Advanced Architectures
Object Detection & Segmentation
Bright & Dark Side of Context

Prof. Dr. Mario Fritz
CISPA Helmholtz Center for Information Security

Research

CISPA HELMHOLTZ CENTER FOR INFORMATION SECURITY

Computer Vision

- Scene Understanding
- Segmentation
- QA and Captioning

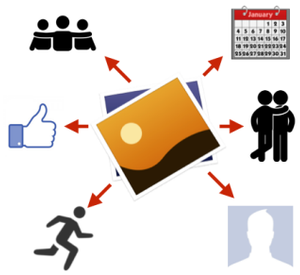
Ours: a skier is headed down a steep slope

What sport is this man enjoying?

snowboarding

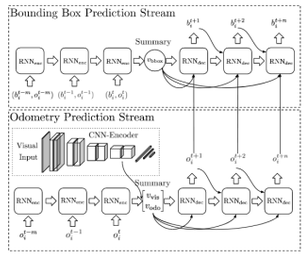
Security & Privacy

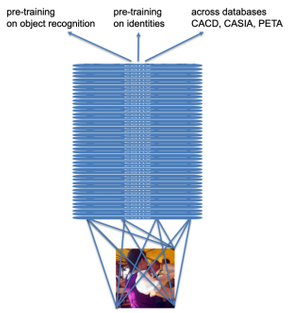
- Trustworthy AI
- Understanding & Controlling Privacy of Data & Models
- Adversarial Machine Learning/Attacks
- AI/ML for Cybersecurity
- Detecting Deep Fakes



Machine Learning

- Deep Learning
- Domain Adaptation
- Generative Adversarial Networks, Variational Autoencoders
- Uncertainty
- Explainability

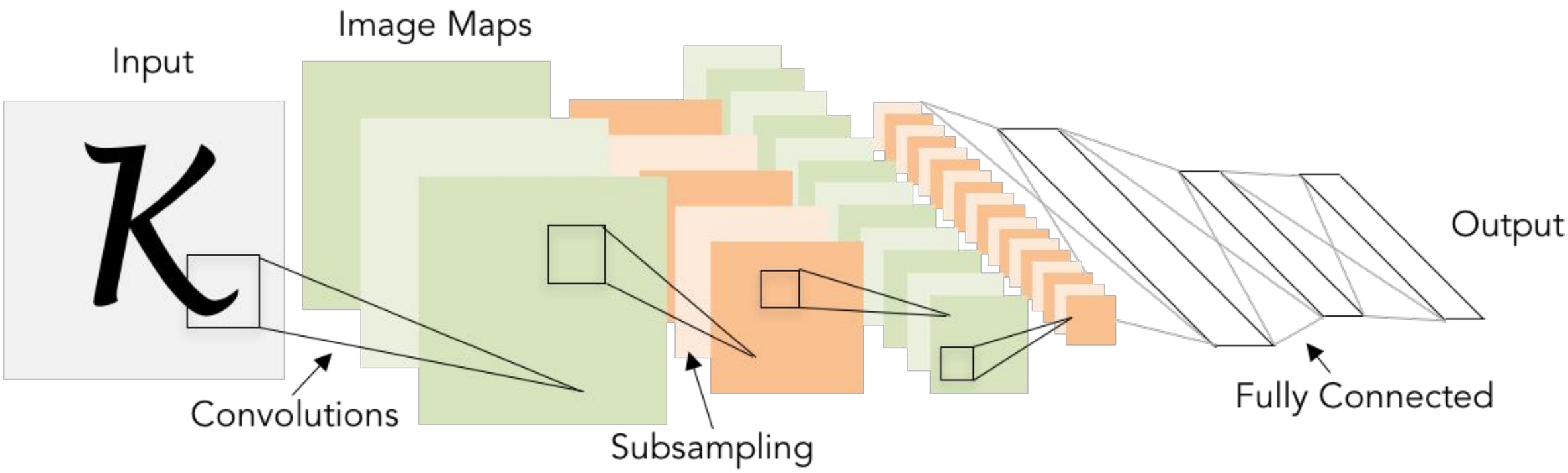




https://fritz.cispa.saarland

Review: LeNet-5

[LeCun et al., 1998]

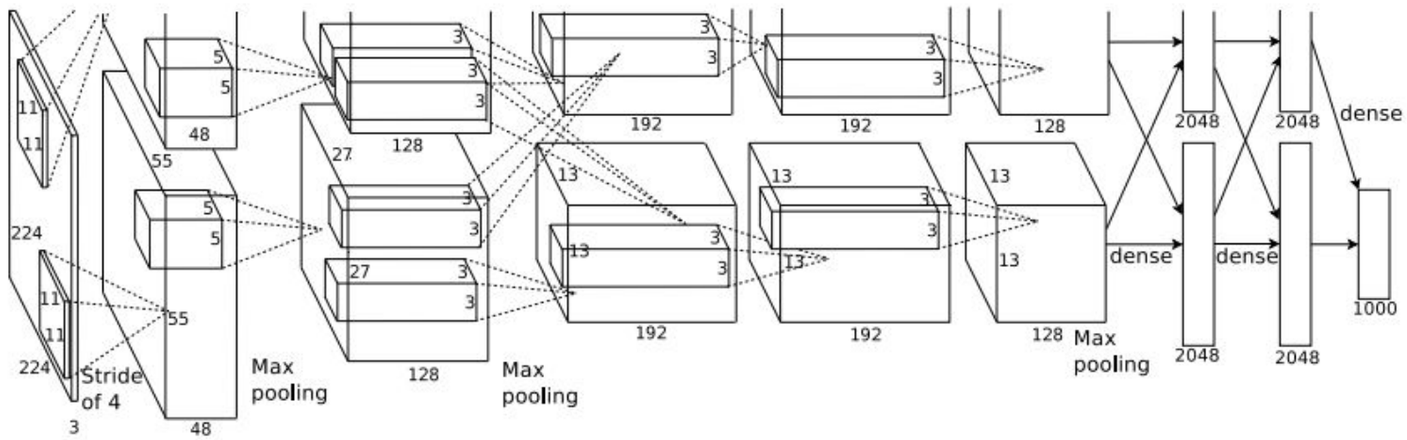


Conv filters were 5x5, applied at stride 1
Subsampling (Pooling) layers were 2x2 applied at stride 2
i.e. architecture is [CONV-POOL-CONV-POOL-FC-FC]

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

Case Study: AlexNet

[Krizhevsky et al. 2012]



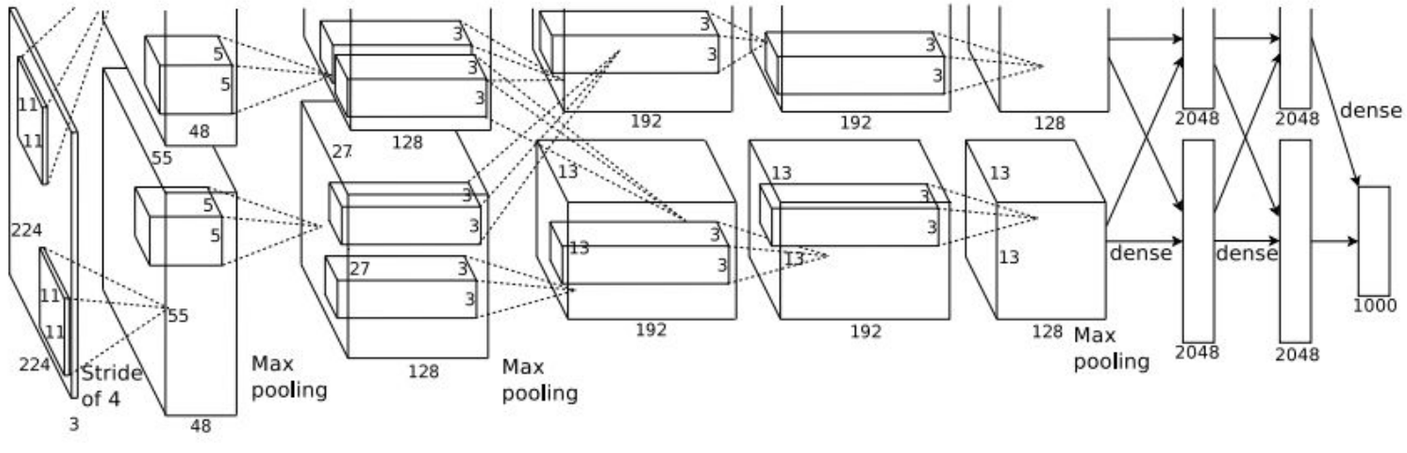
Architecture:

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

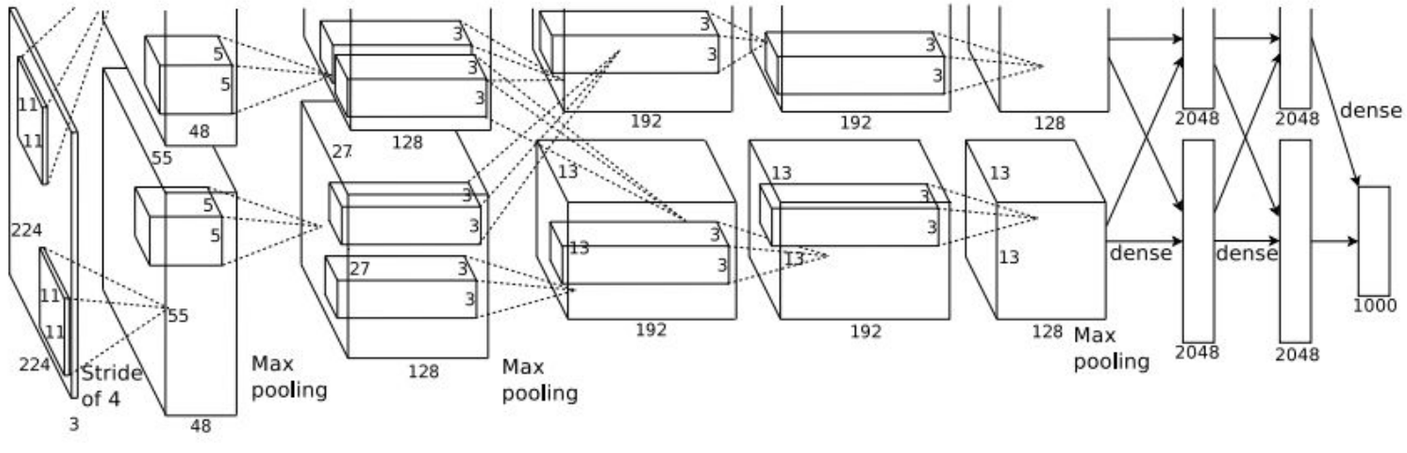
=>

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

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

=>

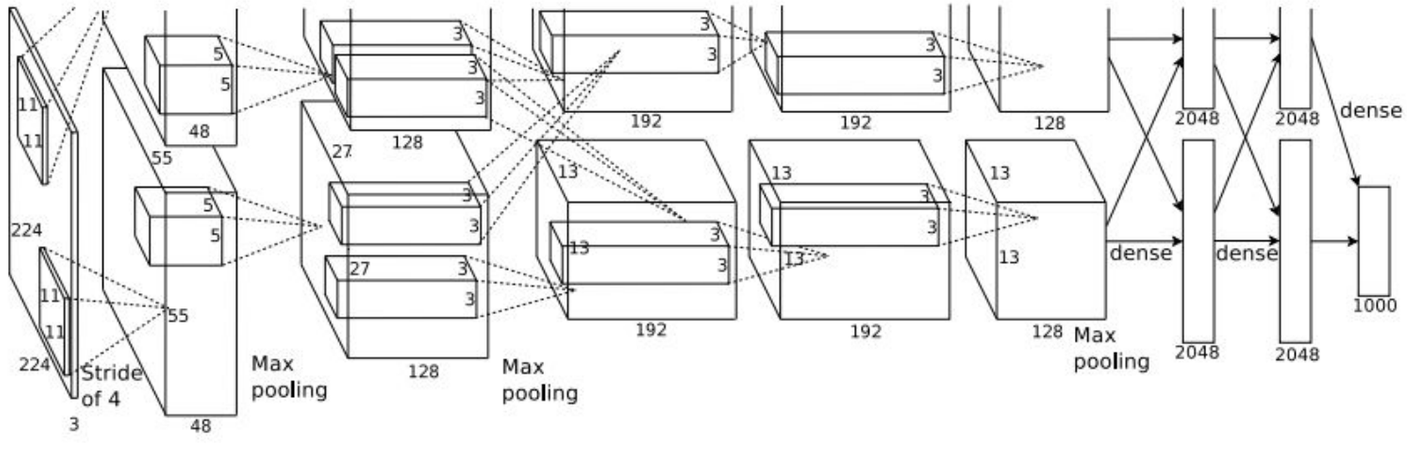
Output volume **[55x55x96]**

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]



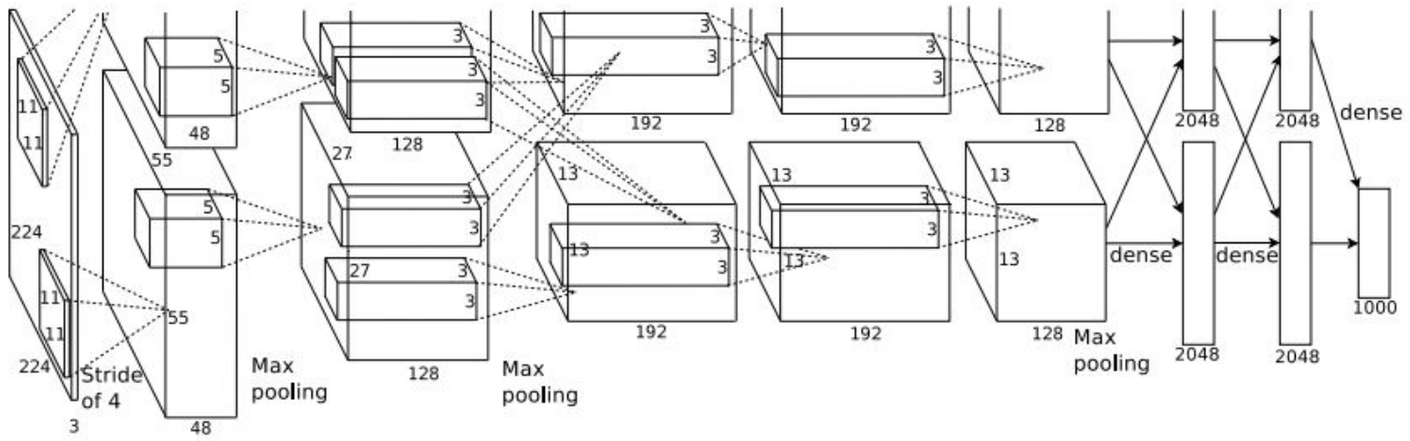
Input: 227x227x3 images

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

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

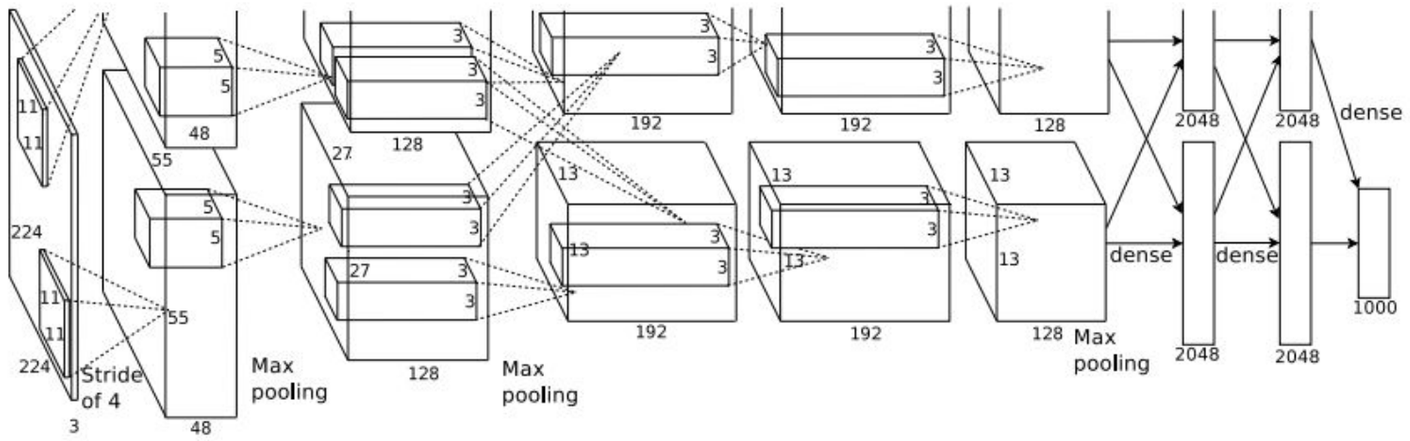
Q: what is the output volume size? Hint: (55-3)/2+1 = 27

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

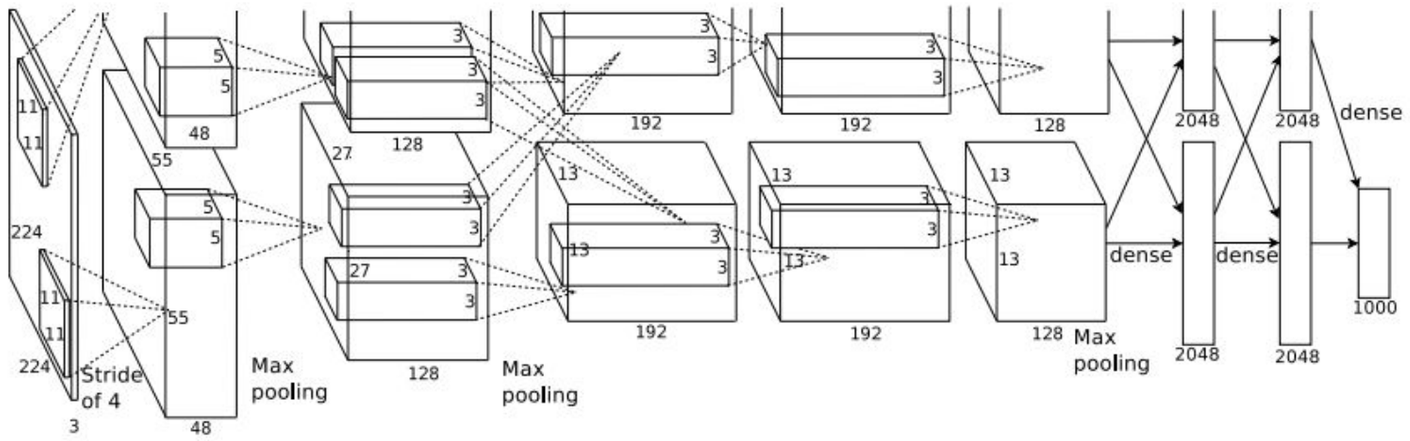
Second layer (POOL1): 3x3 filters applied at stride 2
Output volume: 27x27x96

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

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

Case Study: AlexNet

[Krizhevsky et al. 2012]



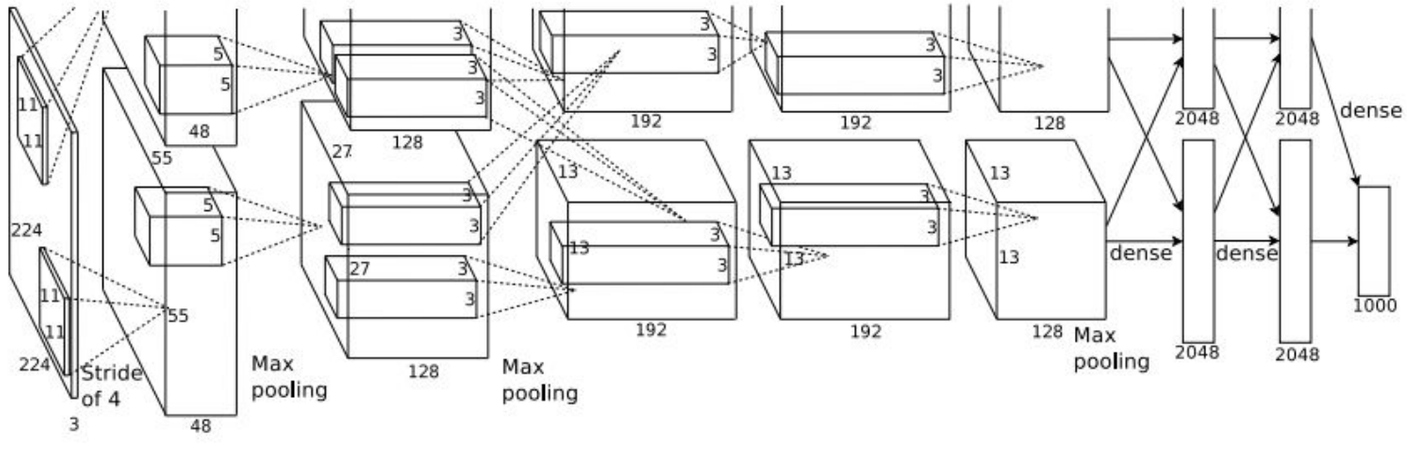
Input: 227x227x3 images
After CONV1: 55x55x96

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

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

Case Study: AlexNet

[Krizhevsky et al. 2012]



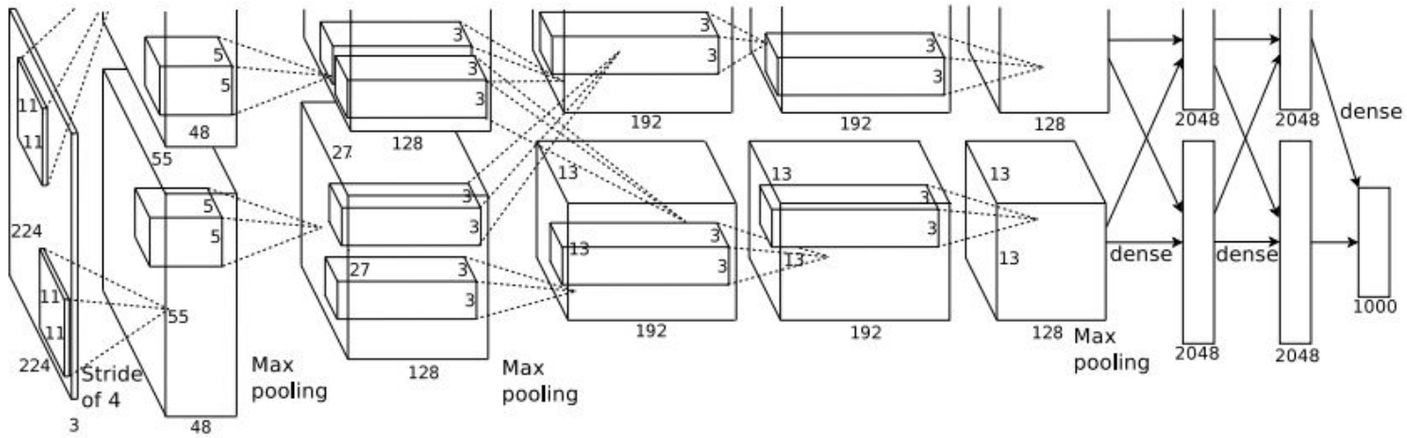
Input: 227x227x3 images
After CONV1: 55x55x96
After POOL1: 27x27x96
...

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]

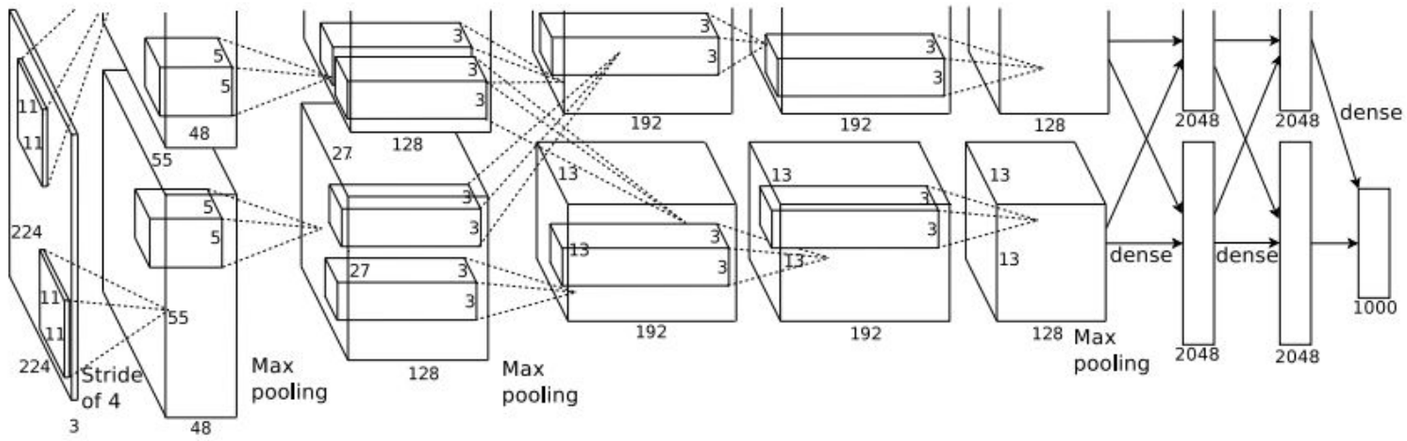


Full (simplified) AlexNet architecture:
[227x227x3] INPUT
[55x55x96] CONV1: 96 11x11 filters at stride 4, pad 0
[27x27x96] MAX POOL1: 3x3 filters at stride 2
[27x27x96] NORM1: Normalization layer
[27x27x256] CONV2: 256 5x5 filters at stride 1, pad 2
[13x13x256] MAX POOL2: 3x3 filters at stride 2
[13x13x256]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

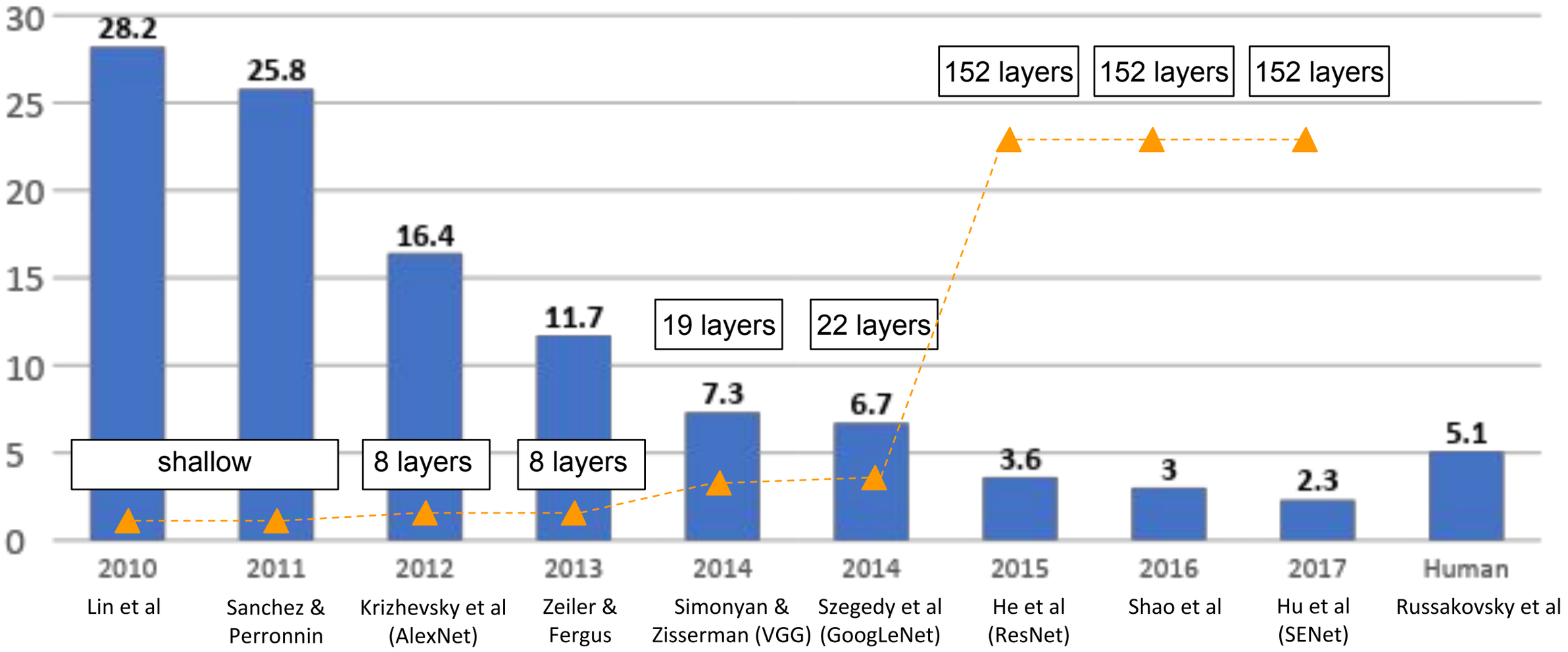


Full (simplified) AlexNet architecture:
[227x227x3] INPUT
[55x55x96] CONV1: 96 11x11 filters at stride 4, pad 0
[27x27x96] MAX POOL1: 3x3 filters at stride 2
[27x27x96] NORM1: Normalization layer
[27x27x256] CONV2: 256 5x5 filters at stride 1, pad 2
[13x13x256] MAX POOL2: 3x3 filters at stride 2
[13x13x256]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

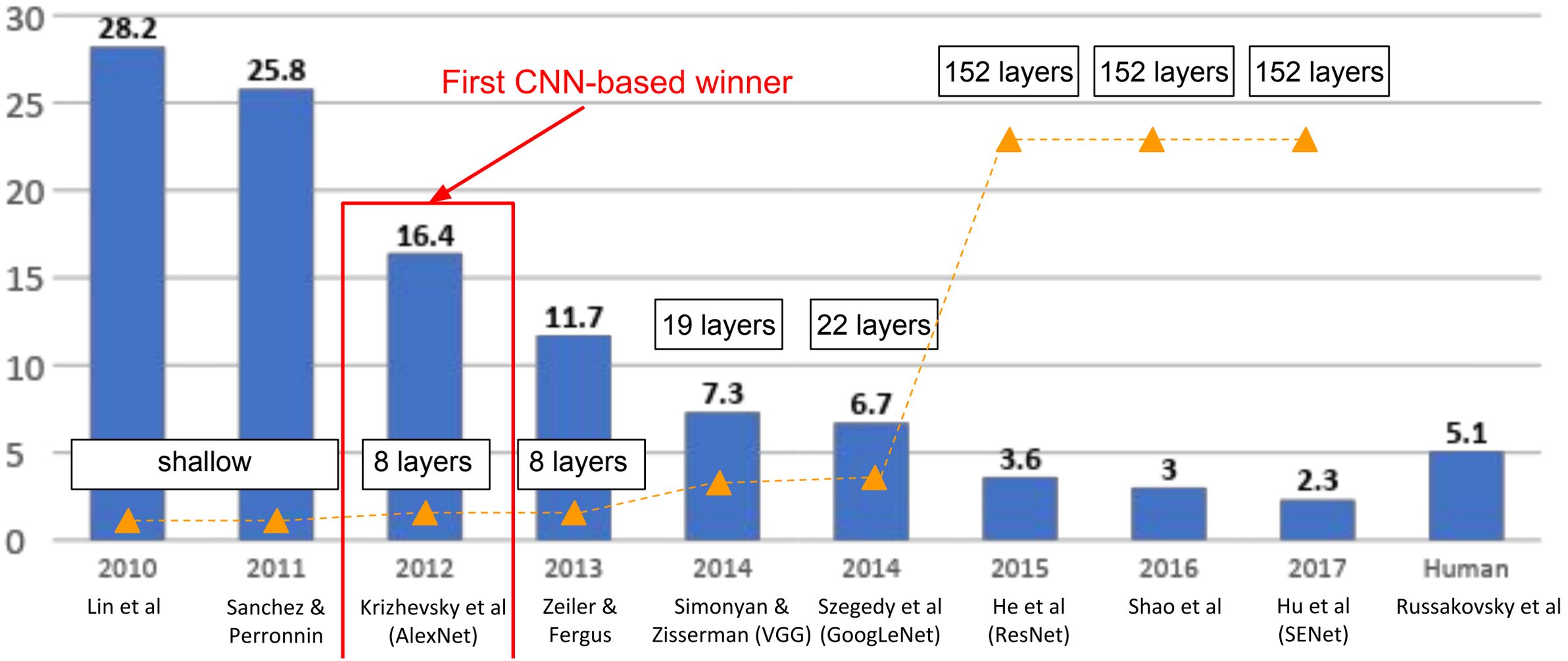
slide credit: Fei-Fei, Justin Johnson, Serena Yeung

ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners



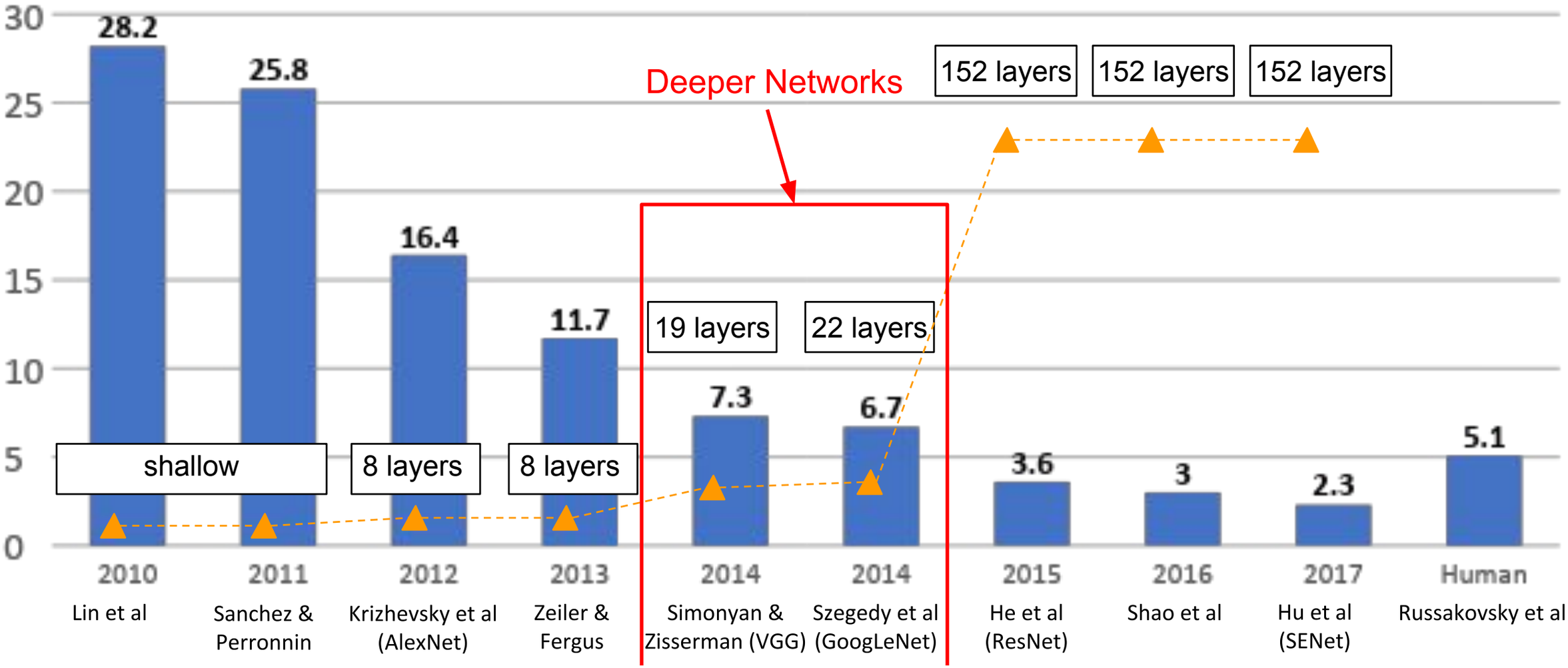
slide credit: Fei-Fei, Justin Johnson, Serena Yeung

ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners



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

ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners



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

Case Study: VGGNet

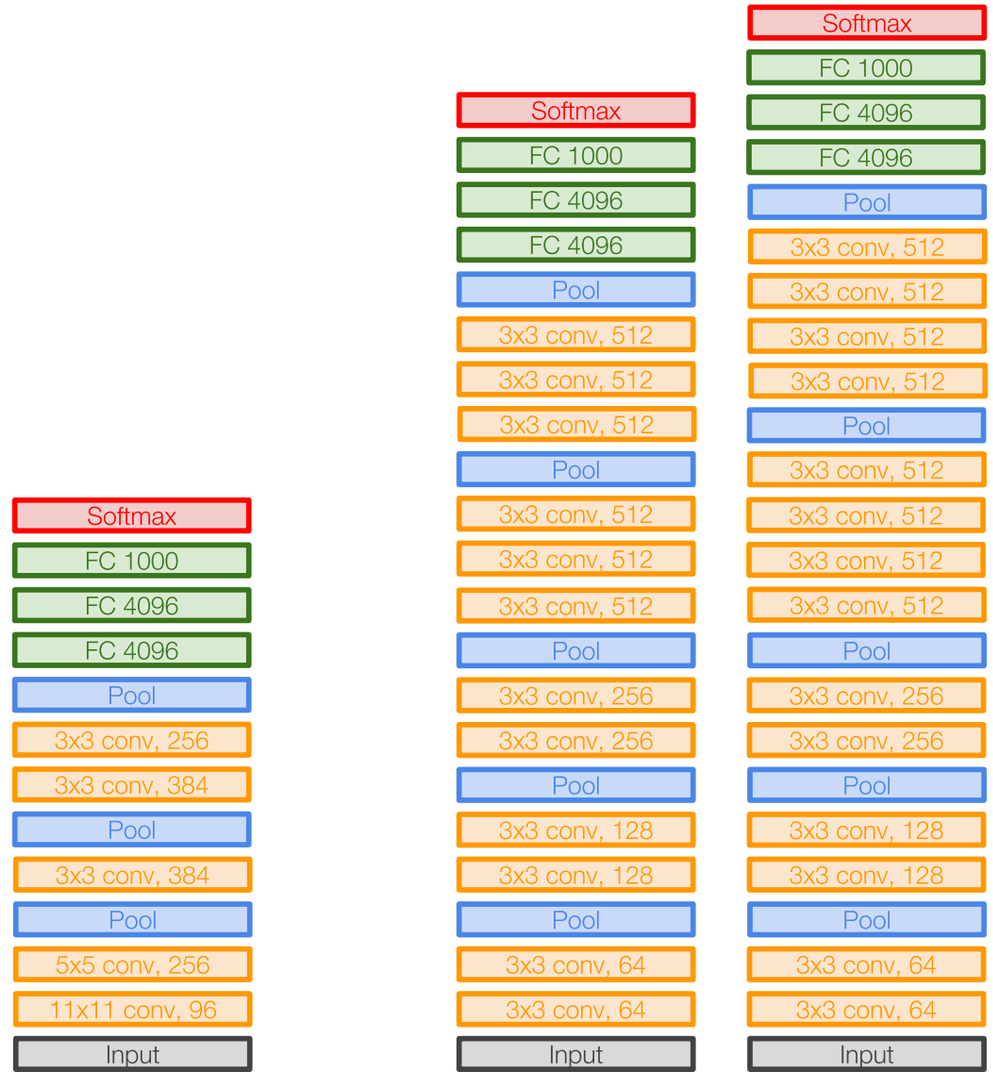
[Simonyan and Zisserman, 2014]

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)

AlexNet

VGG16

VGG19

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

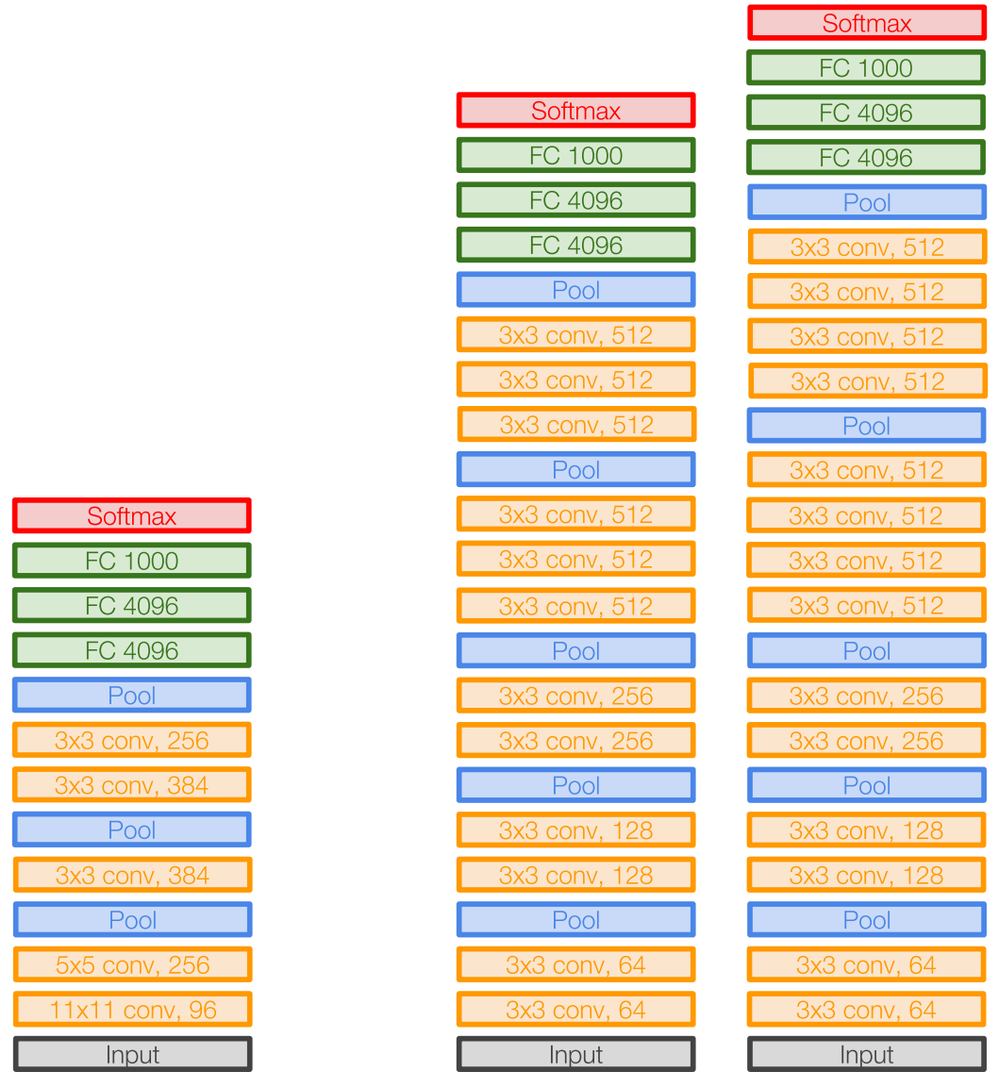
Case Study: VGGNet

[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

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



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

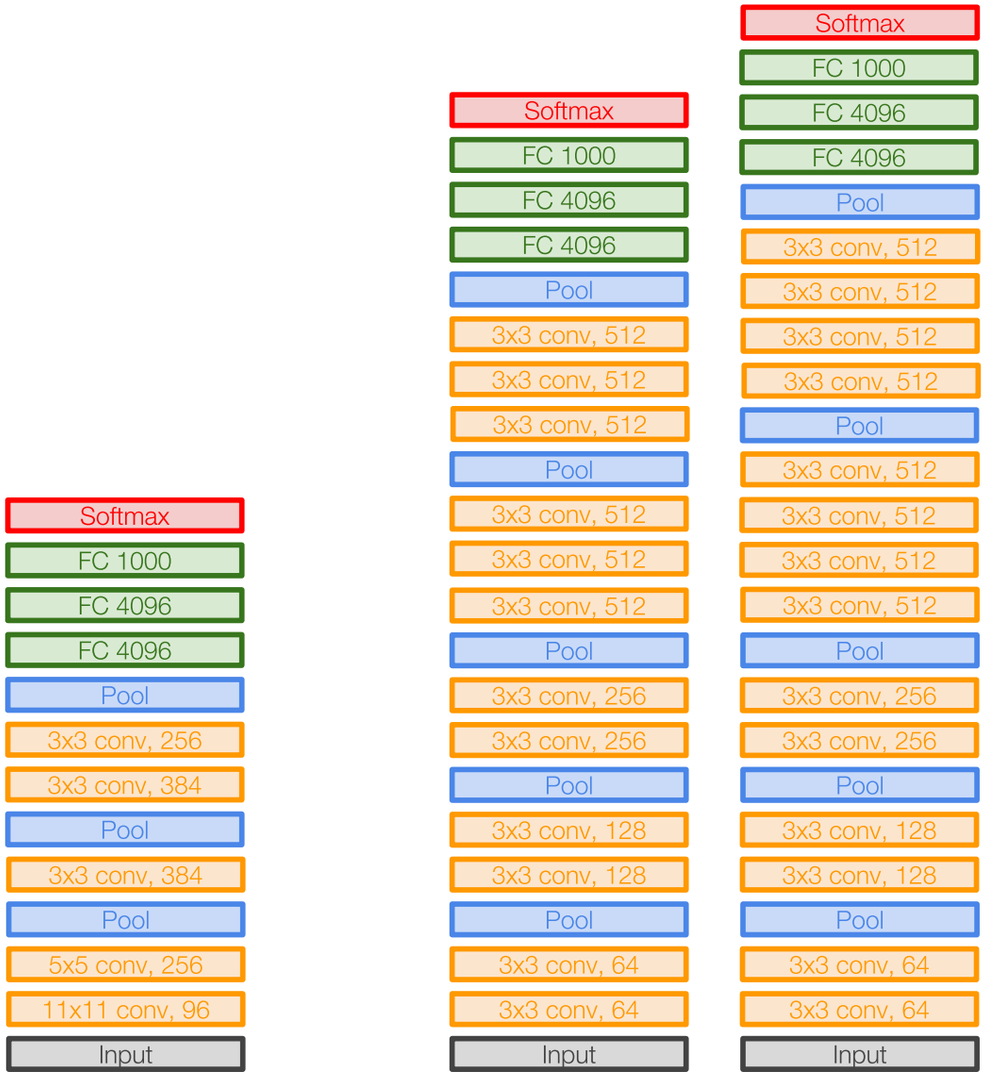
Case Study: VGGNet

[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

[7x7]



