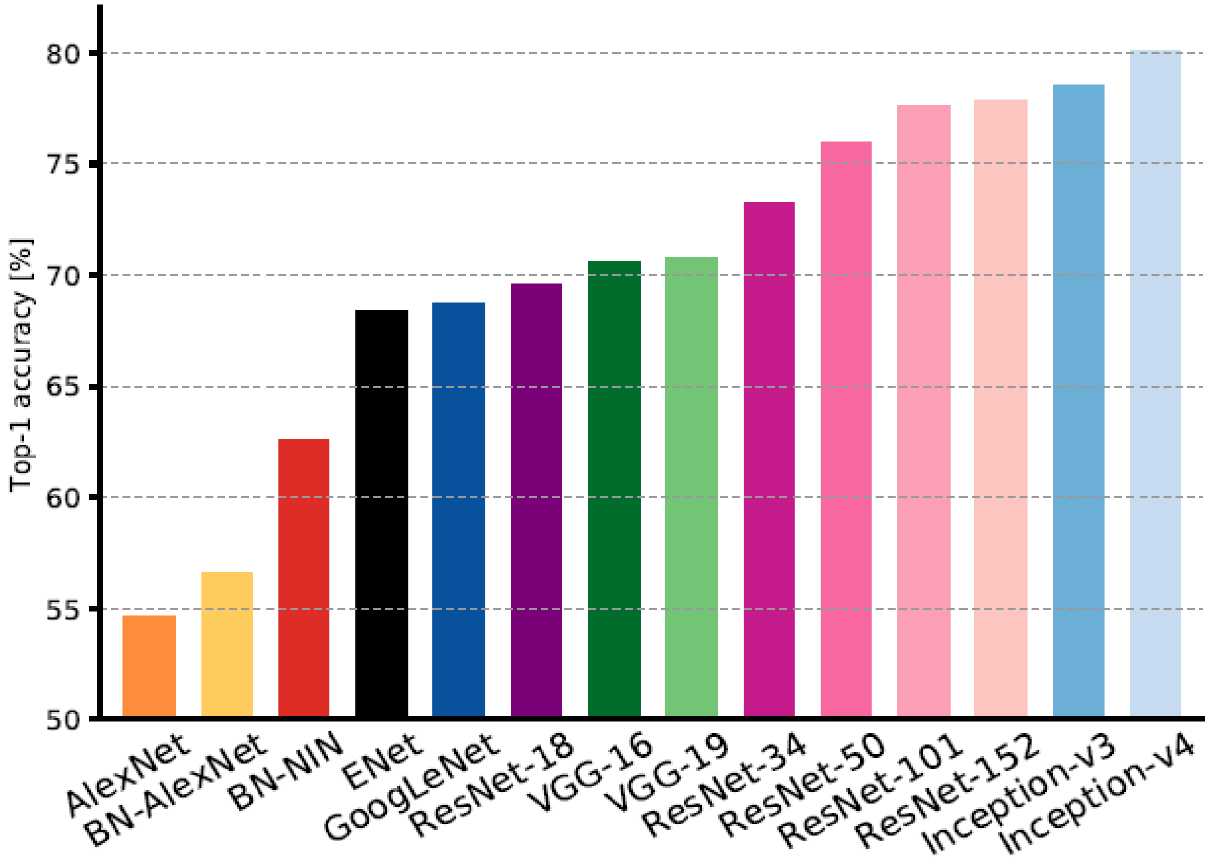


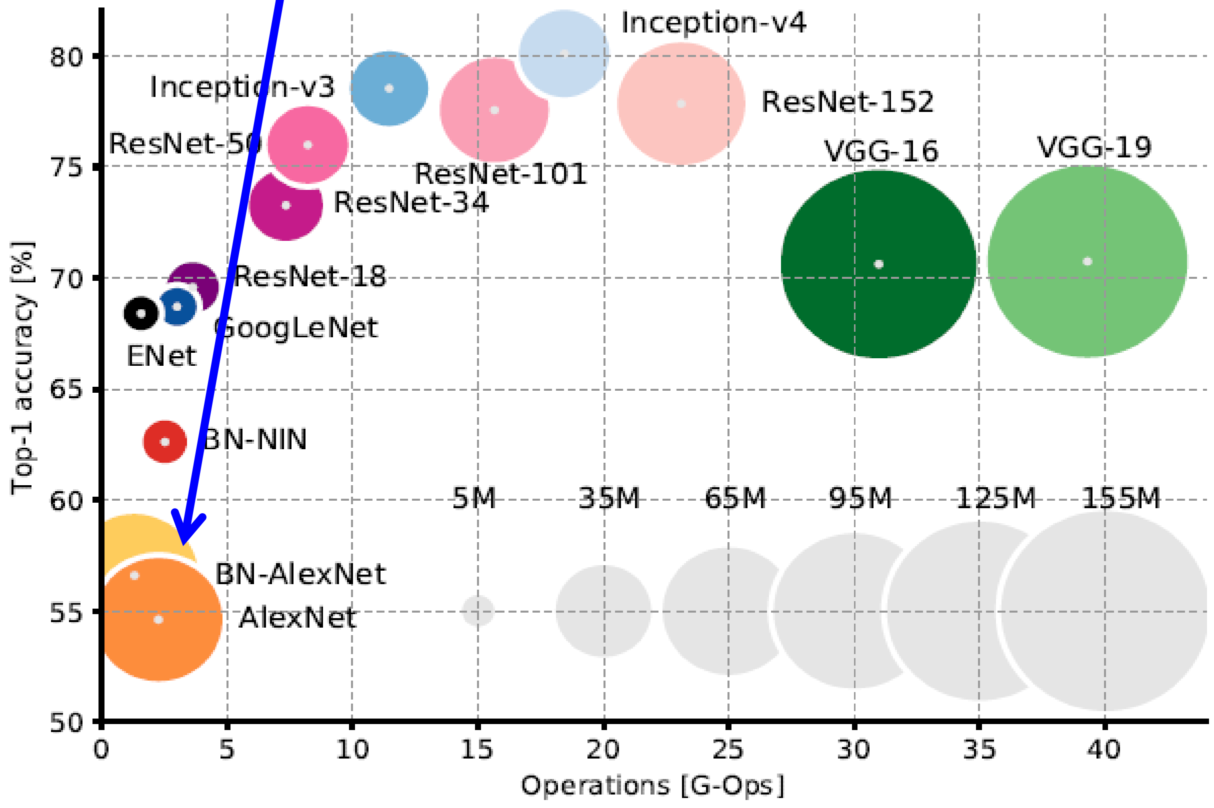
Canziani et al. An Analysis of Deep Neural Network Models for Practical Applications, 2017.

