Convolutional Neural Network

Jaegul Choo (주재걸)
Korea University
https://sites.google.com/site/jaegulchoo/

Most slides made by my student, Yunjey Choi

Contents

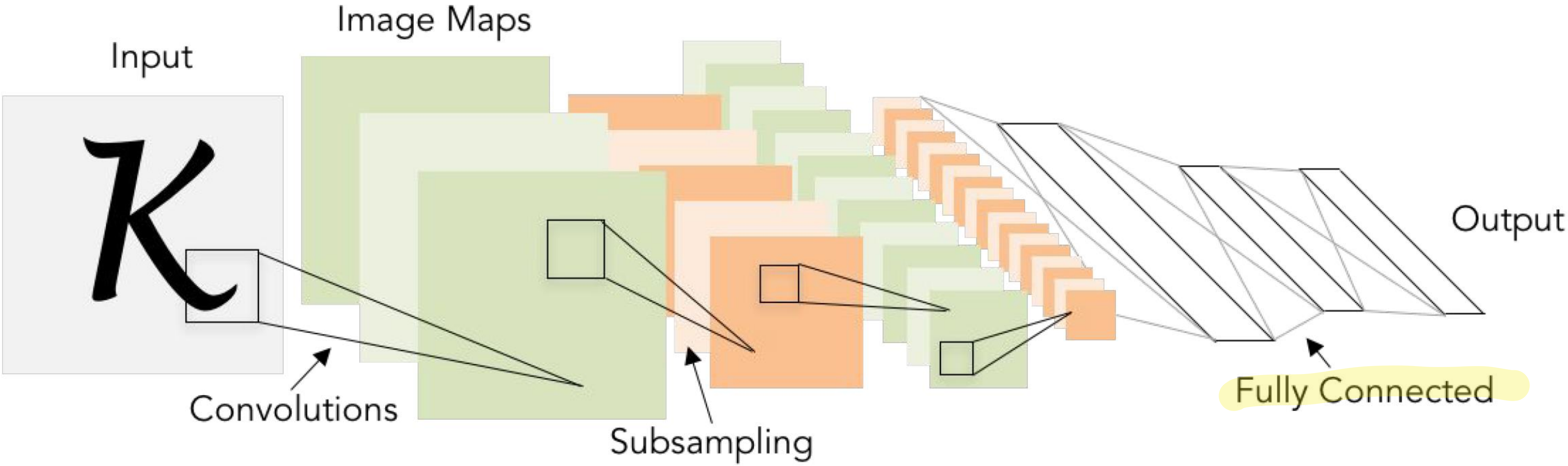
05
Advanced CNN Artitectures

CNN Architectures

Case Studies

- AlexNet
- VGG
- GoogLeNet
- ResNet
- ...

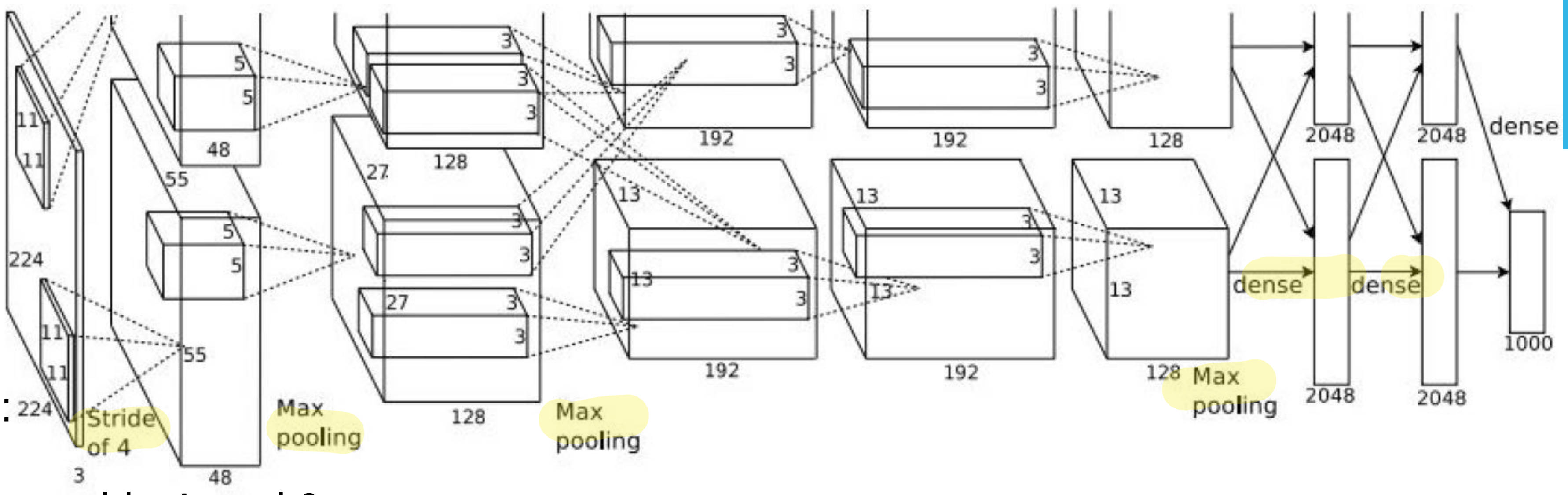
Review: LeNet-5 [LeCun et al., 1998]