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

Case Study: VGGNet

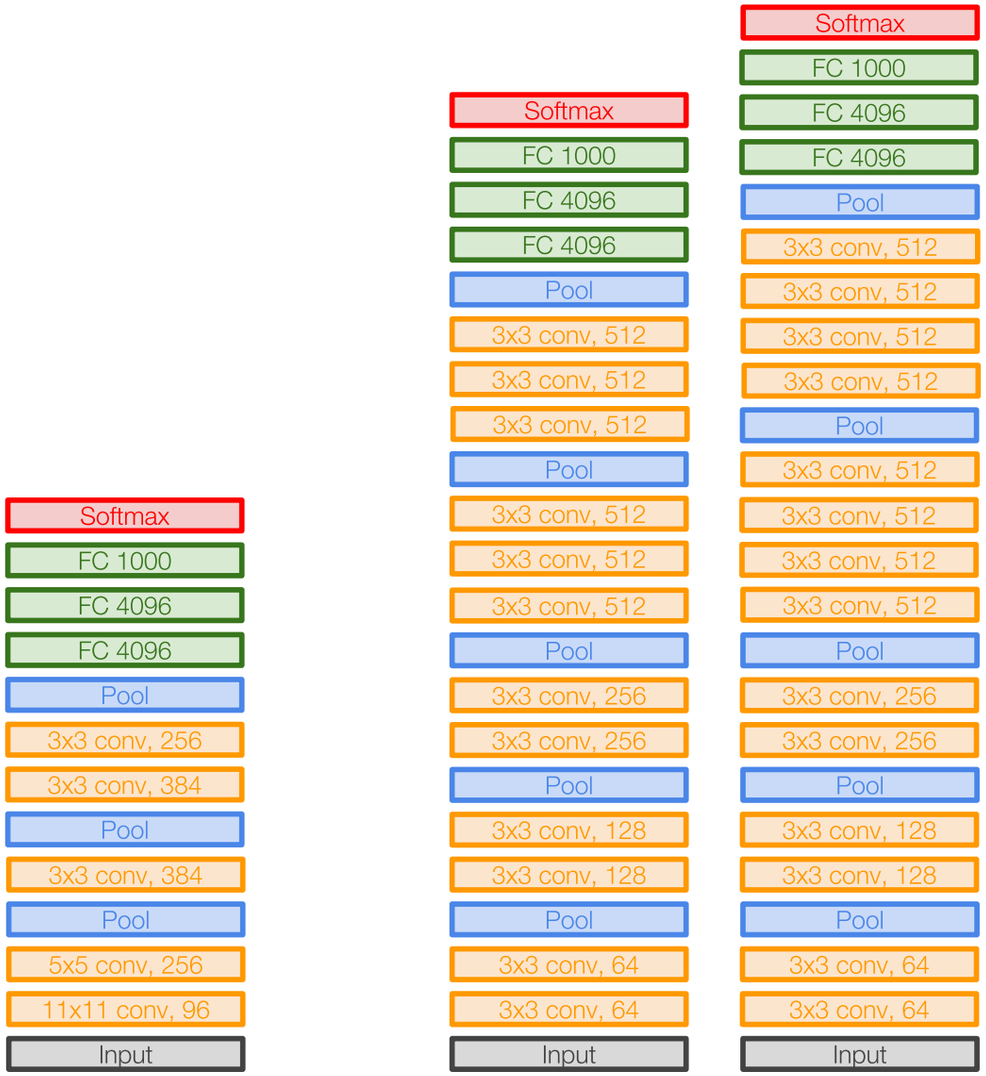
[Simonyan and Zisserman, 2014]

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

(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

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

(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 (for a forward pass)
TOTAL params: 138M parameters

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

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

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

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

VGG16

Common names

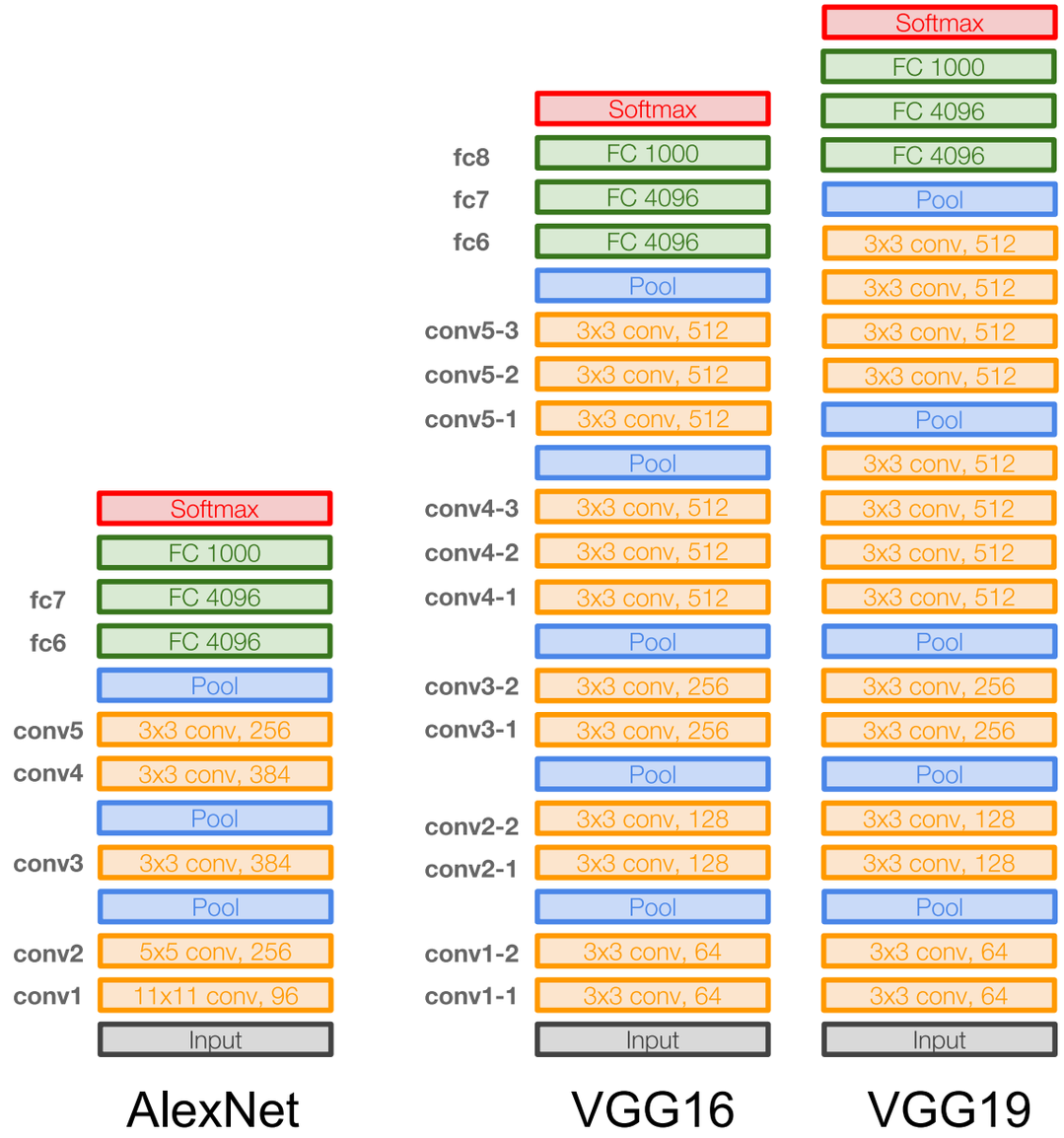
slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Case Study: VGGNet

[Simonyan and Zisserman, 2014]

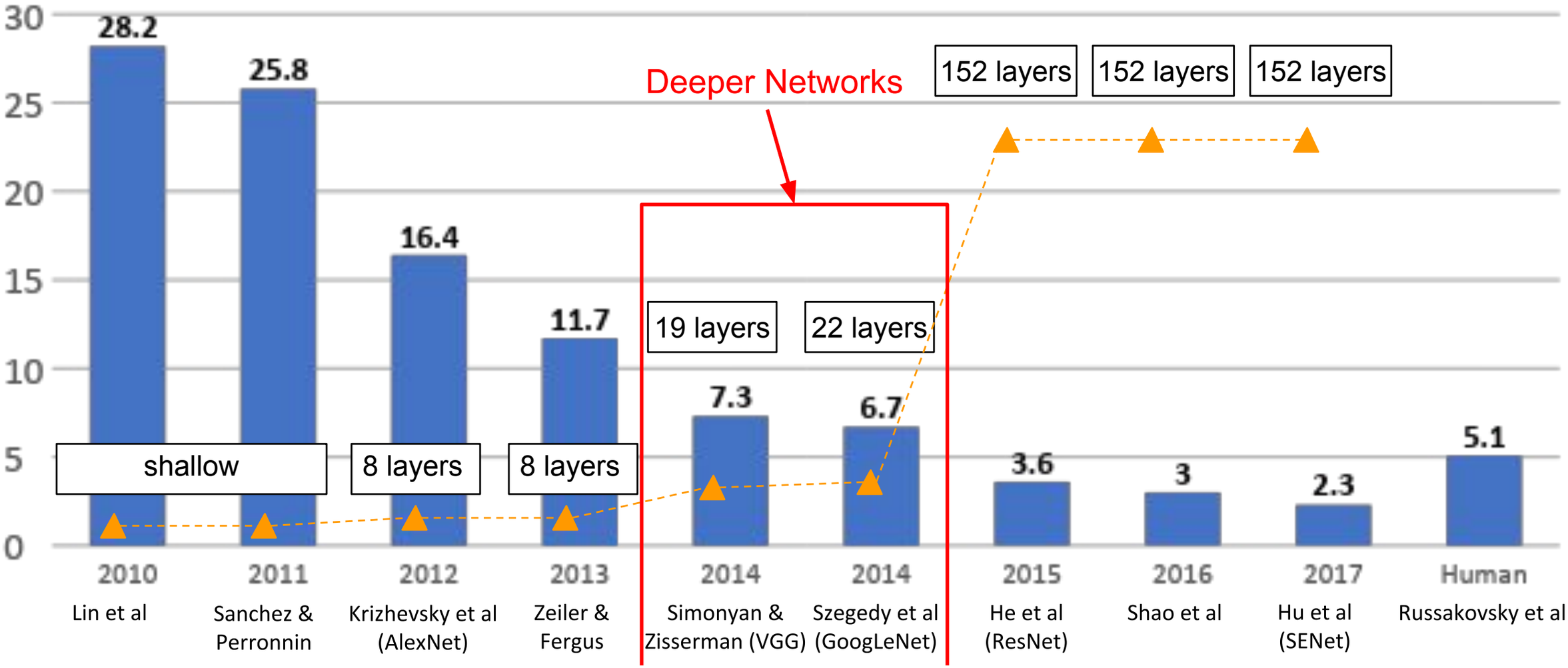
Details:

- 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



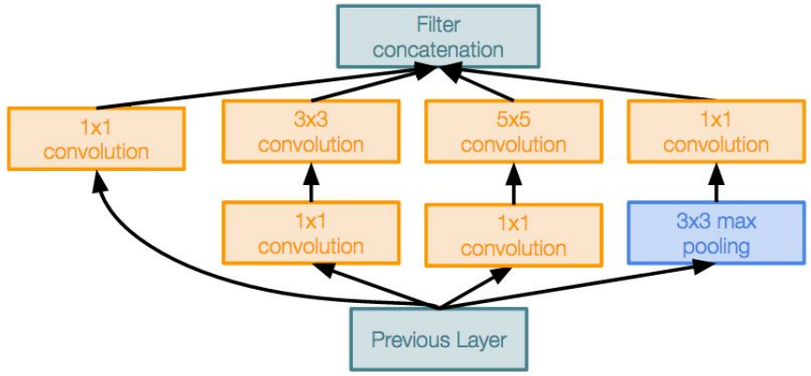
slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Case Study: GoogLeNet

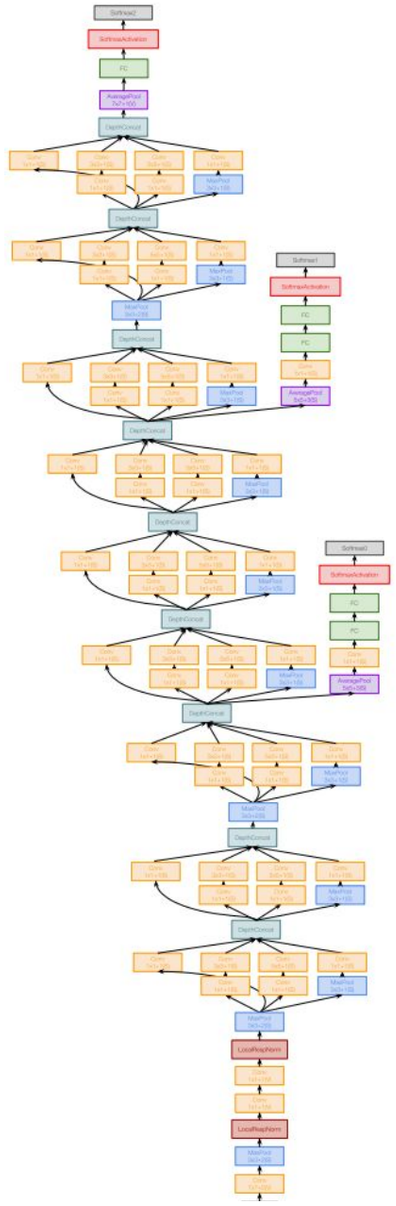
[Szegedy et al., 2014]

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

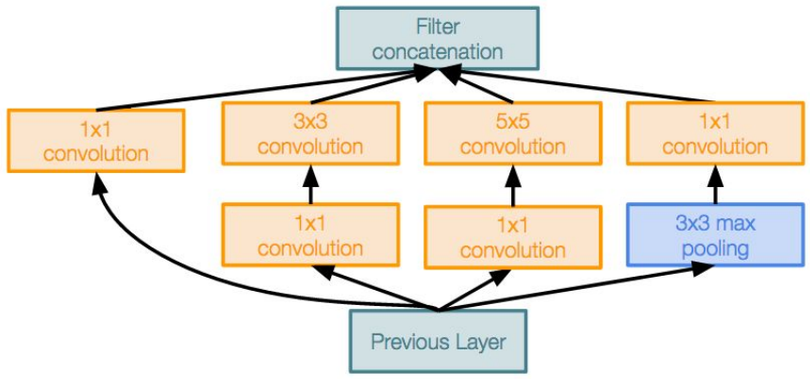


slide credit: Fei-Fei, Justin Johnson, Serena Yeung

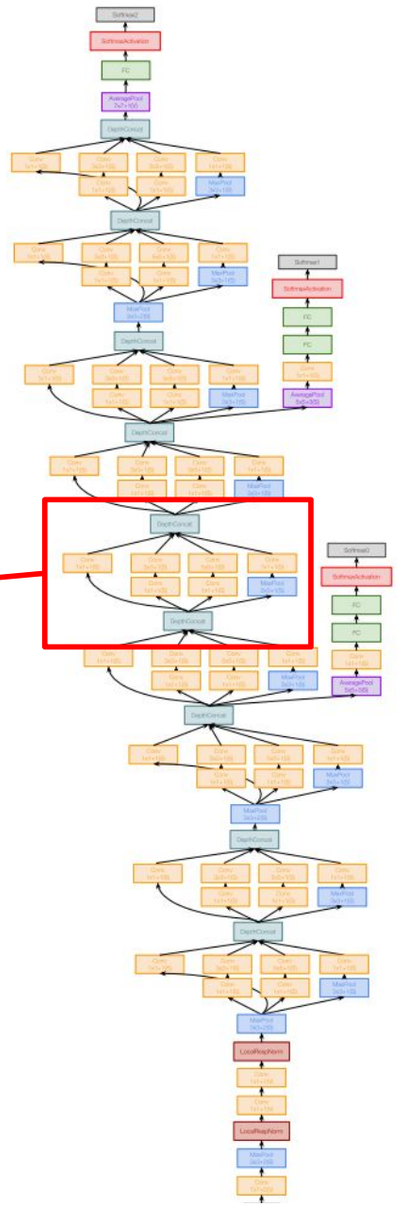
Case Study: GoogLeNet

[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



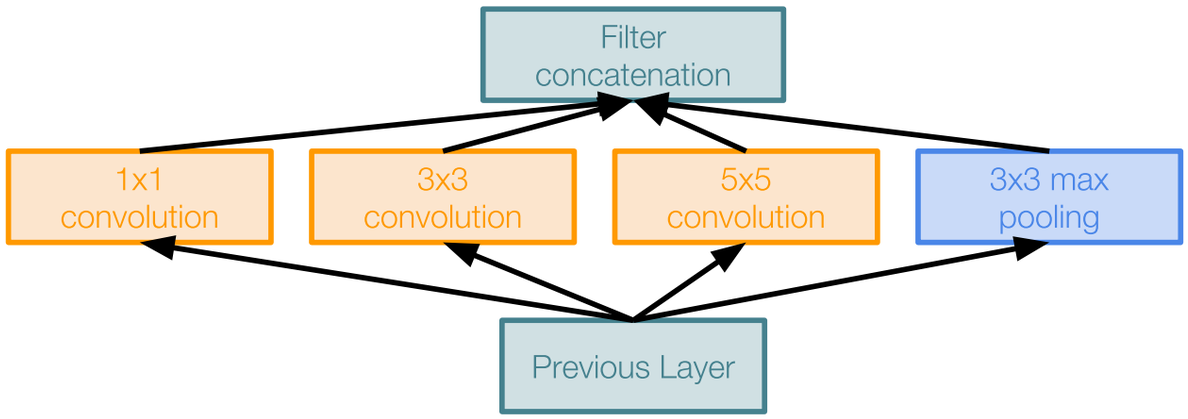
Inception module



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:

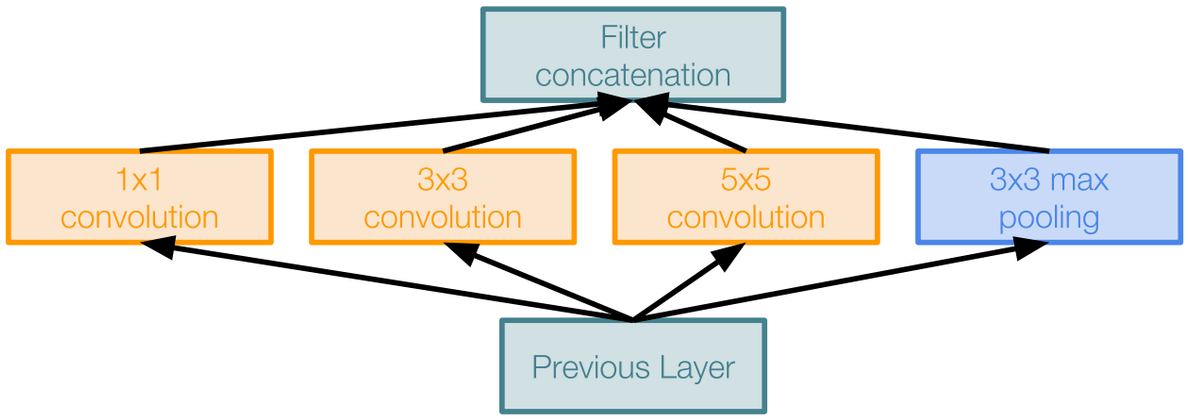
- Multiple receptive field sizes for convolution (1x1, 3x3, 5x5)
- Pooling operation (3x3)

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]

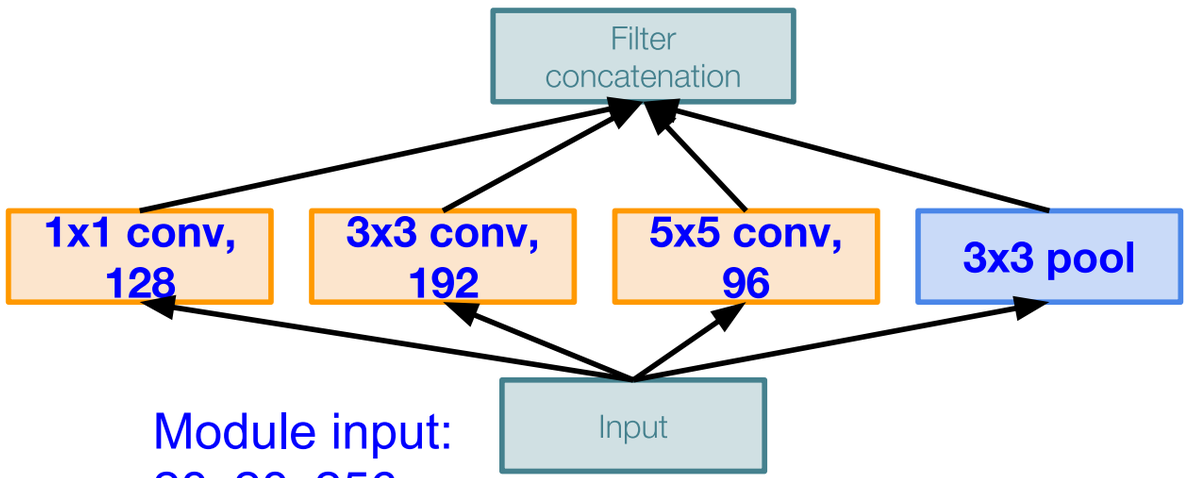
slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Case Study: GoogLeNet

[Szegedy et al., 2014]

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

Example:



Naive Inception module

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

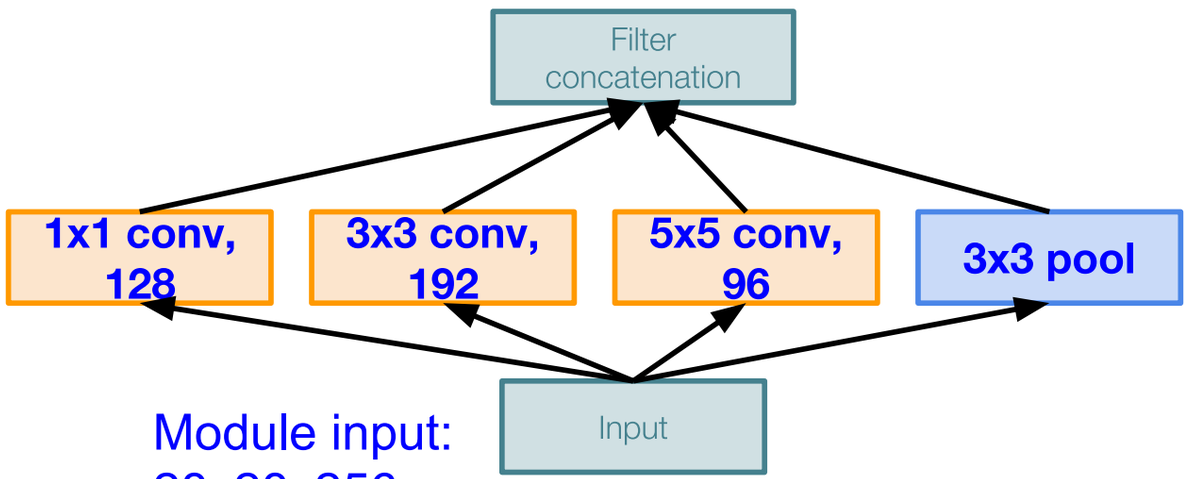
Case Study: GoogLeNet

[Szegedy et al., 2014]

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

Example:

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



Naive Inception module

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

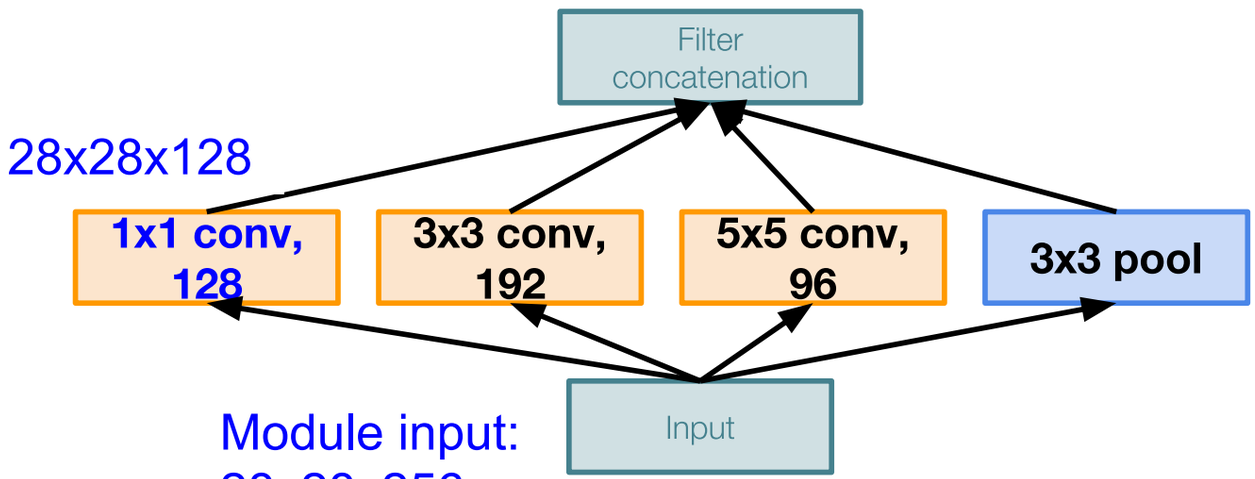
Case Study: GoogLeNet

[Szegedy et al., 2014]

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

Example:

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



Naive Inception module

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

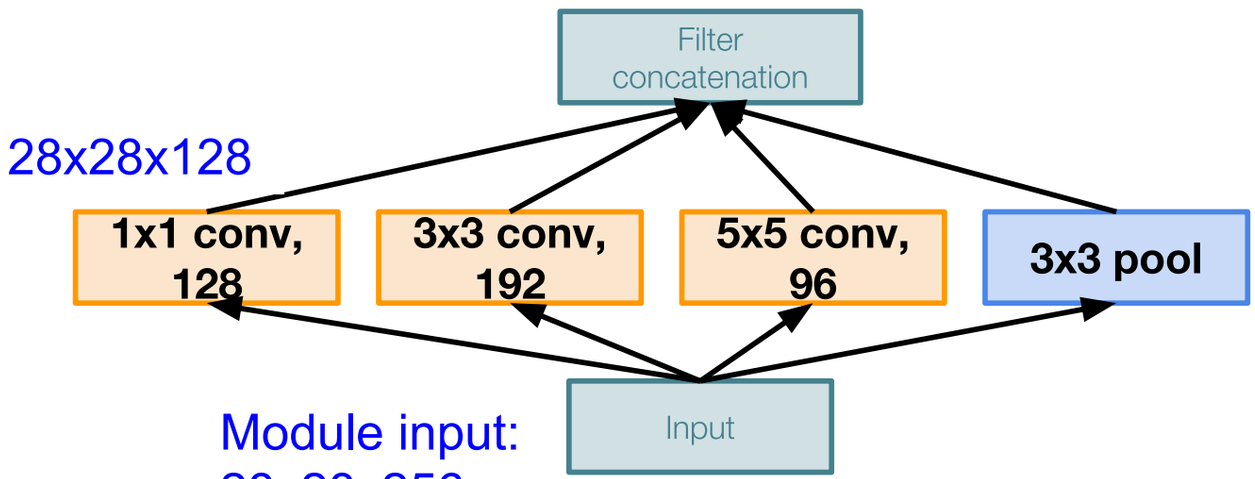
Case Study: GoogLeNet

[Szegedy et al., 2014]

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

Example:

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



Naive Inception module

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

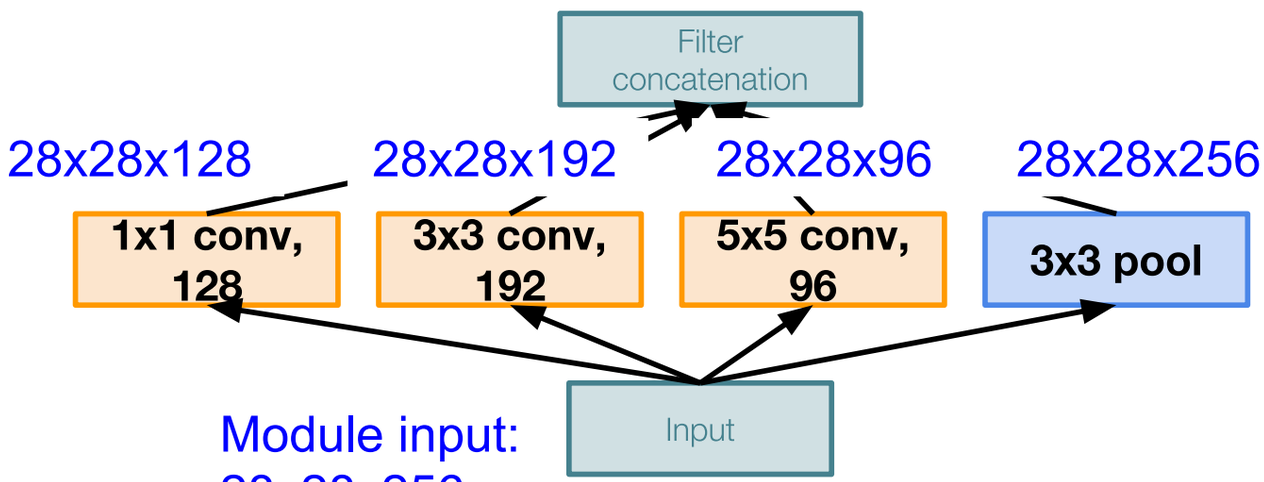
Case Study: GoogLeNet

[Szegedy et al., 2014]

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

Example:

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



Naive Inception module

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

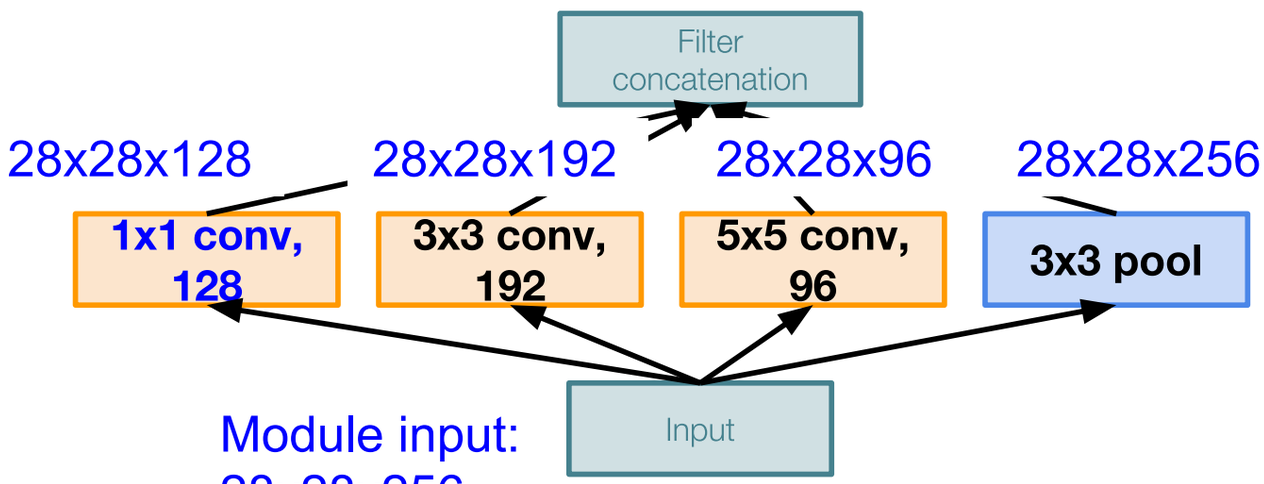
Case Study: GoogLeNet

[Szegedy et al., 2014]

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

Example:

Q3:What is output size after filter concatenation?



Naive Inception module

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

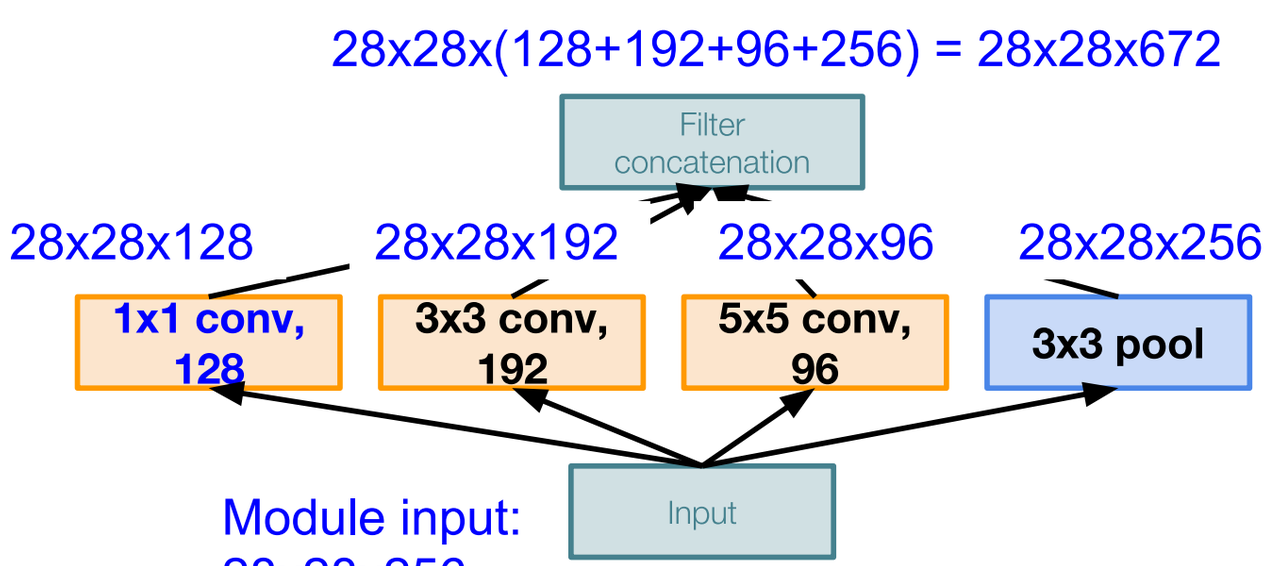
Case Study: GoogLeNet

[Szegedy et al., 2014]

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

Example:

Q3:What is output size after filter concatenation?



Naive Inception module

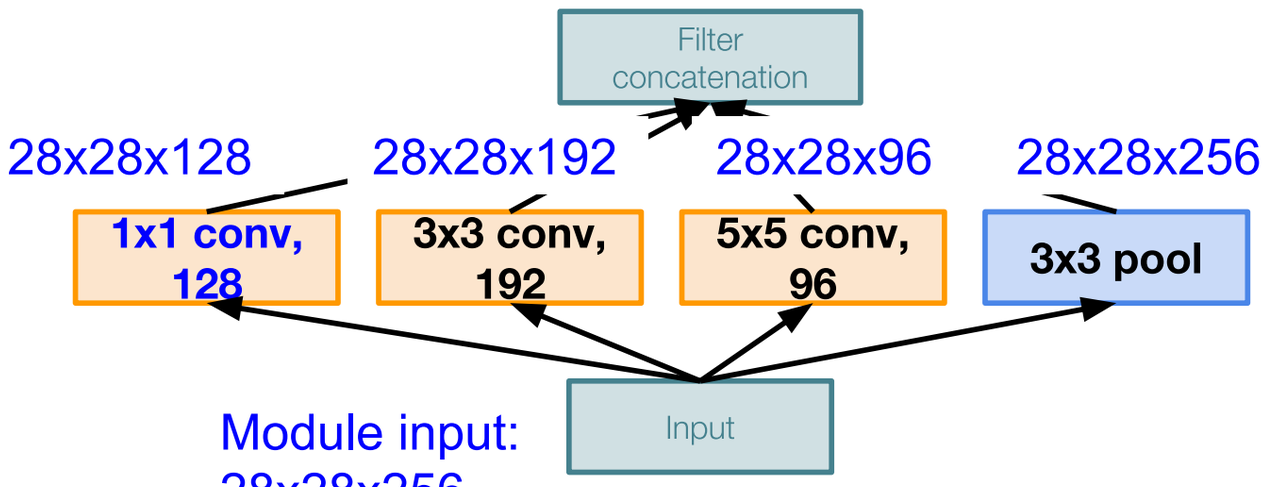
slide credit: Fei-Fei, Justin Johnson, Serena Yeung

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

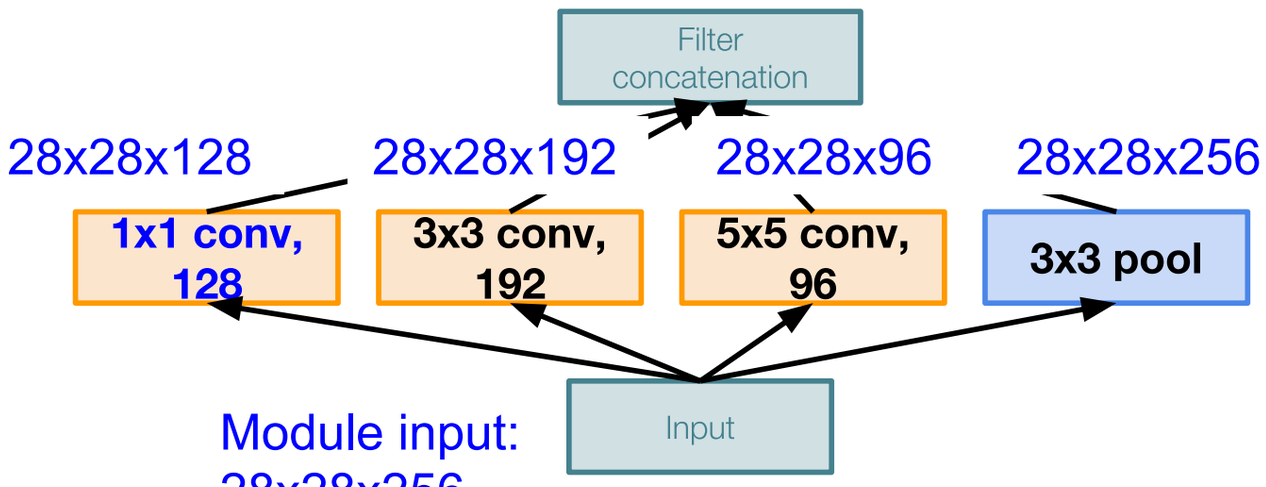
slide credit: Fei-Fei, Justin Johnson, Serena Yeung

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

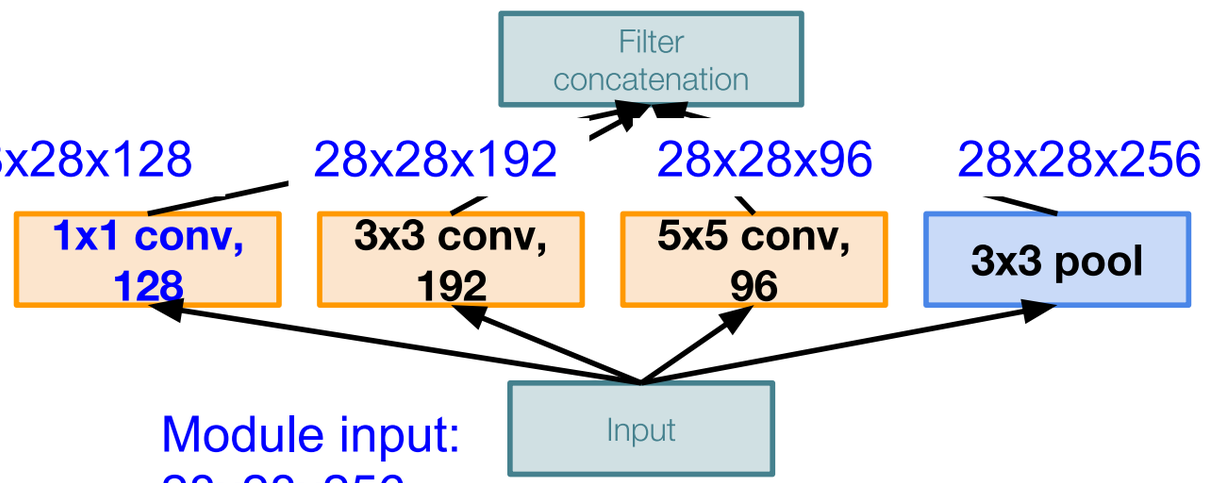
[Szegedy et al., 2014]

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

Example:

Q3:What is output size after filter concatenation?

28x28x(128+192+96+256) = **529k**

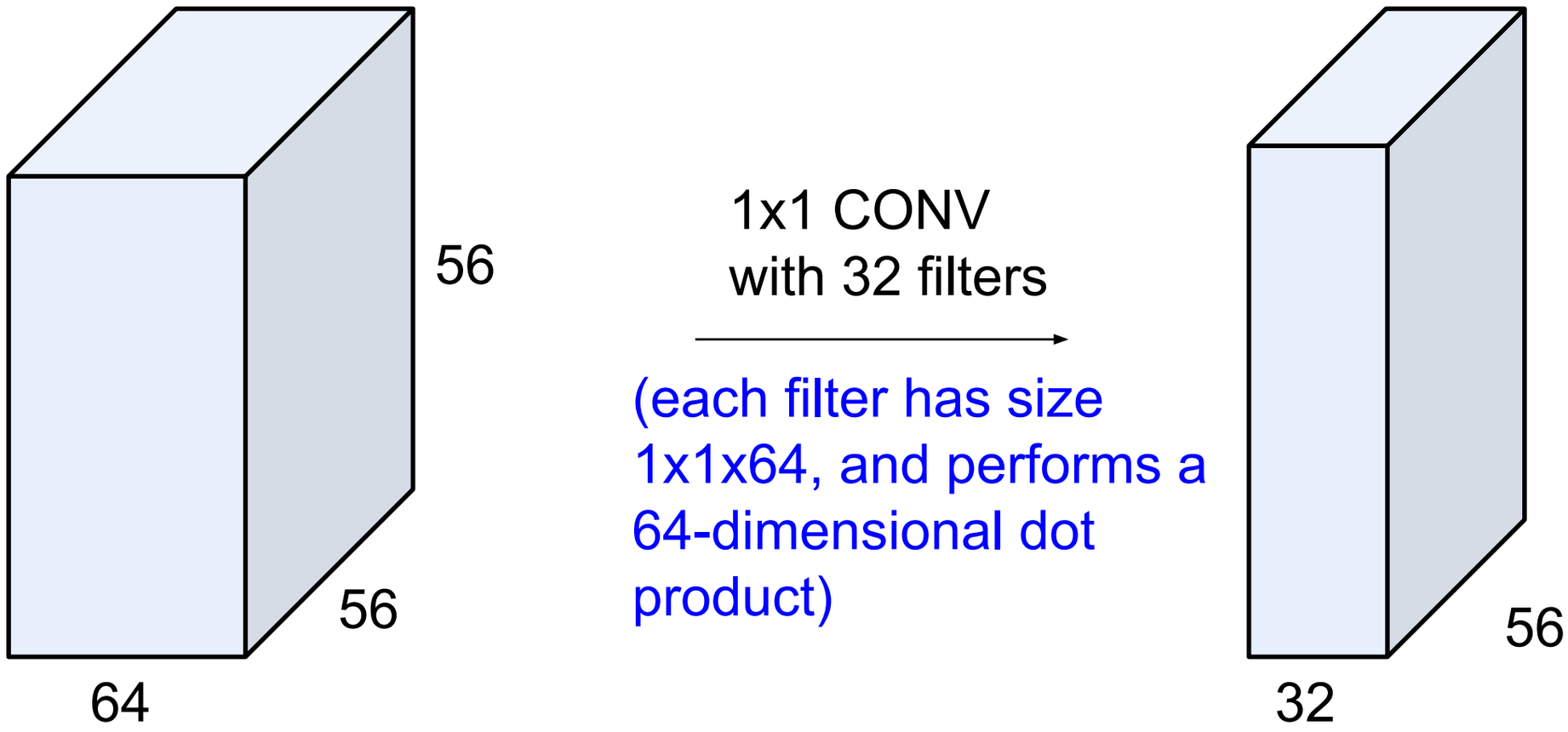


Naive Inception module

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

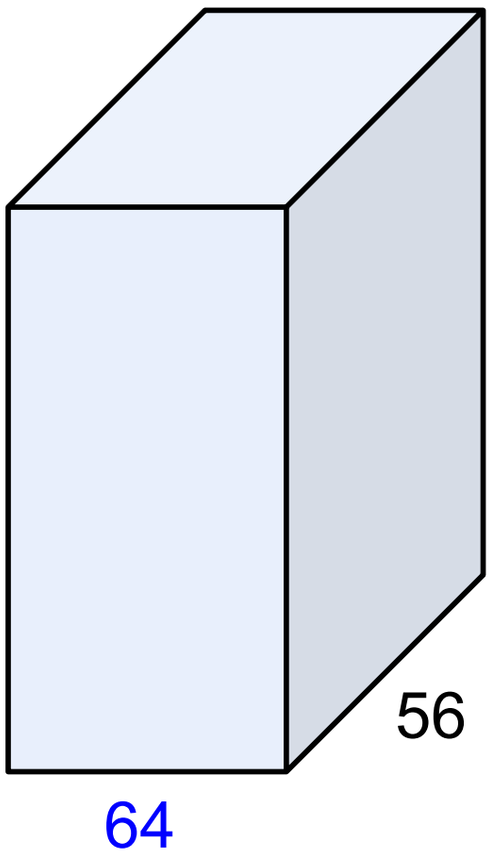
slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Reminder: 1x1 convolutions



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

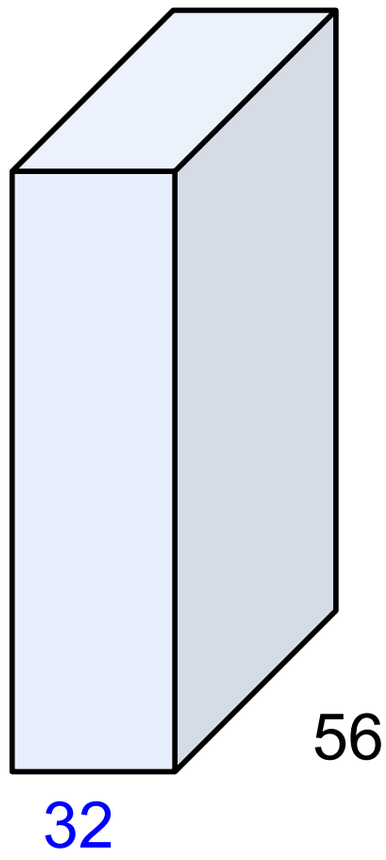
Reminder: 1x1 convolutions



1x1 CONV
with 32 filters

preserves spatial dimensions, reduces depth!