Source: cs231n

COMPARING COMPLEXITY …

AlexNet:
Smaller compute, still memory heavy, lower accuracy



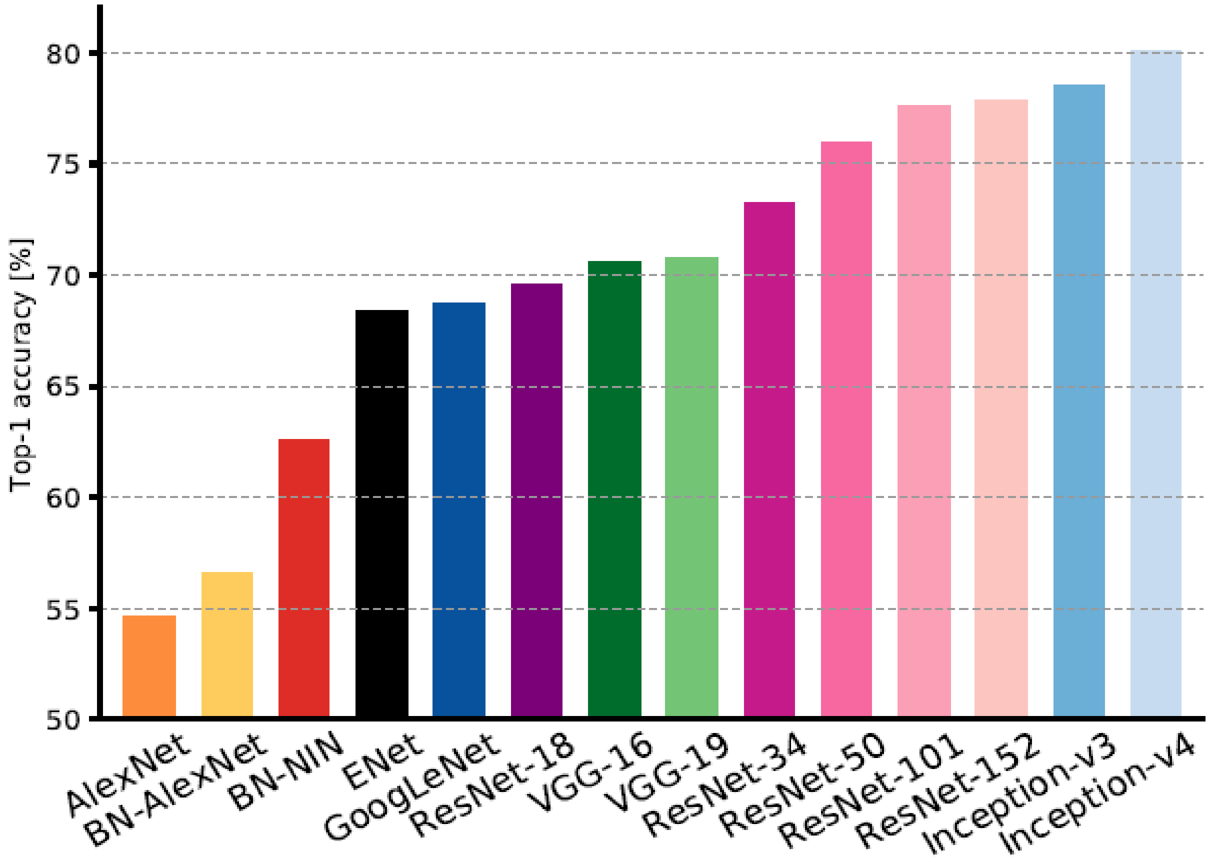


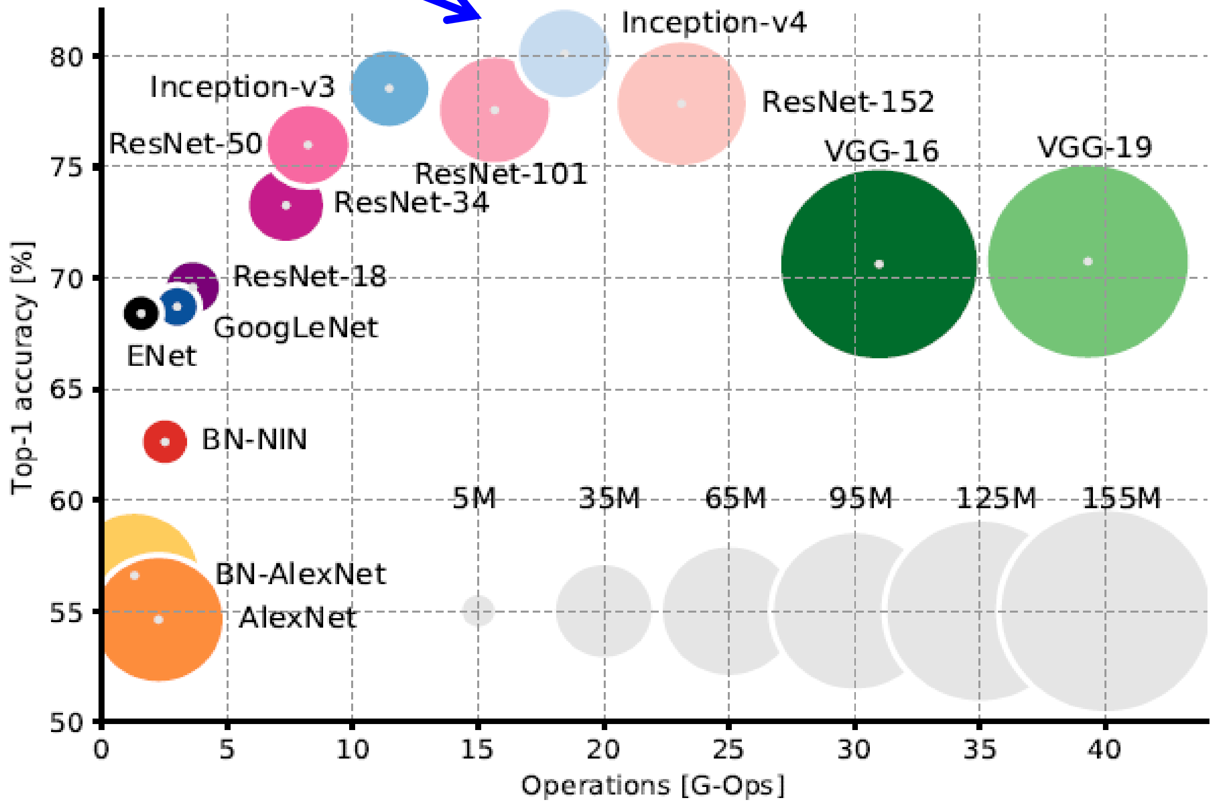
Canziani et al. An Analysis of Deep Neural Network Models for Practical Applications, 2017.

Source: cs231n

COMPARING COMPLEXITY ...

ResNet:
Moderate efficiency depending on model, highest accuracy

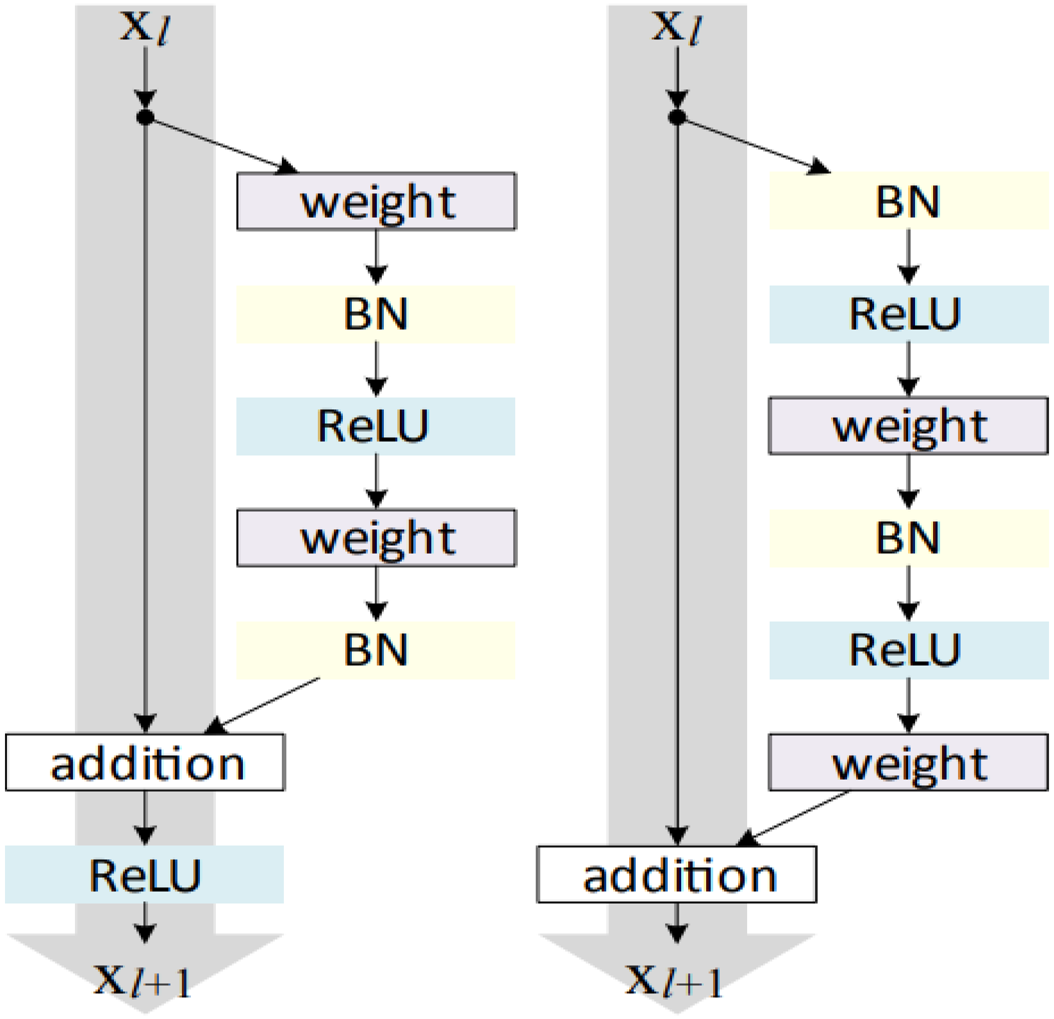




Canziani et al. An Analysis of Deep Neural Network Models for Practical Applications, 2017.