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]

Case Study: AlexNet [Krizhevsky et al. 2012]

딥러닝 ~ 불꽃쏘기



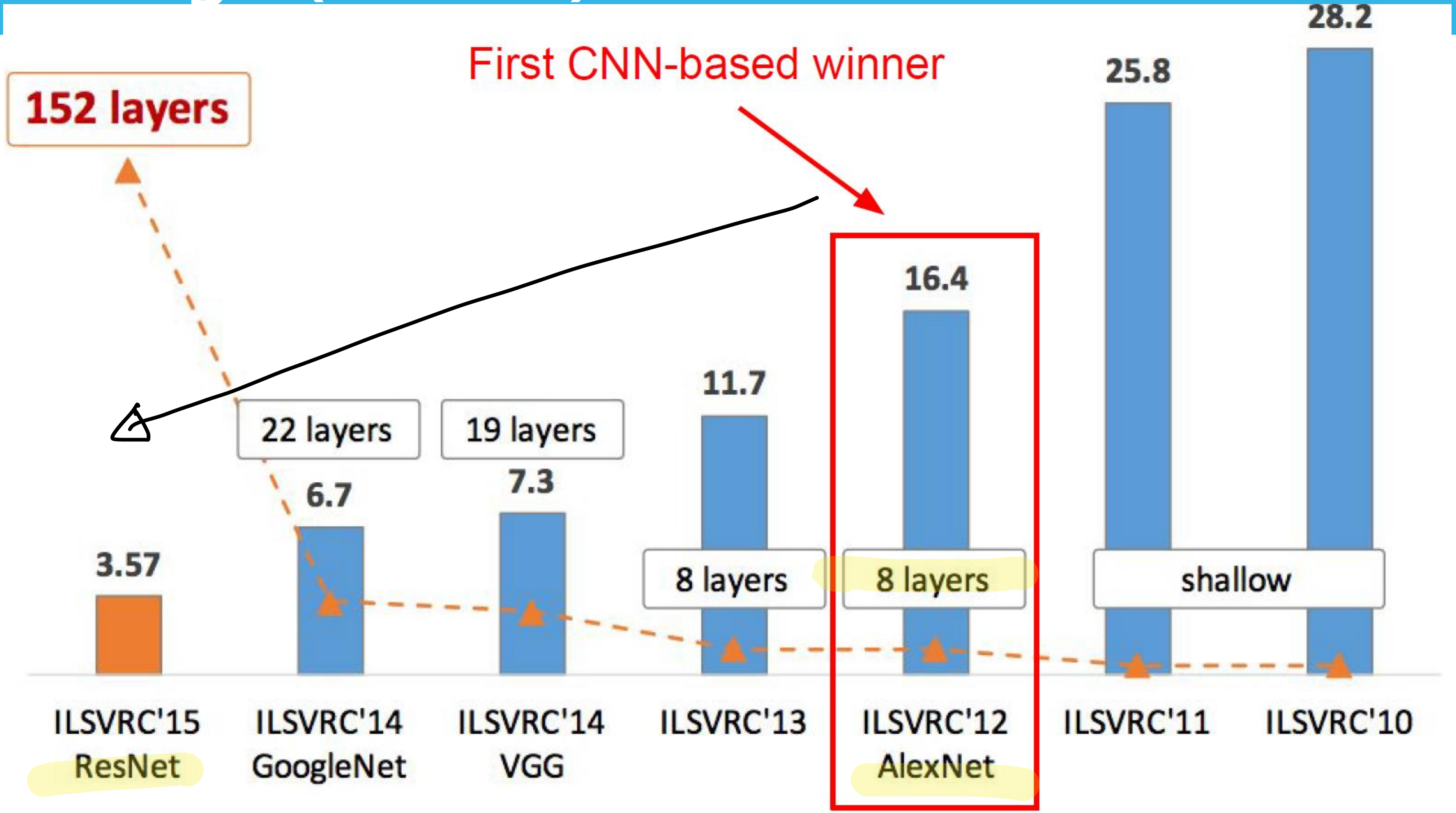
Input 개수

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)

1000개 중 하나고르기

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 ≒ ADAM optimizer
- Learning rate 1e-2, reduced by 10 manually when val accuracy plateaus