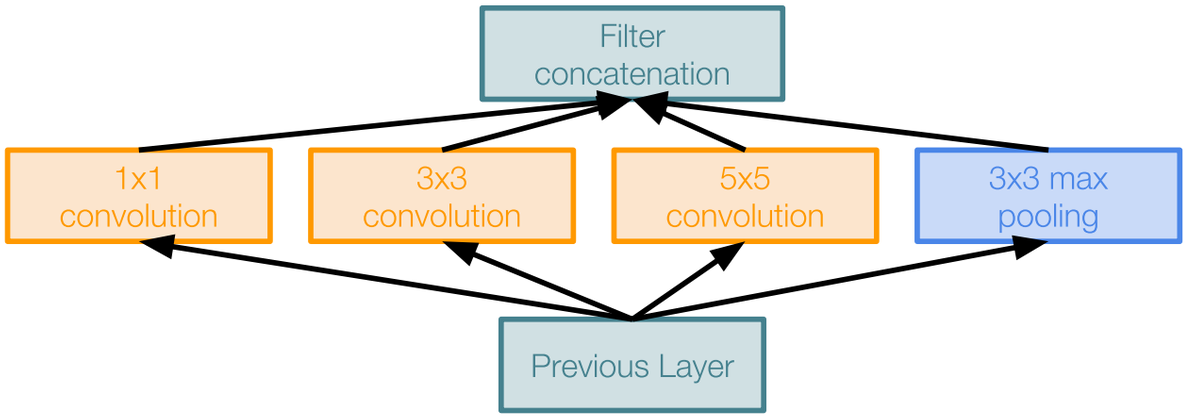
Projects depth to lower dimension (combination of feature maps)



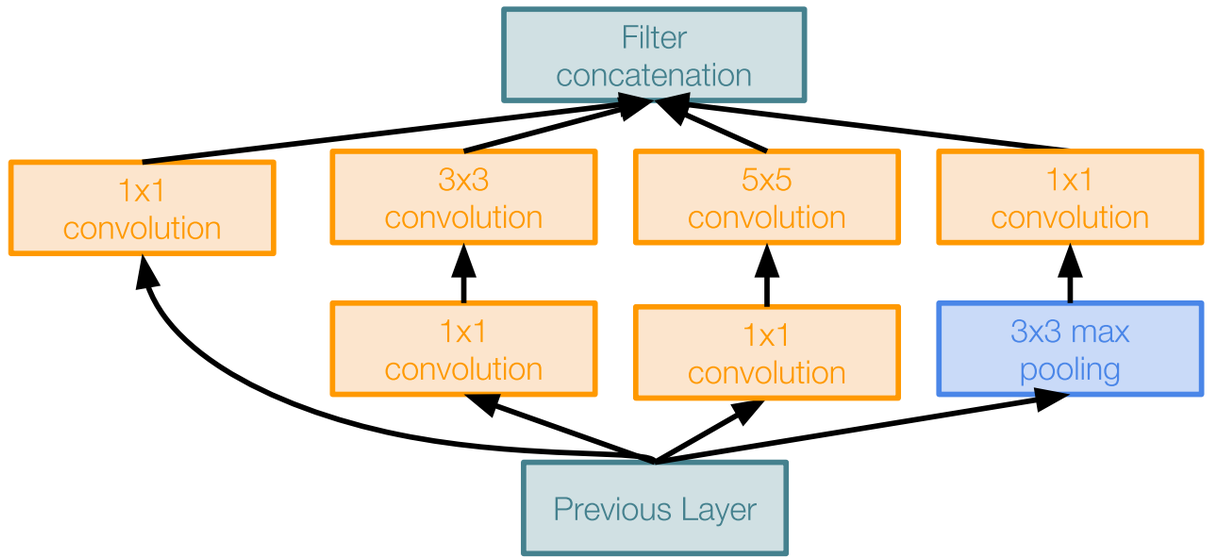
slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Case Study: GoogLeNet

[Szegedy et al., 2014]



Naive Inception module

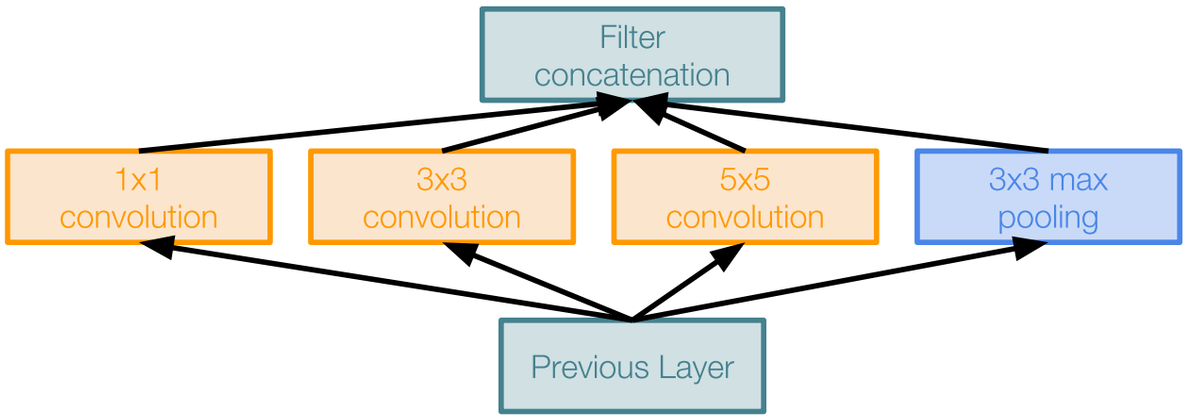


Inception module with dimension reduction

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

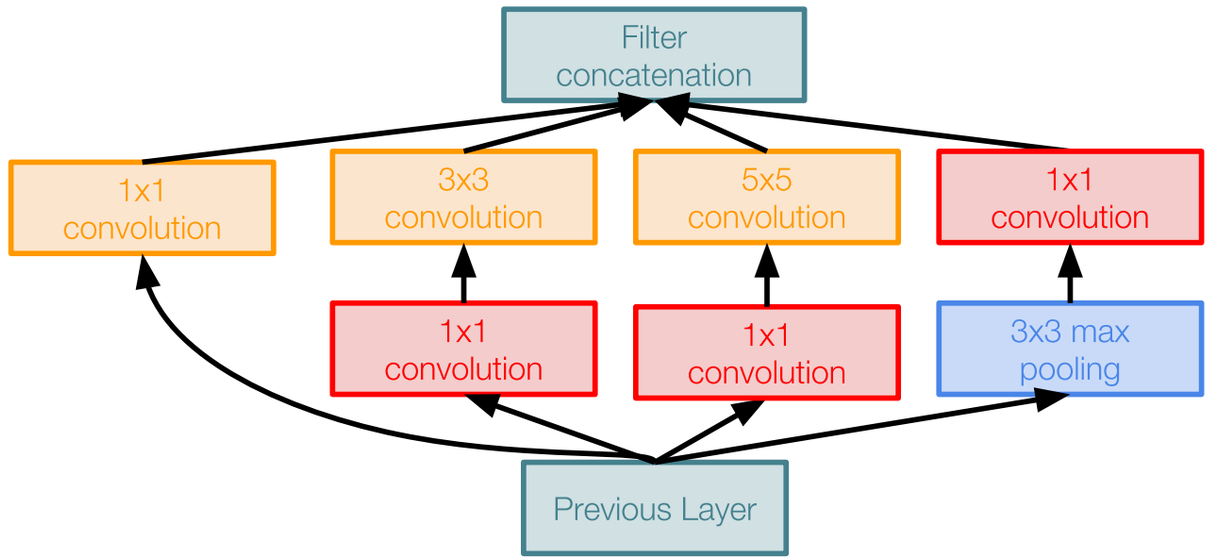
Case Study: GoogLeNet

[Szegedy et al., 2014]



Naive Inception module

1x1 conv “bottleneck” layers

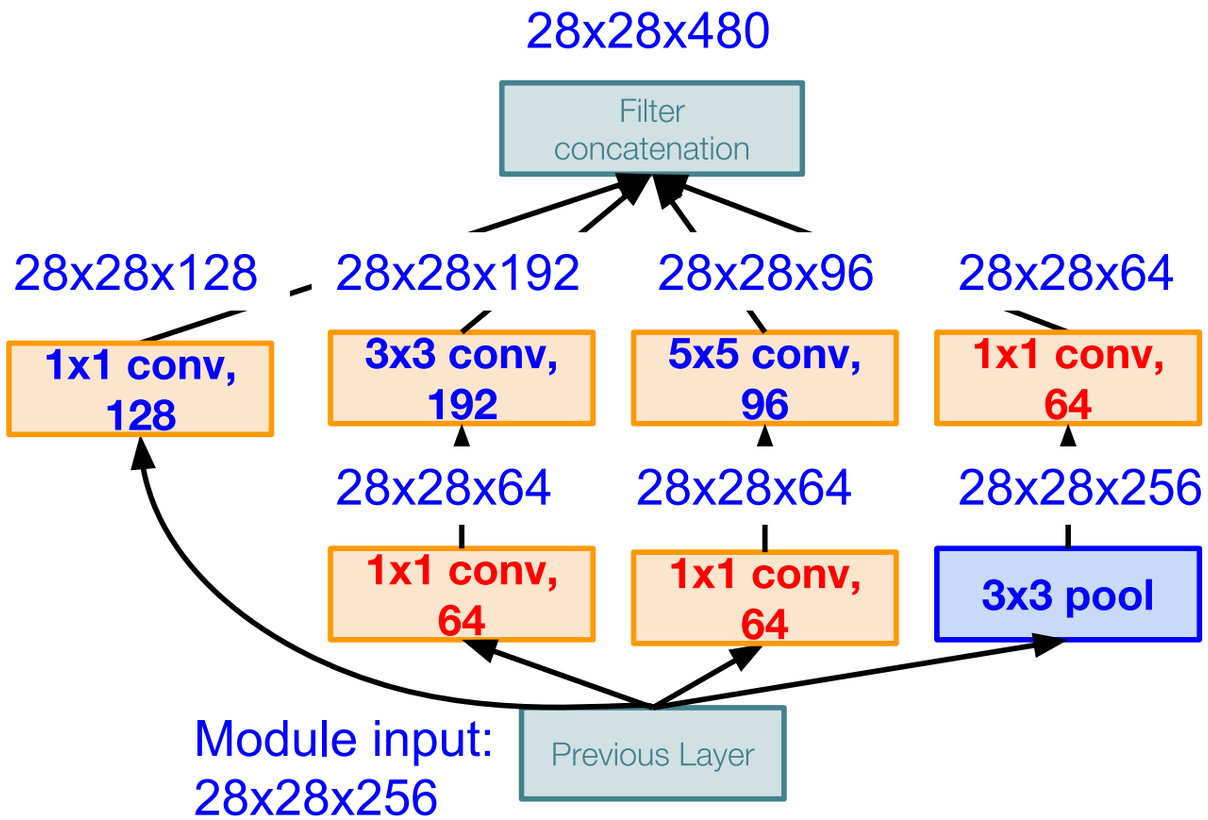


Inception module with dimension reduction

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

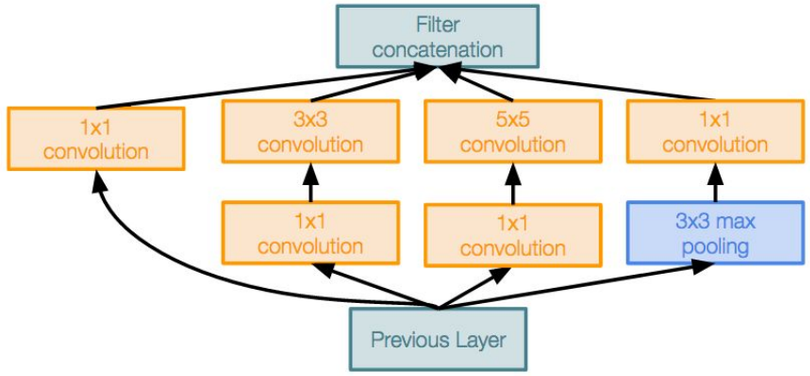
Compared to 854M ops for naive version
Bottleneck can also reduce depth after pooling layer

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

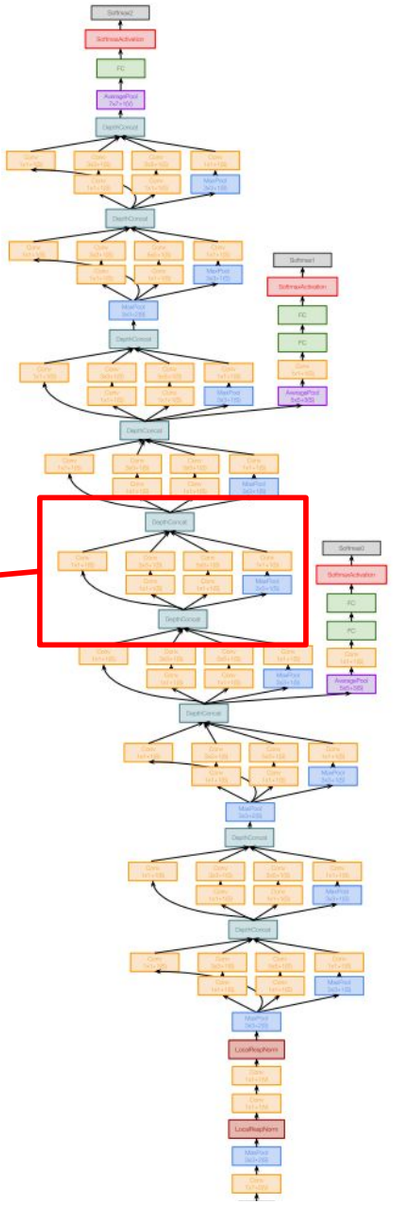
Case Study: GoogLeNet

[Szegedy et al., 2014]

Stack Inception modules with dimension reduction on top of each other



Inception module

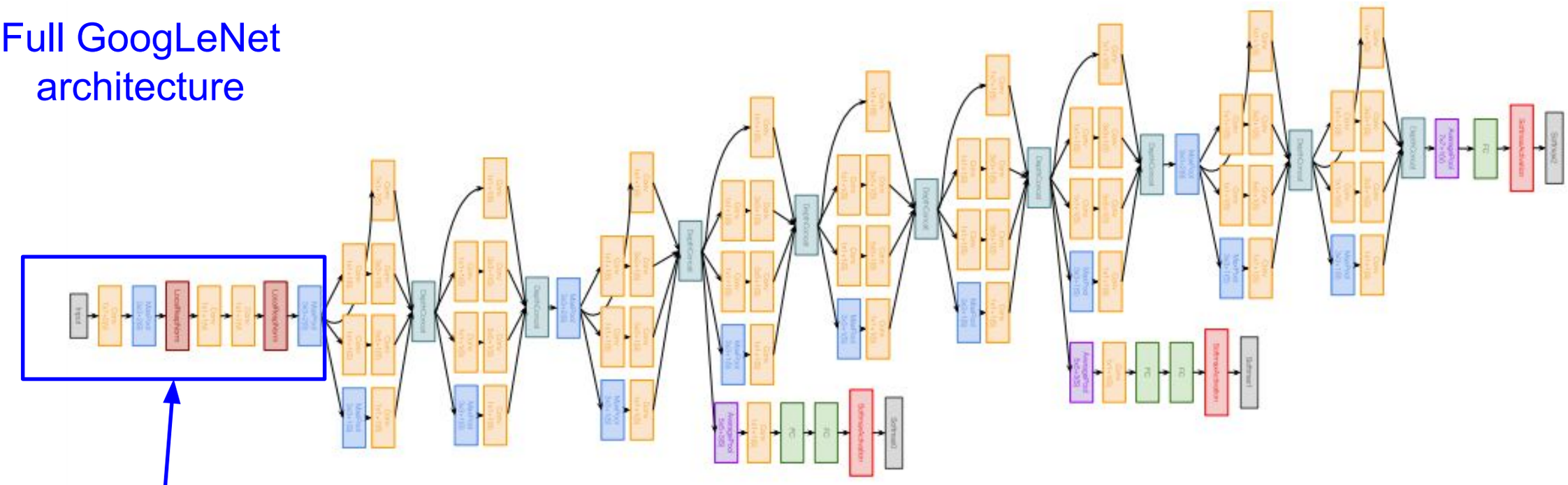


slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Case Study: GoogLeNet

[Szegedy et al., 2014]

Full GoogLeNet architecture



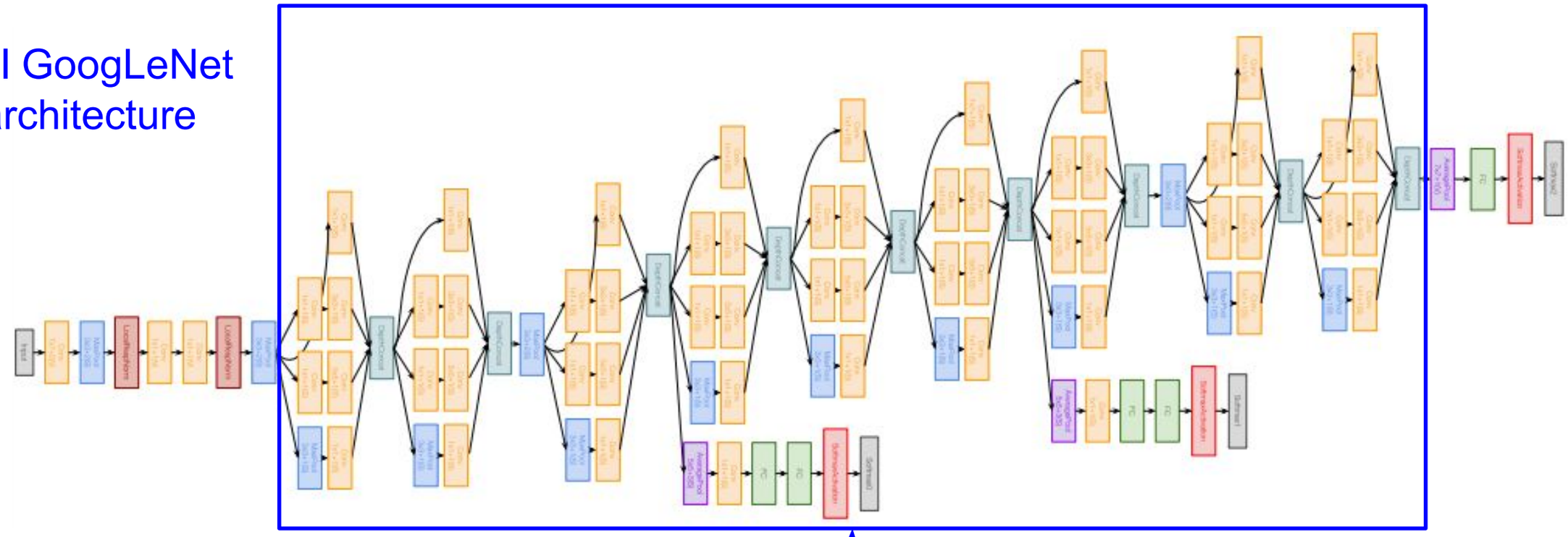
Stem Network: Conv-Pool- 2x Conv-Pool

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

Case Study: GoogLeNet

[Szegedy et al., 2014]

Full GoogLeNet architecture



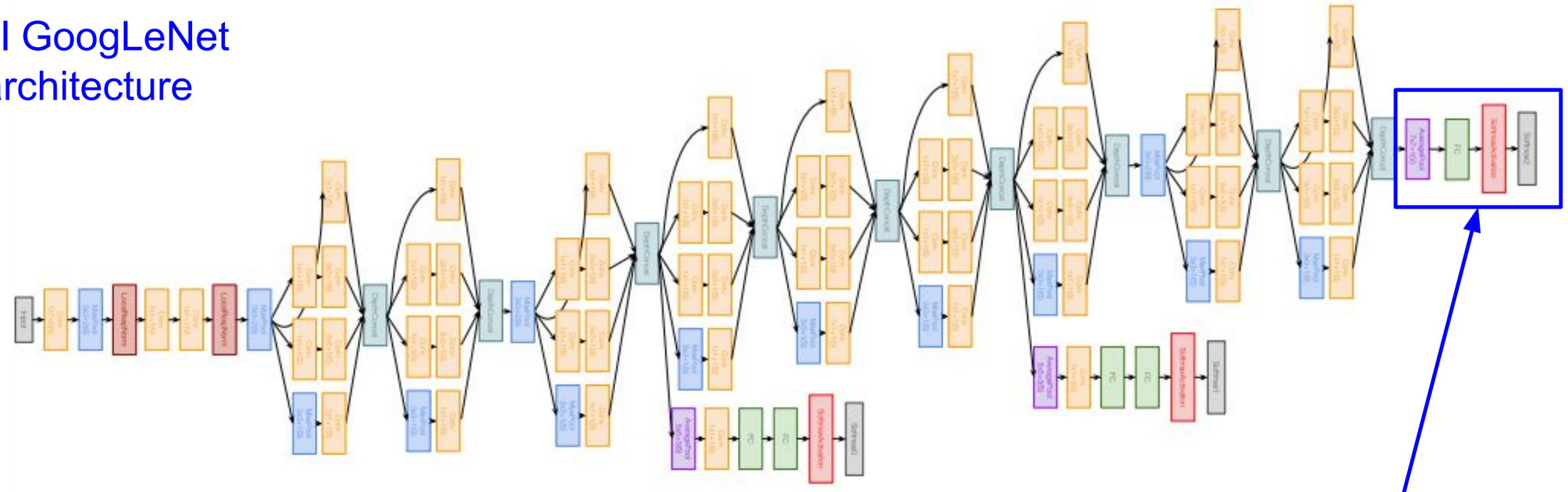
Stacked Inception Modules

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

Case Study: GoogLeNet

[Szegedy et al., 2014]

Full GoogLeNet architecture



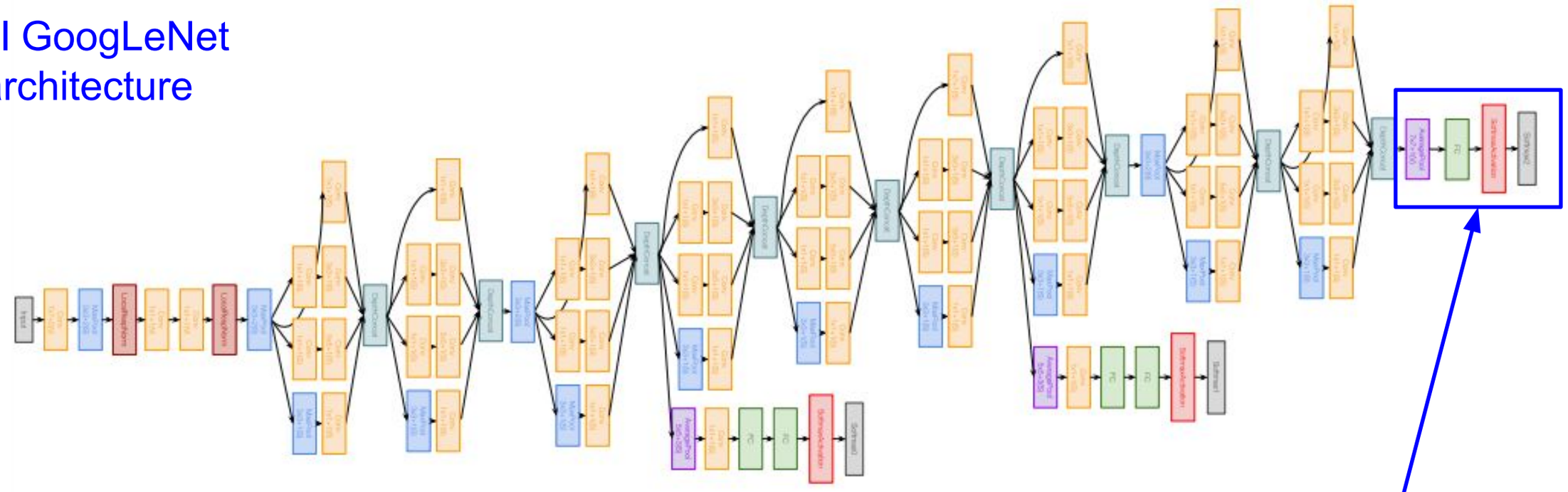
Classifier output

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

Case Study: GoogLeNet

[Szegedy et al., 2014]

Full GoogLeNet architecture



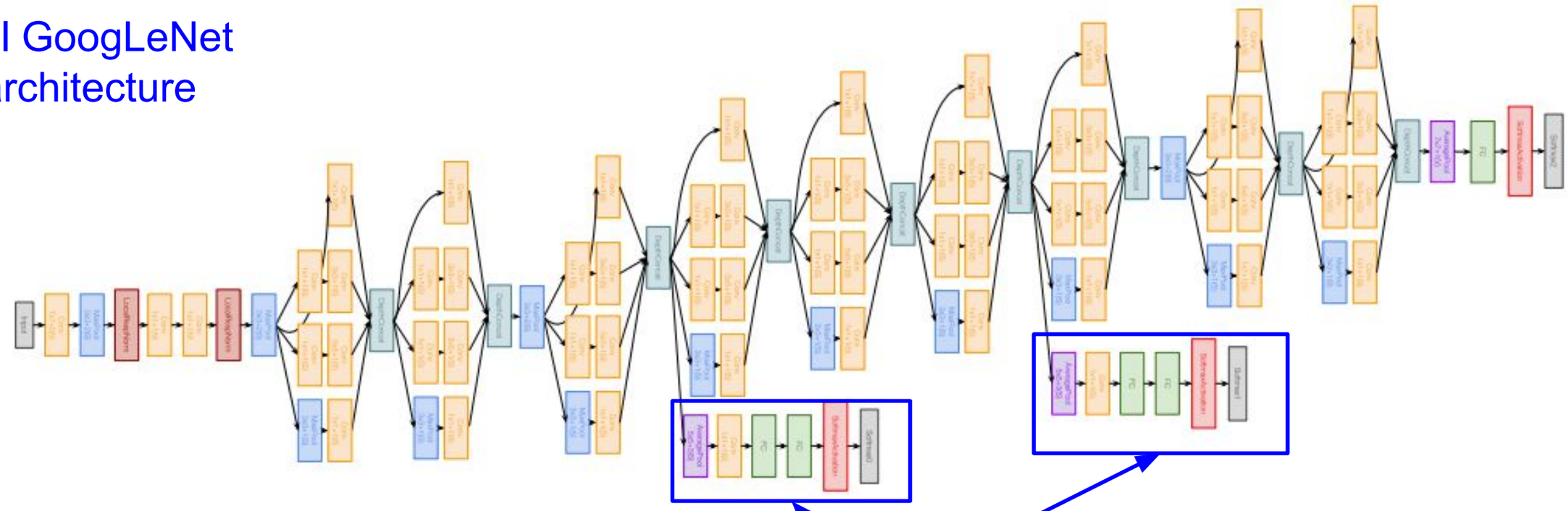
Classifier output
(removed expensive FC layers!)

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

Case Study: GoogLeNet

[Szegedy et al., 2014]

Full GoogLeNet architecture



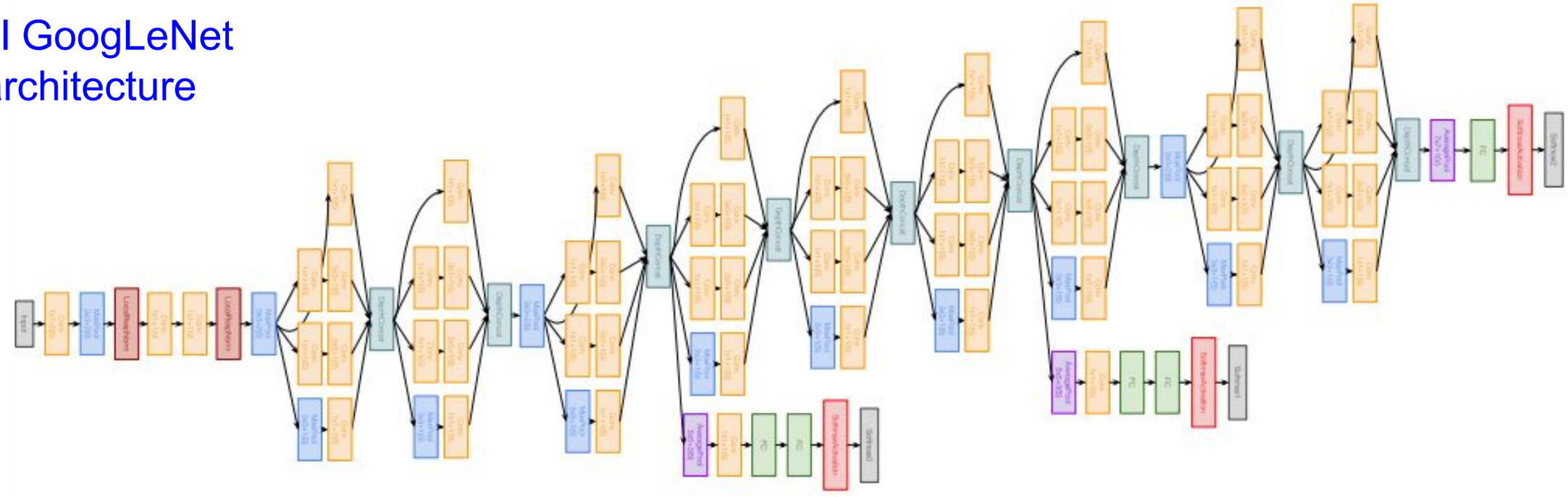
Auxiliary classification outputs to inject additional gradient at lower layers (AvgPool-1x1Conv-FC-FC-Softmax)

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)

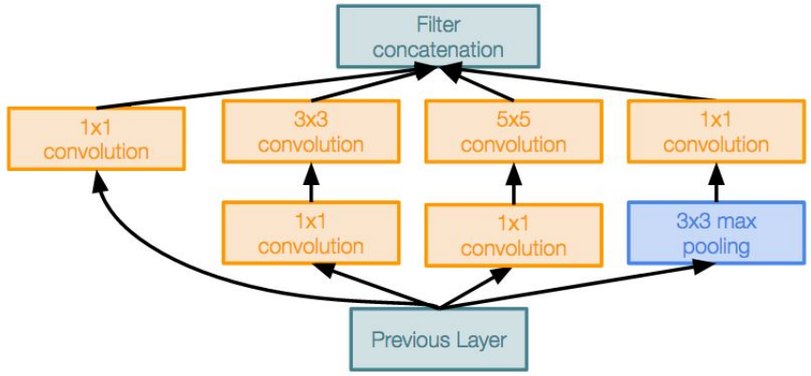
slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Case Study: GoogLeNet

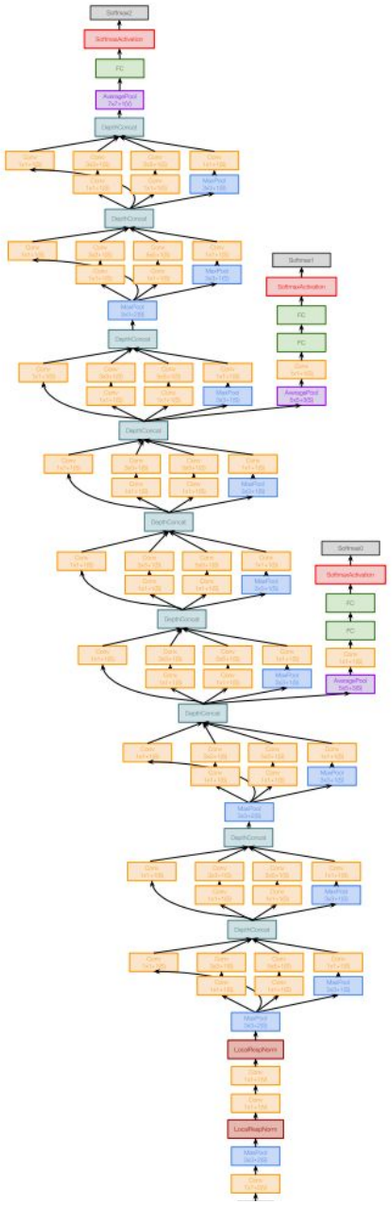
[Szegedy et al., 2014]

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)

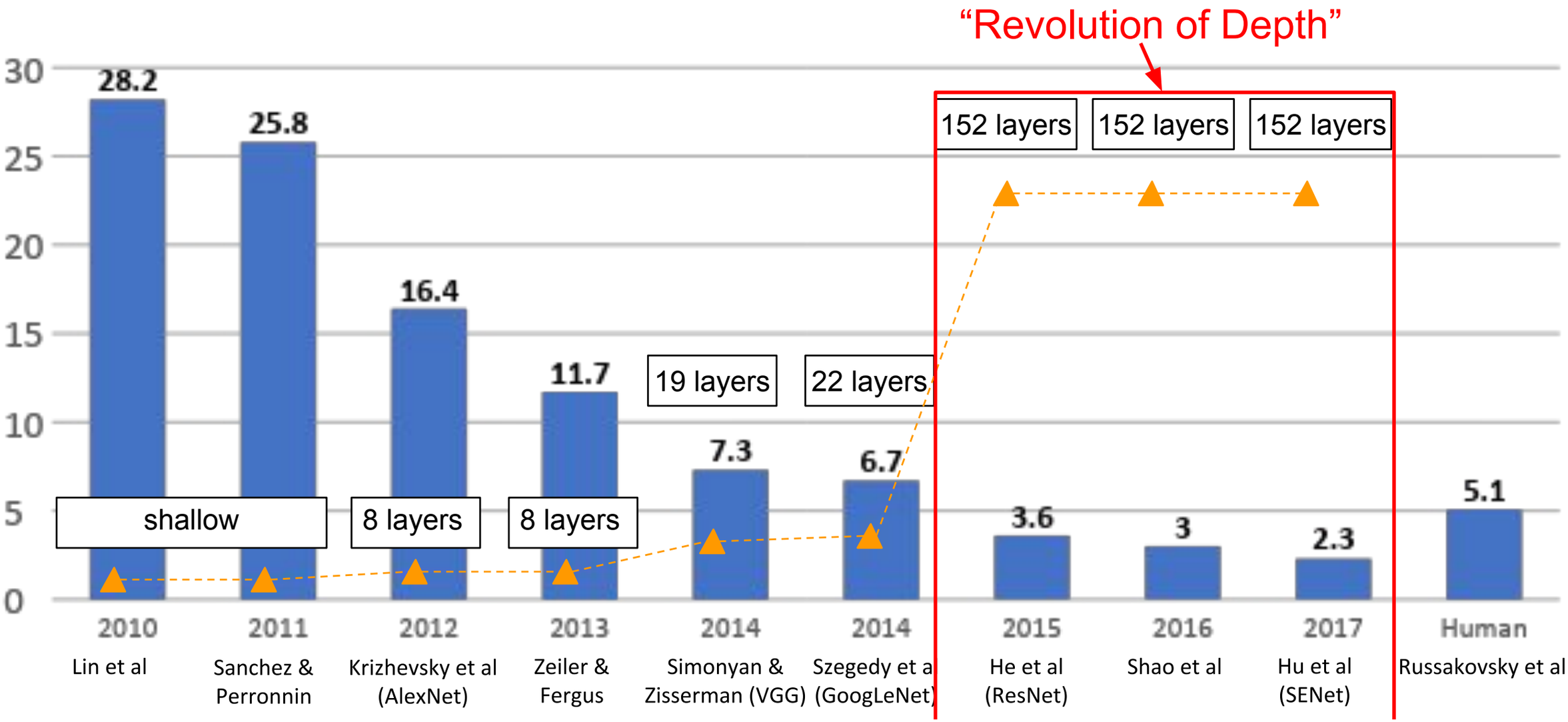


Inception module



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

ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners



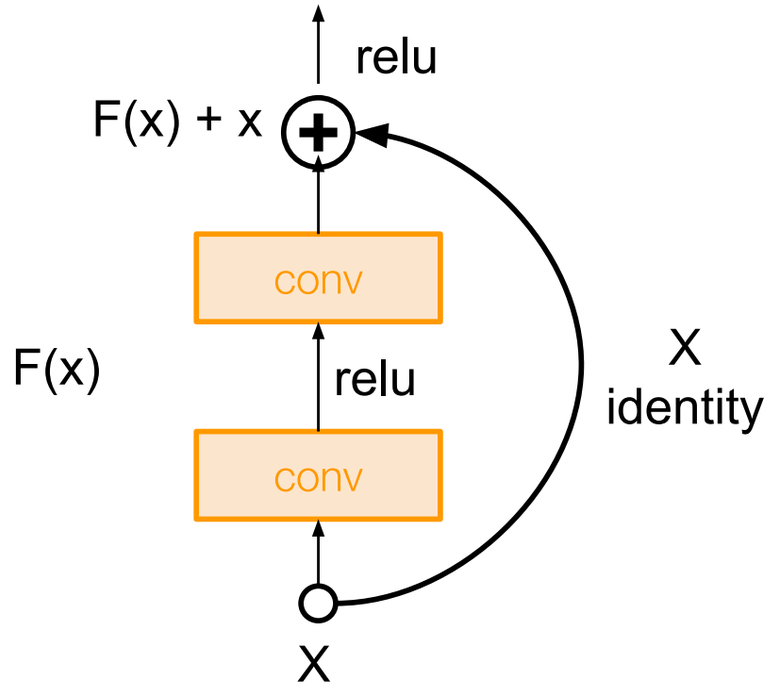
slide credit: Fei-Fei, Justin Johnson, Serena Yeung

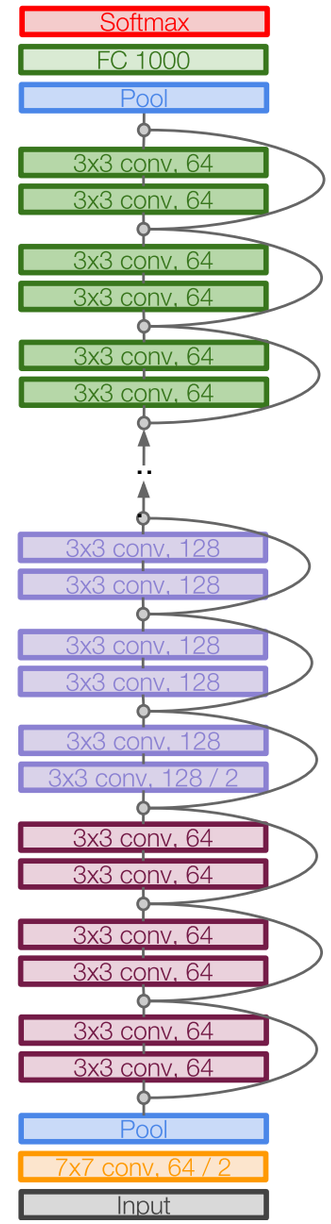
Case Study: ResNet

[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!





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

Case Study: ResNet

[He et al., 2015]

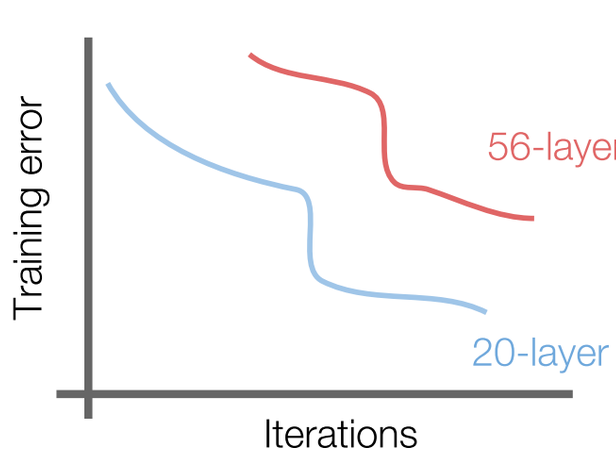
What happens when we continue stacking deeper layers on a “plain” convolutional neural network?

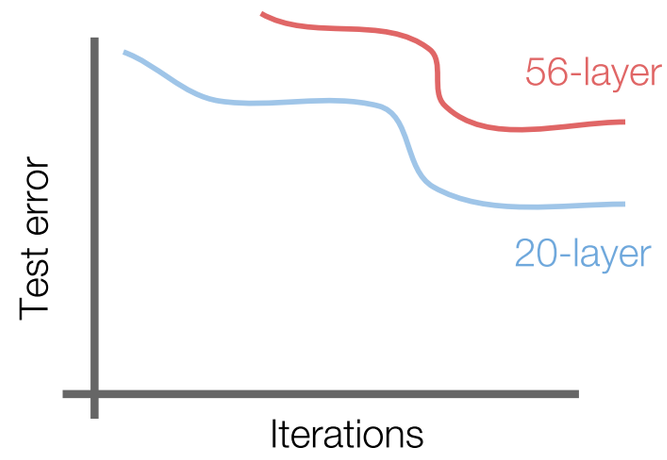
slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Case Study: ResNet

[He et al., 2015]

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





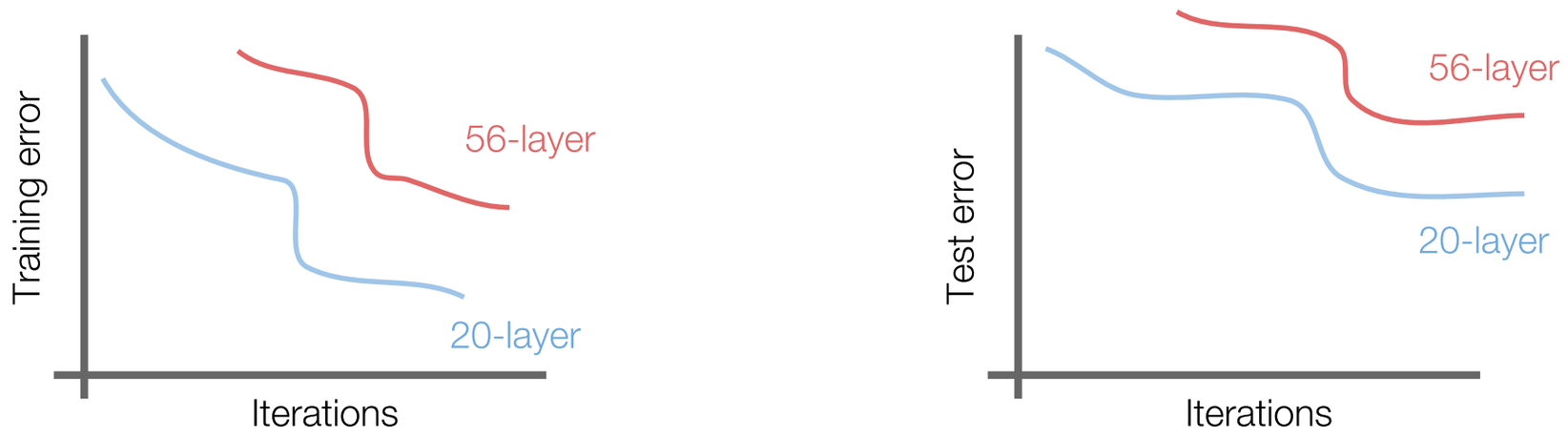
Q: What’s strange about these training and test curves?
[Hint: look at the order of the curves]

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

Case Study: ResNet

[He et al., 2015]

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!