Source: cs231n

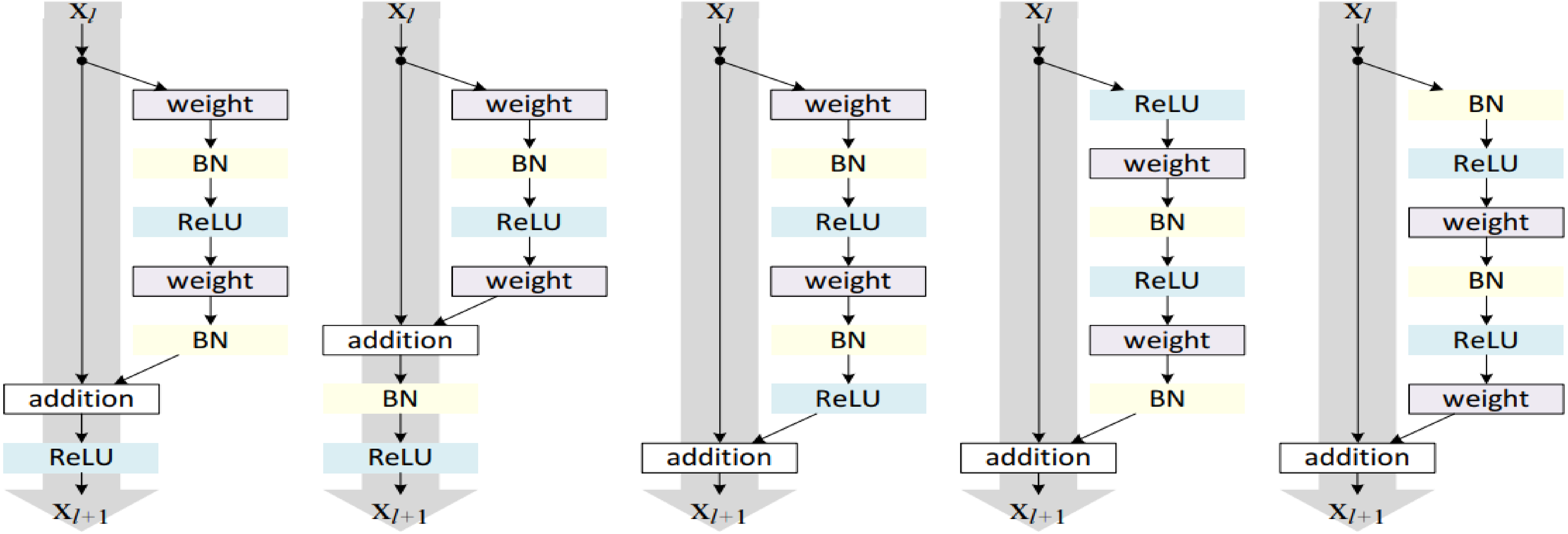
PRE-ACTIVATED RESNET



(a) original (b) proposed

He, Kaiming, et al. "Identity mappings in deep residual networks." *Europ. conf. on computer vision* (ECCV), 2016.

PRE-ACTIVATED RESNET



(a) original (b) BN after addition (c) ReLU before addition (d) ReLU-only pre-activation (e) **full pre-activation**

He, Kaiming, et al. "Identity mappings in deep residual networks." *Europ. conf. on computer vision* (ECCV), 2016.

PRE-ACTIVATED RESNET

Classification error (%) on the CIFAR-10 test set using different activation functions.

case	ResNet-110	ResNet-164
original Residual Unit [1]	6.61	5.93
BN after addition	8.17	6.50
ReLU before addition	7.84	6.14
ReLU-only pre-activation	6.71	5.91
full pre-activation	**6.37**	**5.46**

He, Kaiming, et al. "Identity mappings in deep residual networks." *Europ. conf. on computer vision* (ECCV), 2016.

SENET (SQUEEZE AND EXCITATION NETWORK)

2017 ImageNet Challenge Winner

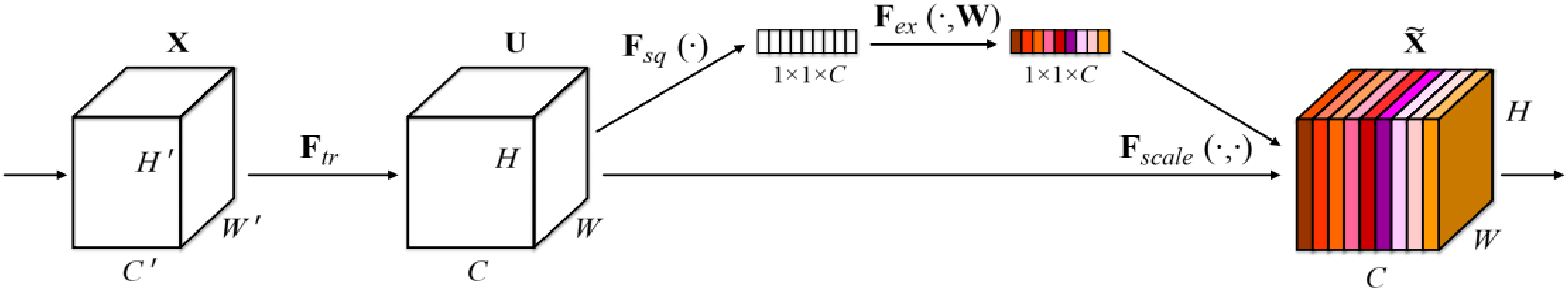
Top-5 Error: 2.251%

Hu et al. Squeeze-and-Excitation Networks, CVPR 2018.

SENET (SQUEEZE AND EXCITATION NETWORK)

2017 ImageNet Challenge Winner

Top-5 Error: 2.251%



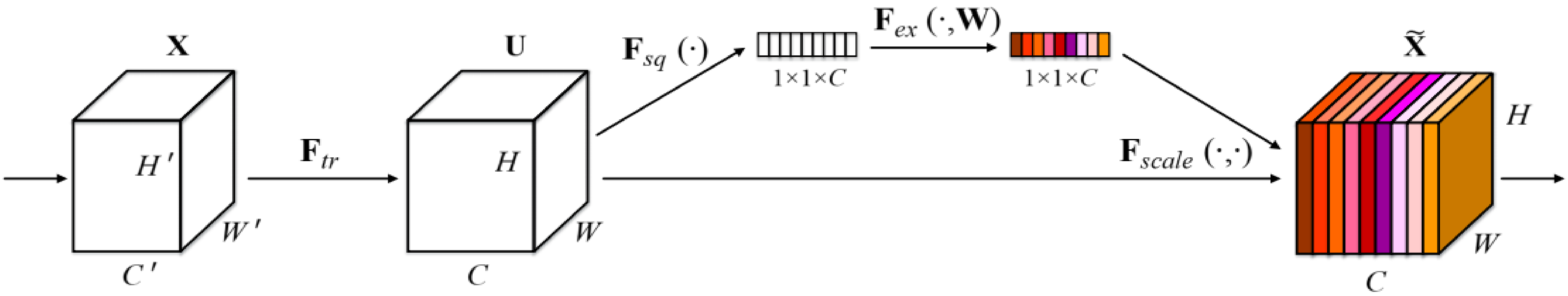
"Squeeze-and-Excitation"(SE) block adaptively recalibrates channel-wise feature responses by explicitly modelling interdependencies between channels.

Hu et al. Squeeze-and-Excitation Networks, CVPR 2018.