- L2 weight decay 5e-4
- 7 CNN ensemble: 18.2% -> 15.4%

ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners



Case Study: VGGNet [Simonyan and Zisserman, 2014]

AlexNet
11x11 필터 ≒ 3x3 필터

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

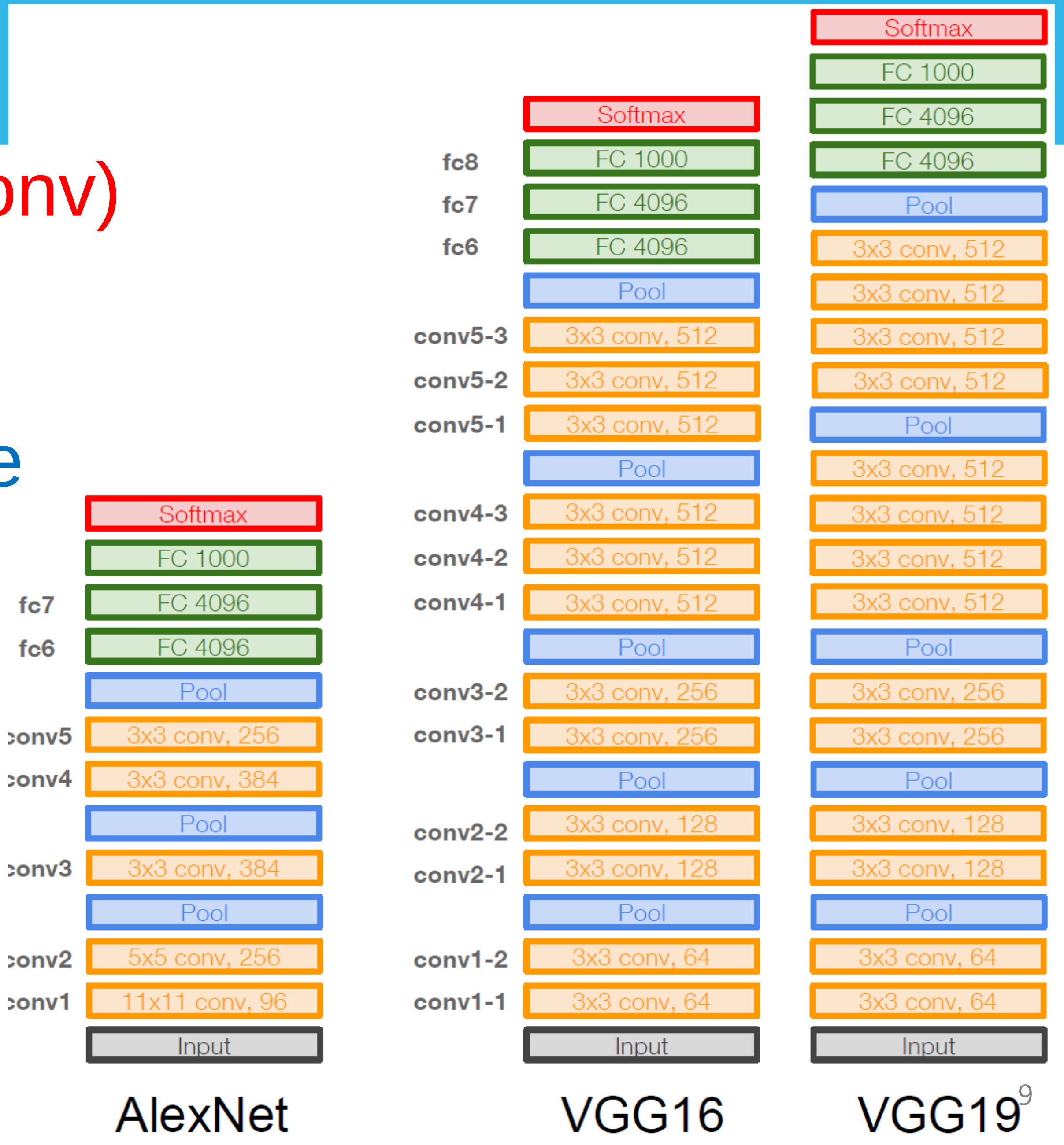
Case Study: VGGNet [Simonyan and Zisserman, 2014]