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

Case Study: ResNet

[He et al., 2015]

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

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

Case Study: ResNet

[He et al., 2015]

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

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

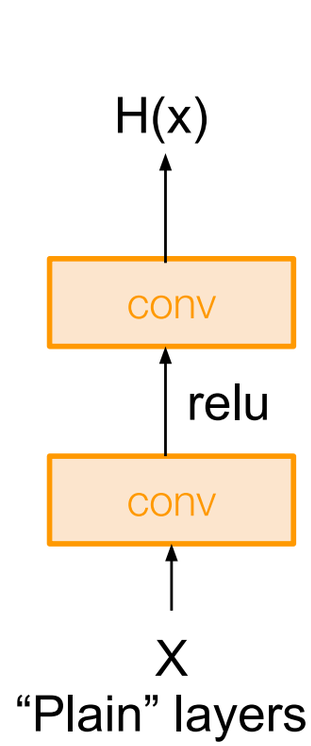
A solution by construction is copying the learned layers from the shallower model and setting additional layers to identity mapping.

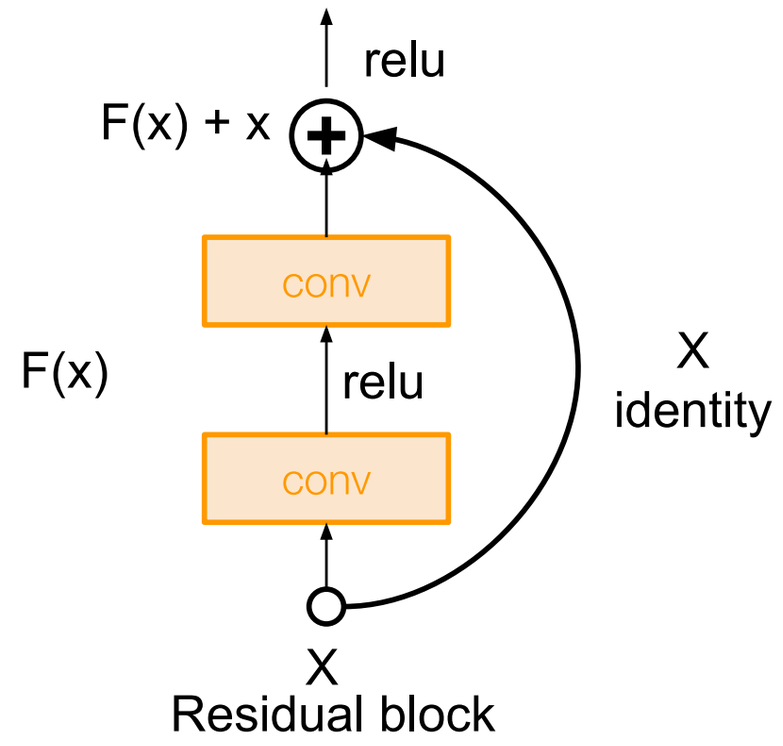
slide credit: Fei-Fei, Justin Johnson, Serena Yeung

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



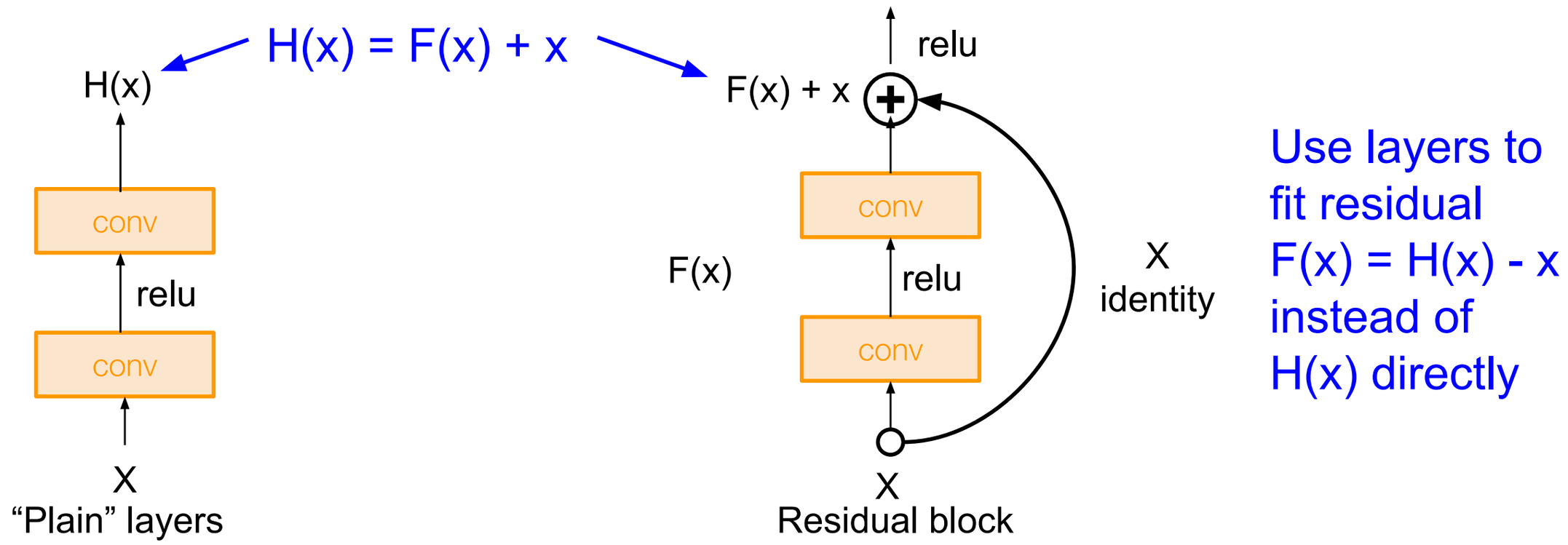


slide credit: Fei-Fei, Justin Johnson, Serena Yeung

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



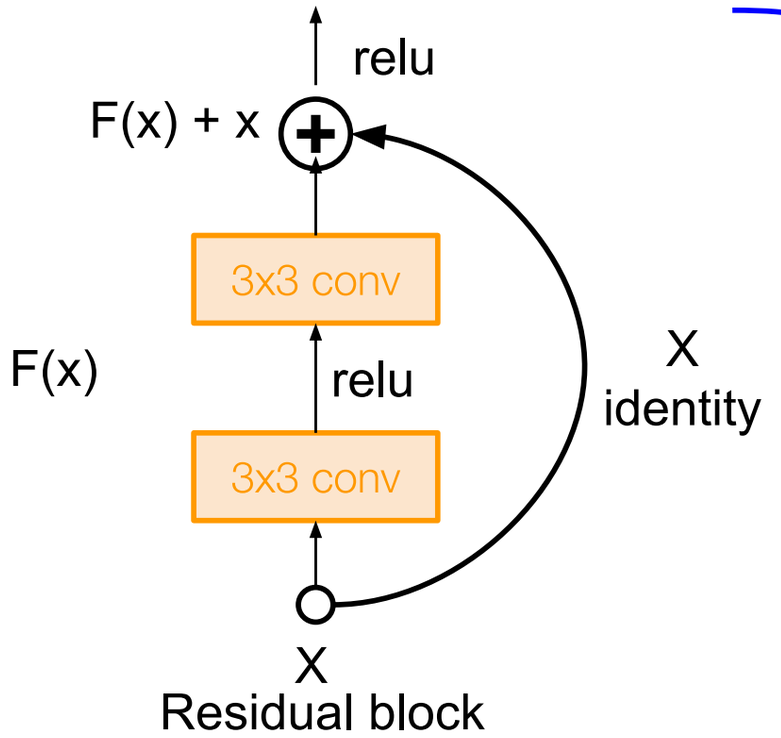
slide credit: Fei-Fei, Justin Johnson, Serena Yeung

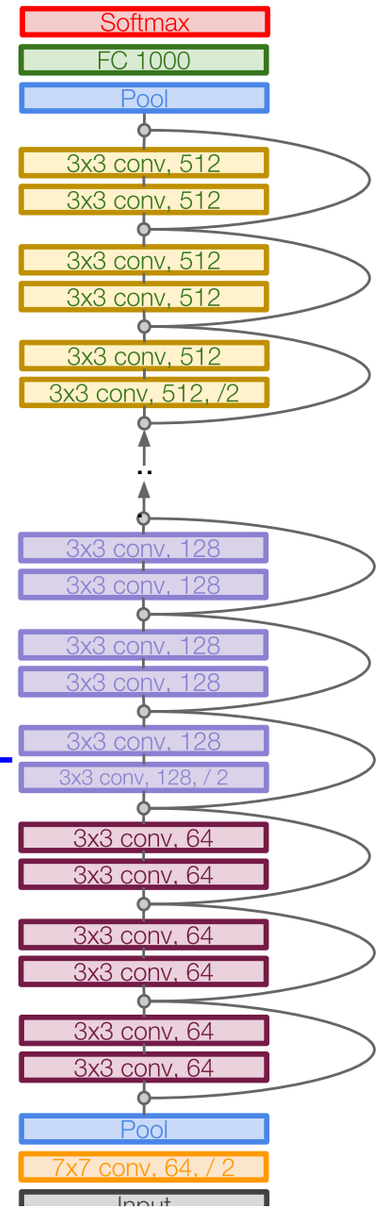
Case Study: ResNet

[He et al., 2015]

Full ResNet architecture:

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





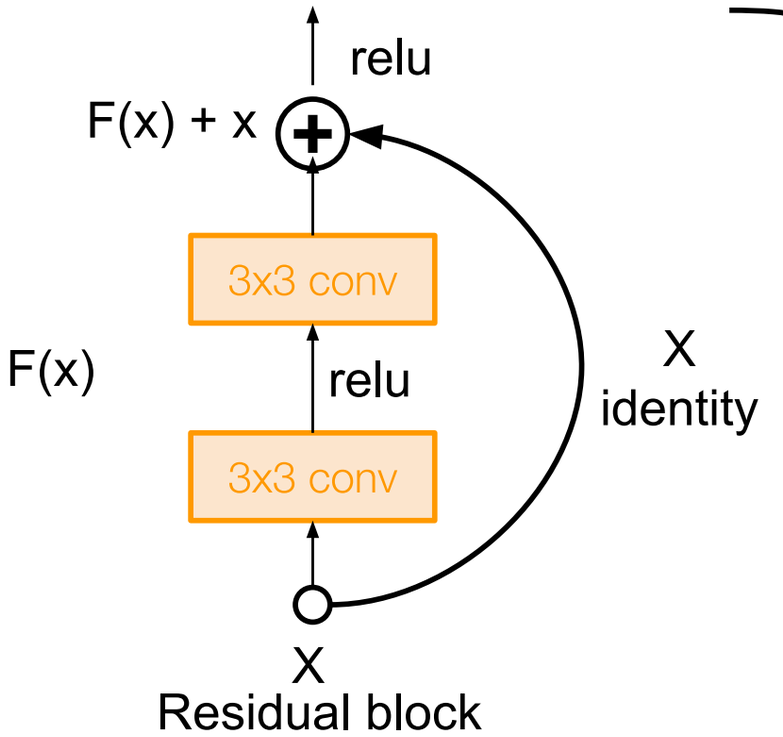
slide credit: Fei-Fei, Justin Johnson, Serena Yeung

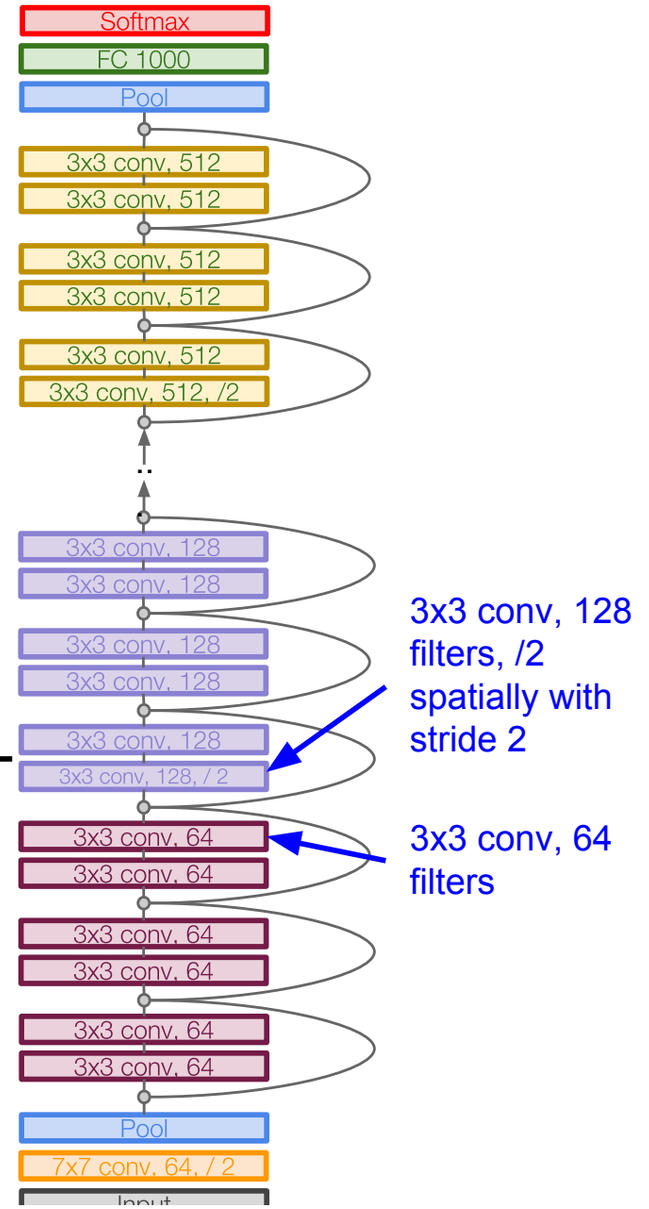
Case Study: ResNet

[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)





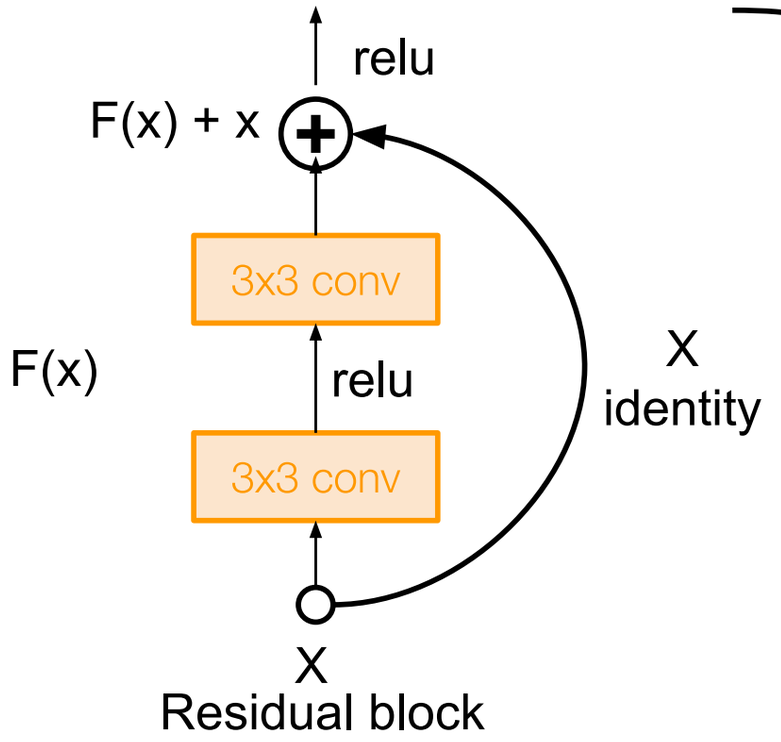
slide credit: Fei-Fei, Justin Johnson, Serena Yeung

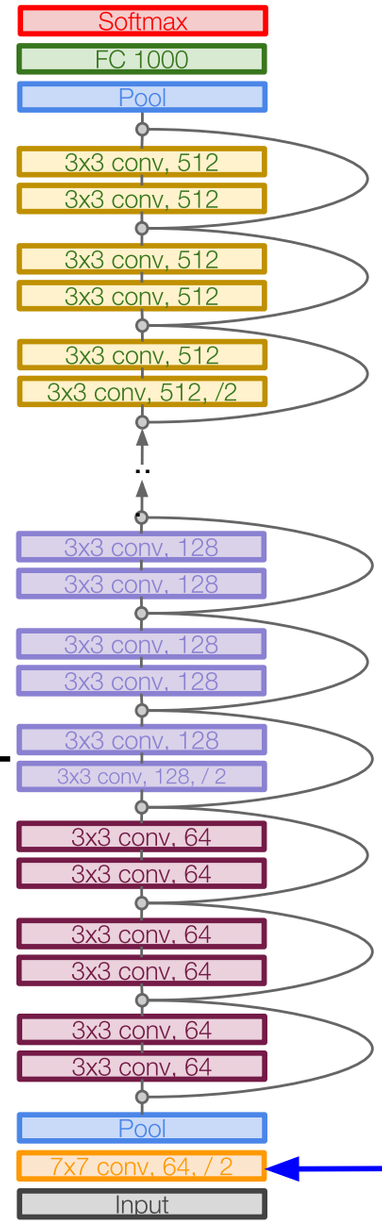
Case Study: ResNet

[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





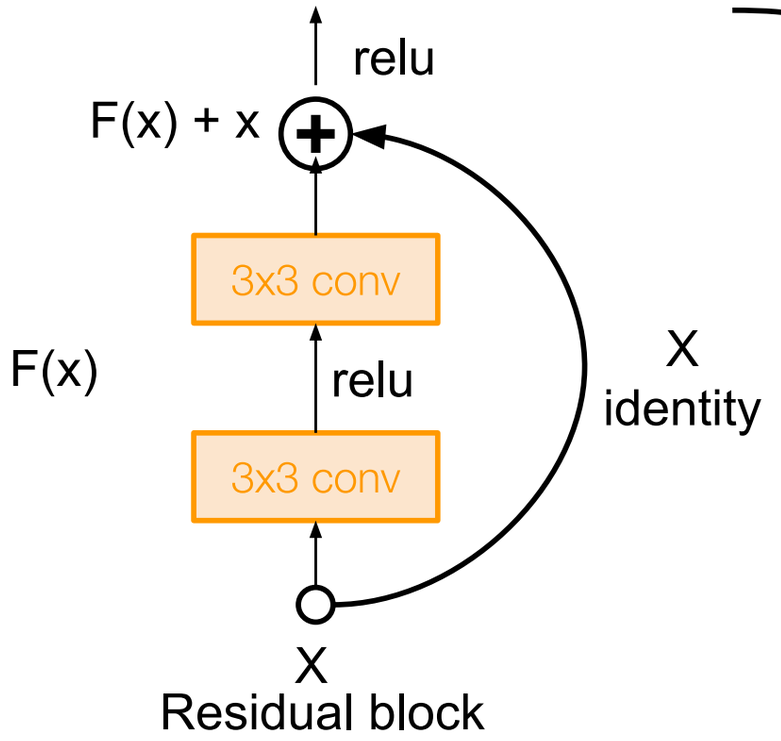
slide credit: Fei-Fei, Justin Johnson, Serena Yeung

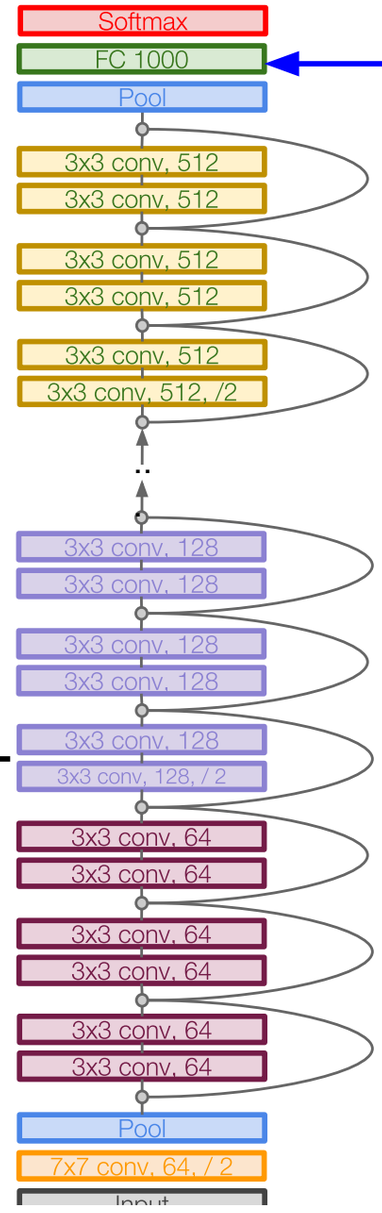
Case Study: ResNet

[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)





No FC layers besides FC 1000 to output classes

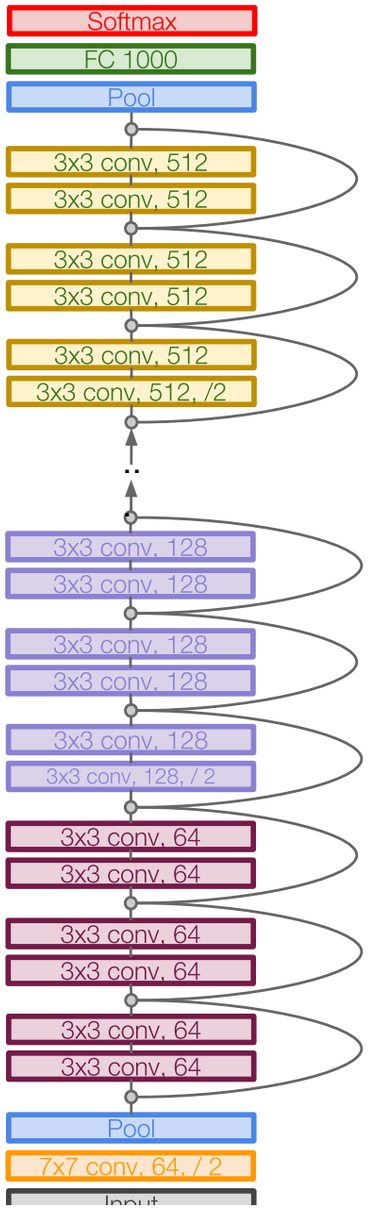
Global average pooling layer after last conv layer

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

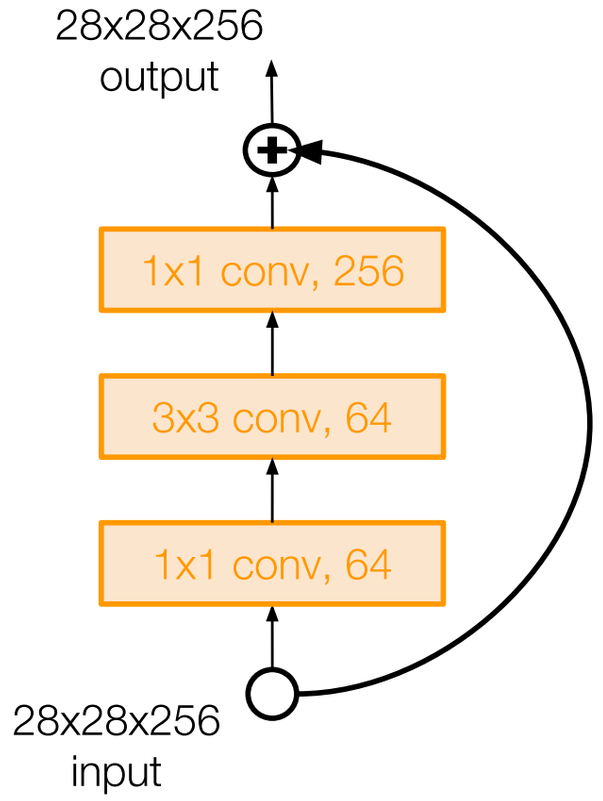


slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Case Study: ResNet

[He et al., 2015]

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

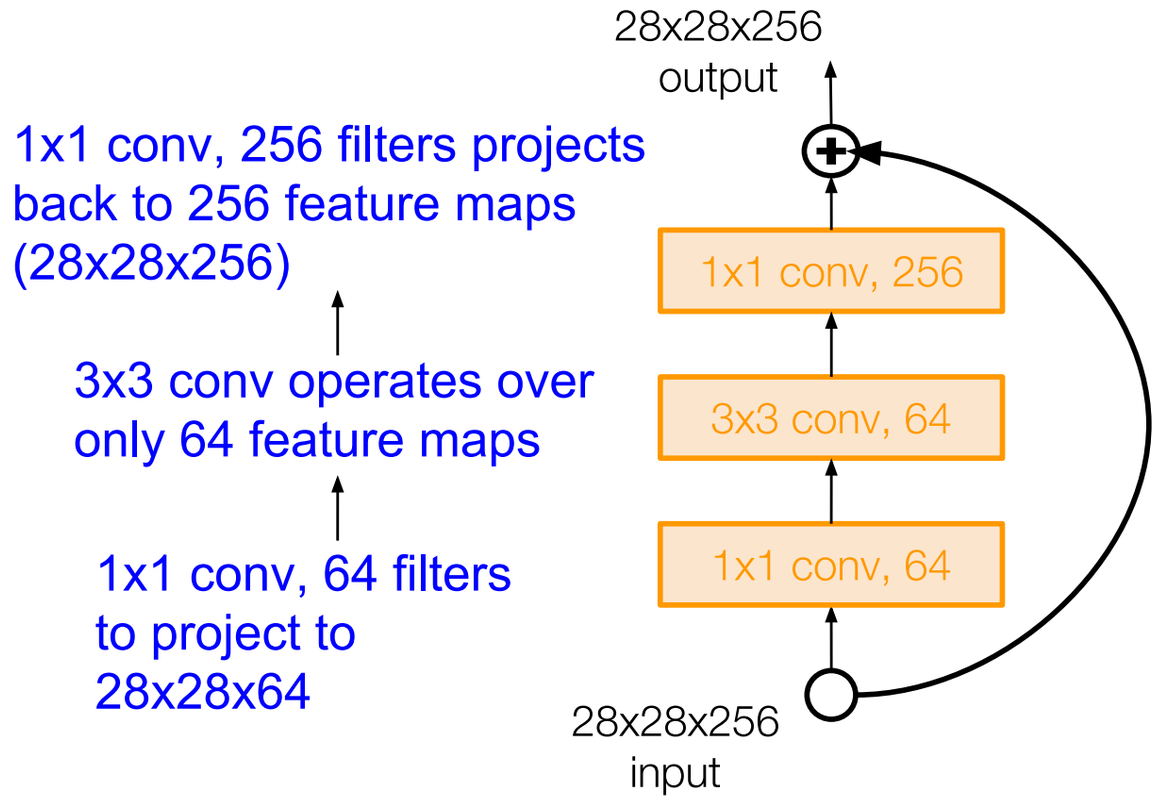


slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Case Study: ResNet

[He et al., 2015]

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



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

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

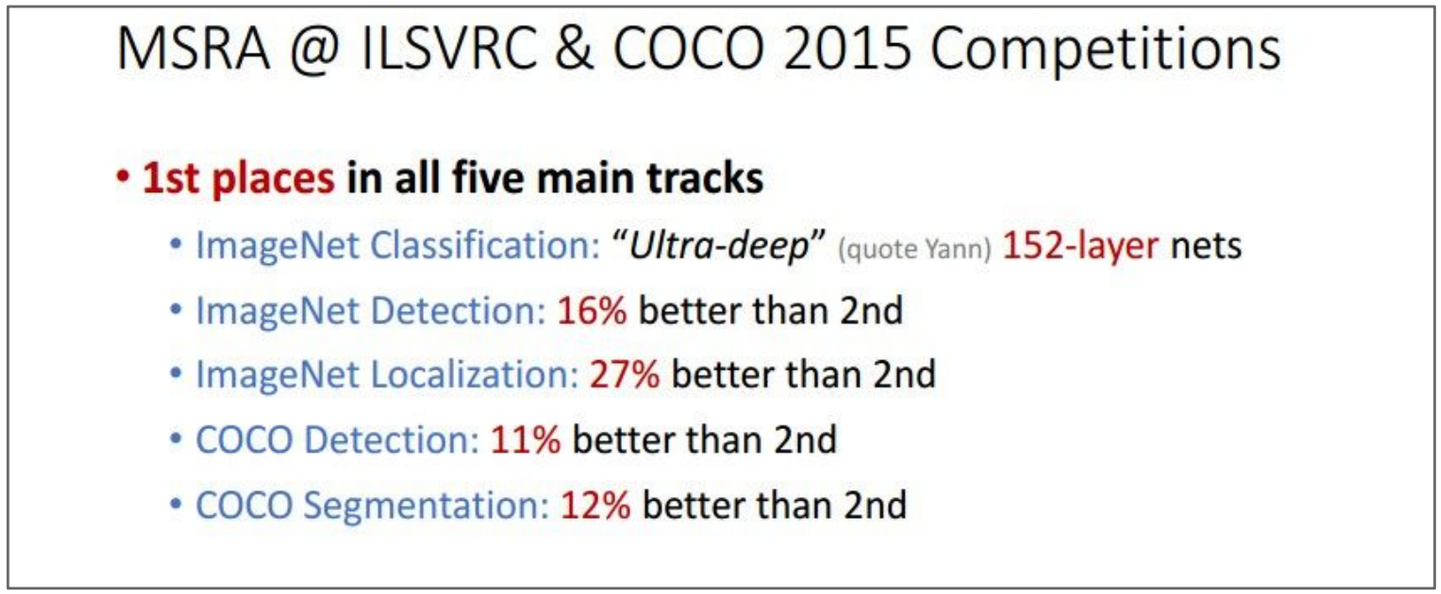
slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Case Study: ResNet

[He et al., 2015]

Experimental Results

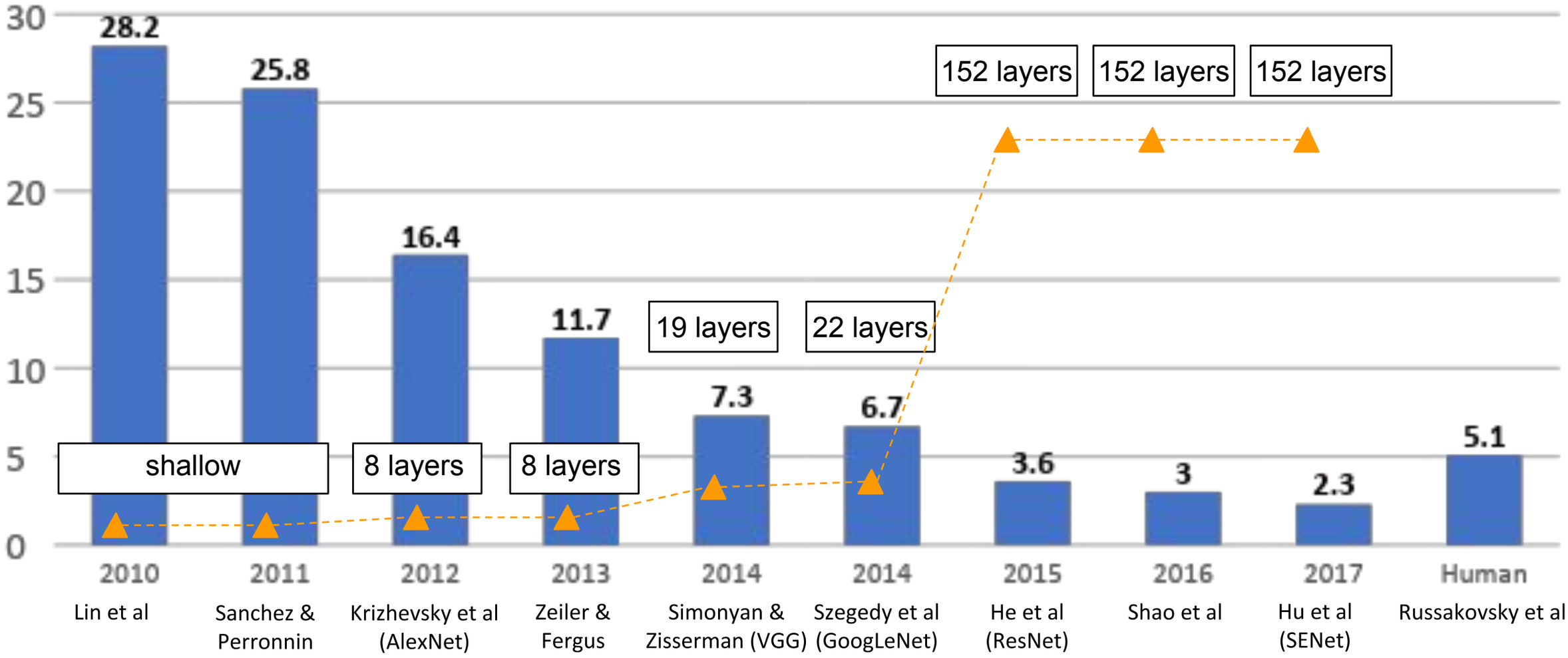
- Able to train very deep networks without degrading (152 layers on ImageNet, 1202 on Cifar)
- Deeper networks now achieve lowing training error as expected
- Swept 1st place in all ILSVRC and COCO 2015 competitions



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

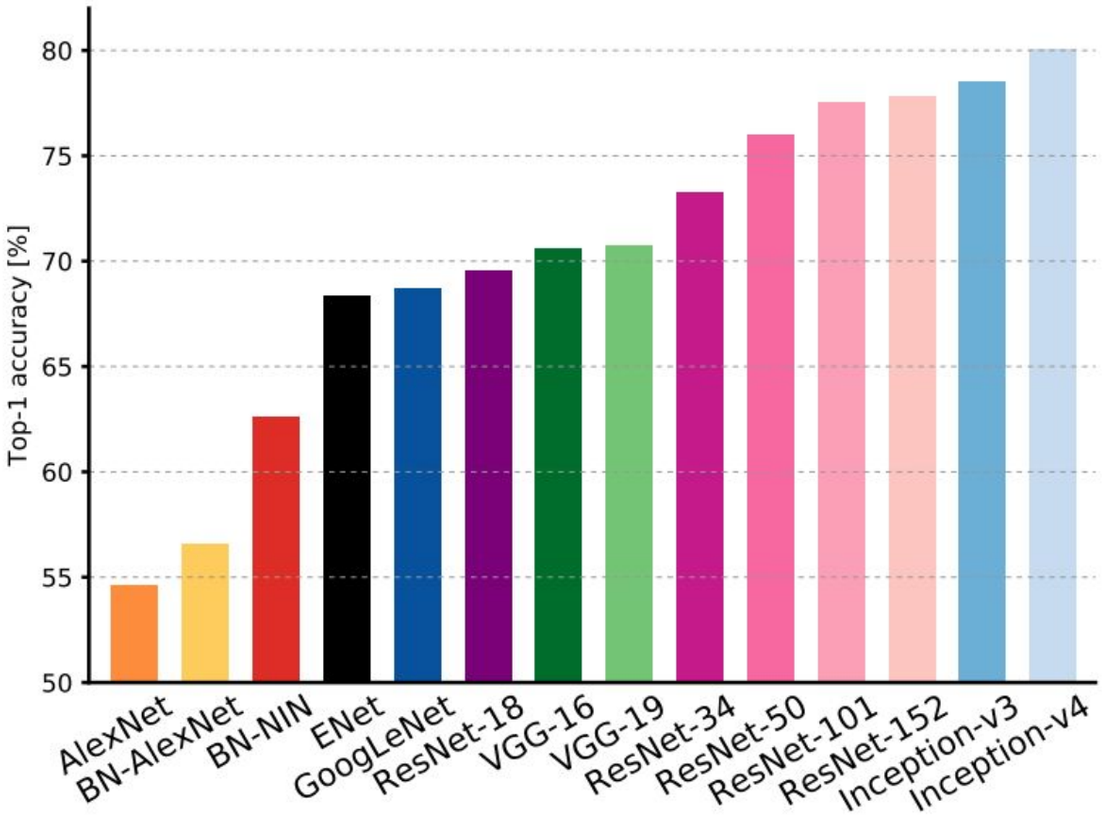
slide credit: Fei-Fei, Justin Johnson, Serena Yeung

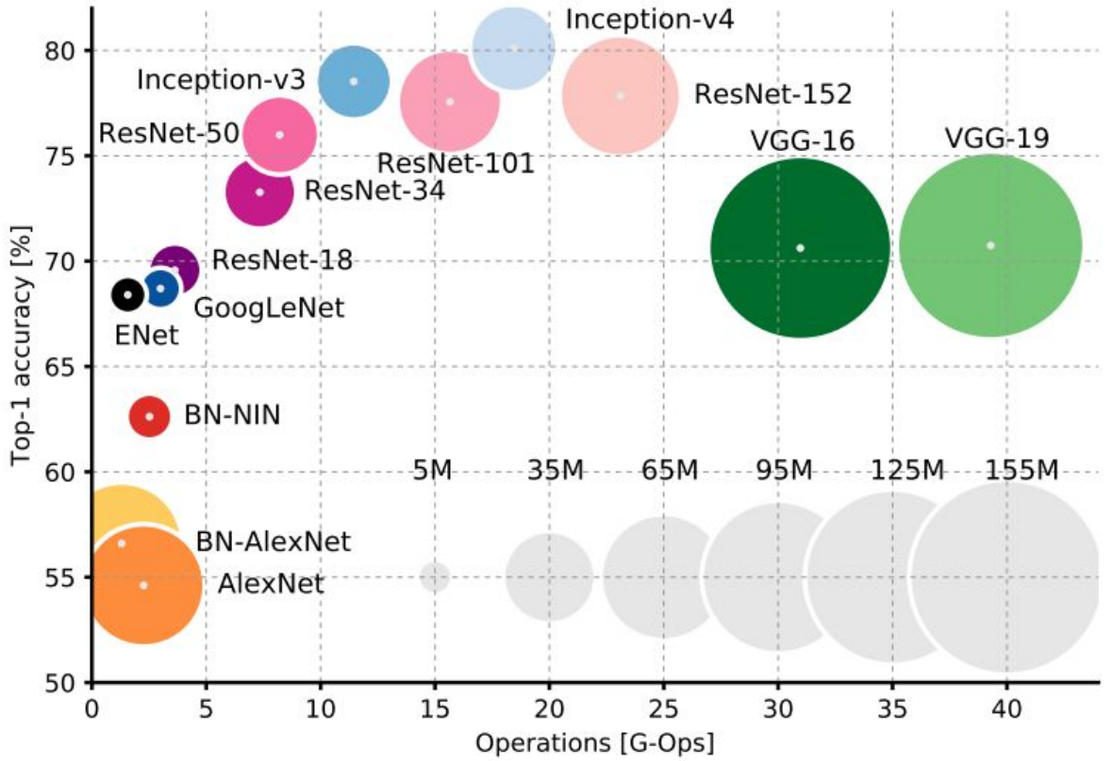
ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners



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

Comparing complexity...



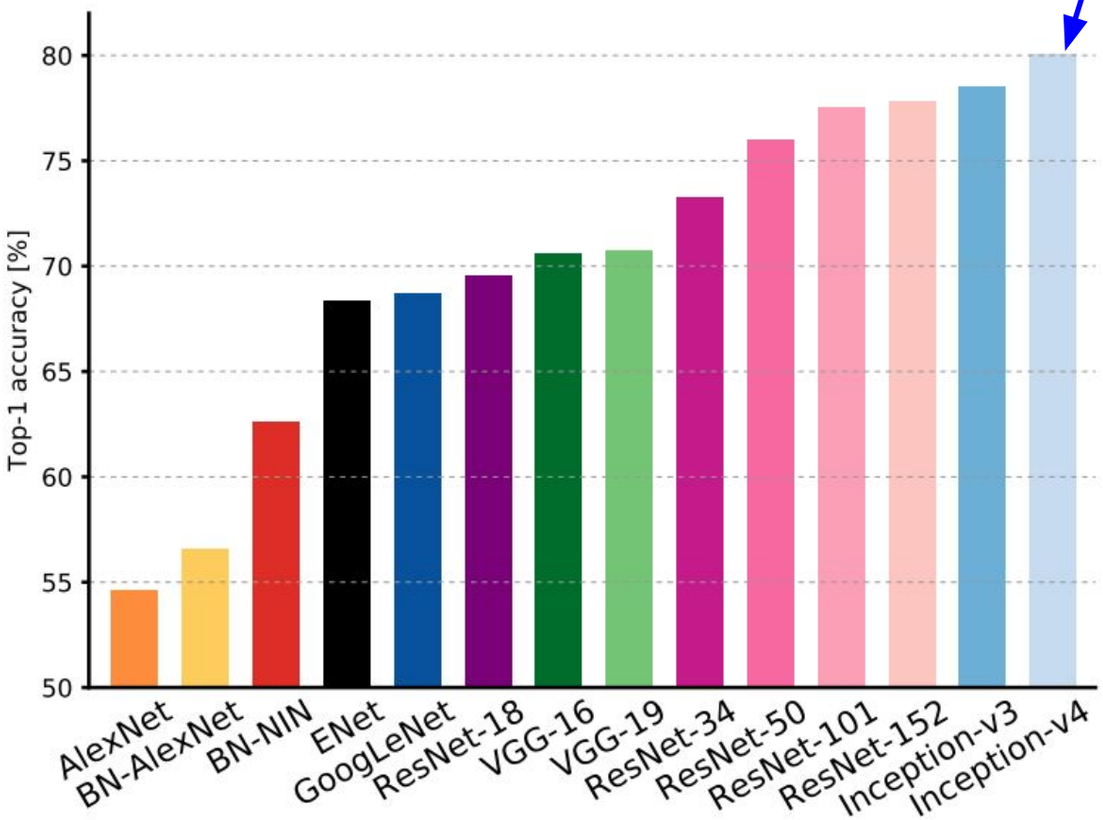


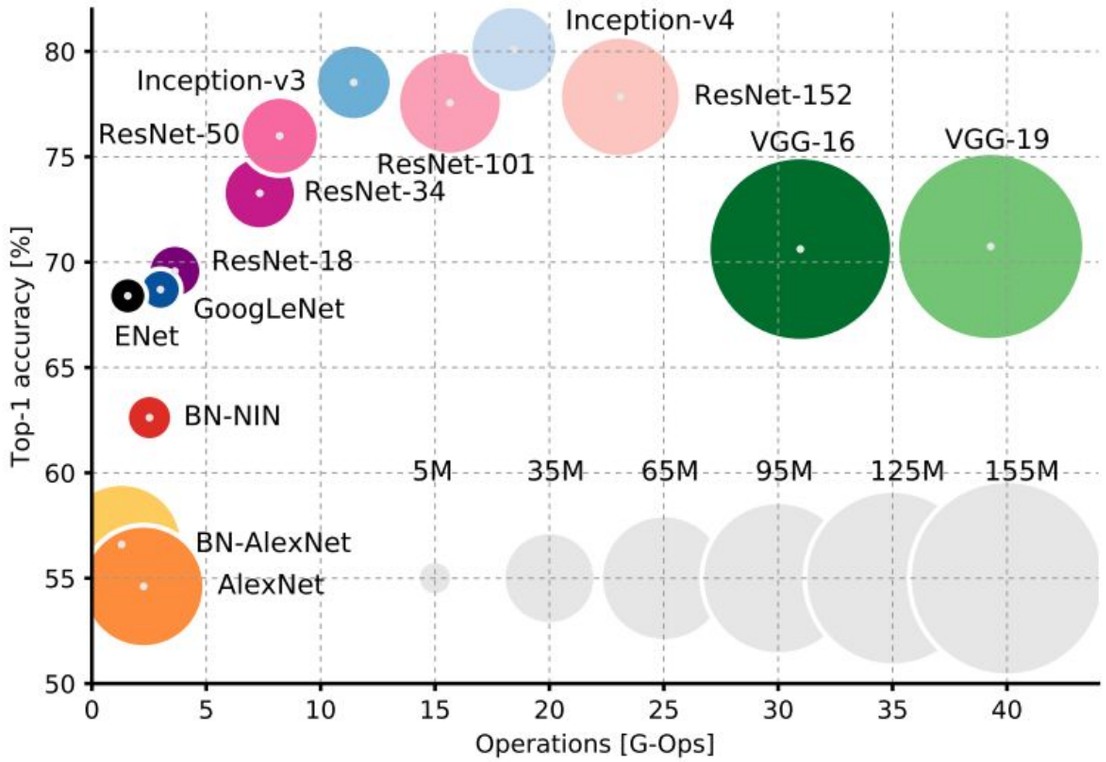
An Analysis of Deep Neural Network Models for Practical Applications, 2017.

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

Comparing complexity...

Inception-v4: Resnet + Inception!