SENET (SQUEEZE AND EXCITATION NETWORK)

2017 ImageNet Challenge Winner

Top-5 Error: 2.251%



Squeeze: Average Global Pooling

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

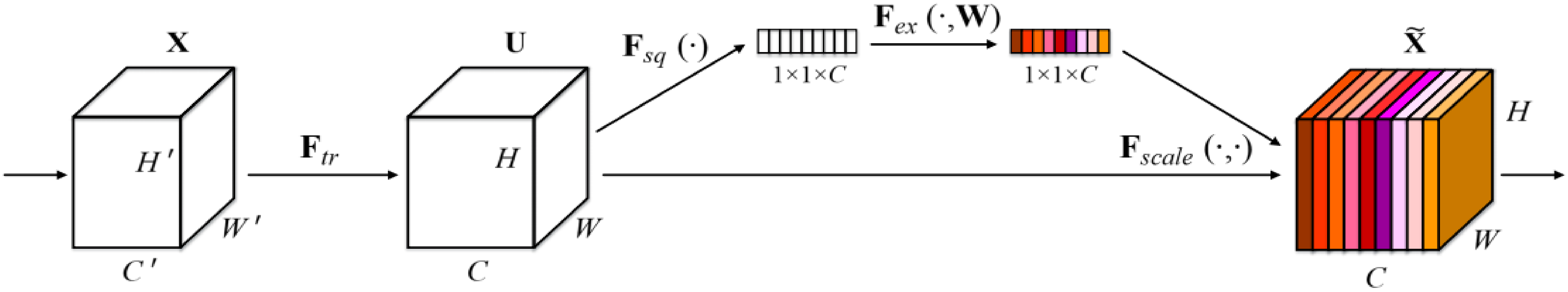
c^{th} channel of C

Hu et al. Squeeze-and-Excitation Networks, CVPR 2018.

SENET (SQUEEZE AND EXCITATION NETWORK)

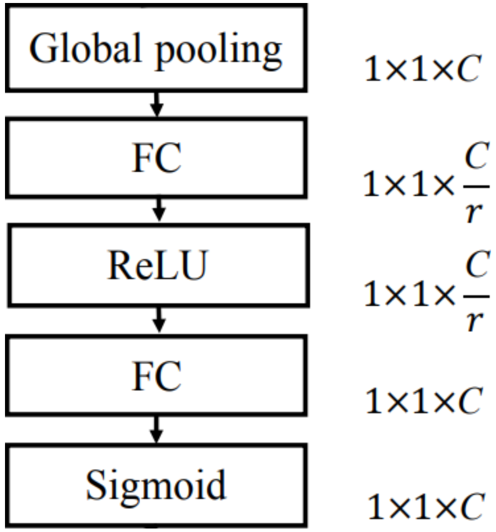
2017 ImageNet Challenge Winner

Top-5 Error: 2.251%



Excitation:

Adaptive Recalibration

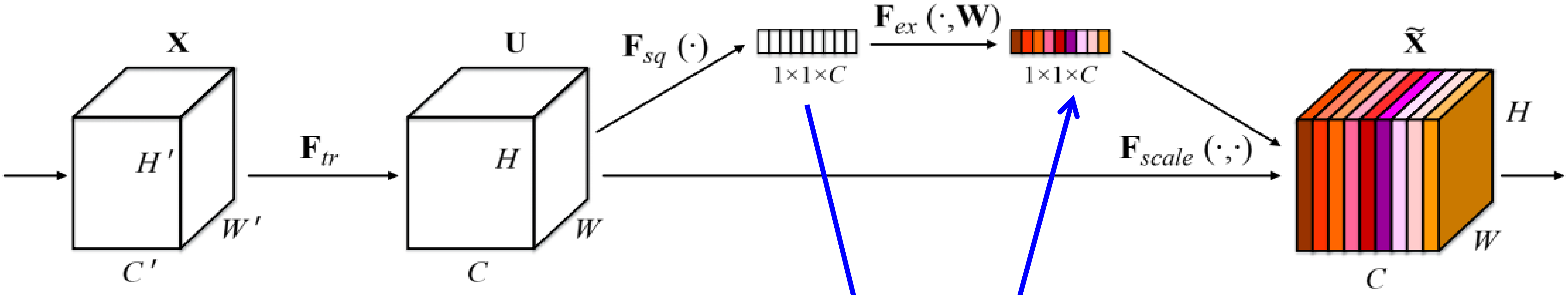


Hu et al. Squeeze-and-Excitation Networks, CVPR 2018.

SENET (SQUEEZE AND EXCITATION NETWORK)

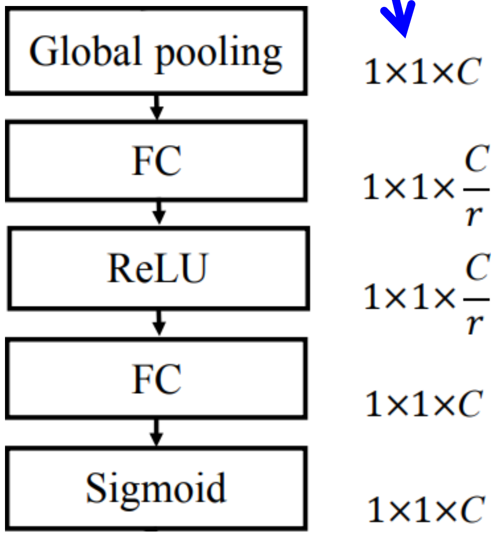
2017 ImageNet Challenge Winner

Top-5 Error: 2.251%



Excitation:

Adaptive Recalibration

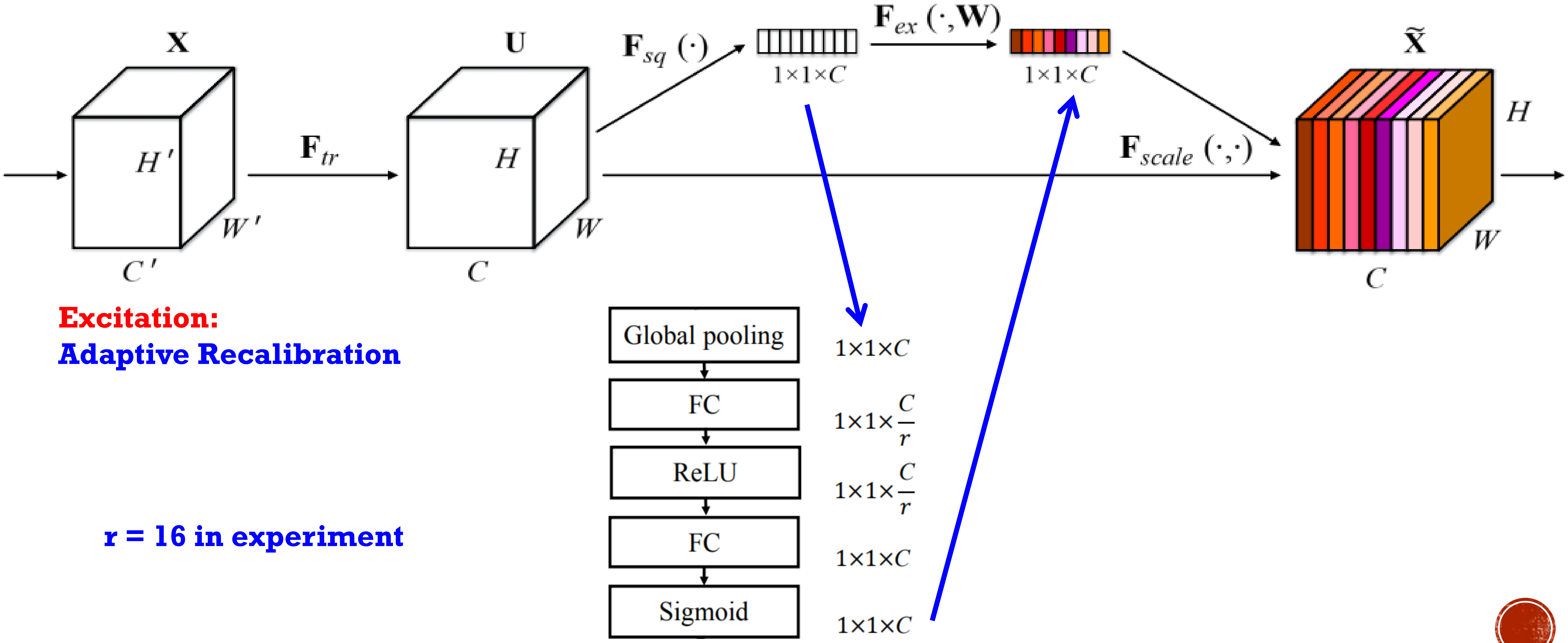


Hu et al. Squeeze-and-Excitation Networks, CVPR 2018.