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

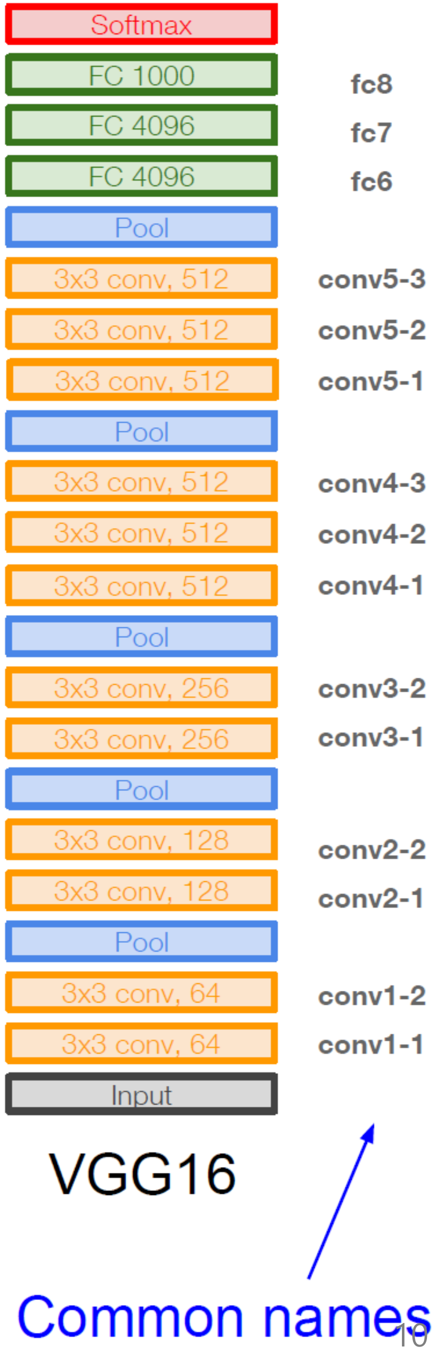
Stack of three 3x3 conv (stride 1) layers has same effective receptive field as one 7x7 conv layer