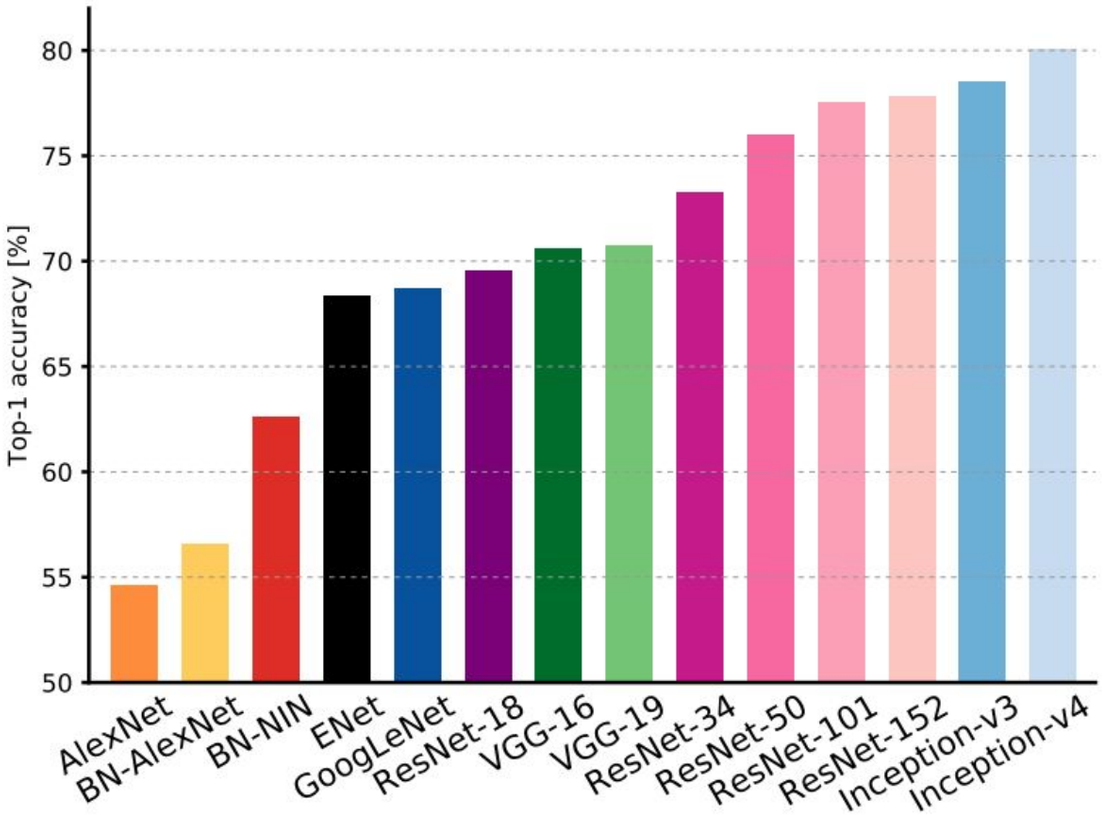


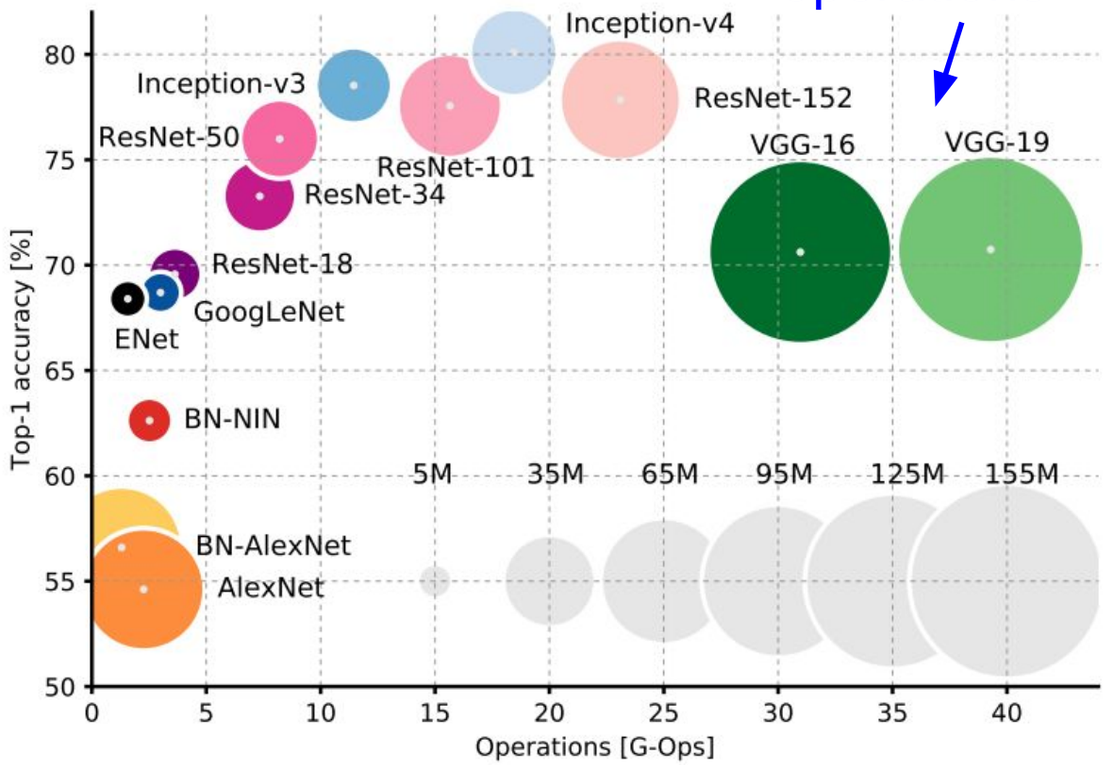
An Analysis of Deep Neural Network Models for Practical Applications, 2017.

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

Comparing complexity...

VGG: Highest memory, most operations



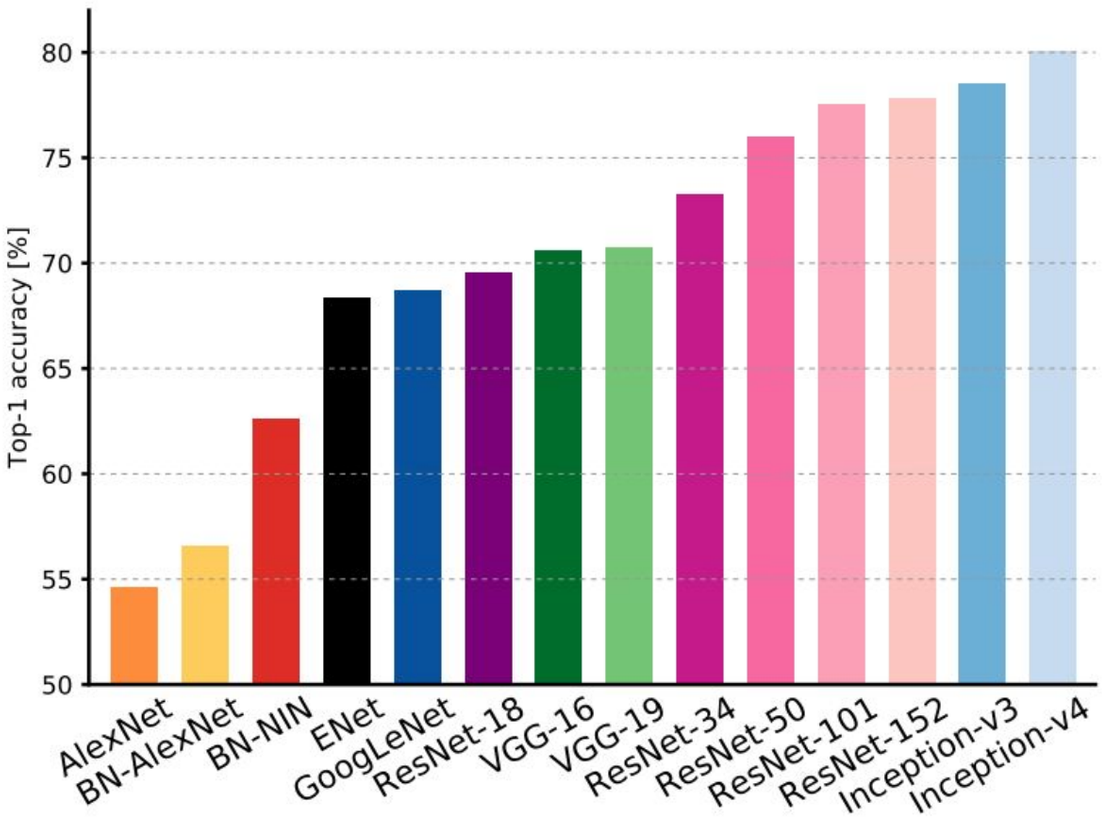


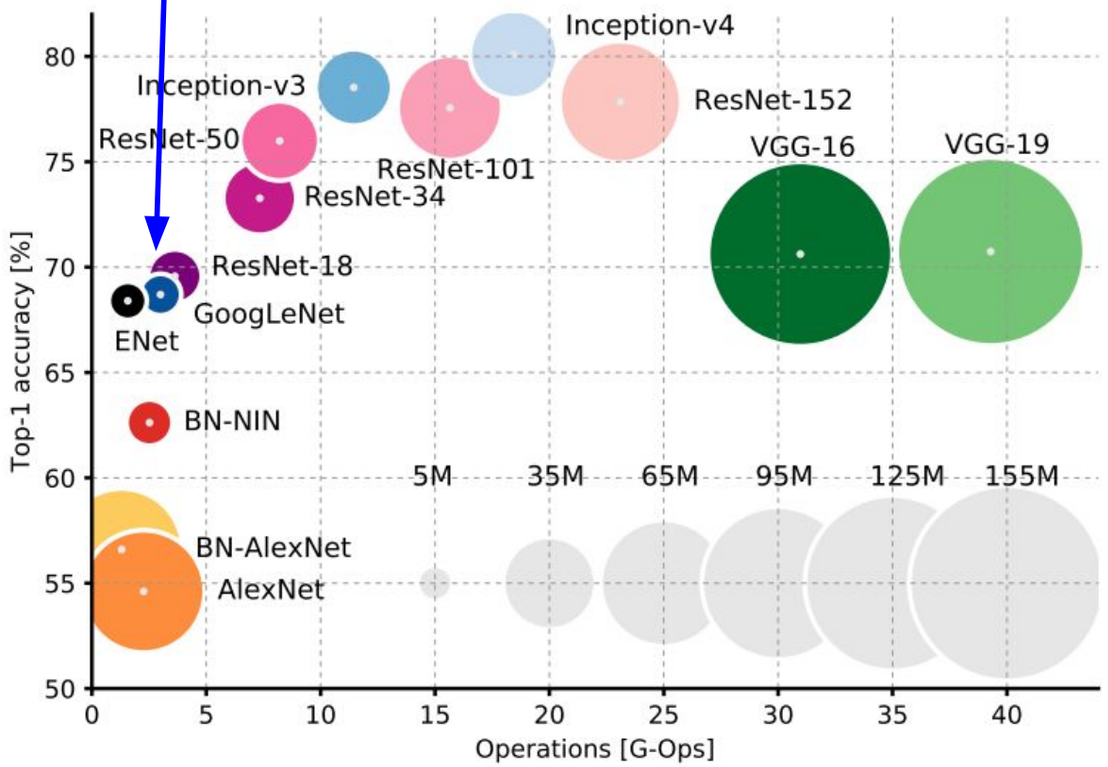
An Analysis of Deep Neural Network Models for Practical Applications, 2017.

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

Comparing complexity...

GoogLeNet:
most efficient



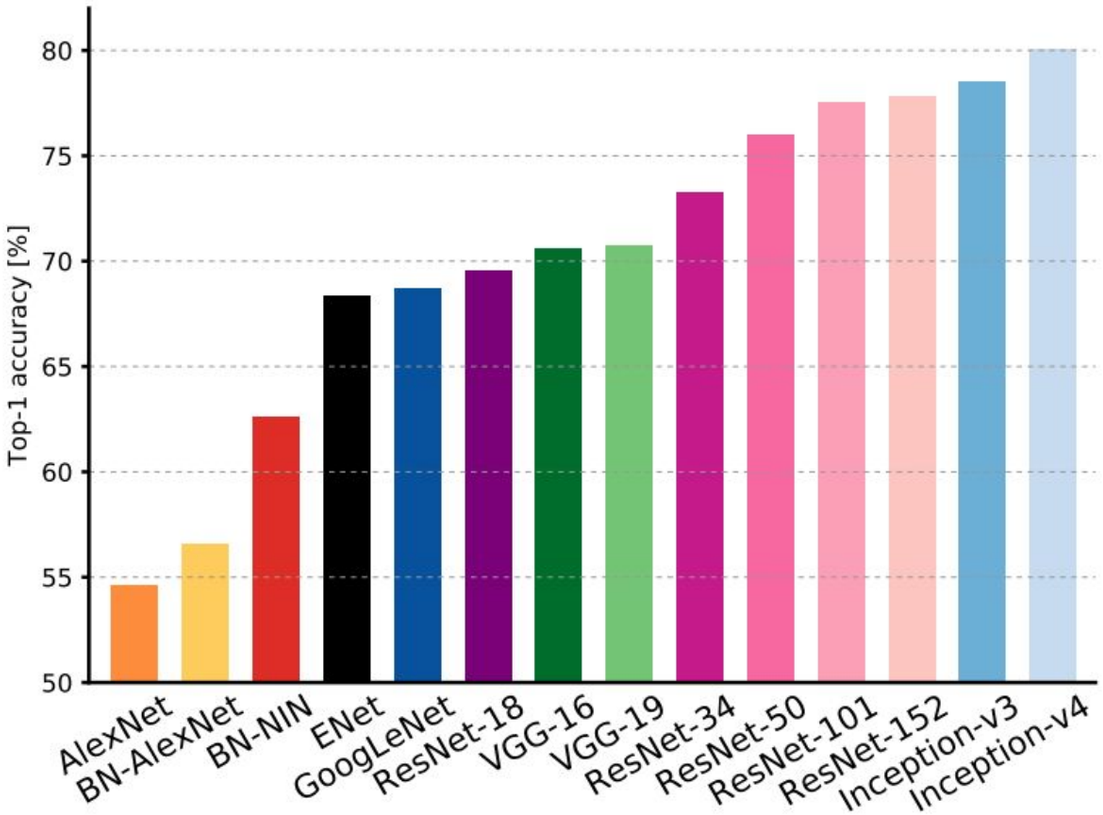


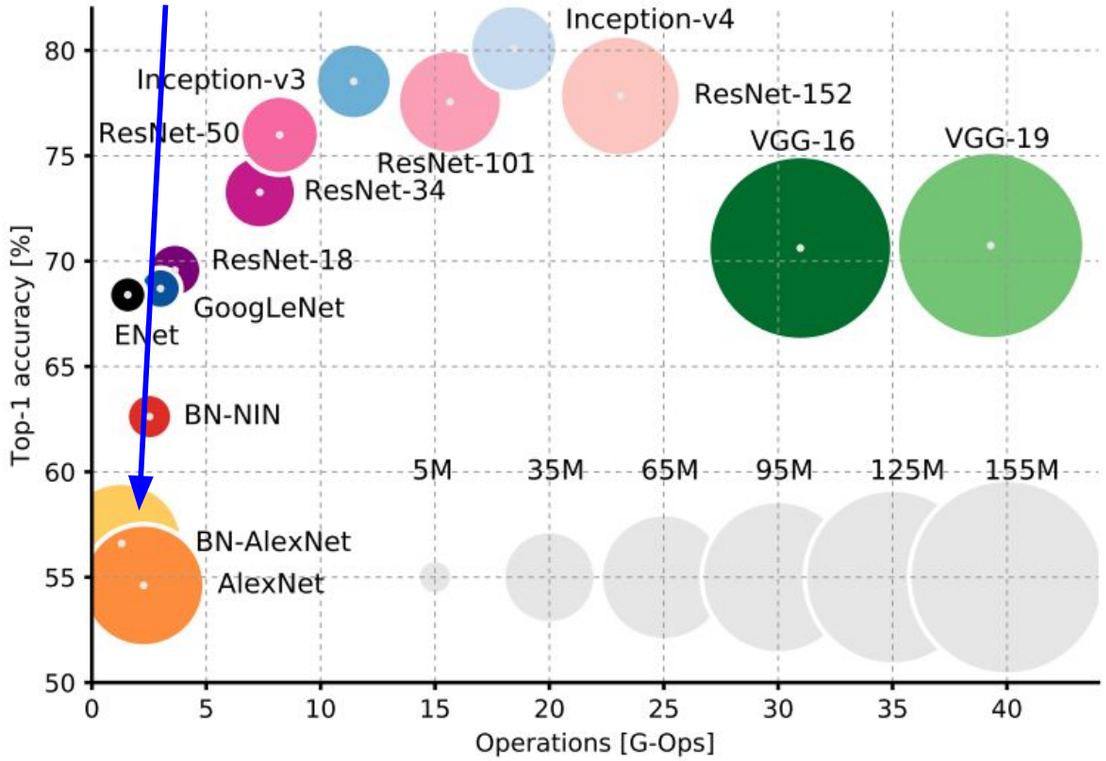
An Analysis of Deep Neural Network Models for Practical Applications, 2017.

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

Comparing complexity...

AlexNet:
Smaller compute, still memory heavy, lower accuracy



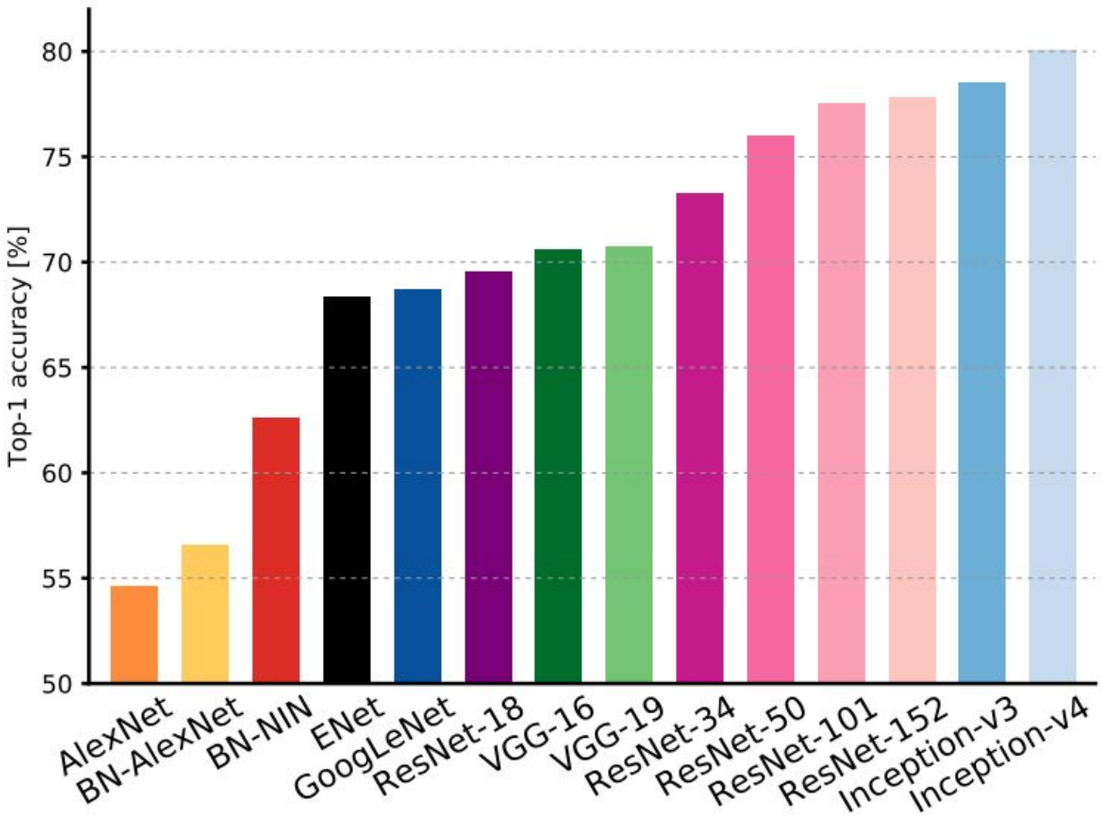


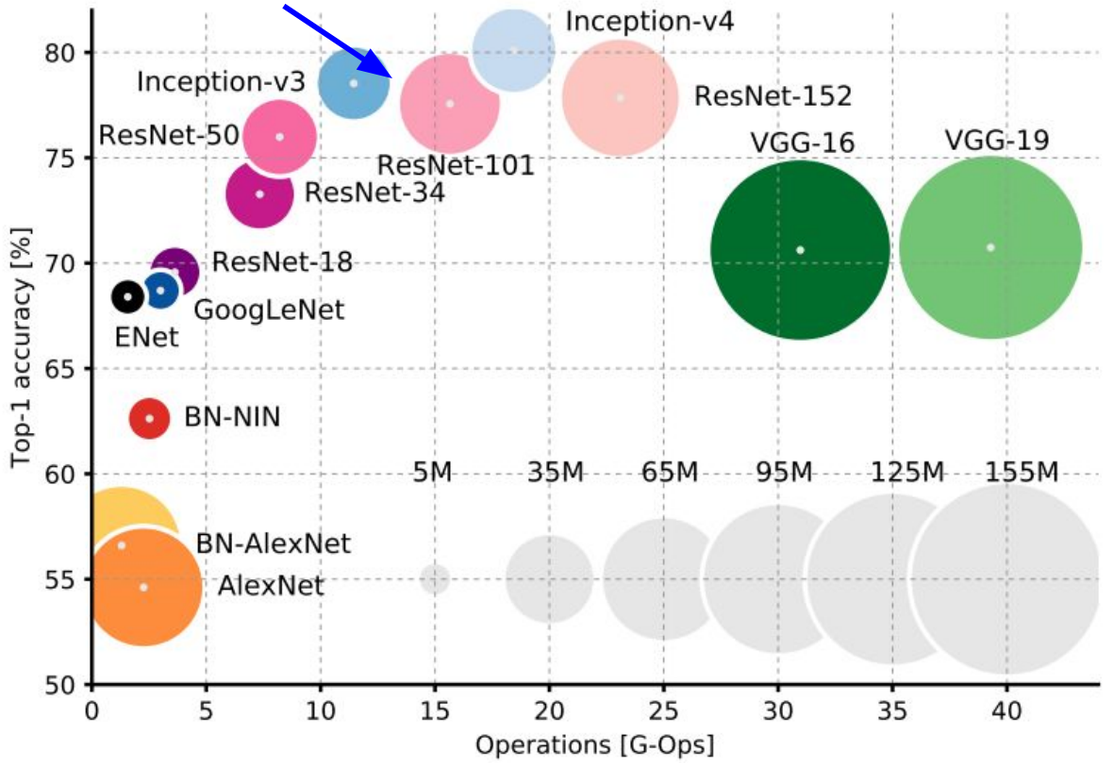
An Analysis of Deep Neural Network Models for Practical Applications, 2017.

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

Comparing complexity...

ResNet:
Moderate efficiency depending on model, highest accuracy

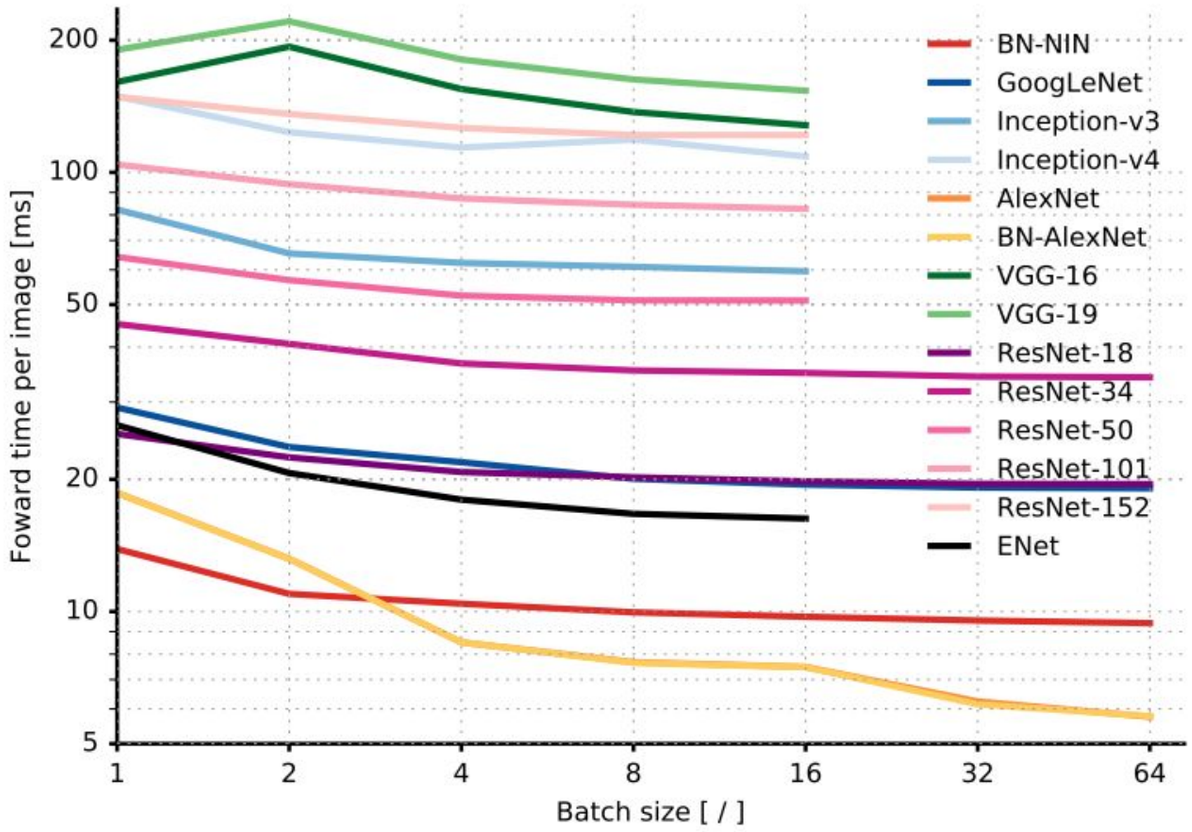


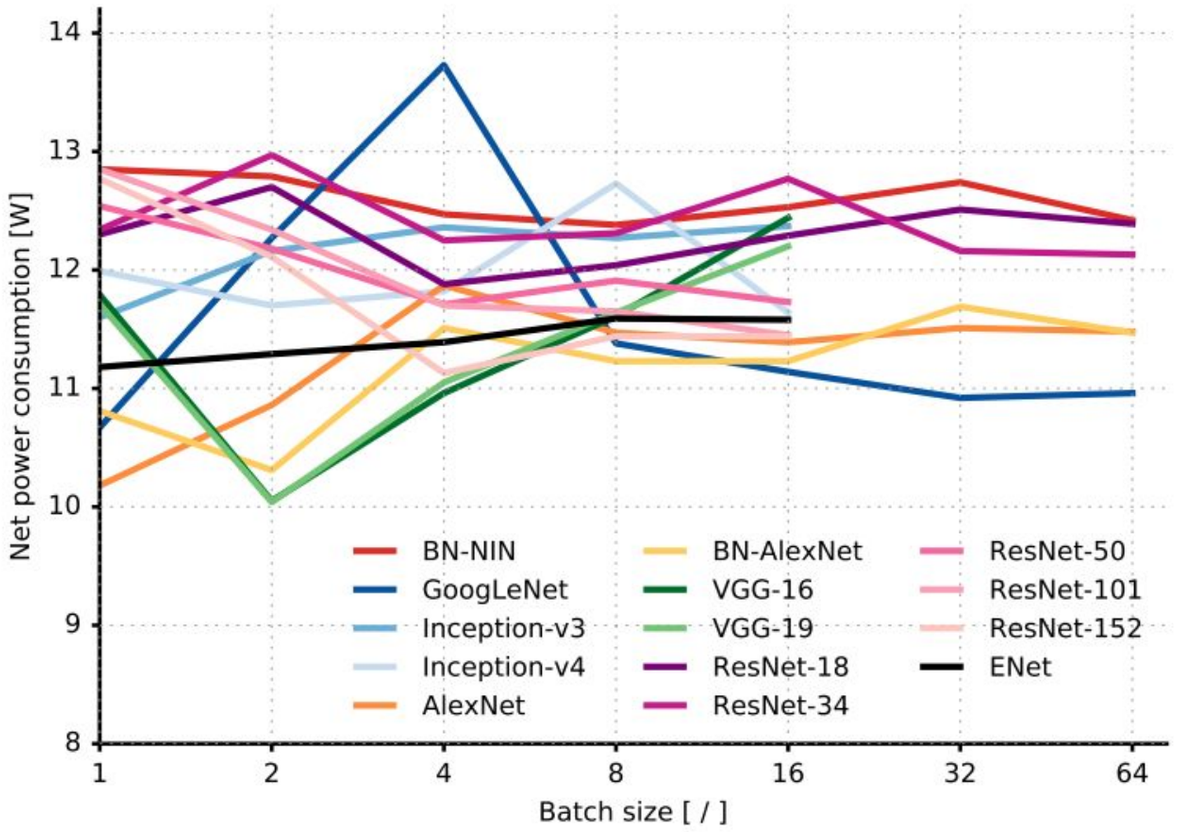


An Analysis of Deep Neural Network Models for Practical Applications, 2017.

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

Forward pass time and power consumption



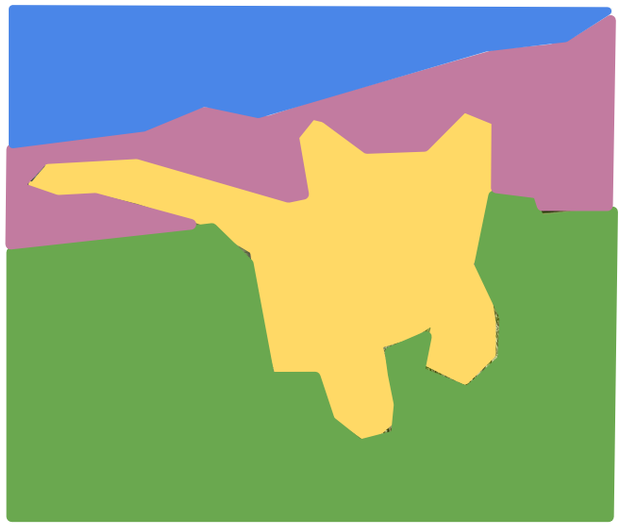


An Analysis of Deep Neural Network Models for Practical Applications, 2017.

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

Other Computer Vision Tasks

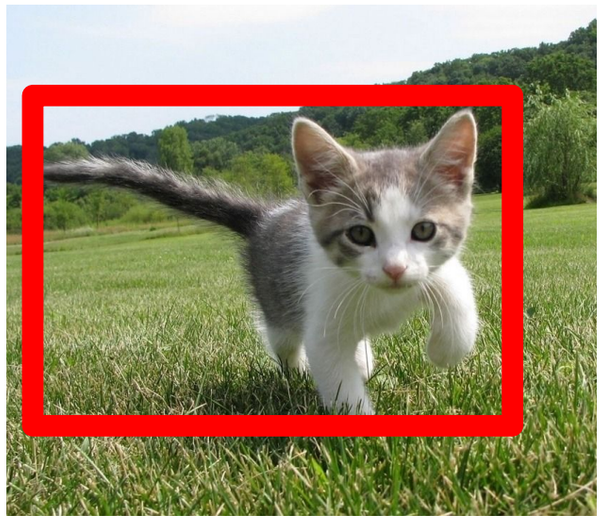
Semantic Segmentation



GRASS, CAT, TREE, SKY

No objects, just pixels

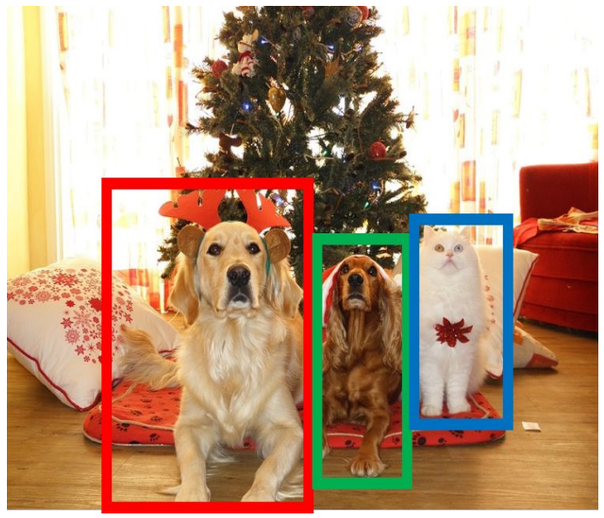
Classification + Localization



CAT

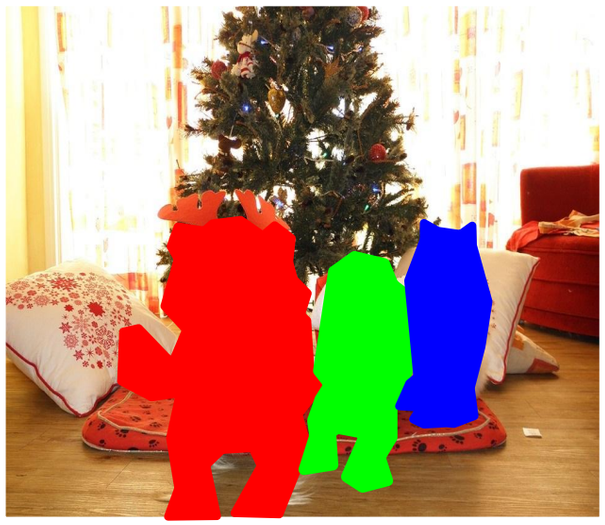
Single Object

Object Detection



DOG, DOG, CAT

Instance Segmentation



DOG, DOG, CAT

Multiple Object

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

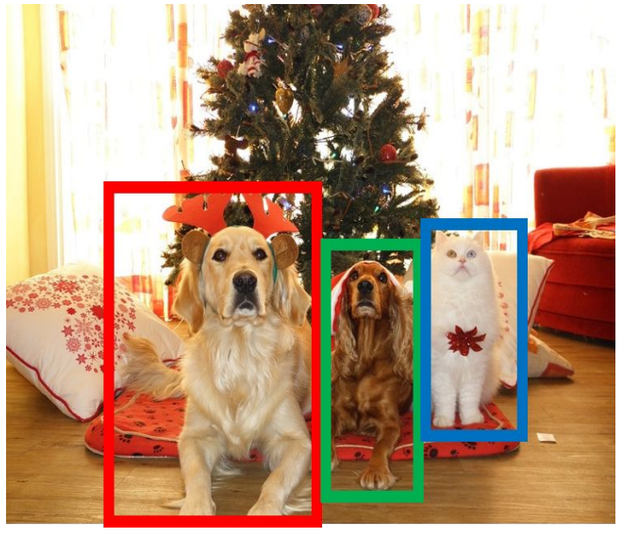
Other Computer Vision Tasks

Semantic Segmentation

GRASS, **CAT**, **TREE**, **SKY**

No objects, just pixels

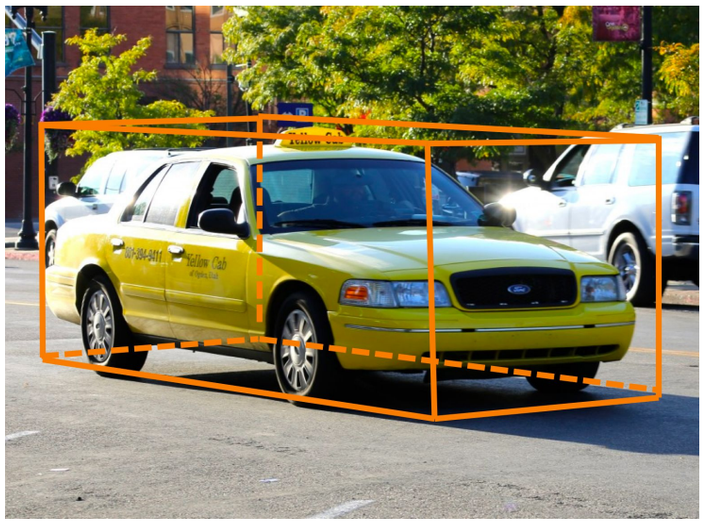
2D Object Detection



DOG, **DOG**, **CAT**

Object categories +
2D bounding boxes

3D Object Detection



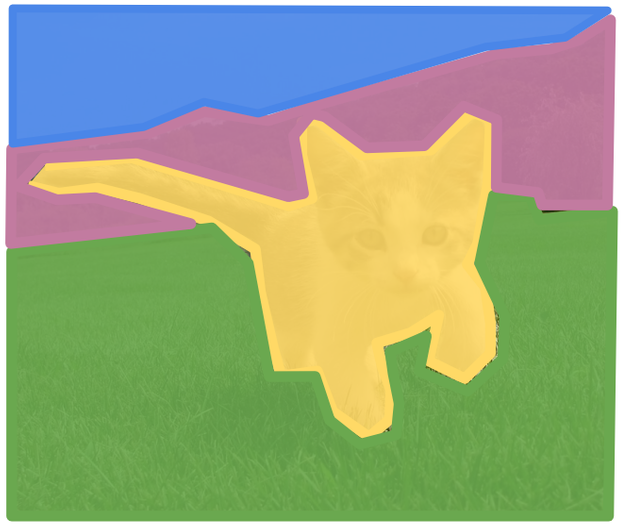
Car

Object categories +
3D bounding boxes

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

Semantic Segmentation

Semantic Segmentation



GRASS, CAT, TREE, SKY

No objects, just pixels

2D Object Detection

DOG, DOG, CAT

Object categories + 2D bounding boxes

3D Object Detection

Car

Object categories + 3D bounding boxes

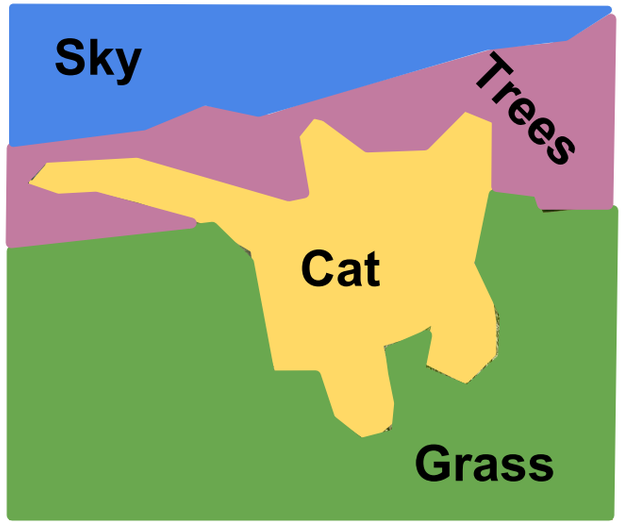
slide credit: Fei-Fei, Justin Johnson, Serena Yeung

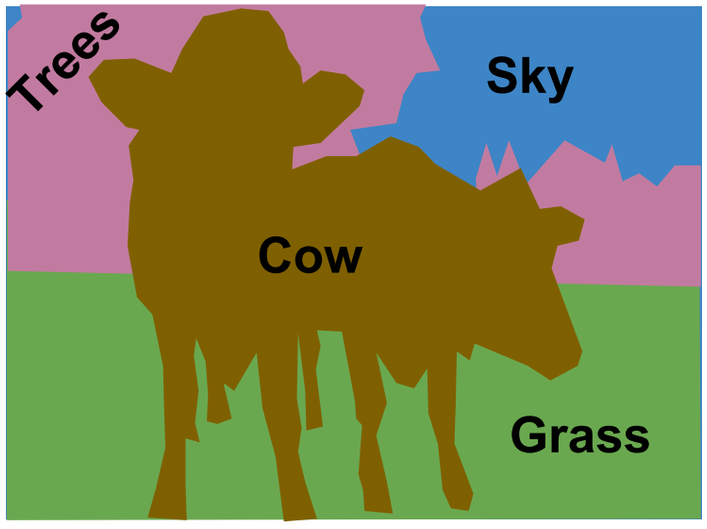
Semantic Segmentation

This image is CC0 public domain

Label each pixel in the image with a category label

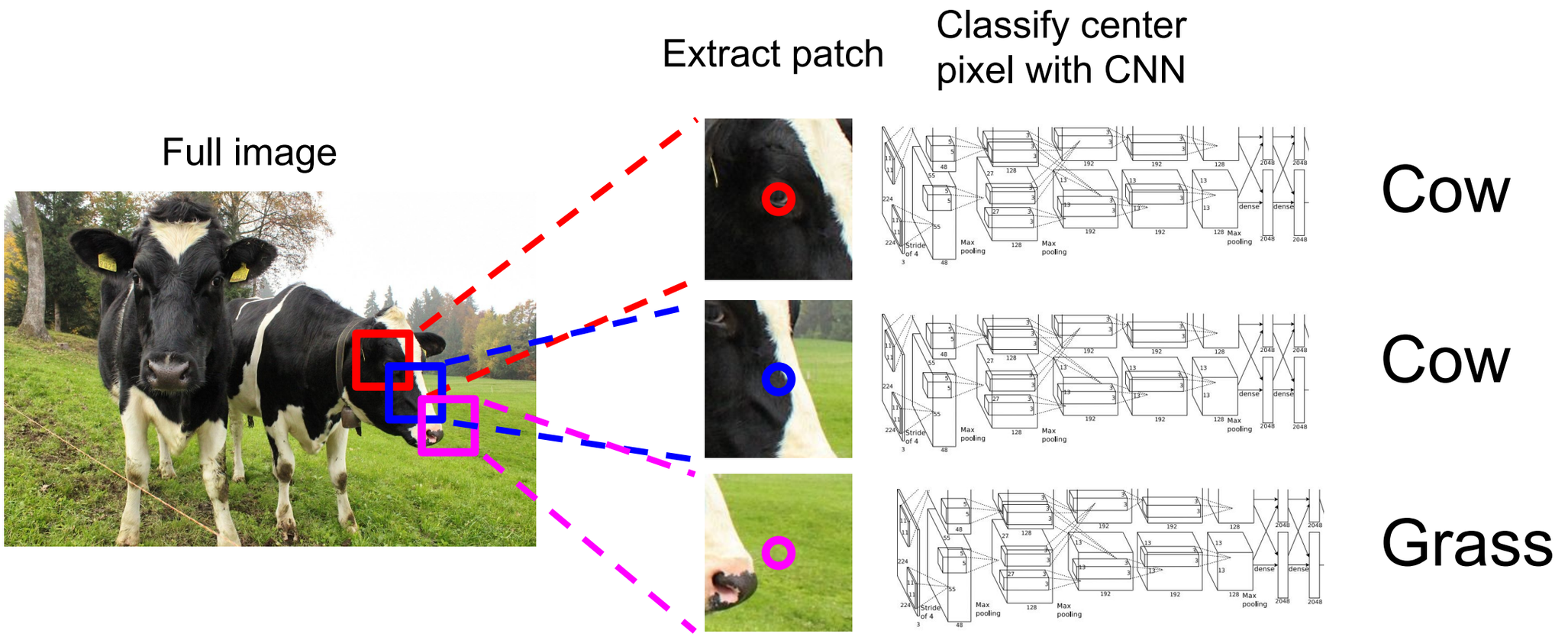
Don’t differentiate instances, only care about pixels





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

Semantic Segmentation Idea: Sliding Window

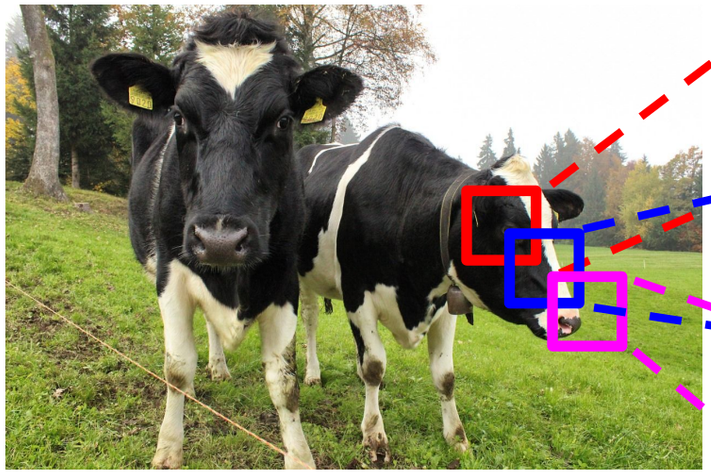


Farabet et al, “Learning Hierarchical Features for Scene Labeling,” TPAMI 2013
Pinheiro and Collobert, “Recurrent Convolutional Neural Networks for Scene Labeling”, ICML 2014

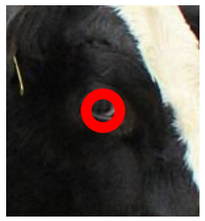
slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Semantic Segmentation Idea: Sliding Window

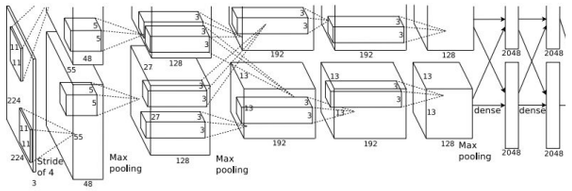
Full image



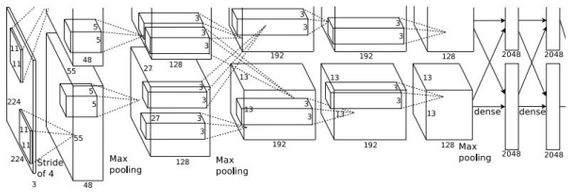
Extract patch



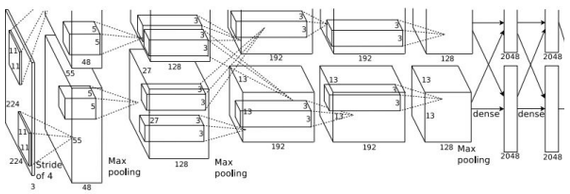
Classify center pixel with CNN



Cow



Cow



Grass

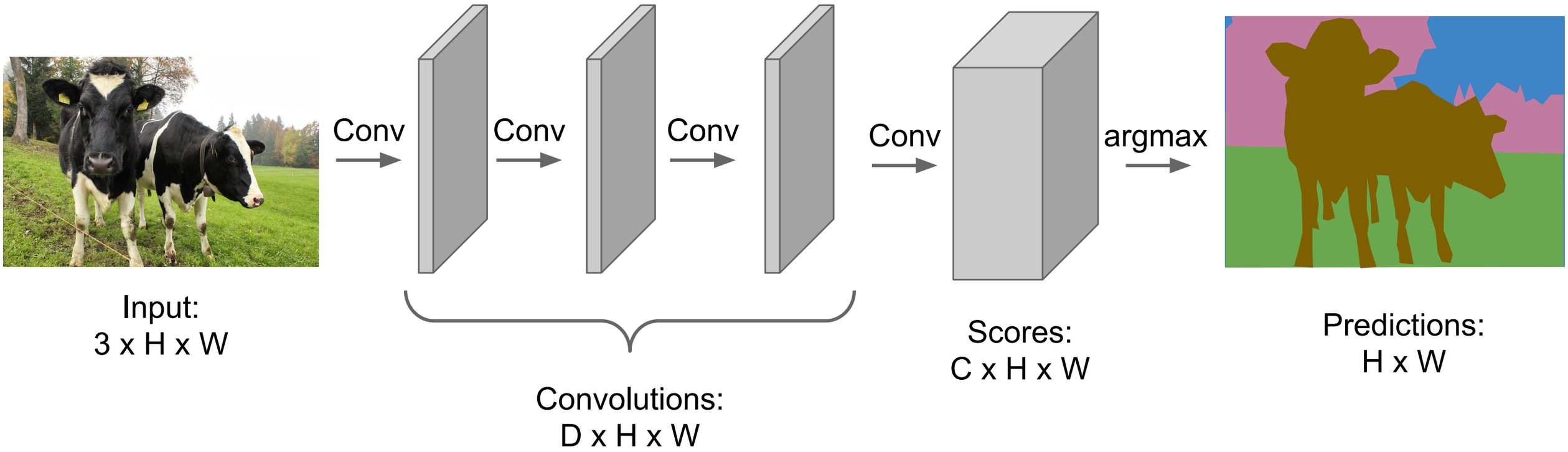
Problem: Very inefficient! Not reusing shared features between overlapping patches

Farabet et al, “Learning Hierarchical Features for Scene Labeling,” TPAMI 2013
Pinheiro and Collobert, “Recurrent Convolutional Neural Networks for Scene Labeling”, ICML 2014

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

Semantic Segmentation Idea: Fully Convolutional

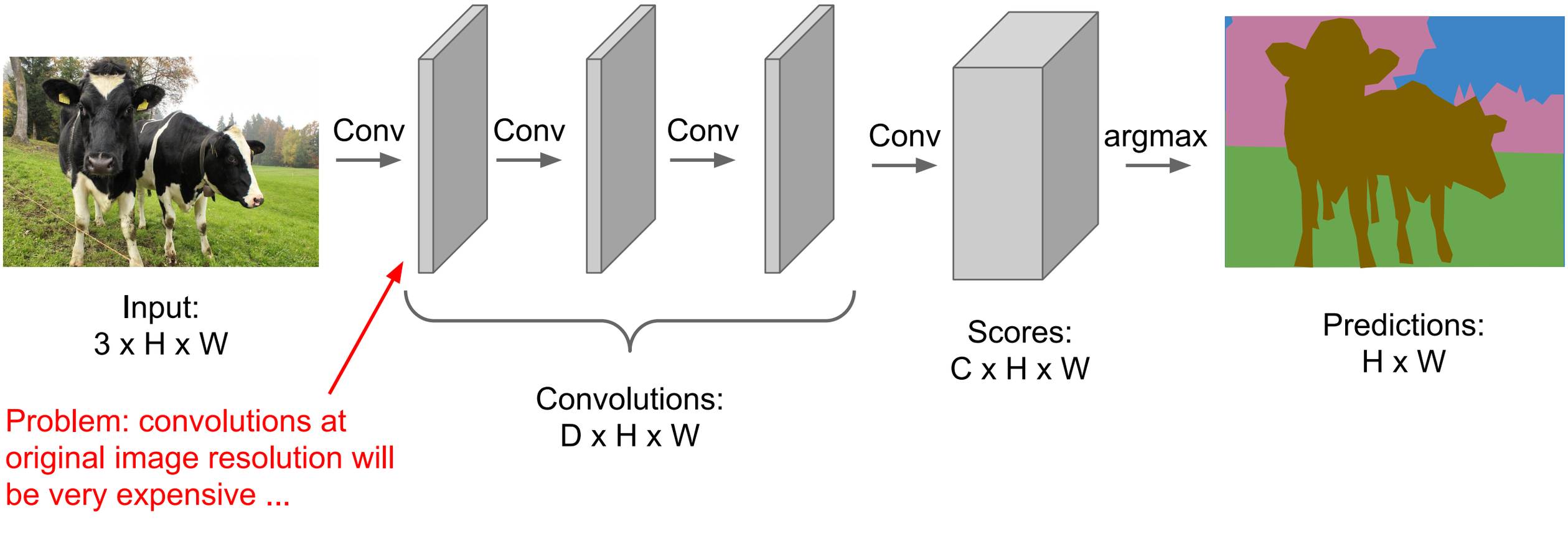
Design a network as a bunch of convolutional layers
to make predictions for pixels all at once!