SENET (SQUEEZE AND EXCITATION NETWORK)

2017 ImageNet Challenge Winner

Top-5 Error: 2.251%



Excitation:

Adaptive Recalibration

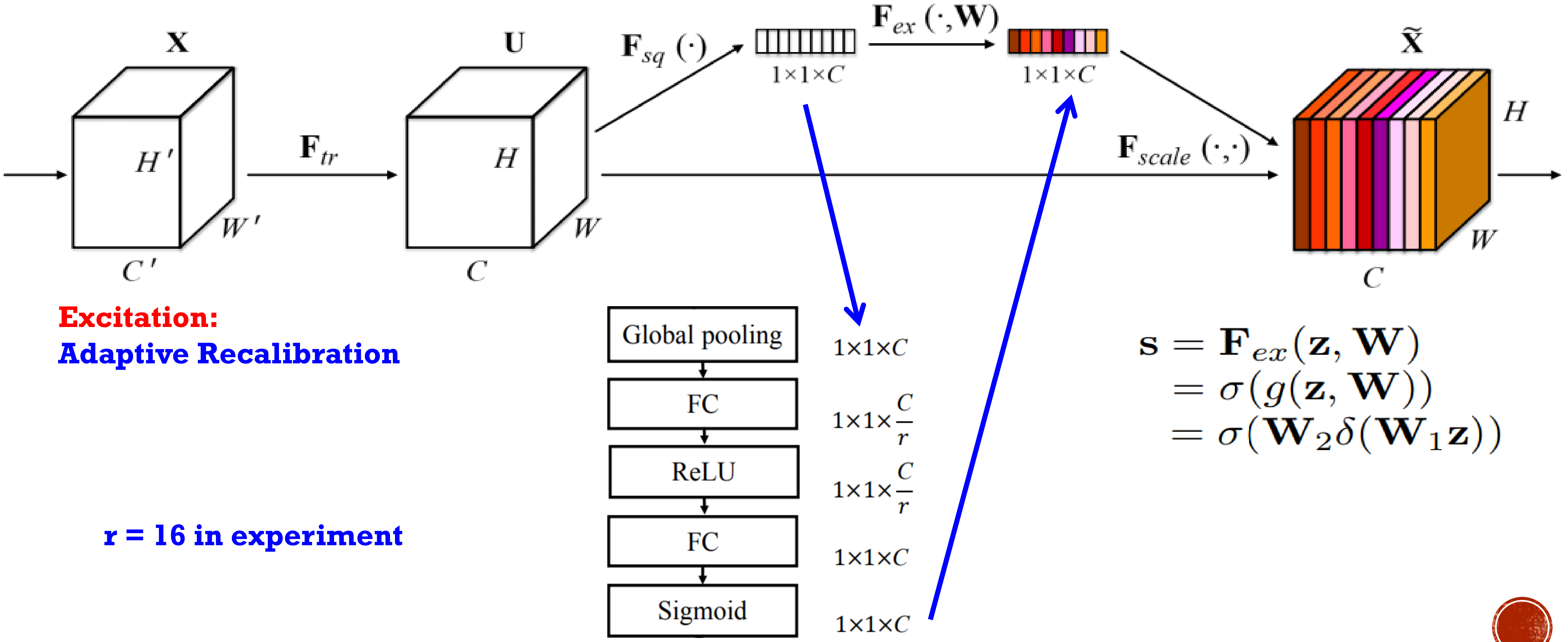
r = 16 in experiment

Hu et al. Squeeze-and-Excitation Networks, CVPR 2018.

SENET (SQUEEZE AND EXCITATION NETWORK)

2017 ImageNet Challenge Winner

Top-5 Error: 2.251%



Excitation:

Adaptive Recalibration

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

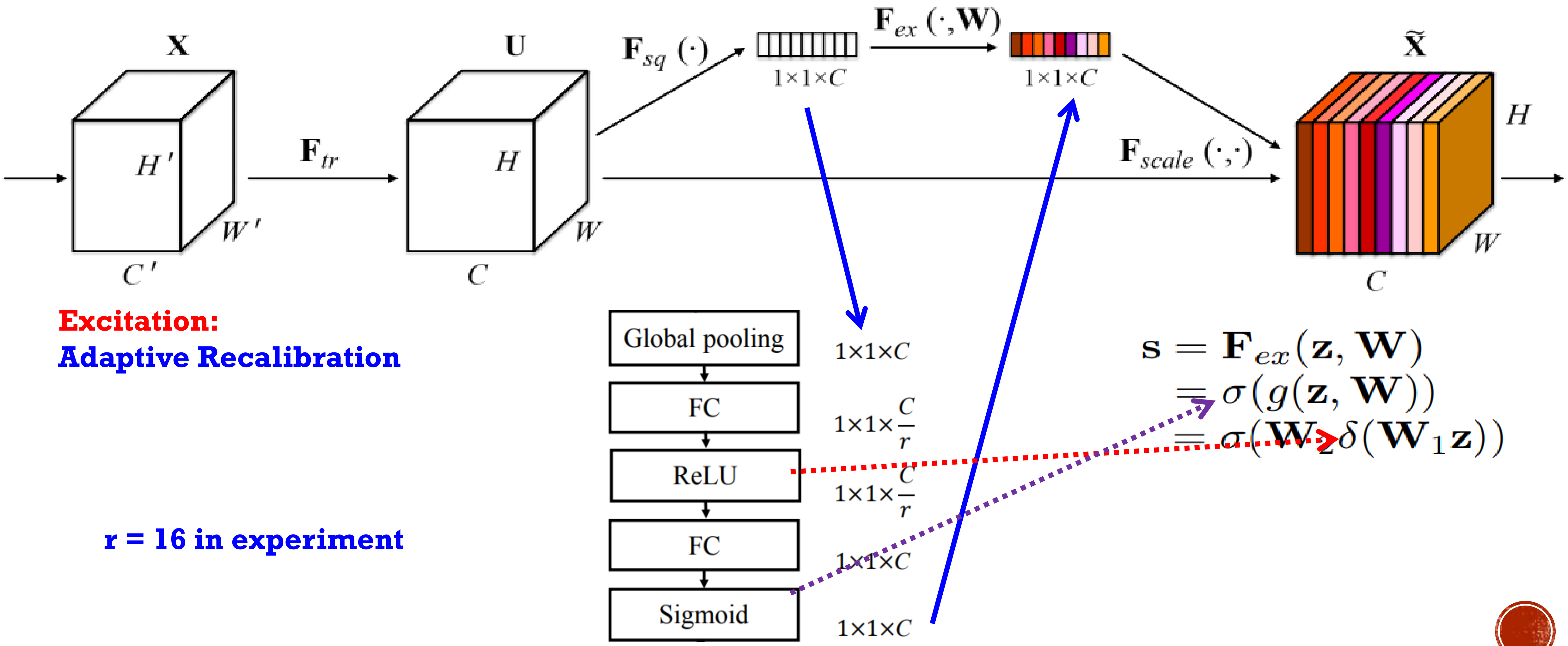
r = 16 in experiment

Hu et al. Squeeze-and-Excitation Networks, CVPR 2018.

SENET (SQUEEZE AND EXCITATION NETWORK)

2017 ImageNet Challenge Winner

Top-5 Error: 2.251%



Excitation:

Adaptive Recalibration

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

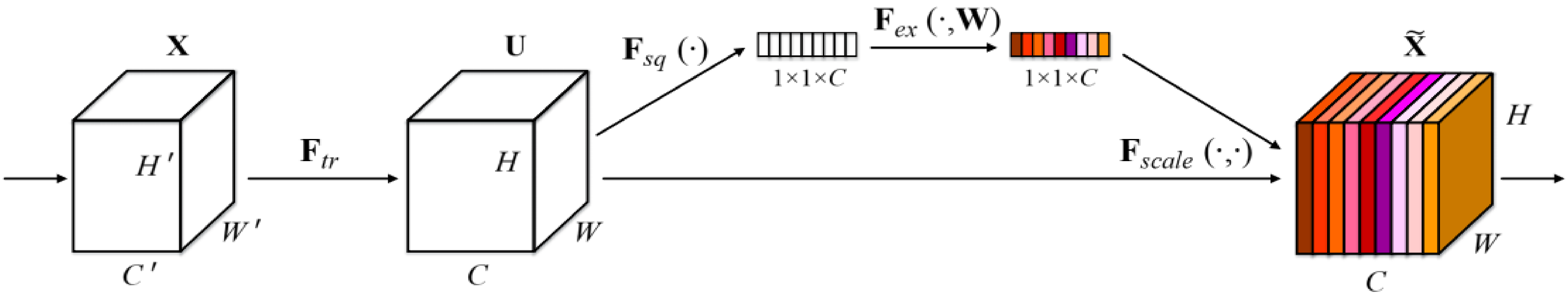
r = 16 in experiment

Hu et al. Squeeze-and-Excitation Networks, CVPR 2018.