But deeper, more non-linearities

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



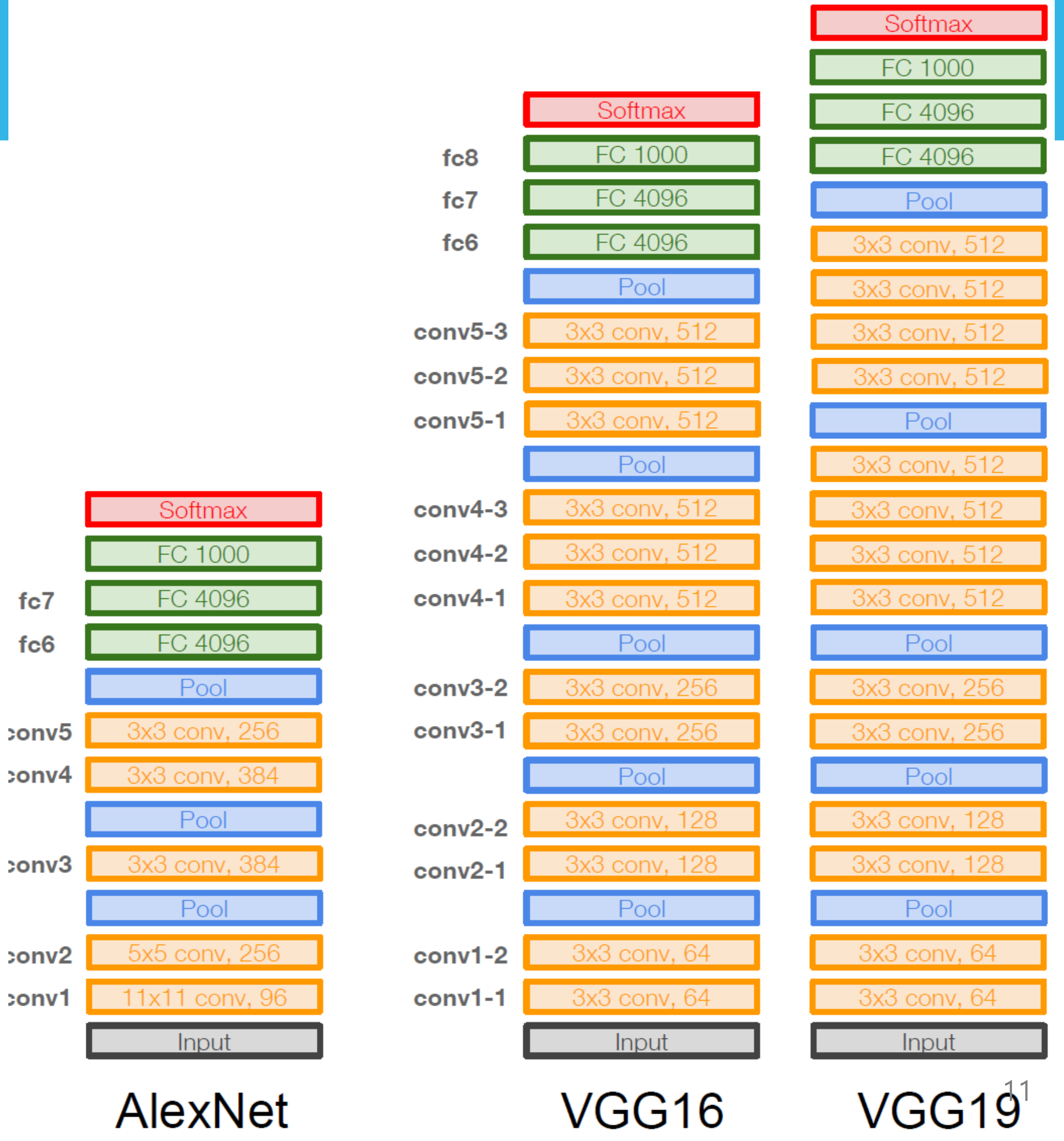
INPUT: [224x224x3] memory: 224*224*3=150K params: 0
CONV3-64: [224x224x64] memory: 224*224*64=3.2M params: (3*3*3)*64 = 1,728
CONV3-64: [224x224x64] memory: 224*224*64=3.2M params: (3*3*64)*64 = 36,864
POOL2: [112x112x64] memory: 112*112*64=800K params: 0
CONV3-128: [112x112x128] memory: 112*112*128=1.6M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



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

Case Study: VGGNet [Simonyan and Zisserman, 2014]