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

Semantic Segmentation Idea: Fully Convolutional

Design a network as a bunch of convolutional layers
to make predictions for pixels all at once!



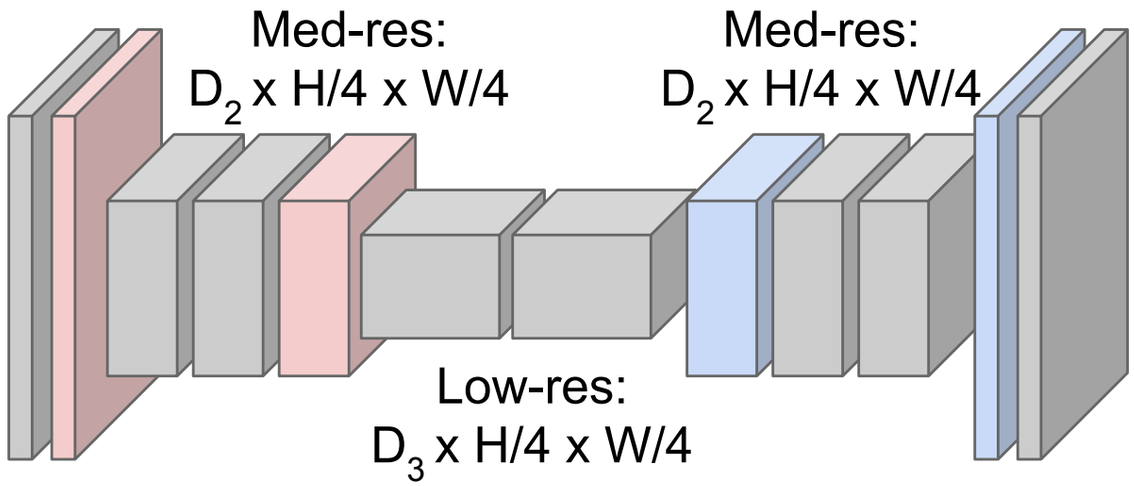
Problem: convolutions at original image resolution will be very expensive ...

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

Semantic Segmentation Idea: Fully Convolutional

Design network as a bunch of convolutional layers, with **downsampling** and **upsampling** inside the network!

Input:
3 x H x W



High-res:
D_1 x H/2 x W/2

High-res:
D_1 x H/2 x W/2

Predictions:
H x W

Long, Shelhamer, and Darrell, “Fully Convolutional Networks for Semantic Segmentation”, CVPR 2015
Noh et al, “Learning Deconvolution Network for Semantic Segmentation”, ICCV 2015

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

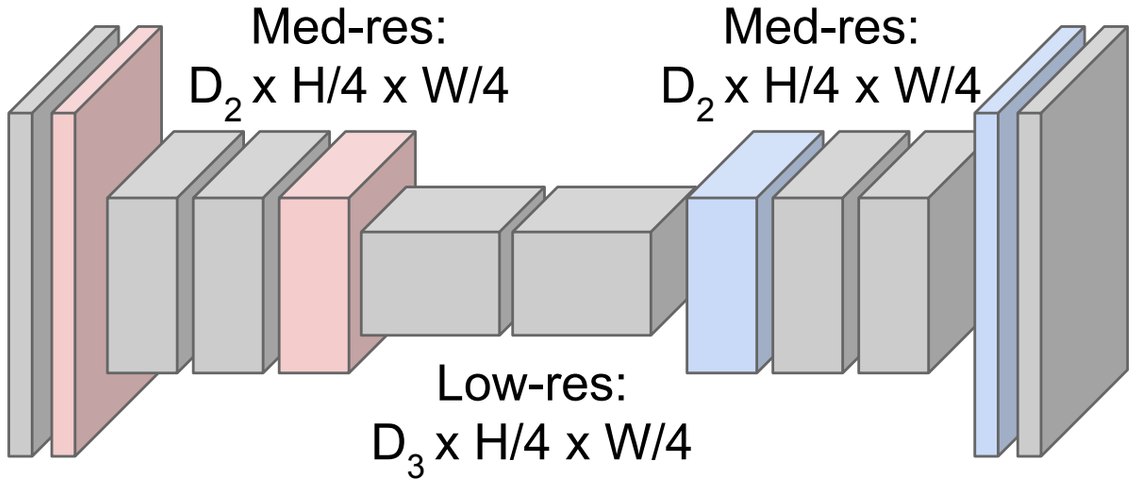
Semantic Segmentation Idea: Fully Convolutional

Downsampling:
Pooling, strided convolution

Design network as a bunch of convolutional layers, with **downsampling** and **upsampling** inside the network!

Upsampling:
???

Input:
3 x H x W



High-res:
D_1 x H/2 x W/2

High-res:
D_1 x H/2 x W/2

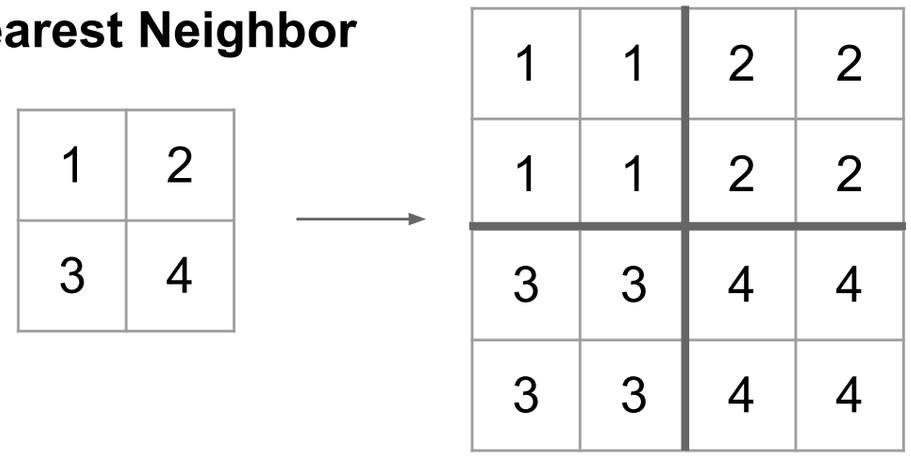
Predictions:
H x W

Long, Shelhamer, and Darrell, “Fully Convolutional Networks for Semantic Segmentation”, CVPR 2015
Noh et al, “Learning Deconvolution Network for Semantic Segmentation”, ICCV 2015

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

In-Network upsampling: “Unpooling”

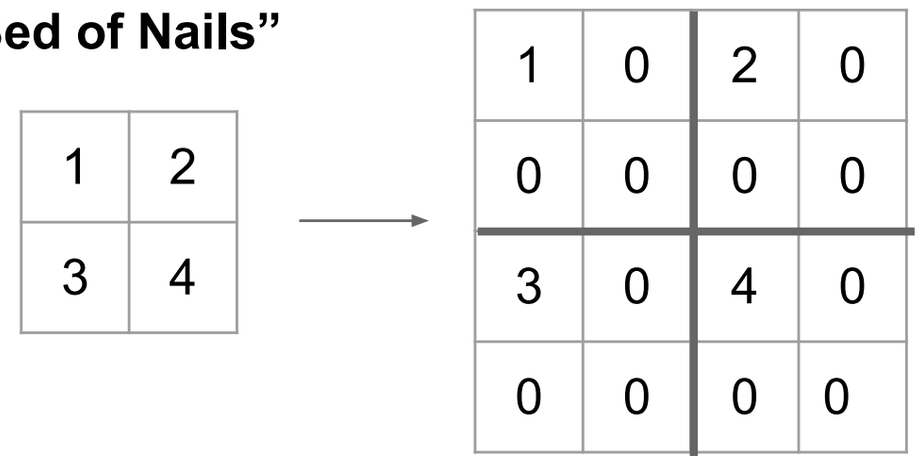
Nearest Neighbor



Input: 2 x 2

Output: 4 x 4

“Bed of Nails”



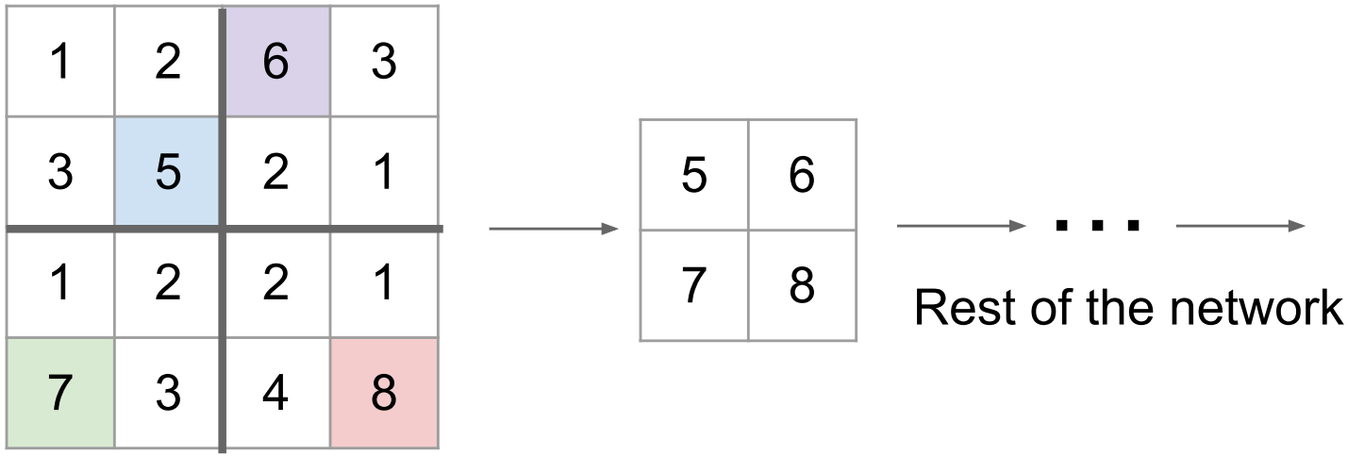
Input: 2 x 2

Output: 4 x 4

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

In-Network upsampling: “Max Unpooling”

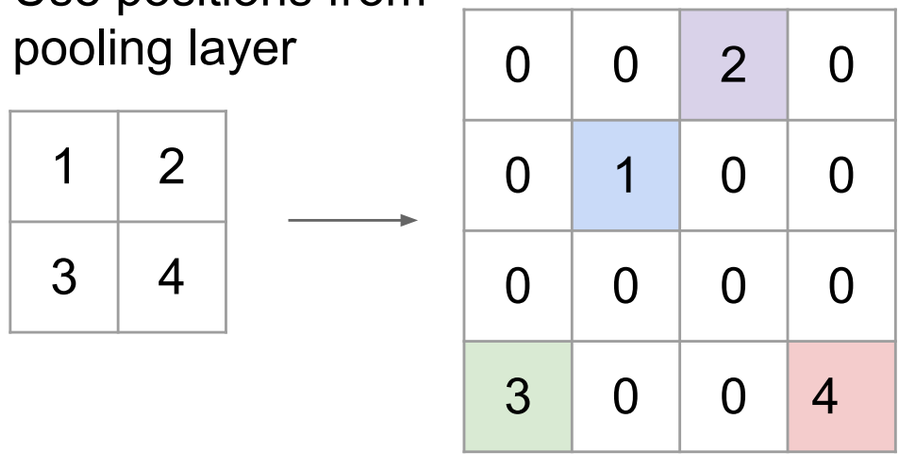
Max Pooling
Remember which element was max!



Input: 4 x 4

Output: 2 x 2

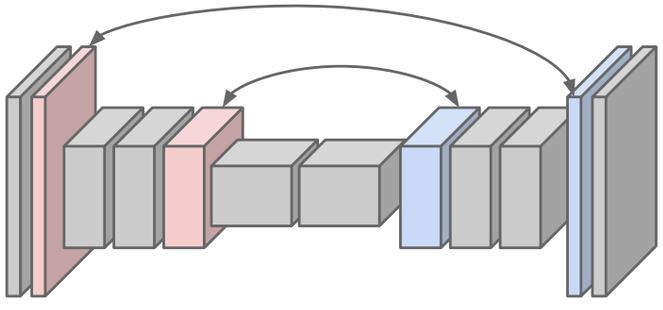
Max Unpooling
Use positions from pooling layer



Input: 2 x 2

Output: 4 x 4

Corresponding pairs of downsampling and upsampling layers



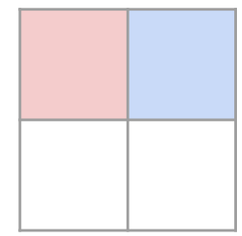
slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Convolution Transposed

Learnable Upsampling: Transpose Convolution

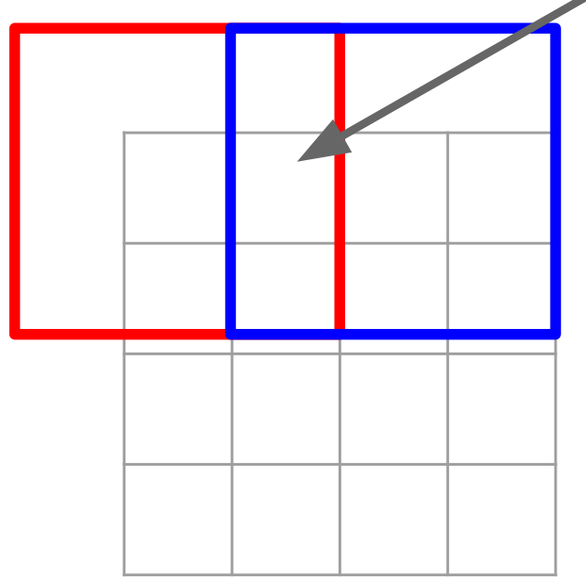
Other names:
-Deconvolution (bad)
-Upconvolution
-Fractionally strided convolution
-Backward strided convolution

3 x 3 **transpose** convolution, stride 2 pad 1



Input: 2 x 2

Input gives weight for filter



Output: 4 x 4

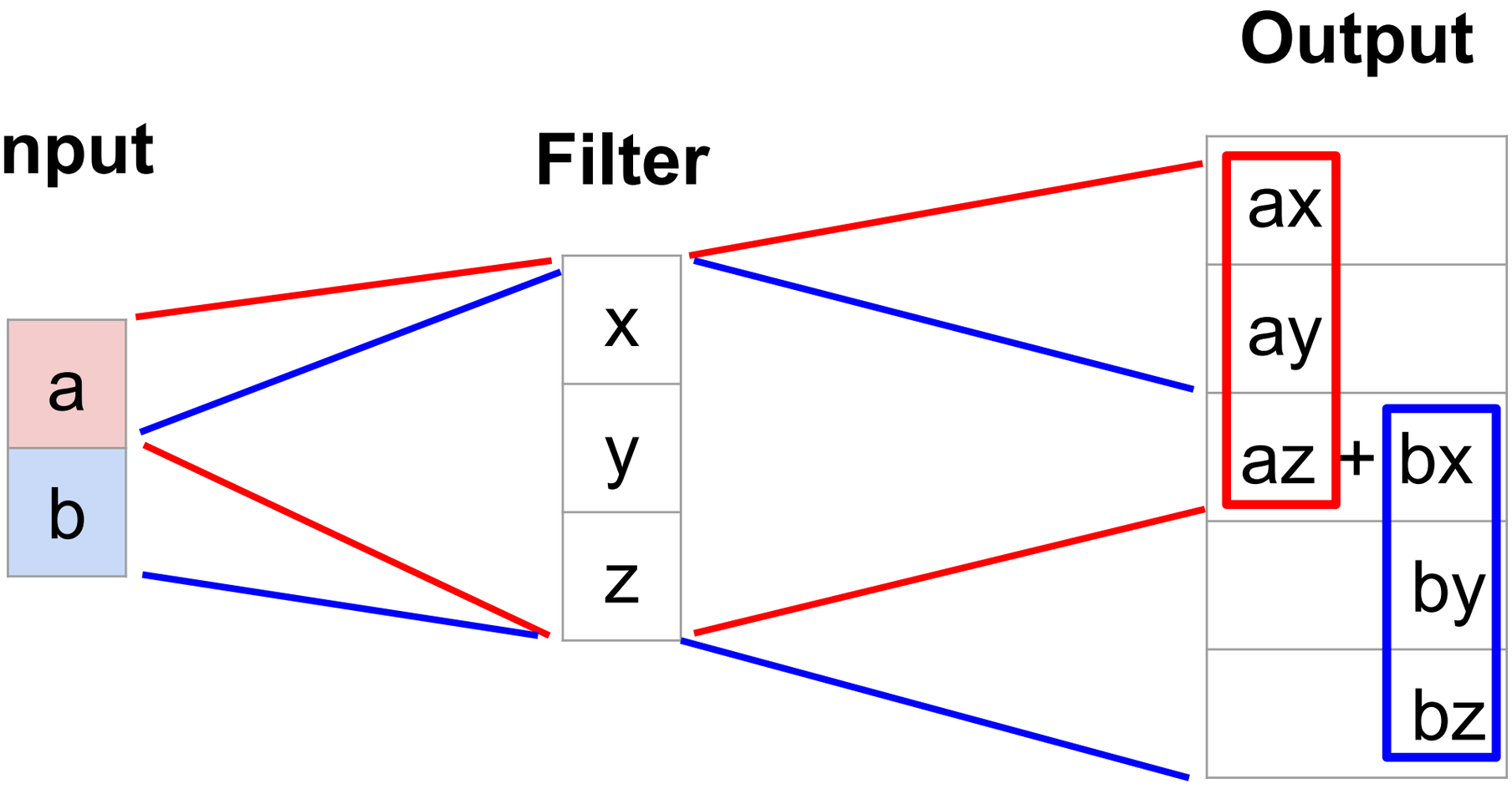
Sum where output overlaps

Filter moves 2 pixels in the output for every one pixel in the input

Stride gives ratio between movement in output and input

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

Learnable Upsampling: 1D Example



Output contains copies of the filter weighted by the input, summing at where at overlaps in the output

Need to crop one pixel from output to make output exactly 2x input

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

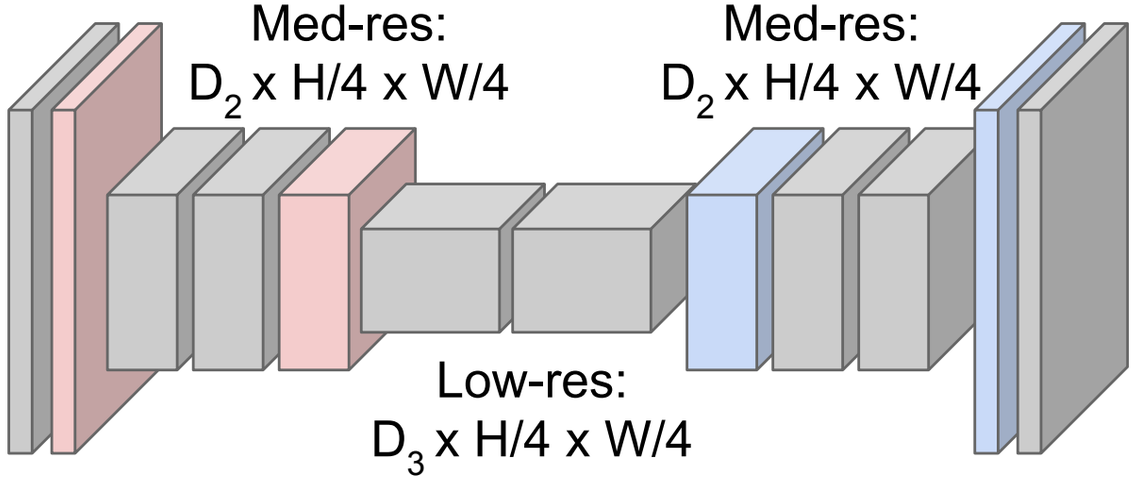
Semantic Segmentation Idea: Fully Convolutional

Downsampling:
Pooling, strided convolution

Design network as a bunch of convolutional layers, with **downsampling** and **upsampling** inside the network!

Upsampling:
Unpooling or strided transpose convolution

Input:
3 x H x W



High-res:
D_1 x H/2 x W/2

High-res:
D_1 x H/2 x W/2

Predictions:
H x W

Long, Shelhamer, and Darrell, “Fully Convolutional Networks for Semantic Segmentation”, CVPR 2015
Noh et al, “Learning Deconvolution Network for Semantic Segmentation”, ICCV 2015

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

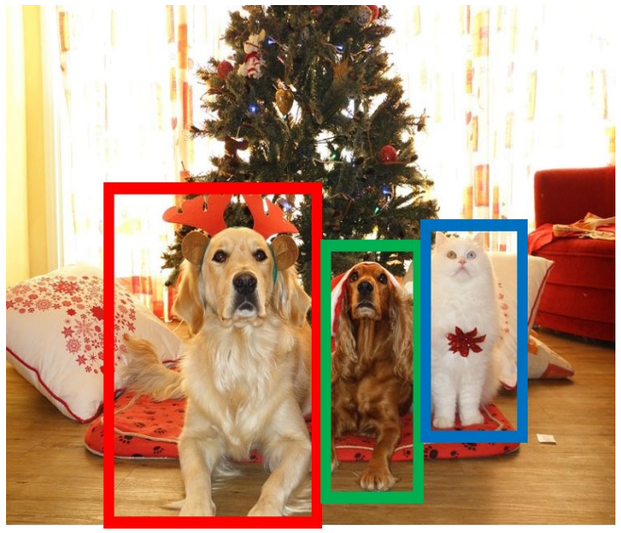
2D Object Detection

Semantic Segmentation

GRASS, CAT, TREE, SKY

No objects, just pixels

2D Object Detection



DOG, DOG, CAT

Object categories + 2D bounding boxes

3D Object Detection

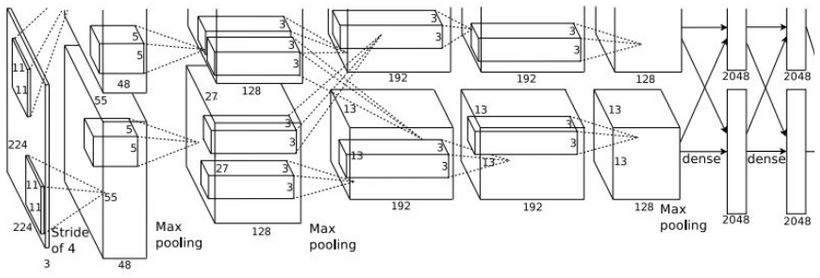
Car

Object categories + 3D bounding boxes

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

Classification + Localization

This image is CC0 public domain



Fully Connected: 4096 to 1000

Class Scores
Cat: 0.9
Dog: 0.05
Car: 0.01
...

Vector: 4096

Fully Connected: 4096 to 4

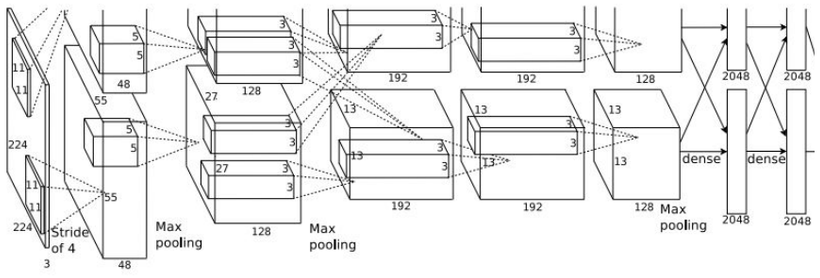
Box Coordinates
(x, y, w, h)

Treat localization as a regression problem!

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

Classification + Localization

This image is CC0 public domain



Correct label:
Cat

Fully Connected:
4096 to 1000

Class Scores
Cat: 0.9
Dog: 0.05
Car: 0.01
...

Softmax Loss

Vector:
4096

Fully Connected:
4096 to 4

Box Coordinates
(x, y, w, h)

L2 Loss

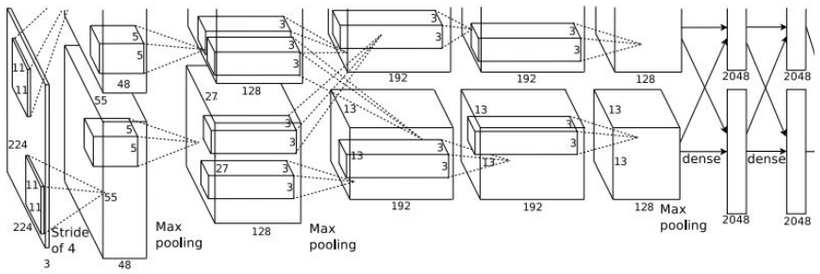
Correct box:
(x', y', w', h')

Treat localization as a regression problem!

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

Classification + Localization

This image is CC0 public domain



Fully Connected: 4096 to 1000

Class Scores
Cat: 0.9
Dog: 0.05
Car: 0.01
...

Softmax Loss

Correct label:
Cat

Multitask Loss

Loss

Vector: 4096

Fully Connected: 4096 to 4

Box Coordinates
(x, y, w, h)

L2 Loss

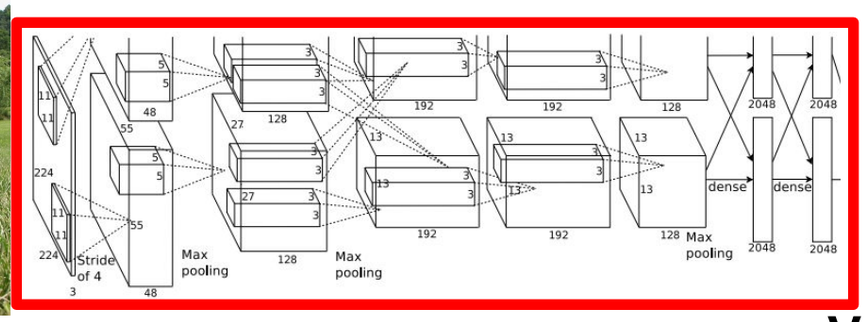
Correct box:
(x', y', w', h')

Treat localization as a regression problem!

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

Classification + Localization

This image is CC0 public domain



Often pretrained on ImageNet
(Transfer learning)

Vector: 4096

Fully Connected: 4096 to 4

Box Coordinates (x, y, w, h)

Fully Connected: 4096 to 1000

Class Scores
Cat: 0.9
Dog: 0.05
Car: 0.01
...

Softmax Loss

Correct label: Cat

L2 Loss

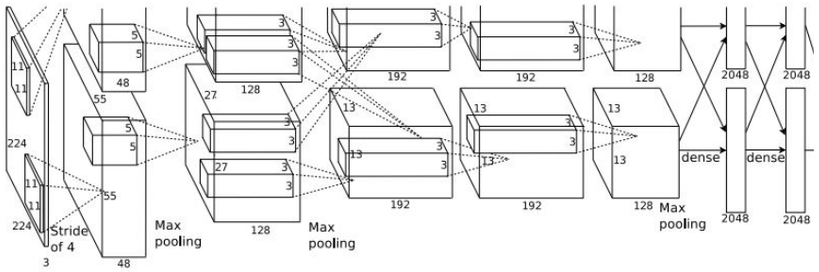
Correct box: (x', y', w', h')

+ → **Loss**

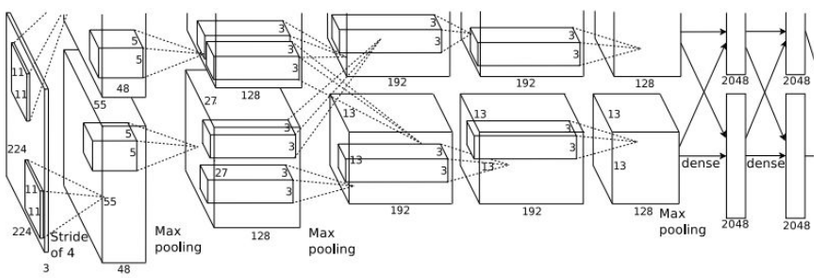
Treat localization as a regression problem!

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

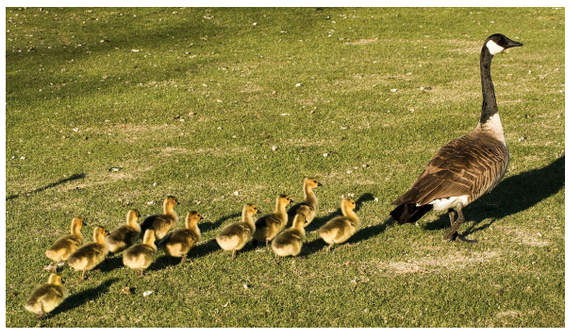
Object Detection as Regression?

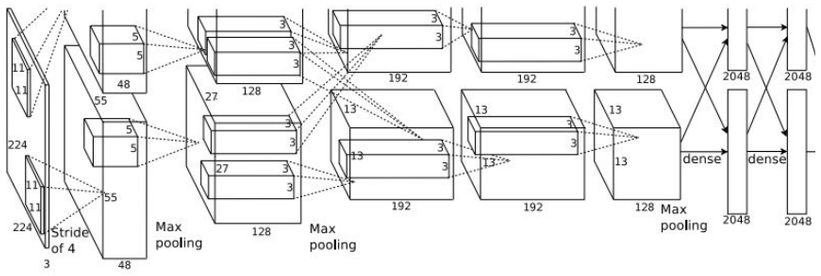


CAT: (x, y, w, h)



DOG: (x, y, w, h)
DOG: (x, y, w, h)
CAT: (x, y, w, h)



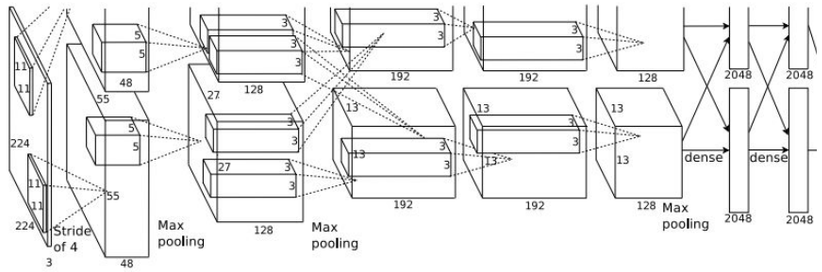


DUCK: (x, y, w, h)
DUCK: (x, y, w, h)
….

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

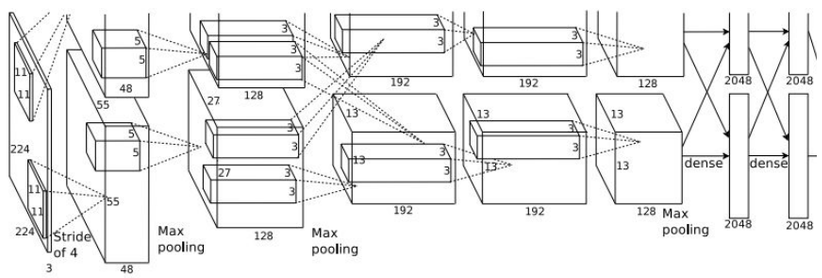
Object Detection as Regression?

Each image needs a different number of outputs!



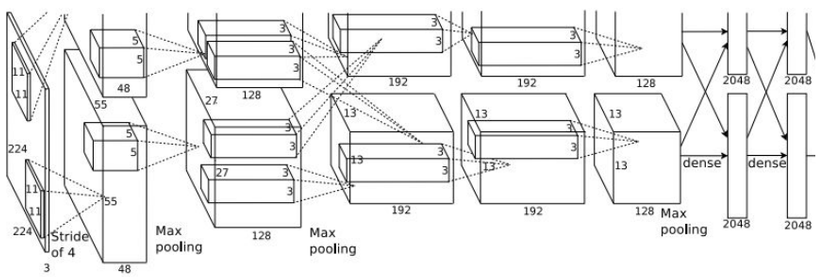
CAT: (x, y, w, h)

4 numbers



DOG: (x, y, w, h)
DOG: (x, y, w, h)
CAT: (x, y, w, h)

16 numbers



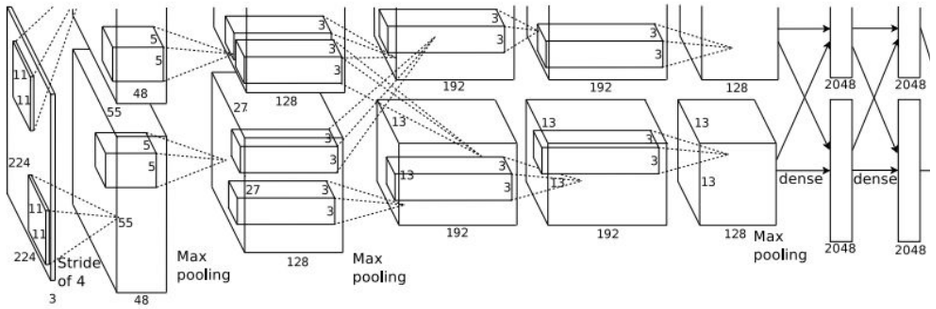
DUCK: (x, y, w, h)
DUCK: (x, y, w, h)
....

Many numbers!

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

Object Detection as Classification: Sliding Window

Apply a CNN to many different crops of the image, CNN classifies each crop as object or background

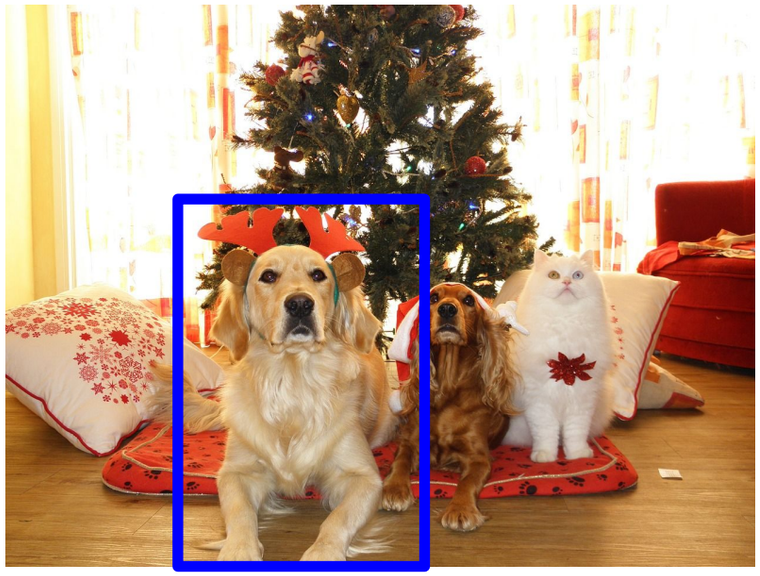


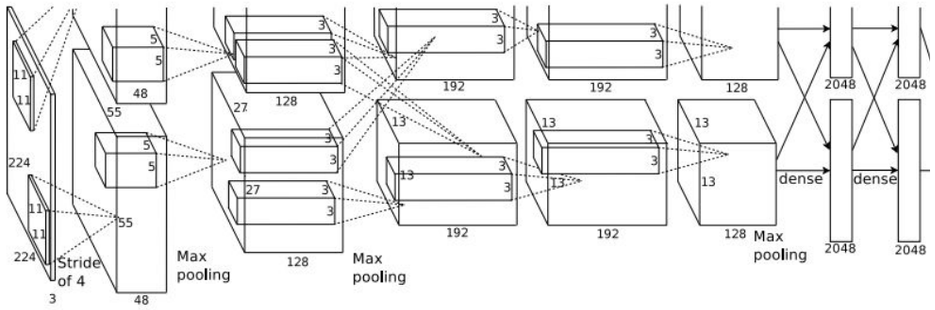
Dog? NO
Cat? NO
Background? YES

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

Object Detection as Classification: Sliding Window

Apply a CNN to many different crops of the image, CNN classifies each crop as object or background



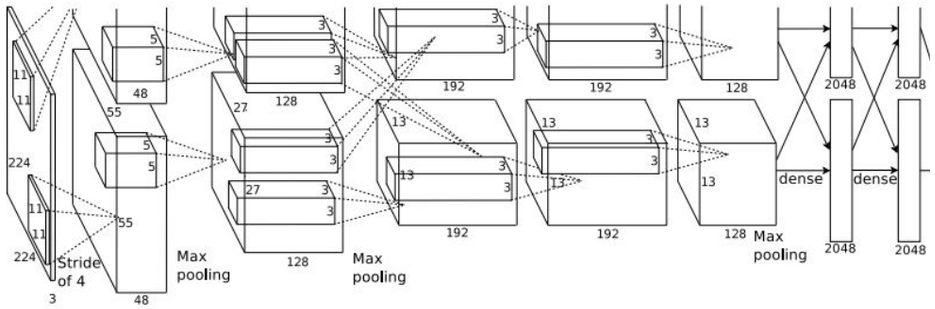


Dog? YES
Cat? NO
Background? NO

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

Object Detection as Classification: Sliding Window

Apply a CNN to many different crops of the image, CNN classifies each crop as object or background

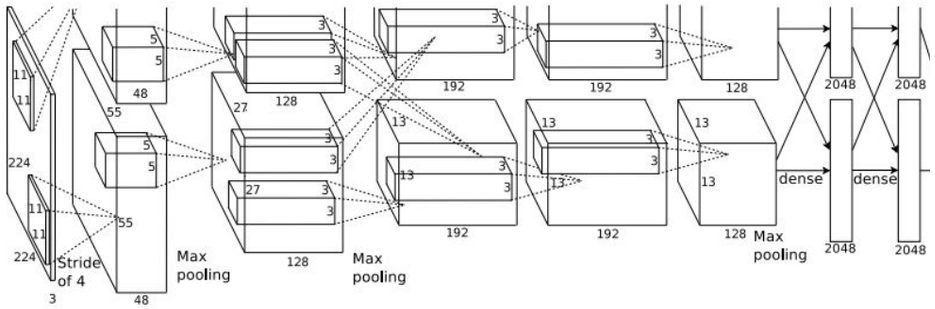


Dog? YES
Cat? NO
Background? NO

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

Object Detection as Classification: Sliding Window

Apply a CNN to many different crops of the image, CNN classifies each crop as object or background

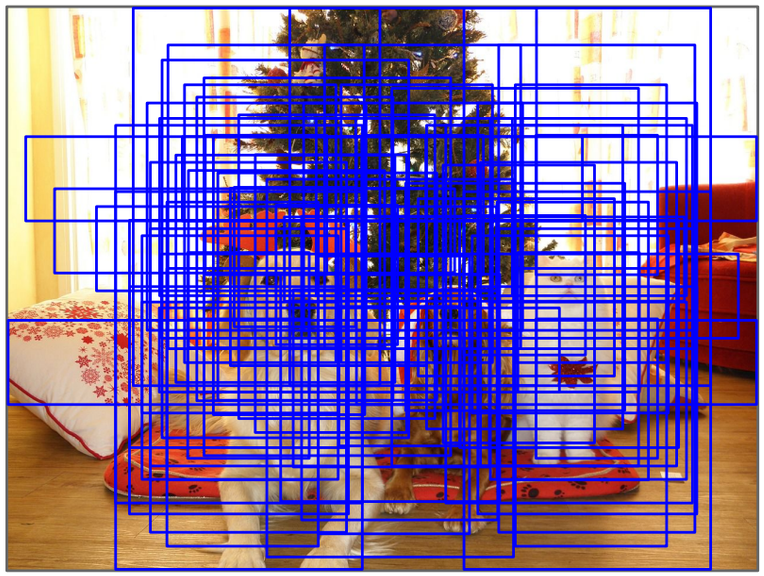


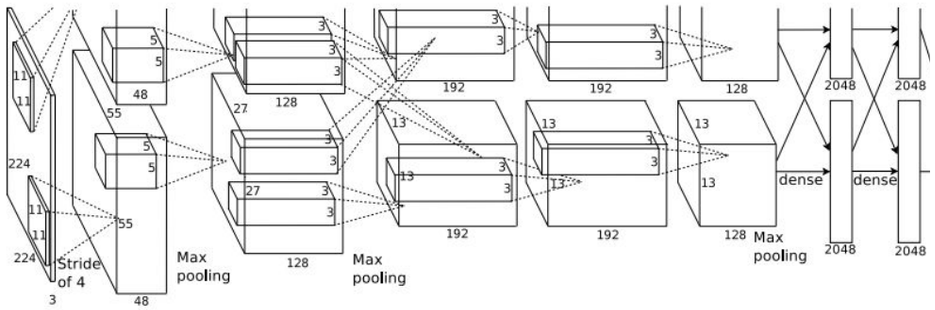
Dog? NO
Cat? YES
Background? NO

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

Object Detection as Classification: Sliding Window

Apply a CNN to many different crops of the image, CNN classifies each crop as object or background





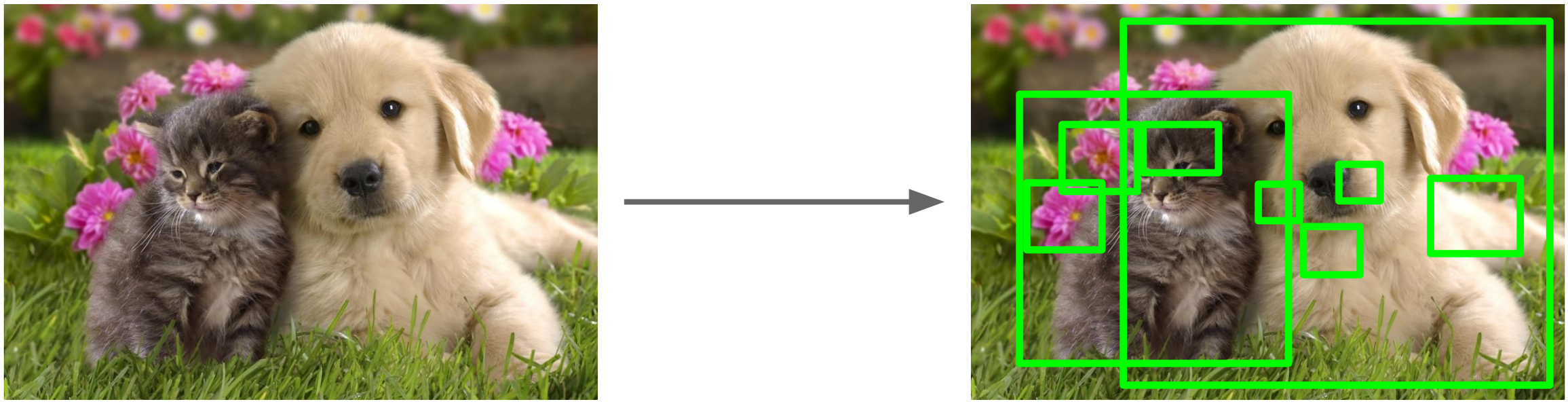
Dog? NO
Cat? YES
Background? NO

Problem: Need to apply CNN to huge number of locations, scales, and aspect ratios, very computationally expensive!