SENET (SQUEEZE AND EXCITATION NETWORK)

2017 ImageNet Challenge Winner

Top-5 Error: 2.251%

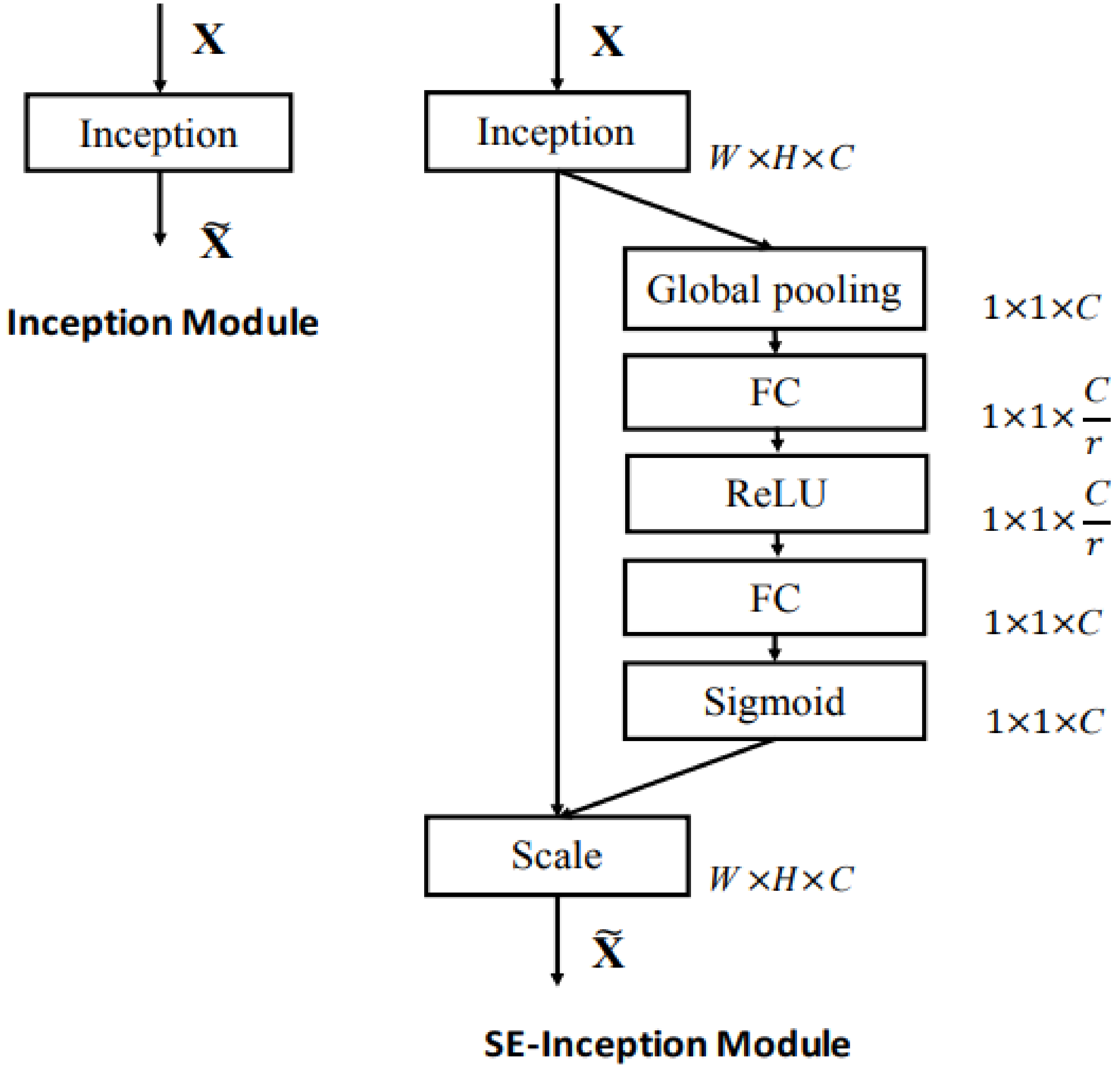


Scaling:

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

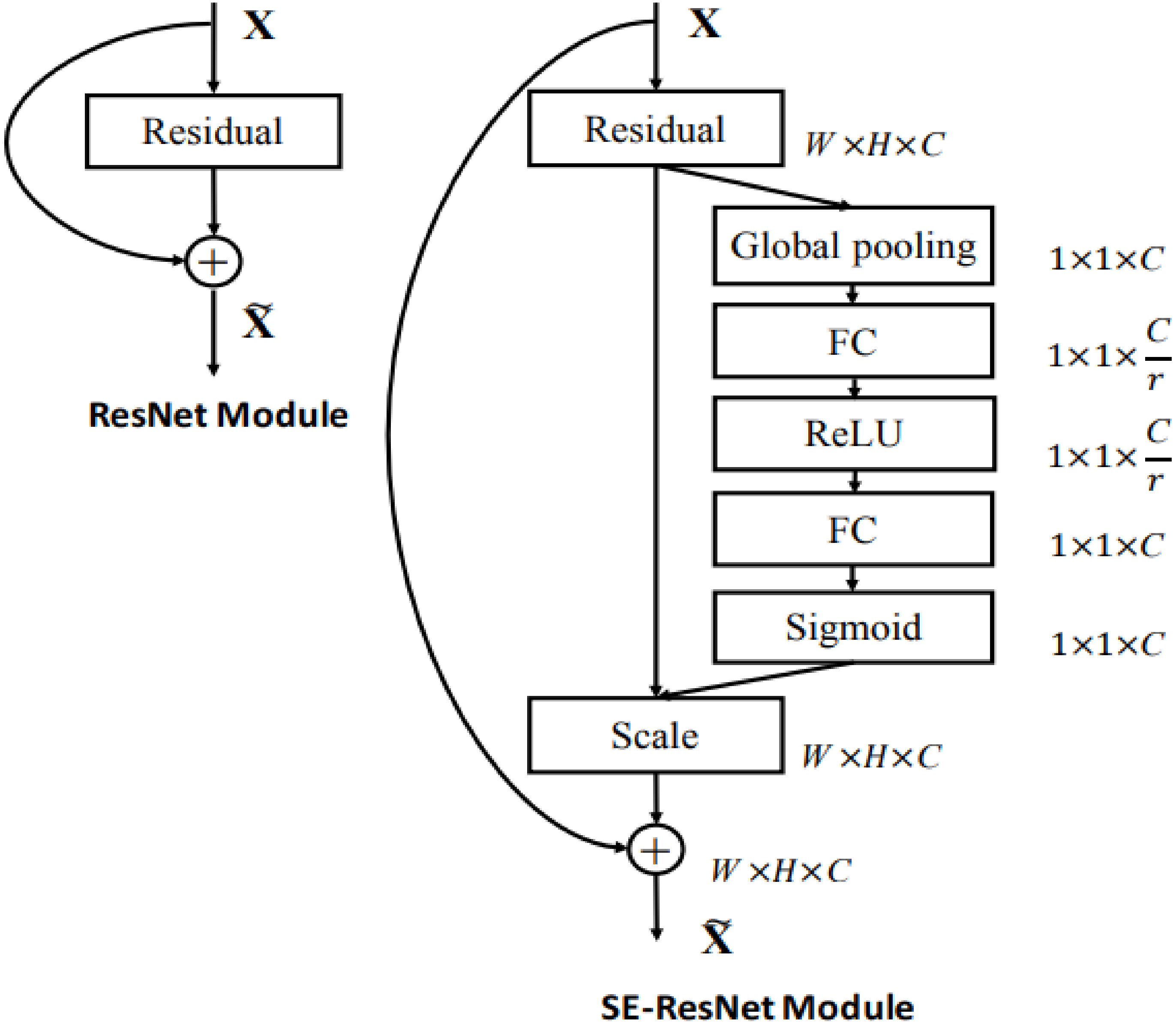
Hu et al. Squeeze-and-Excitation Networks, CVPR 2018.

SE-INCEPTION MODULE



Hu et al. Squeeze-and-Excitation Networks, CVPR 2018.

SE-RESNET MODULE



Hu et al. Squeeze-and-Excitation Networks, CVPR 2018.

OTHER RESNET IMPROVEMENTS TO KNOW …

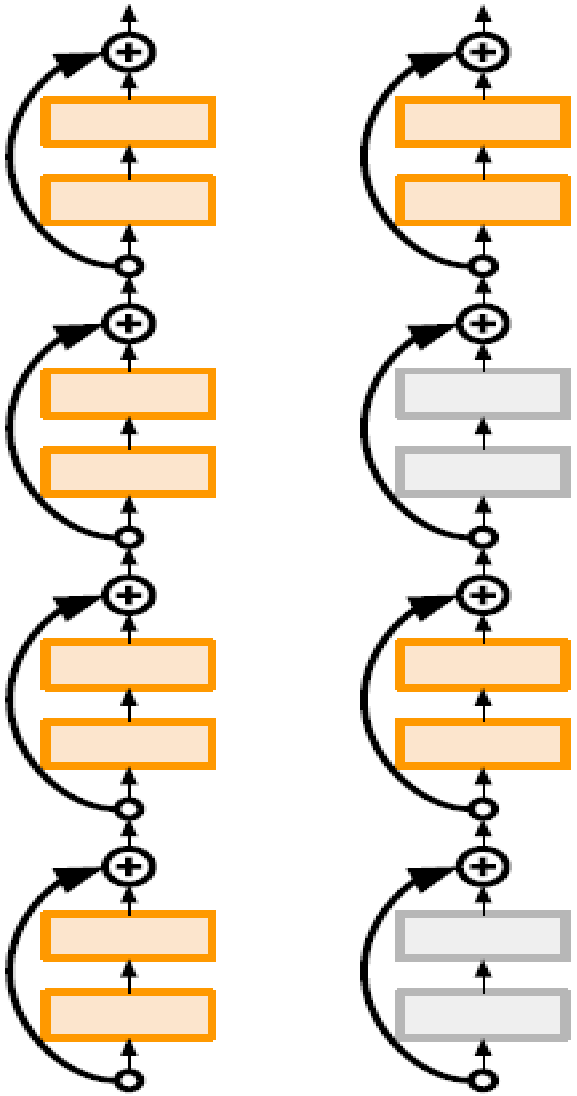
DEEP NETWORKS WITH STOCHASTIC DEPTH

- 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



Huang et al. Deep networks with stochastic depth. ECCV 2016.

Source: cs231n

WIDE RESNET

- Reduce number of residual blocks, but increase number of feature maps in each block
    - More parallelizable, better feature reuse
    - 16-layer WRN outperforms 1000-layer ResNets, though with much larger # of parameters

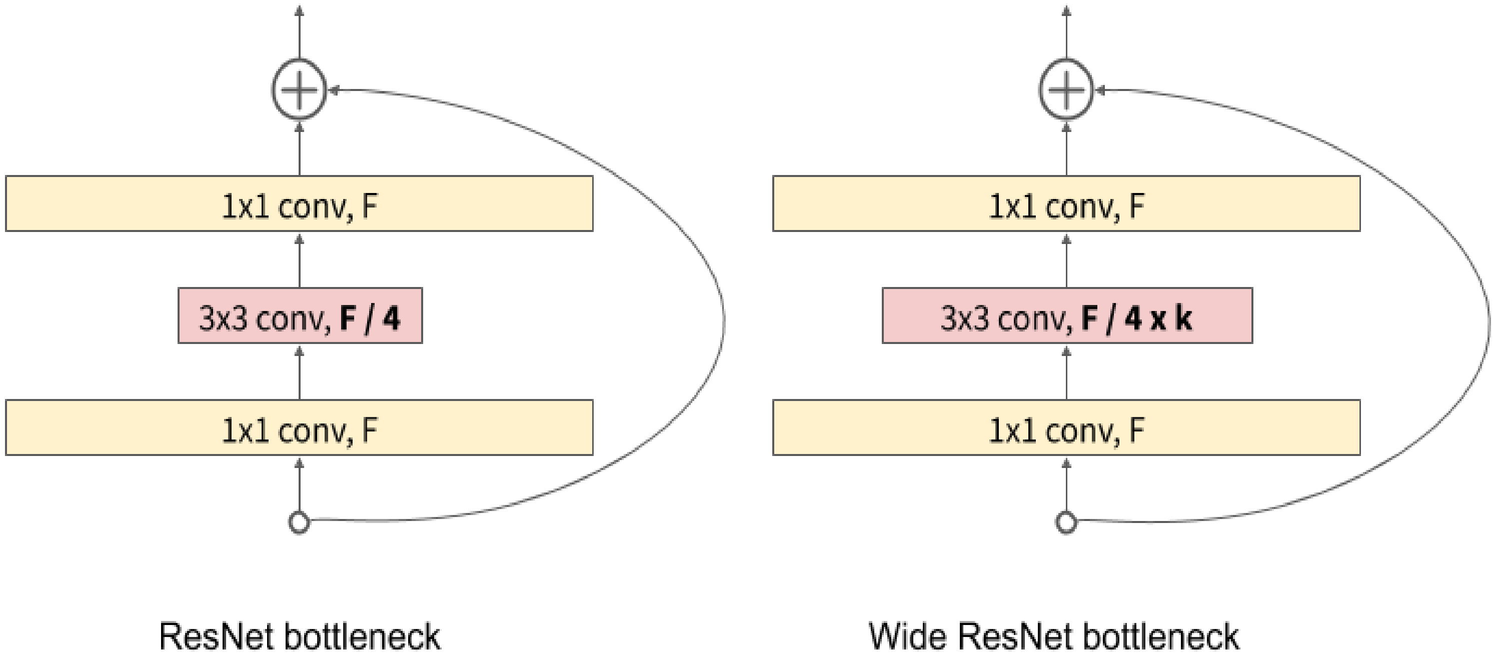
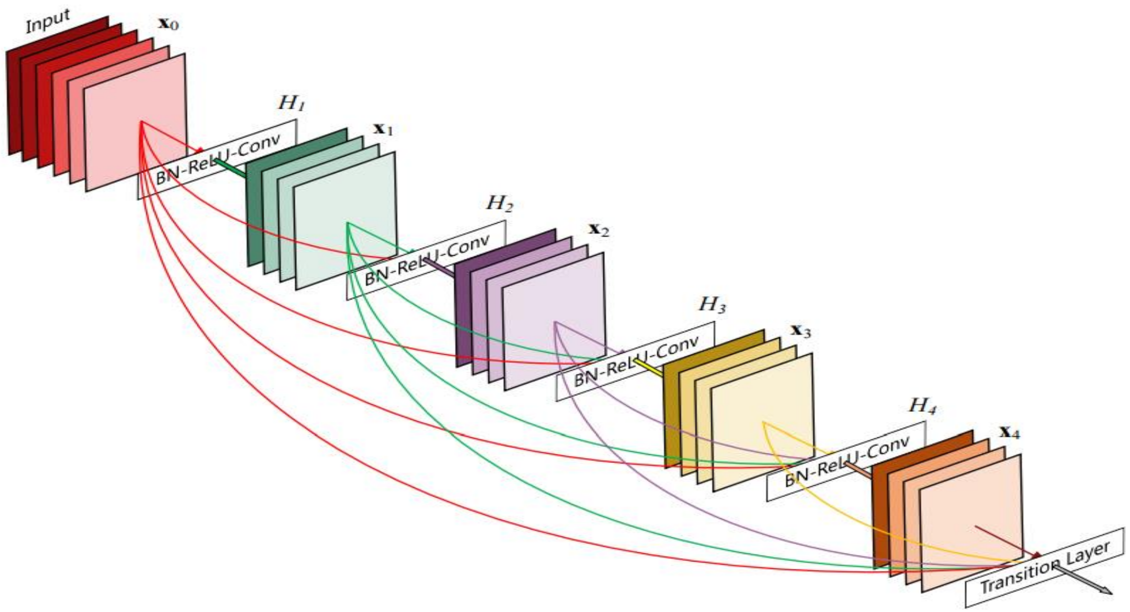