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

Region Proposals / Selective Search

- Find “blobby” image regions that are likely to contain objects
- Relatively fast to run; e.g. Selective Search gives 2000 region proposals in a few seconds on CPU



Alexe et al, “Measuring the objectness of image windows”, TPAMI 2012
Uijlings et al, “Selective Search for Object Recognition”, IJCV 2013
Cheng et al, “BING: Binarized normed gradients for objectness estimation at 300fps”, CVPR 2014
Zitnick and Dollar, “Edge boxes: Locating object proposals from edges”, ECCV 2014

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

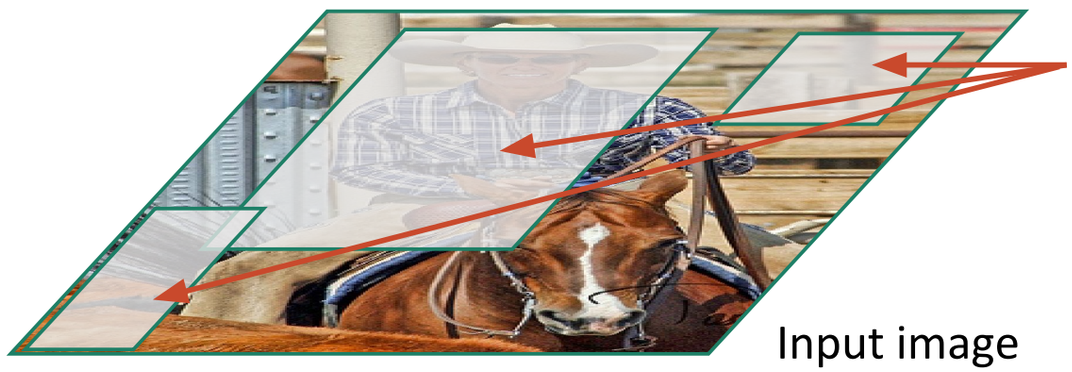
R-CNN

Input image

Girshick et al, “Rich feature hierarchies for accurate object detection and semantic segmentation”, CVPR 2014.
Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

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

R-CNN



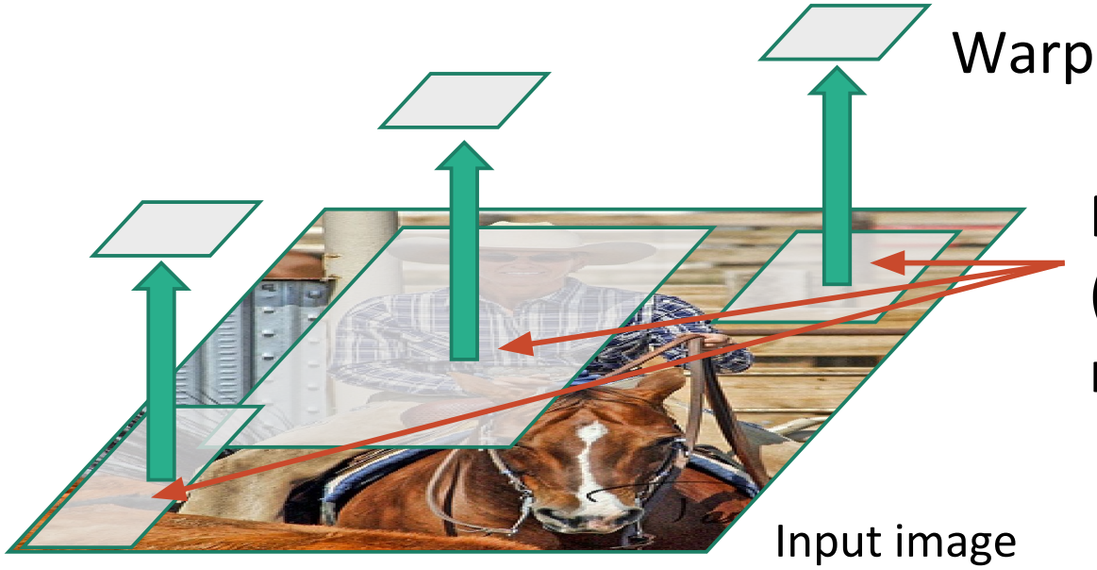
Input image

Regions of Interest (RoI) from a proposal method (~2k)

Girshick et al, “Rich feature hierarchies for accurate object detection and semantic segmentation”, CVPR 2014.
Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

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

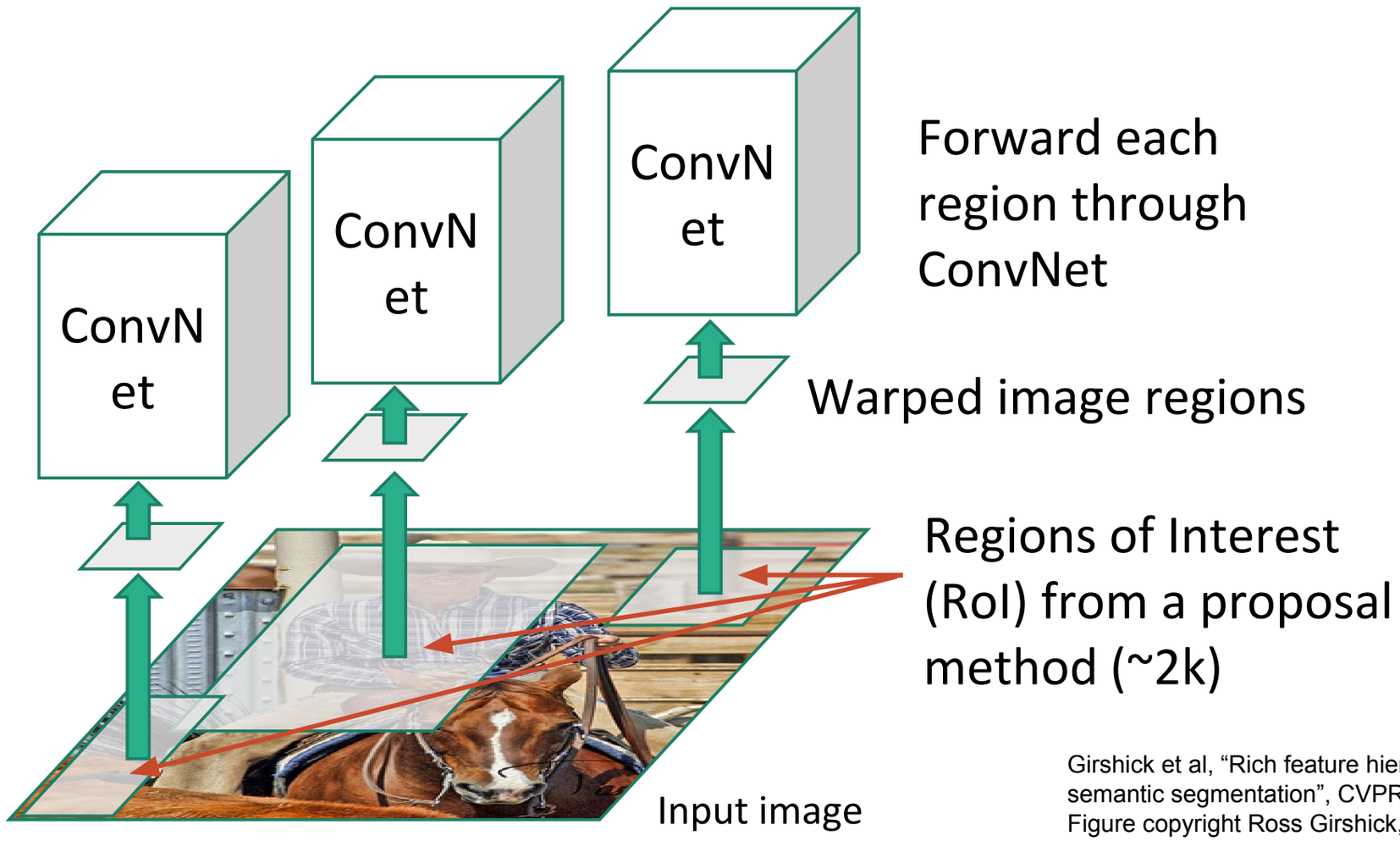
R-CNN



Girshick et al, “Rich feature hierarchies for accurate object detection and semantic segmentation”, CVPR 2014.
Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

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

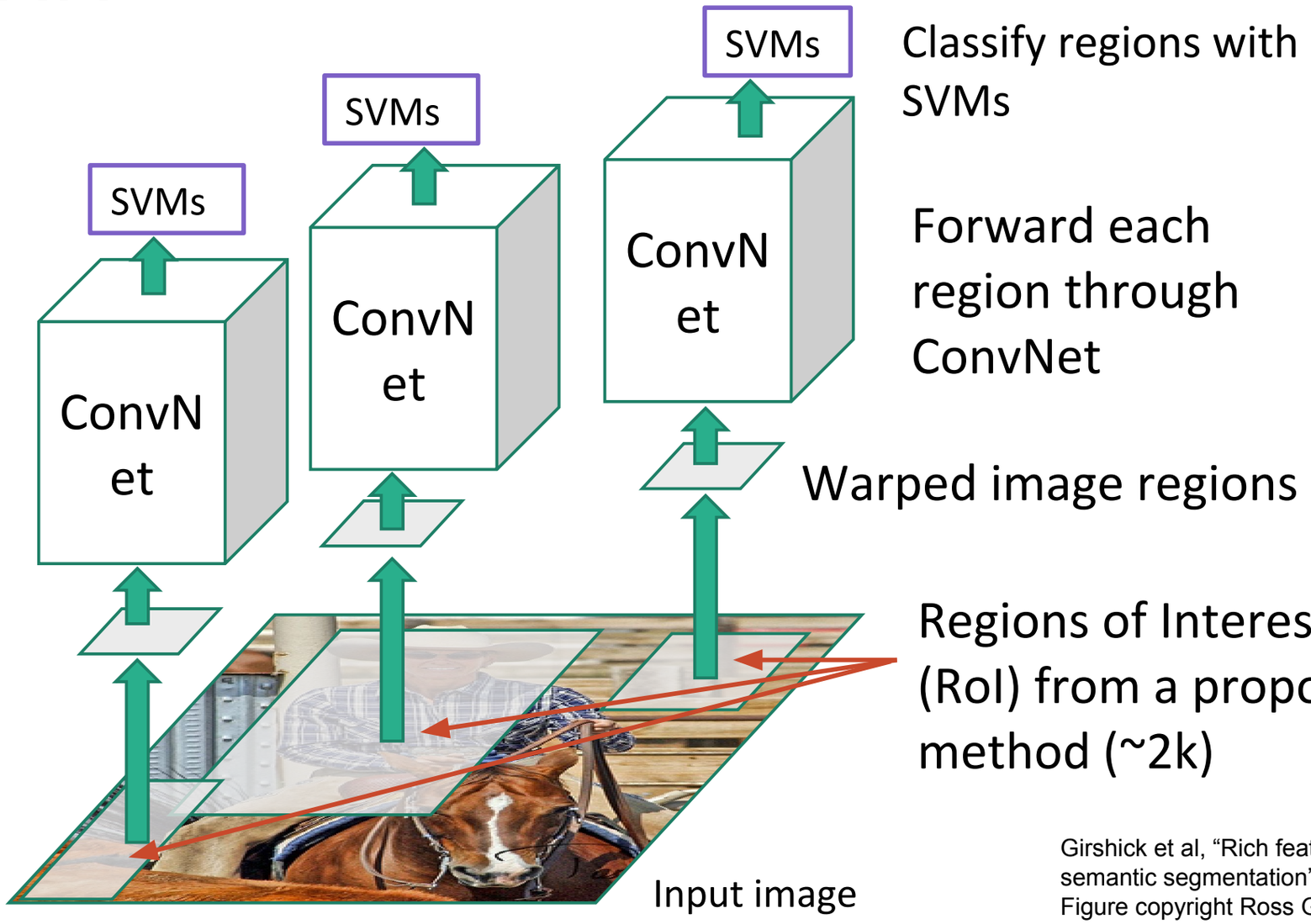
R-CNN



Girshick et al, “Rich feature hierarchies for accurate object detection and semantic segmentation”, CVPR 2014.
Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

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

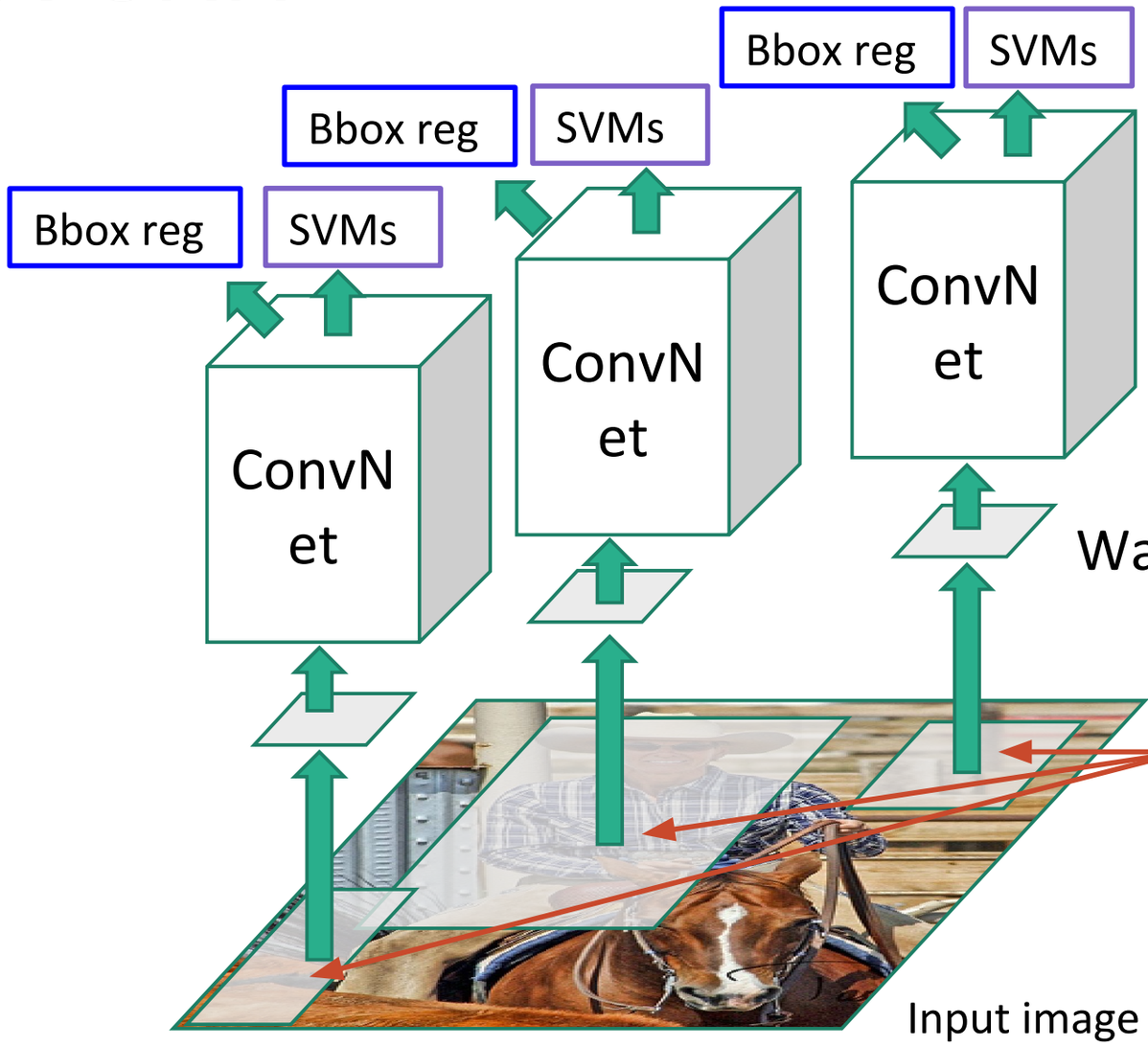
R-CNN



Girshick et al, “Rich feature hierarchies for accurate object detection and semantic segmentation”, CVPR 2014.
Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

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

R-CNN



Linear Regression for bounding box offsets

Classify regions with SVMs

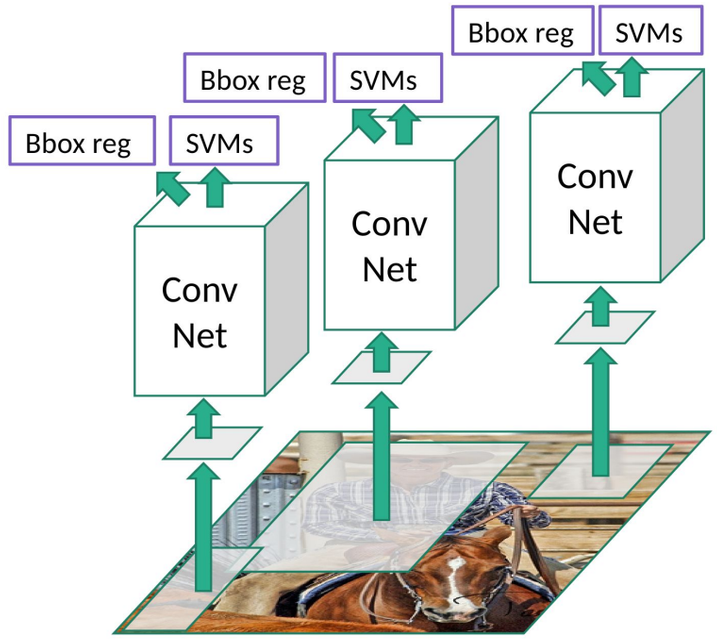
Forward each region through ConvNet

Girshick et al, “Rich feature hierarchies for accurate object detection and semantic segmentation”, CVPR 2014.
Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

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

R-CNN: Problems

- Ad hoc training objectives
  - Fine-tune network with softmax classifier (log loss)
  - Train post-hoc linear SVMs (hinge loss)
  - Train post-hoc bounding-box regressions (least squares)
- Training is slow (84h), takes a lot of disk space
- Inference (detection) is slow
  - 47s / image with VGG16 [Simonyan & Zisserman. ICLR15]
  - Fixed by SPP-net [He et al. ECCV14]



Girshick et al, “Rich feature hierarchies for accurate object detection and semantic segmentation”, CVPR 2014.
Slide copyright Ross Girshick, 2015; source. Reproduced with permission.

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

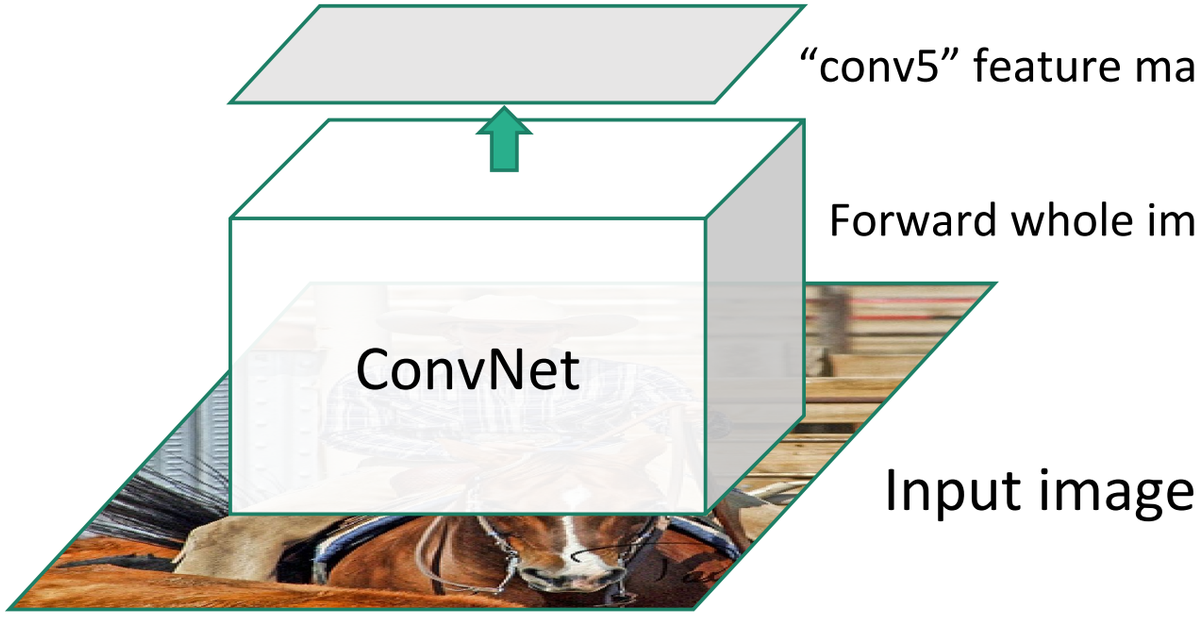
Fast R-CNN

Input image

Girshick, “Fast R-CNN”, ICCV 2015.
Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

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

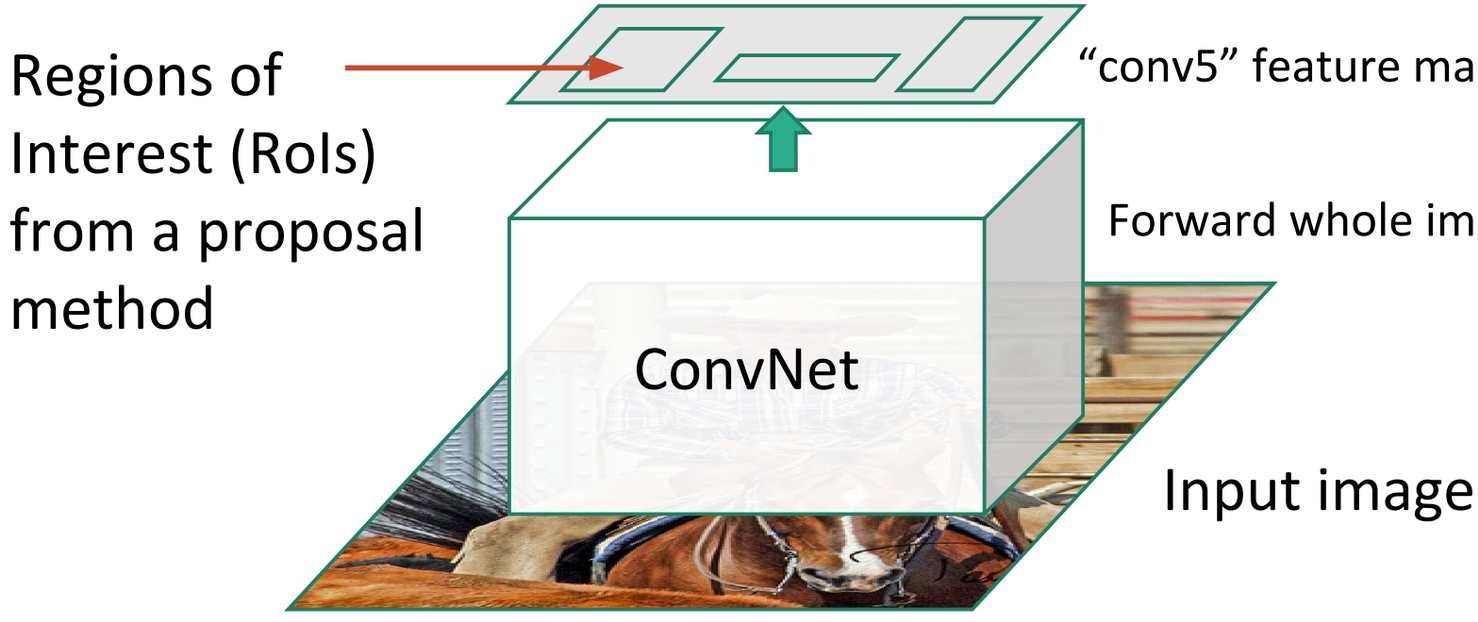
Fast R-CNN



Girshick, “Fast R-CNN”, ICCV 2015.
Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

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

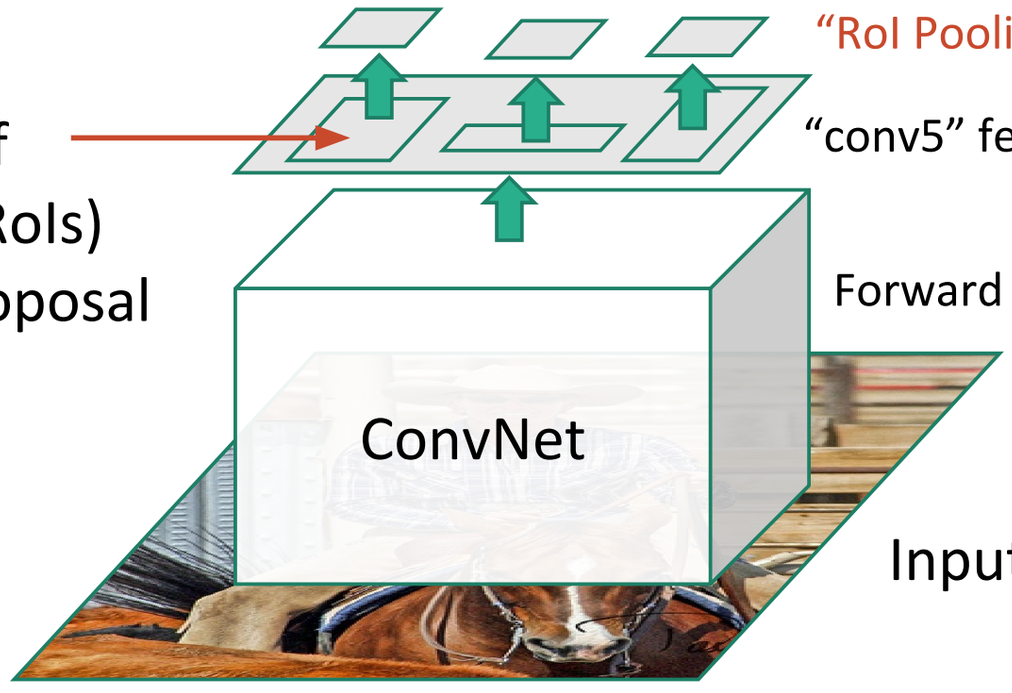
Fast R-CNN



Girshick, “Fast R-CNN”, ICCV 2015.
Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

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

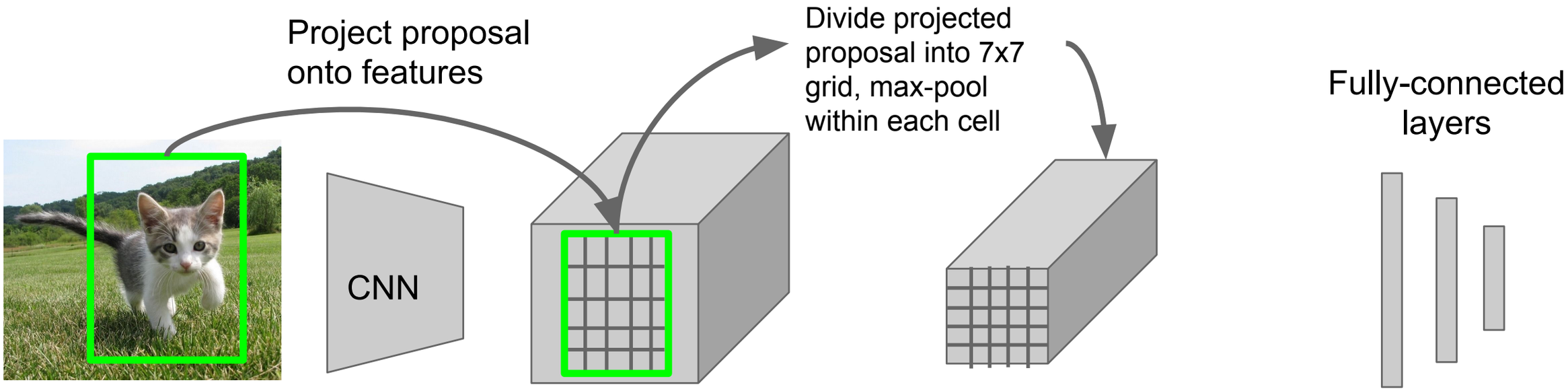
Fast R-CNN



Girshick, “Fast R-CNN”, ICCV 2015.
Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

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

Fast R-CNN: RoI Pooling



Hi-res input image:
3 x 640 x 480
with region proposal

Hi-res conv features:
512 x 20 x 15;

Projected region proposal is e.g.
512 x 18 x 8
(varies per proposal)

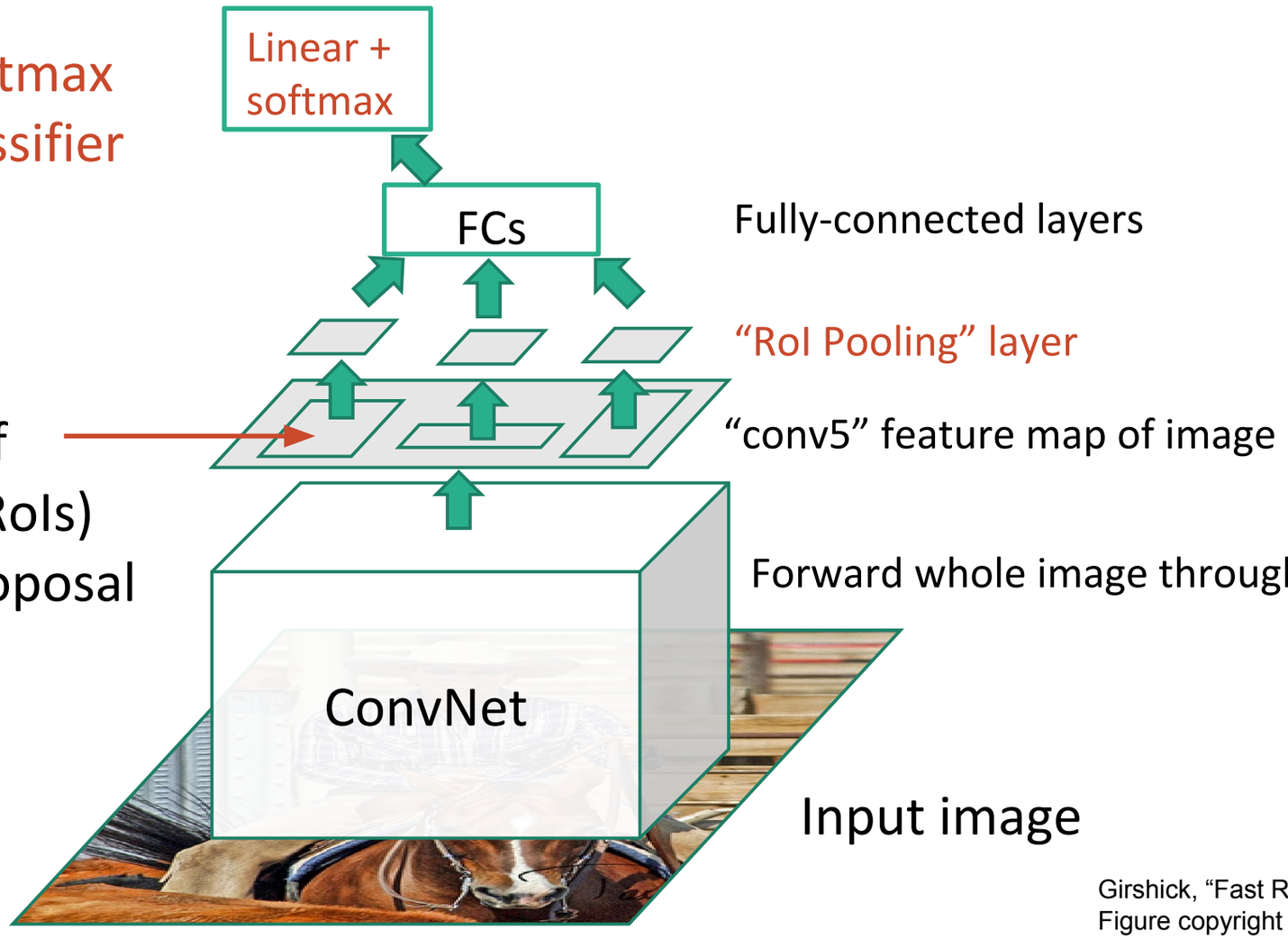
RoI conv features:
512 x 7 x 7
for region proposal

Fully-connected layers expect low-res conv features:
512 x 7 x 7

Girshick, “Fast R-CNN”, ICCV 2015.

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

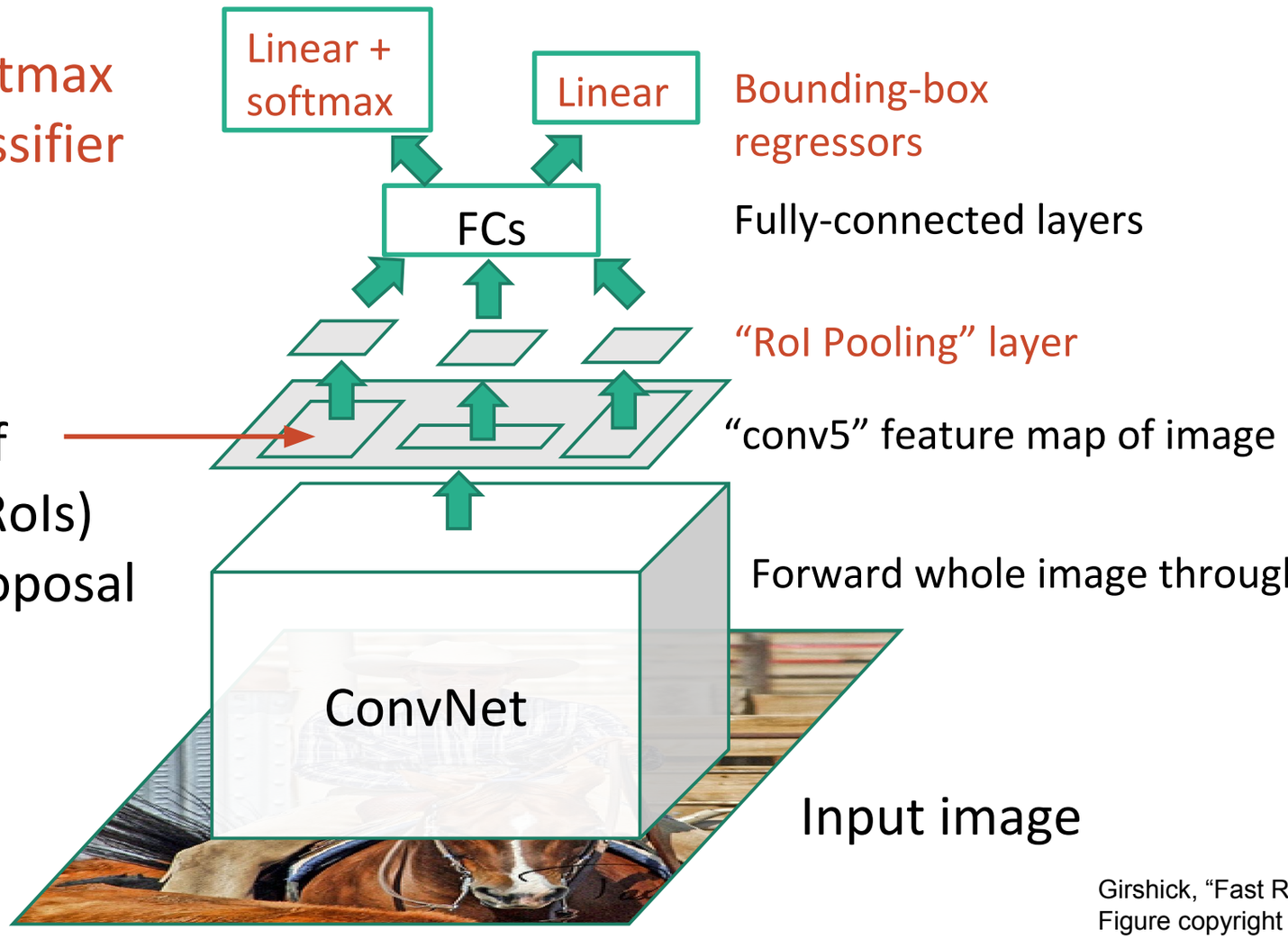
Fast R-CNN



Girshick, "Fast R-CNN", ICCV 2015.
Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

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

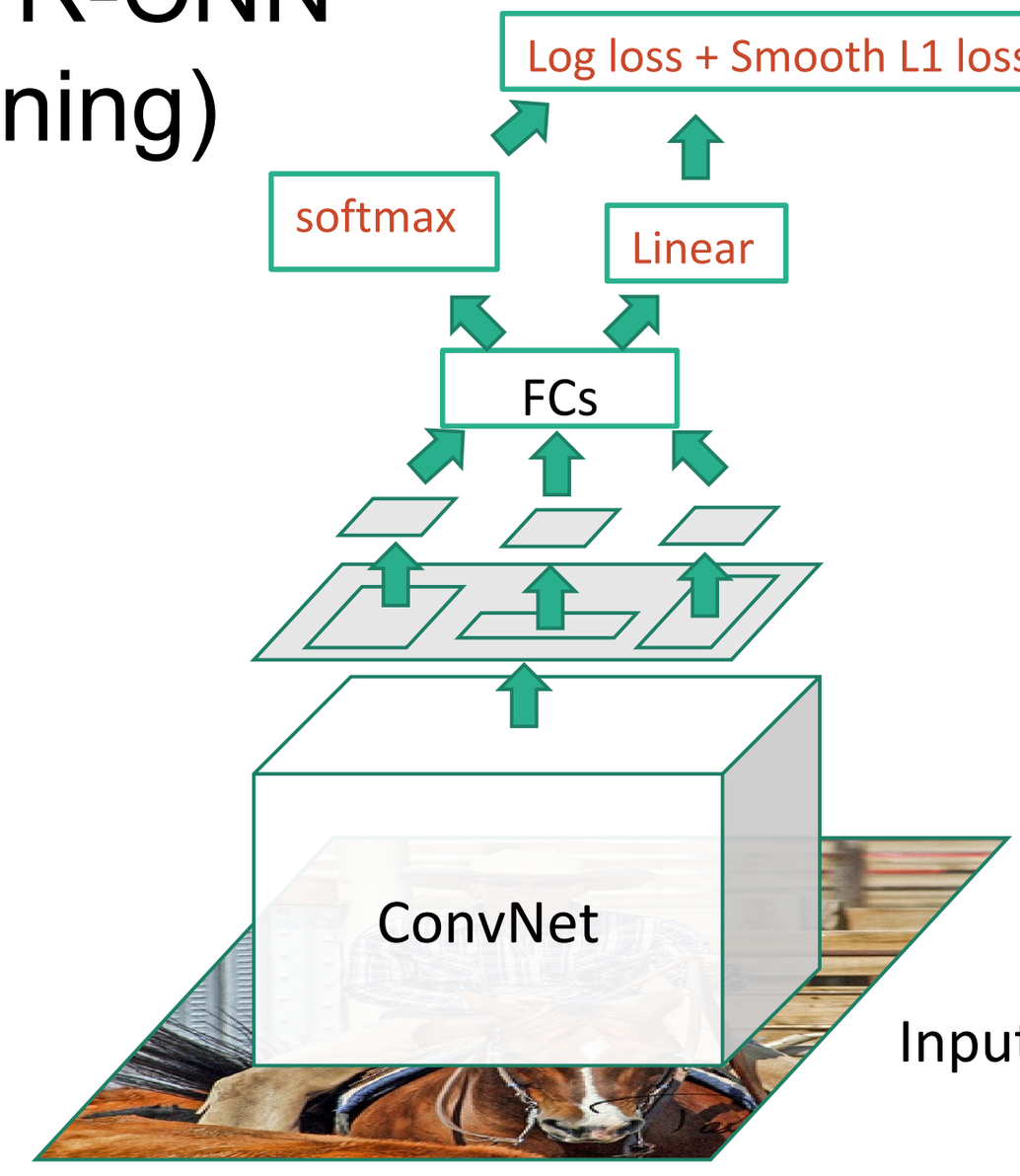
Fast R-CNN



Girshick, "Fast R-CNN", ICCV 2015.
Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

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

Fast R-CNN (Training)