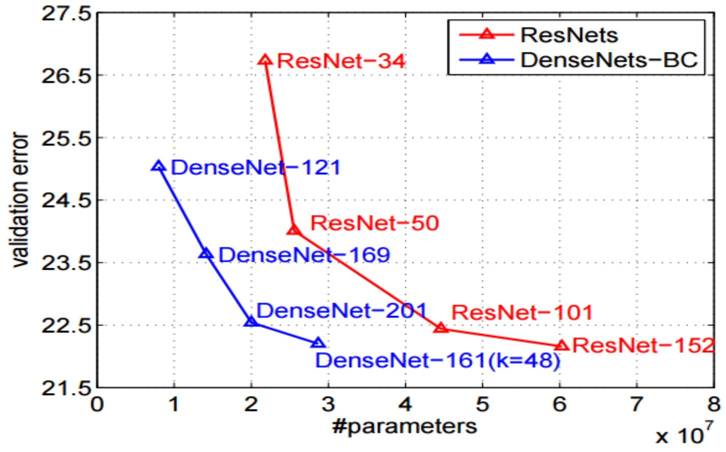
Image source

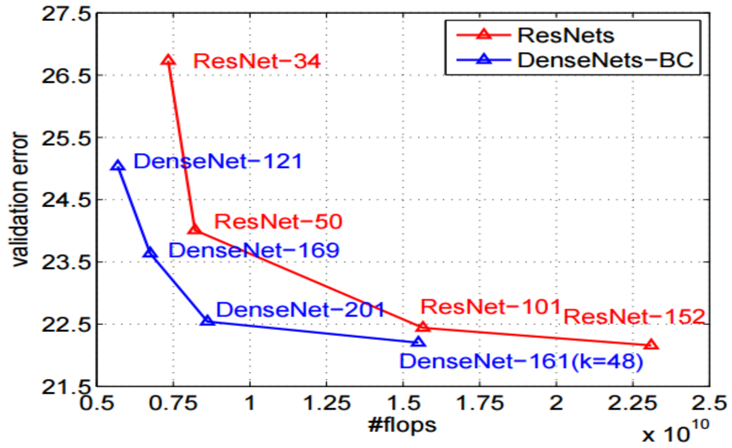
S. Zagoryuko and N. Komodakis, Wide Residual Networks, BMVC 2016

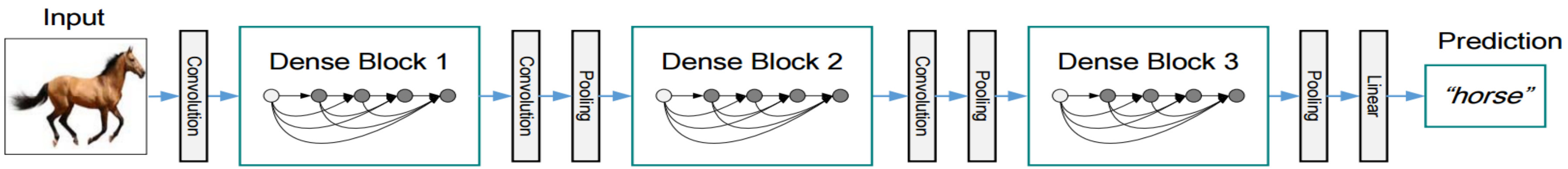
DENSENET

- Shorter connections (like ResNet) help
- Why not just connect them all?





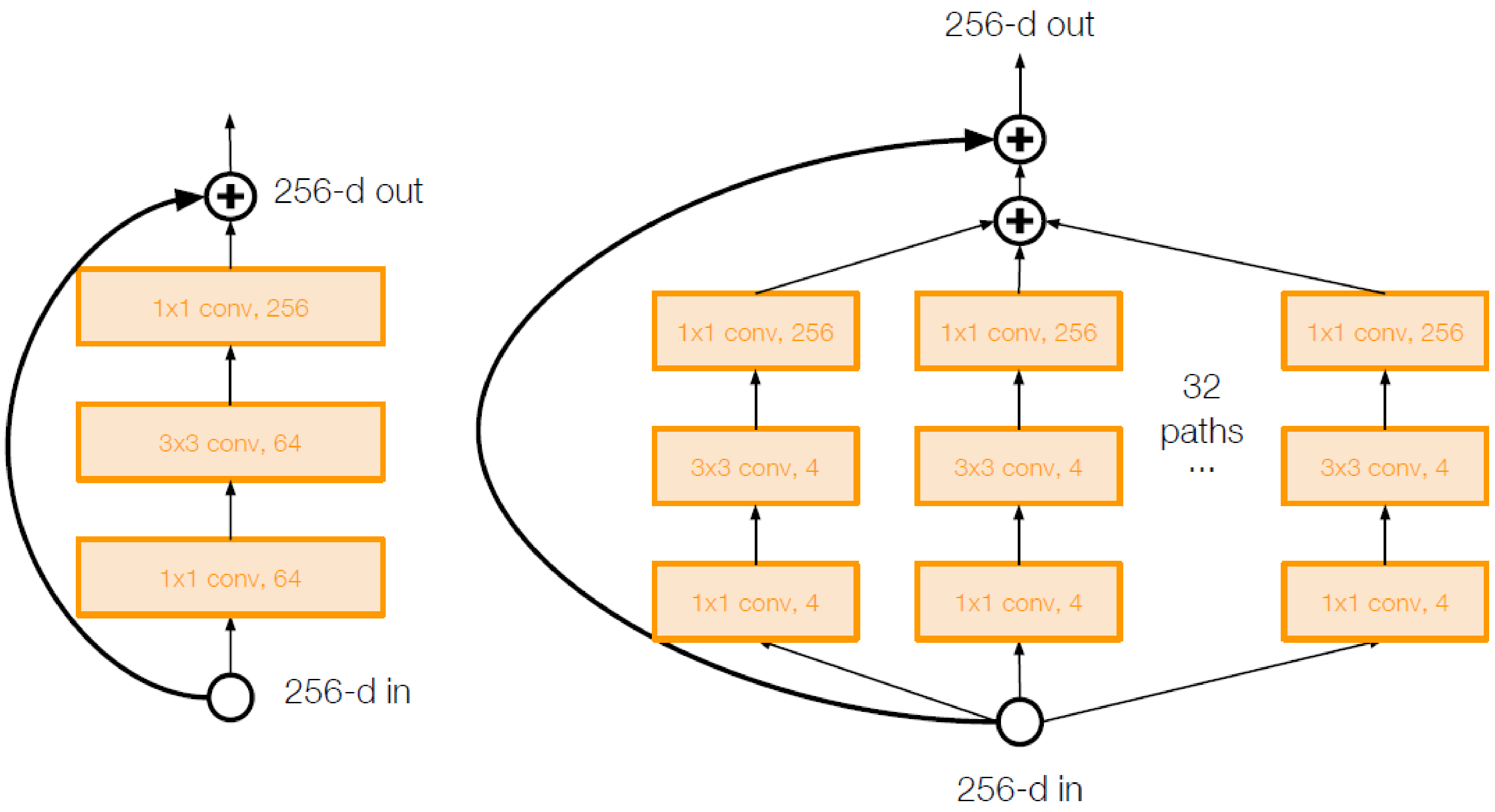




Huang et al. Densely connected convolutional networks. CVPR 2017.

AGGREGATED RESIDUAL TRANSFORMATIONS FOR DEEP NEURAL NETWORKS (RESNEXT)

- Improved ResNet
- Increases width of residual block through multiple parallel pathways ("cardinality")
- Parallel pathways similar in spirit to Inception module



Xie et al. Aggregated residual transformations for deep neural networks. CVPR 2017.

Source: cs231n

DESIGN PRINCIPLES

- Make networks parameter-efficient
  - Reduce filter sizes, factorize filters
  - Use 1x1 convolutions to reduce number of feature maps before more expensive operations
  - Minimize reliance on FC layers
- Reduce spatial resolution gradually, within each level of resolution replicate a given "block" multiple times
- Use skip connections and/or create multiple redundant paths through the network
- Play around with depth vs. width vs. "cardinality"

ACKNOWLEDGEMENT

- Deep Learning, Stanford University
- Introduction to Deep Learning, University of Illinois at Urbana-Champaign
- Introduction to Deep Learning, Carnegie Mellon University
- Convolutional Neural Networks for Visual Recognition, Stanford University
- Natural Language Processing with Deep Learning, Stanford University
- NVDIEA Deep Learning Teaching Kit
- And Many More …