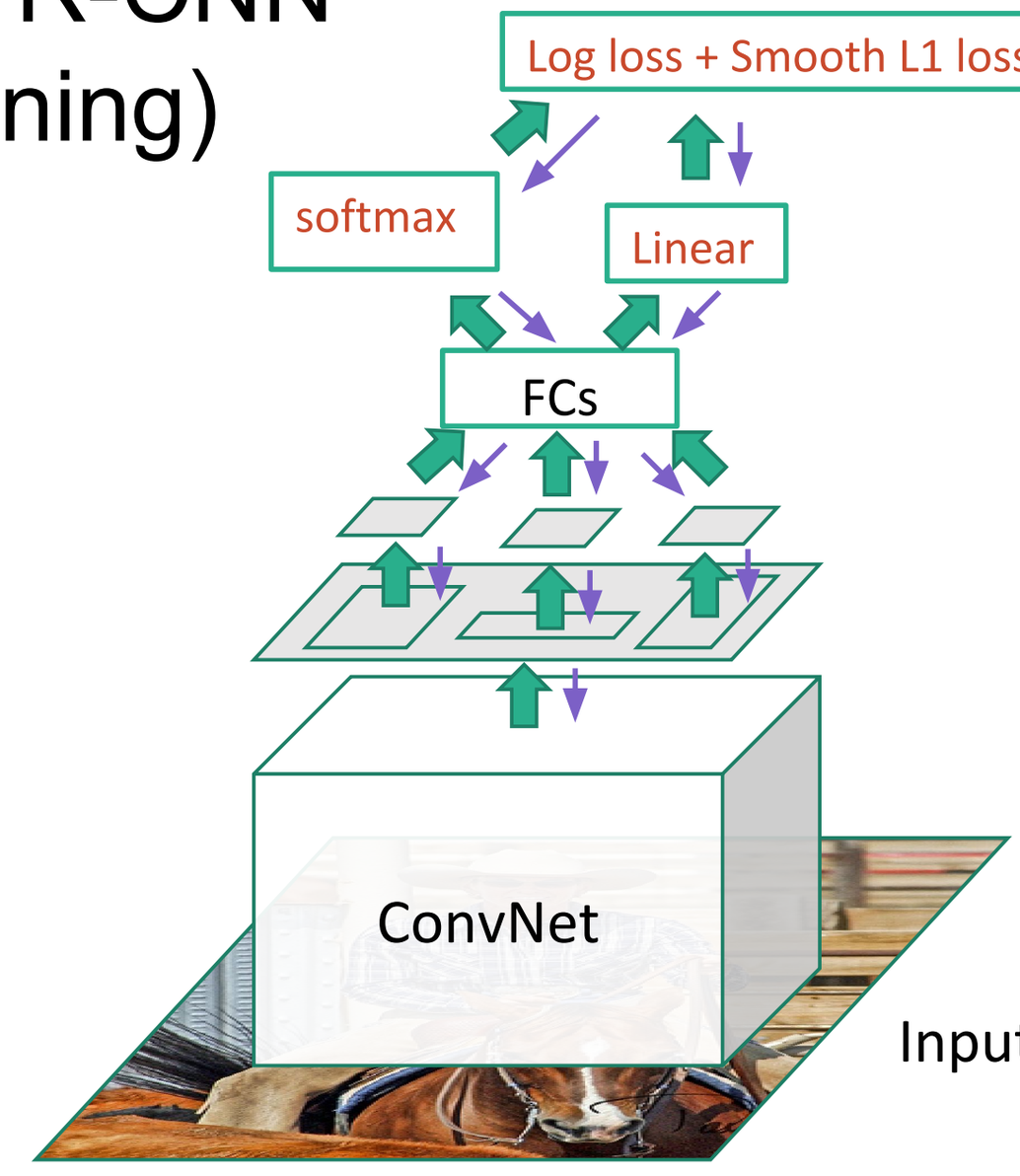


Multi-task loss

Girshick, “Fast R-CNN”, ICCV 2015.
Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

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

Fast R-CNN (Training)

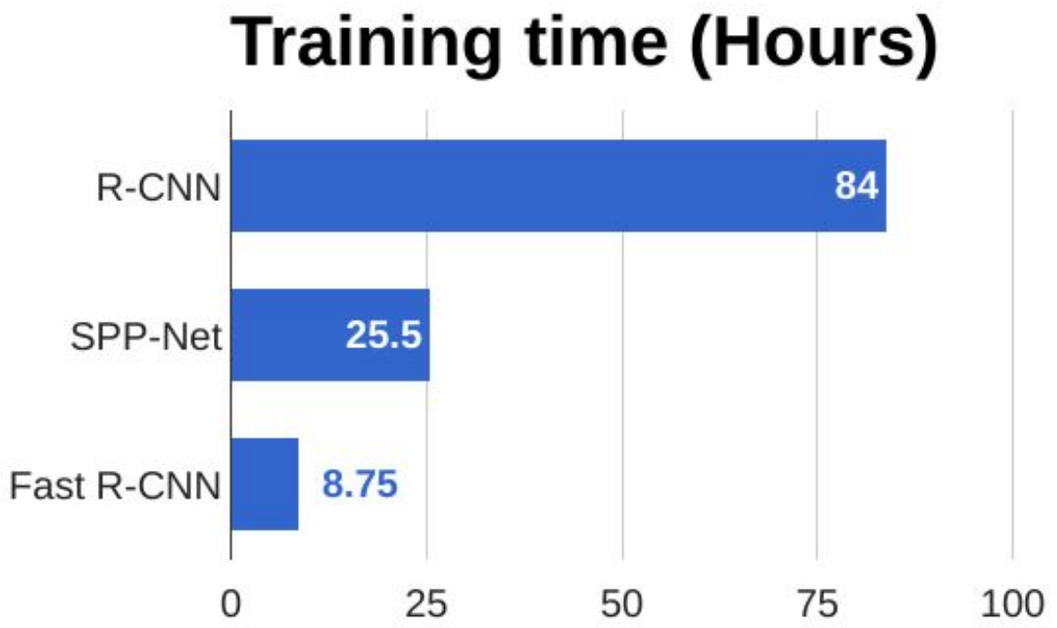


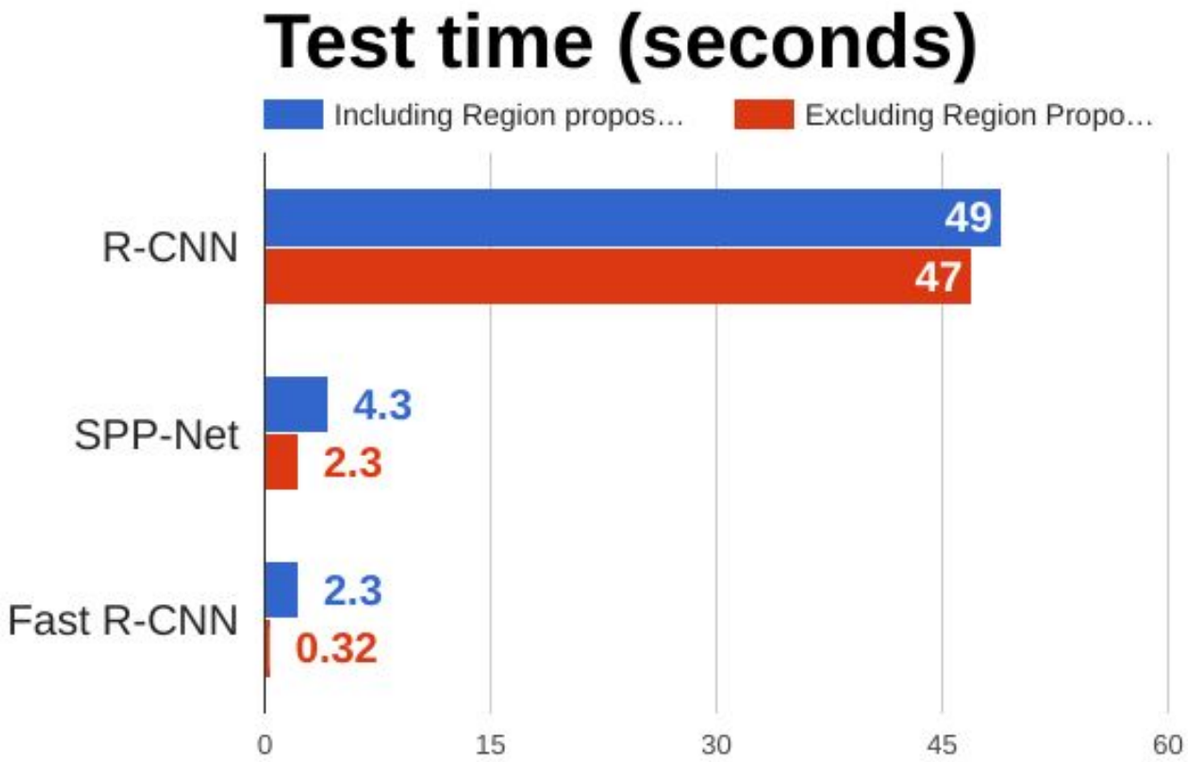
Multi-task loss

Girshick, “Fast R-CNN”, ICCV 2015.
Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

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

R-CNN vs SPP vs Fast R-CNN

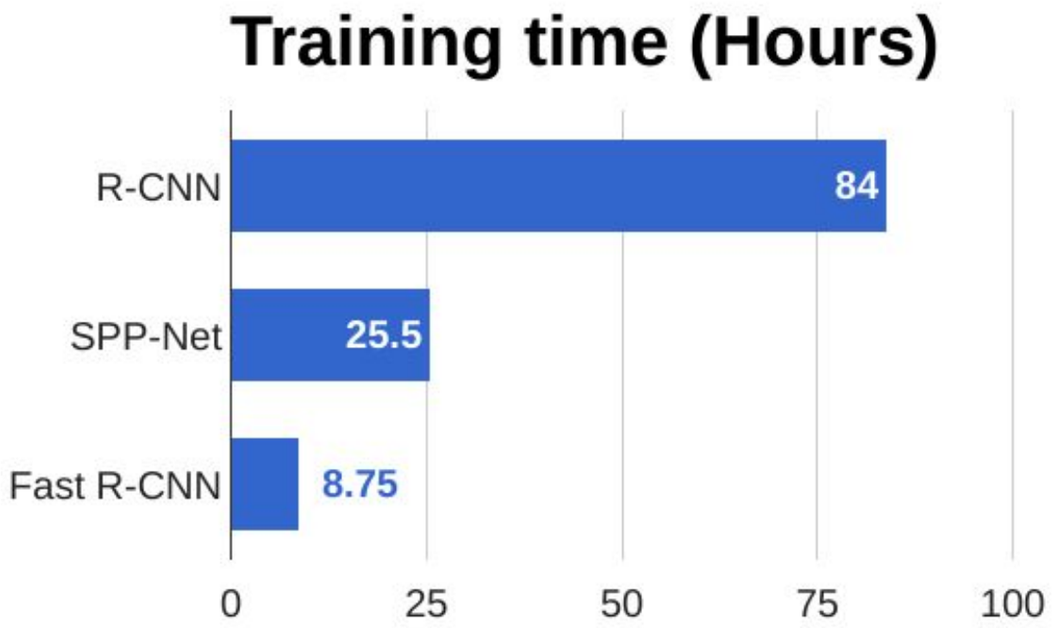


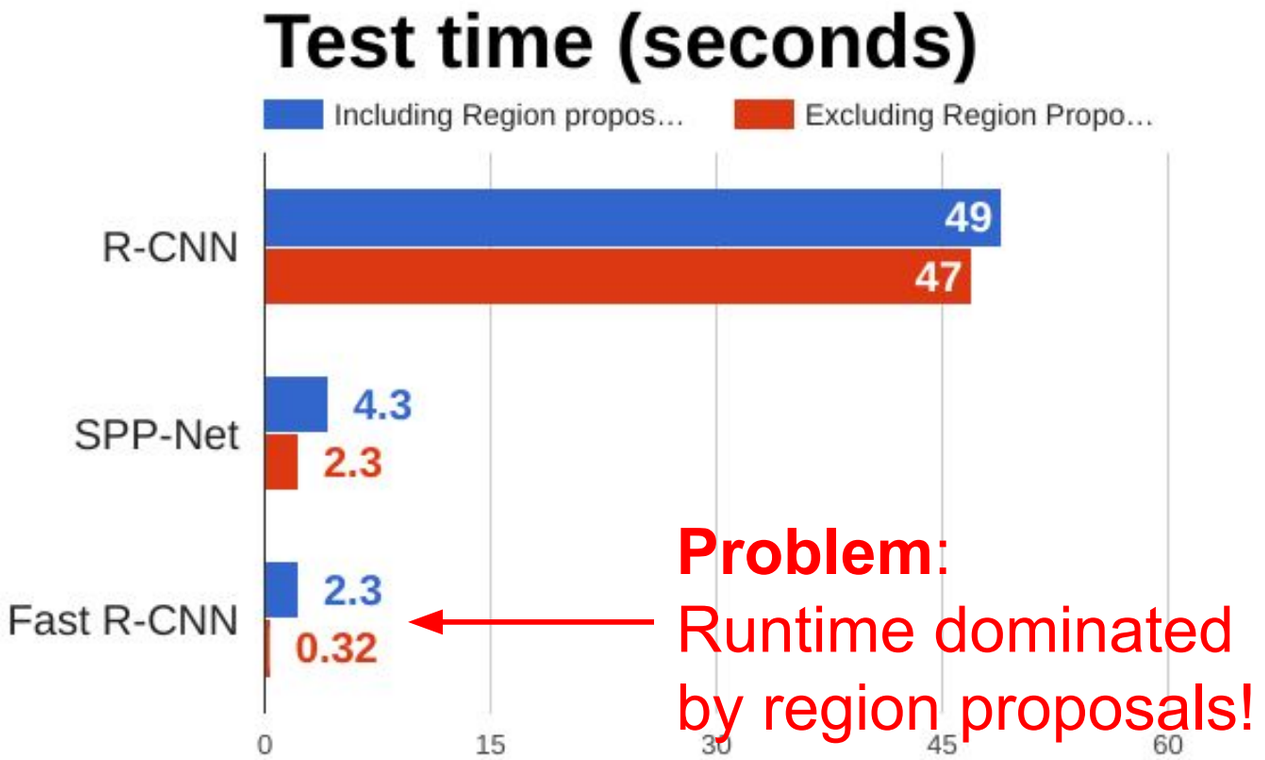


Girshick et al, “Rich feature hierarchies for accurate object detection and semantic segmentation”, CVPR 2014.
He et al, “Spatial pyramid pooling in deep convolutional networks for visual recognition”, ECCV 2014
Girshick, “Fast R-CNN”, ICCV 2015

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

R-CNN vs SPP vs Fast R-CNN





Problem:
Runtime dominated by region proposals!

Girshick et al, “Rich feature hierarchies for accurate object detection and semantic segmentation”, CVPR 2014.
He et al, “Spatial pyramid pooling in deep convolutional networks for visual recognition”, ECCV 2014
Girshick, “Fast R-CNN”, ICCV 2015

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

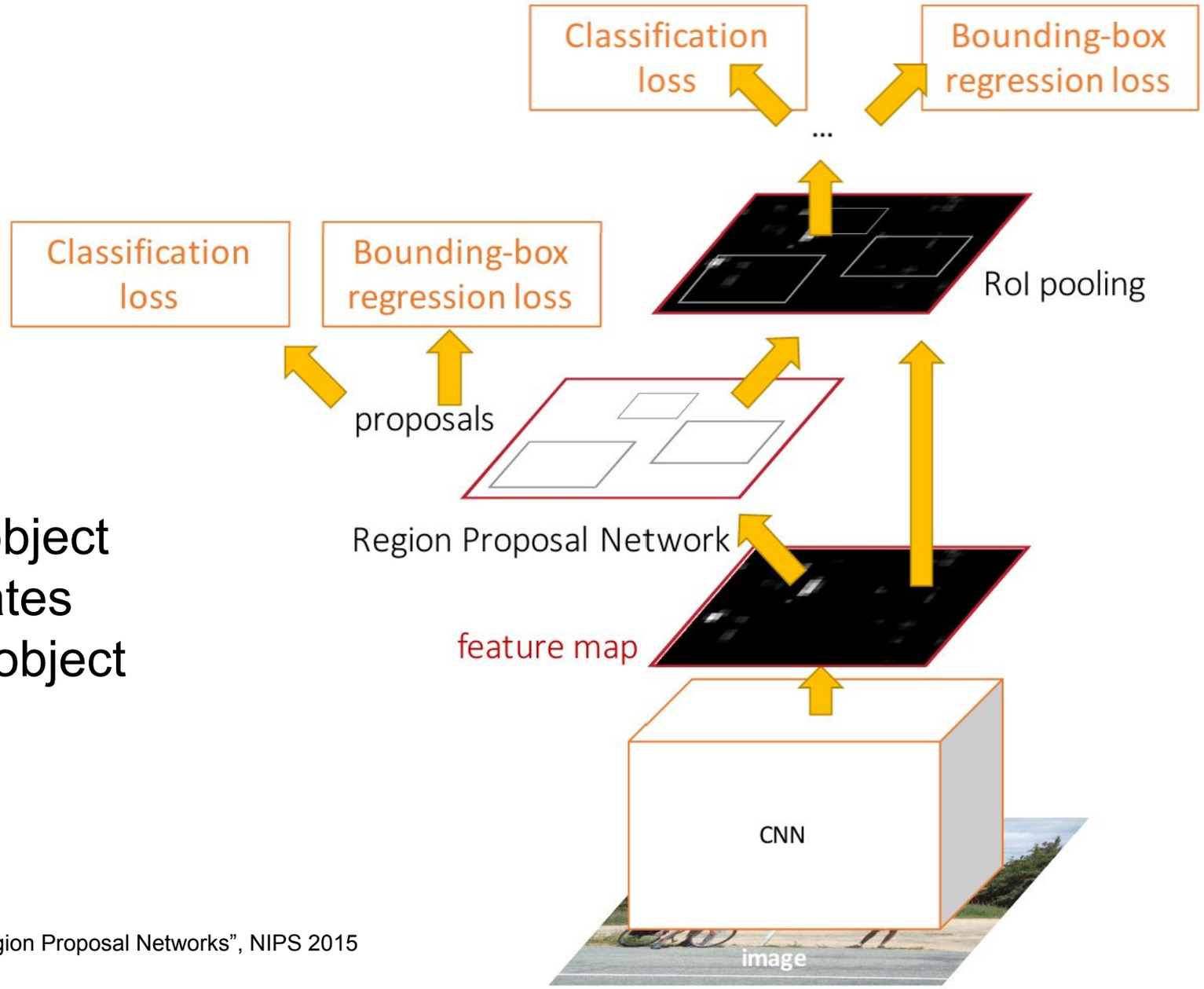
Faster R-CNN:

Make CNN do proposals!

Insert **Region Proposal Network (RPN)** to predict proposals from features

Jointly train with 4 losses:

1. RPN classify object / not object
2. RPN regress box coordinates
3. Final classification score (object classes)
4. Final box coordinates

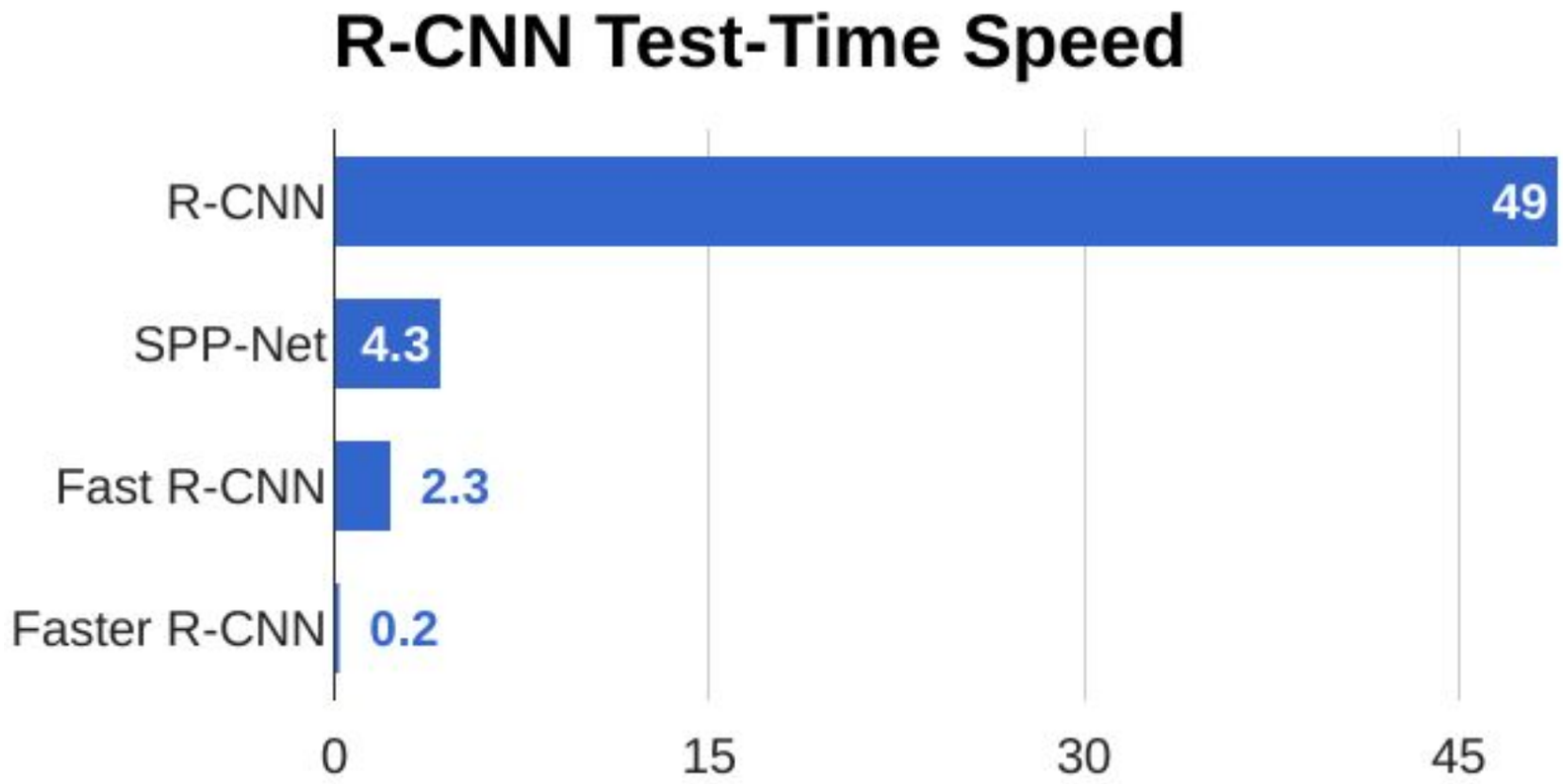


Ren et al, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”, NIPS 2015
Figure copyright 2015, Ross Girshick; reproduced with permission

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

Fast**er** R-CNN:

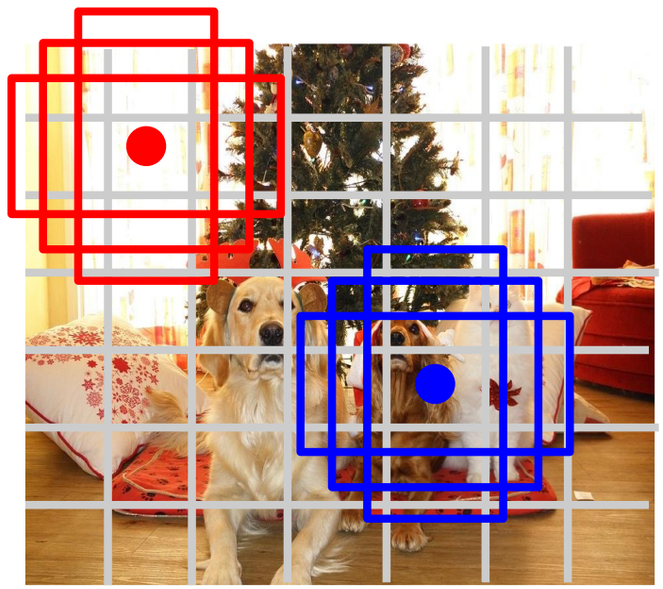
Make CNN do proposals!



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

Detection without Proposals: YOLO / SSD

Input image
3 x H x W



Divide image into grid
7 x 7

Image a set of **base boxes**
centered at each grid cell
Here B = 3

Within each grid cell:

- Regress from each of the B base boxes to a final box with 5 numbers:
  (dx, dy, dh, dw, confidence)
- Predict scores for each of C classes (including background as a class)

Output:
7 x 7 x (5 * B + C)

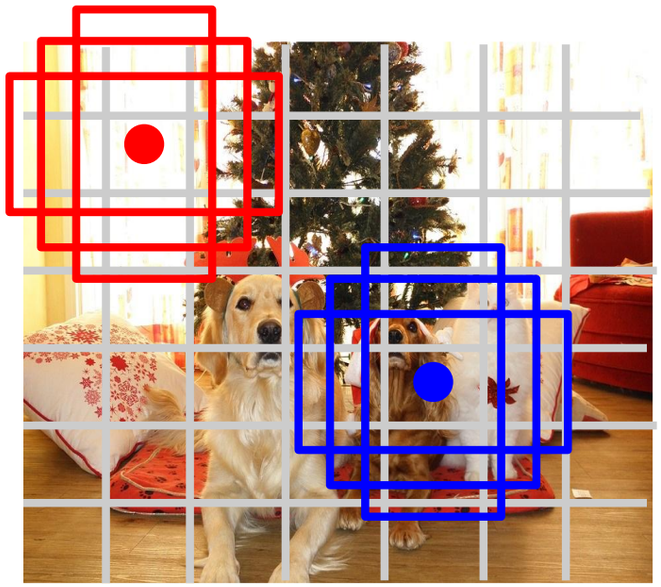
Redmon et al, “You Only Look Once:
Unified, Real-Time Object Detection”, CVPR 2016
Liu et al, “SSD: Single-Shot MultiBox Detector”, ECCV 2016

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

Detection without Proposals: YOLO / SSD

Go from input image to tensor of scores with one big convolutional network!

Input image
3 x H x W



Divide image into grid
7 x 7

Image a set of **base boxes** centered at each grid cell
Here B = 3

Within each grid cell:

- Regress from each of the B base boxes to a final box with 5 numbers:
  (dx, dy, dh, dw, confidence)
- Predict scores for each of C classes (including background as a class)

Output:
7 x 7 x (5 * B + C)

Redmon et al, “You Only Look Once:
Unified, Real-Time Object Detection”, CVPR 2016
Liu et al, “SSD: Single-Shot MultiBox Detector”, ECCV 2016

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

Object Detection: Impact of Deep Learning

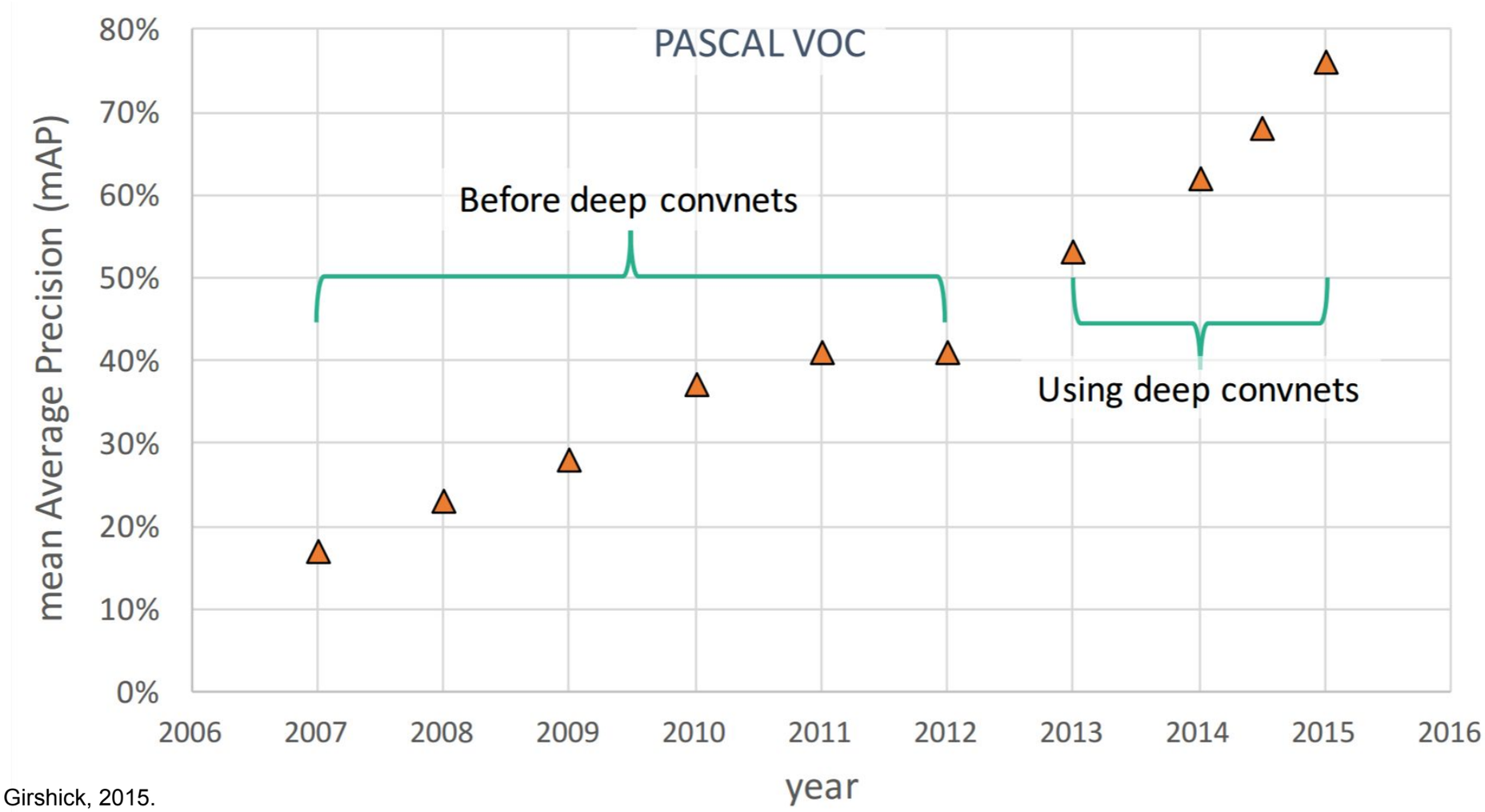


Figure copyright Ross Girshick, 2015.
Reproduced with permission.

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

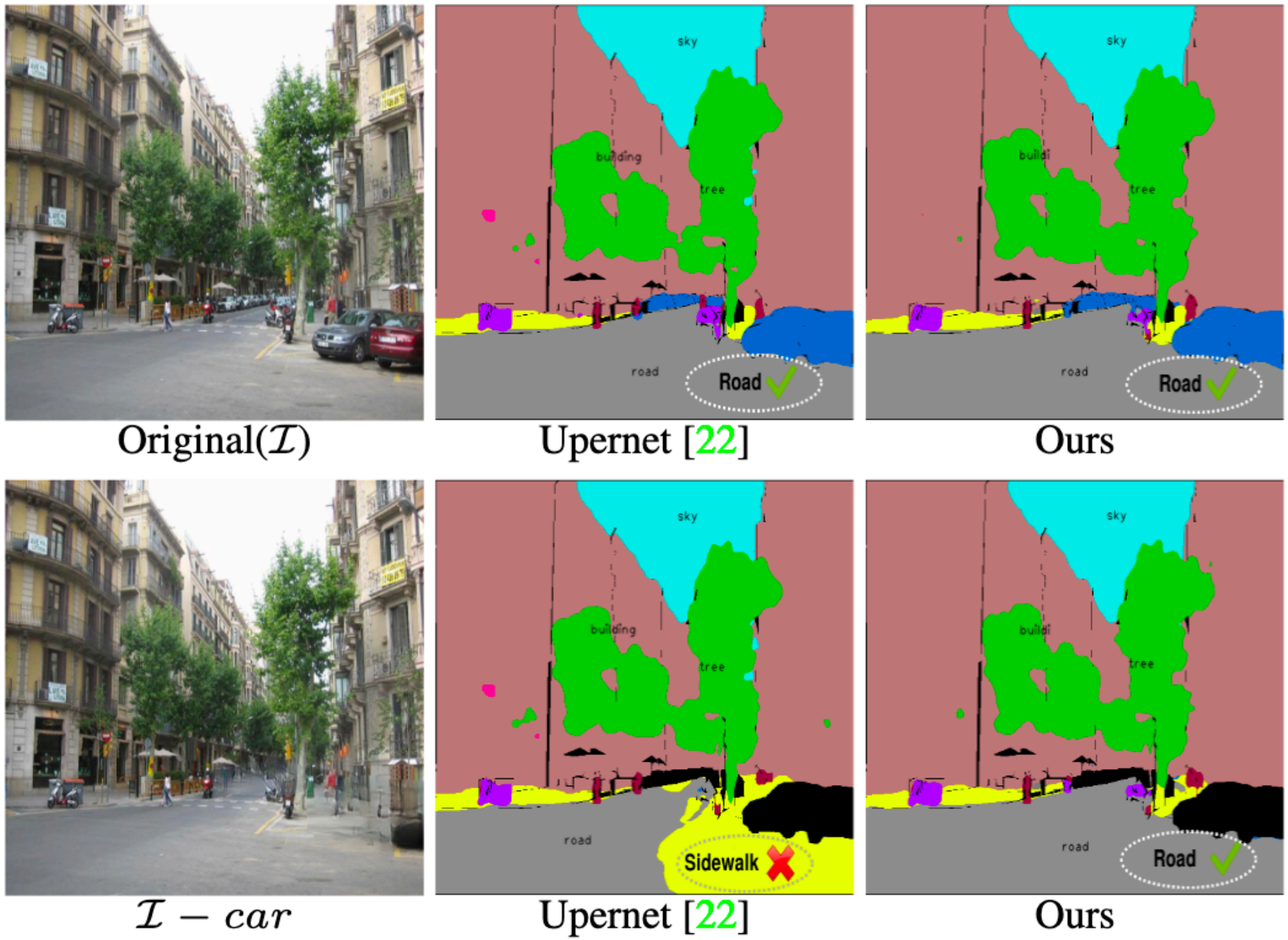
Problems: Bright and Dark Side of Context

[Antonio Torralba]

Problems: Bright and Dark Side of Context

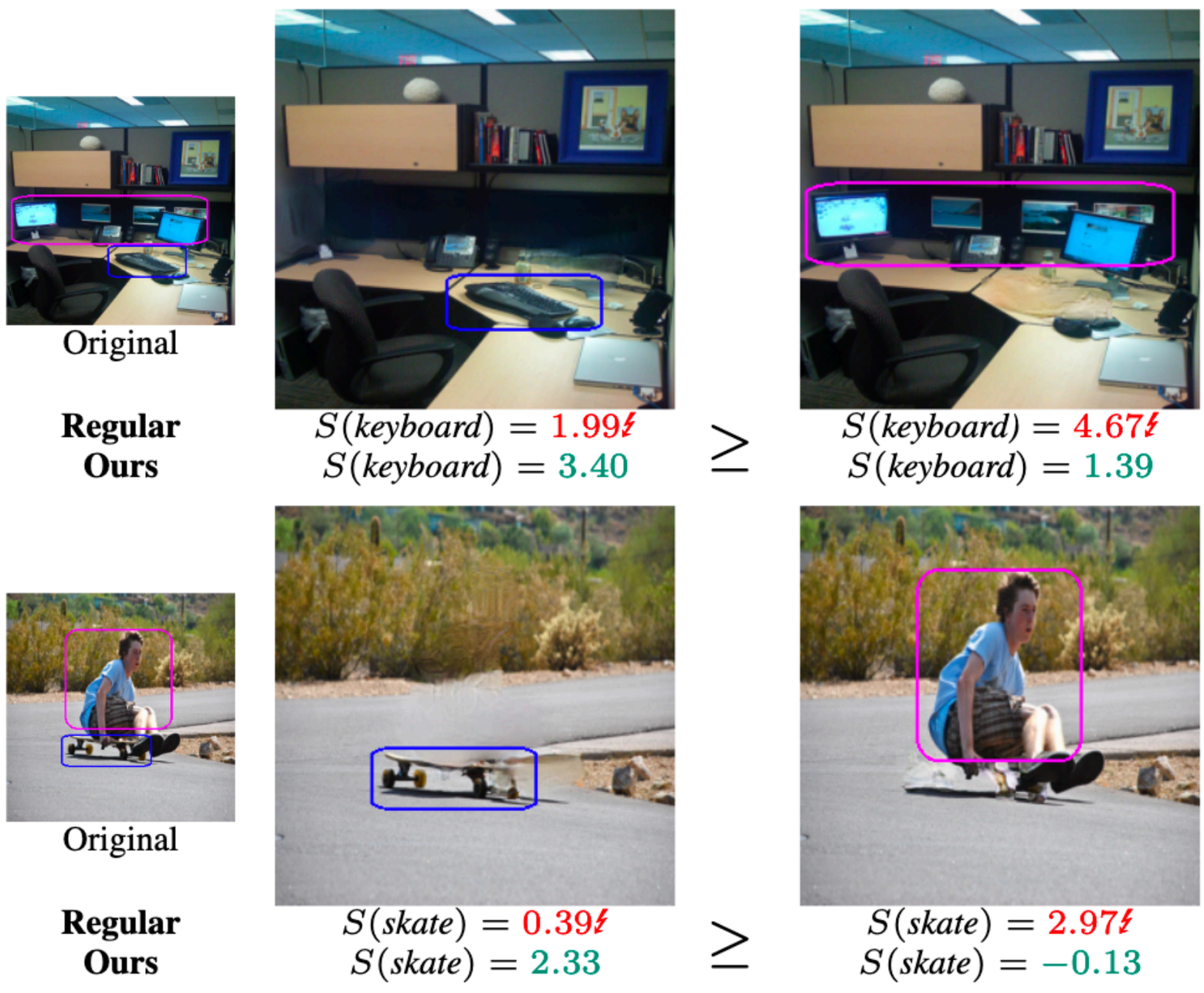
"Context Models and Out-of-context Objects"
Myung Jin Choi, Antonio Torralba, and Alan S. Willsky
Pattern Recognition Letters, 2012.

Problems: Bright and Dark Side of Context

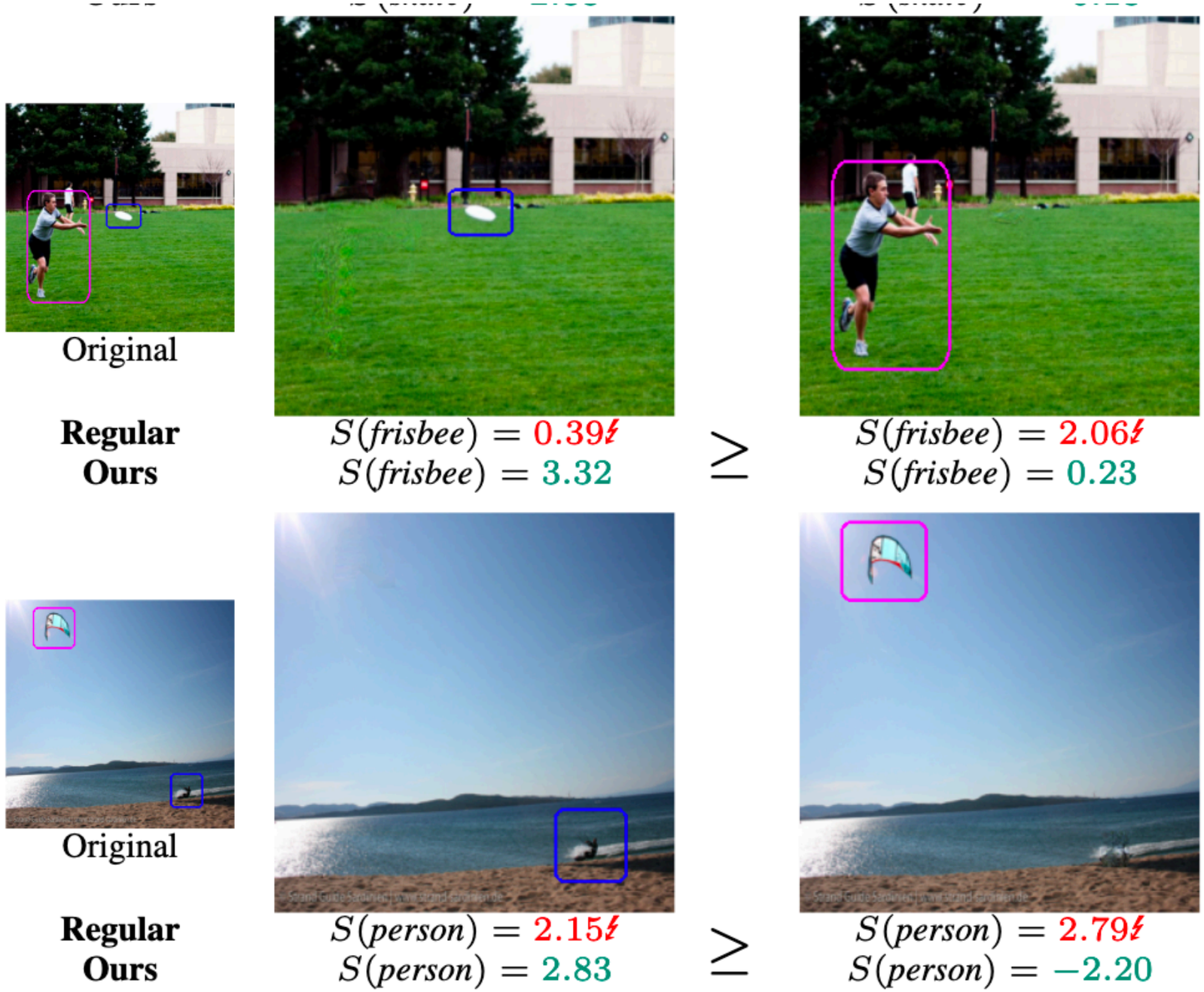


Rakshith Shetty; Bernt Schiele; Mario Fritz; Not Using the Car to See the Sidewalk: Quantifying and Controlling the Effects of Context in Classification and Segmentation ;IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2019.

Problems: Bright and Dark Side of Context



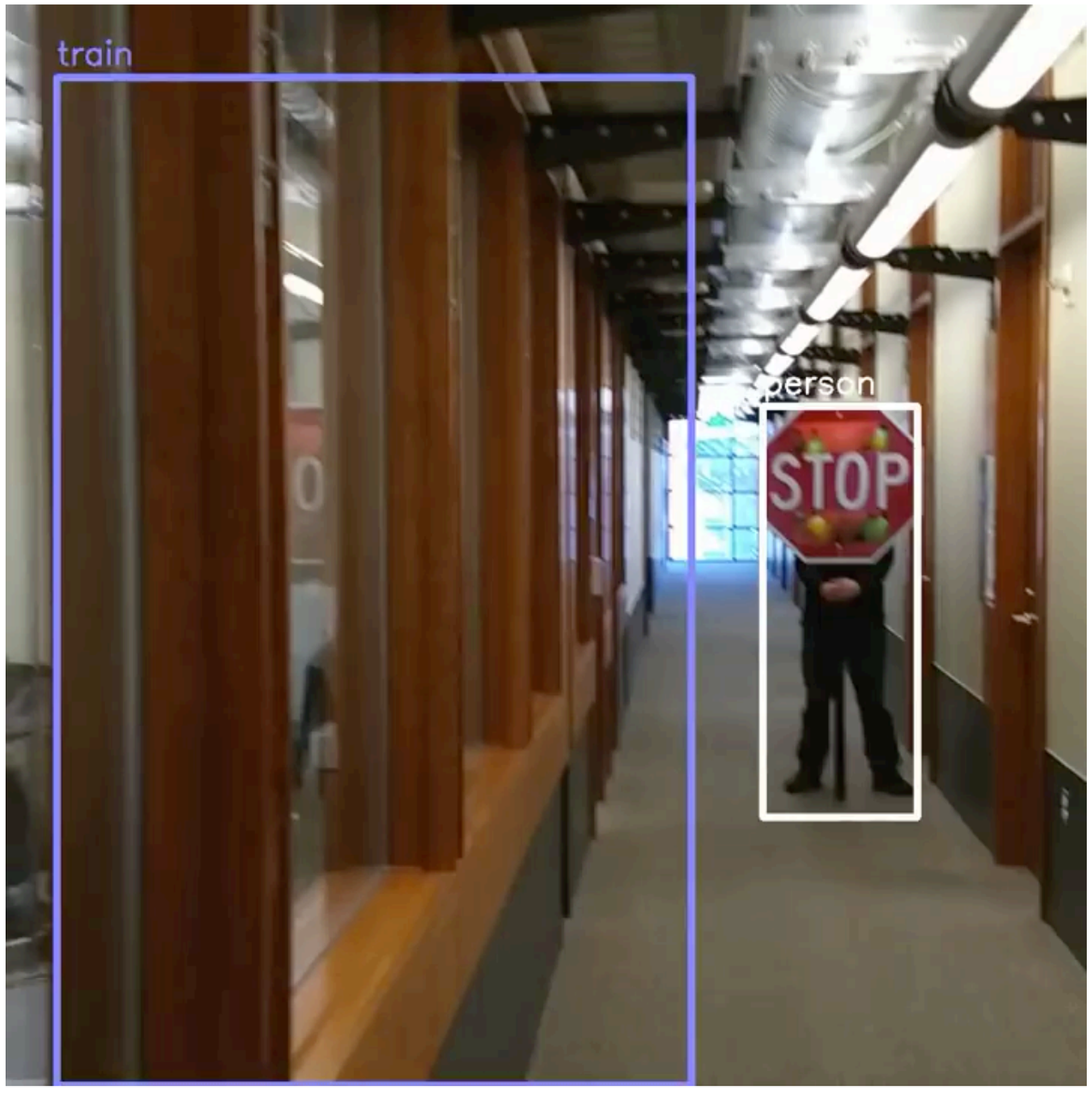
Rakshith Shetty; Bernt Schiele; Mario Fritz; Not Using the Car to See the Sidewalk: Quantifying and Controlling the Effects of Context in Classification and Segmentation ;IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2019.



Rakshith Shetty; Bernt Schiele; Mario Fritz; Not Using the Car to See the Sidewalk: Quantifying and Controlling the Effects of Context in Classification and Segmentation ;IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2019.

Attacks on Object Detectors

- Attacks against ML Models / Object Detectors
- Attacker can craft small change in the image that makes detector
  - Fail
  - Behaves to the liking of adversary



https://bair.berkeley.edu/blog/2017/12/30/yolo-attack/

Advanced Architectures
Object Detection & Segmentation
Bright & Dark Side of Context

Prof. Dr. Mario Fritz
CISPA Helmholtz Center for Information Security