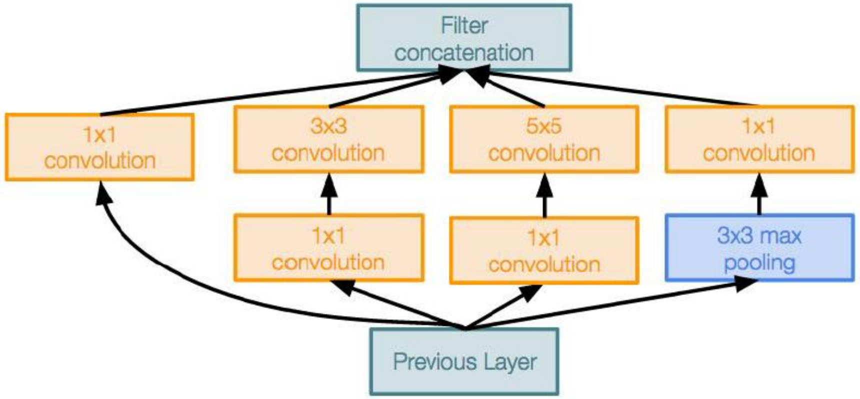
Details:
- ILSVRC’14 2nd in classification, 1st in localization
- Similar training procedure as Krizhevsky 2012
- No Local Response Normalisation (LRN)
- Use VGG16 or VGG19 (VGG19 only slightly better, more memory)
- Use ensembles for best results
- FC7 features generalize well to other tasks



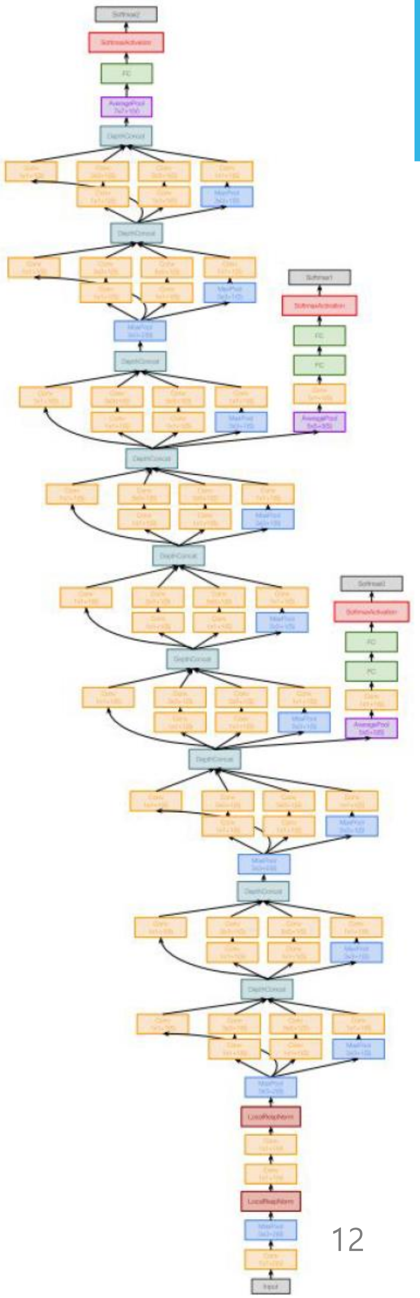
Case Study: GoogLeNet [Szegedy et al., 2014]

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)

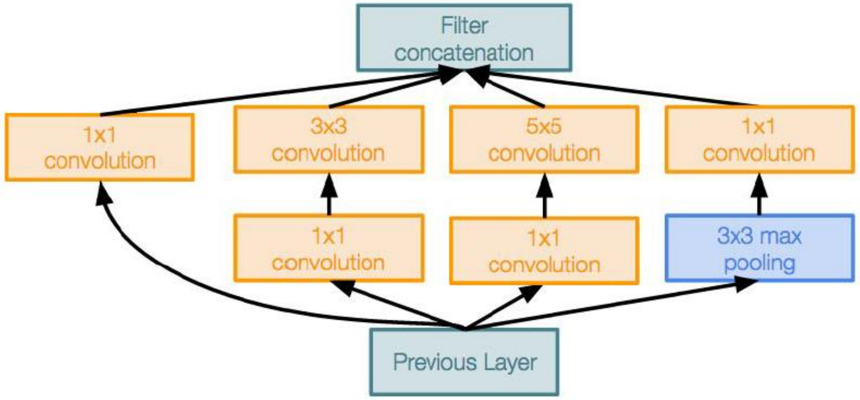


Inception module

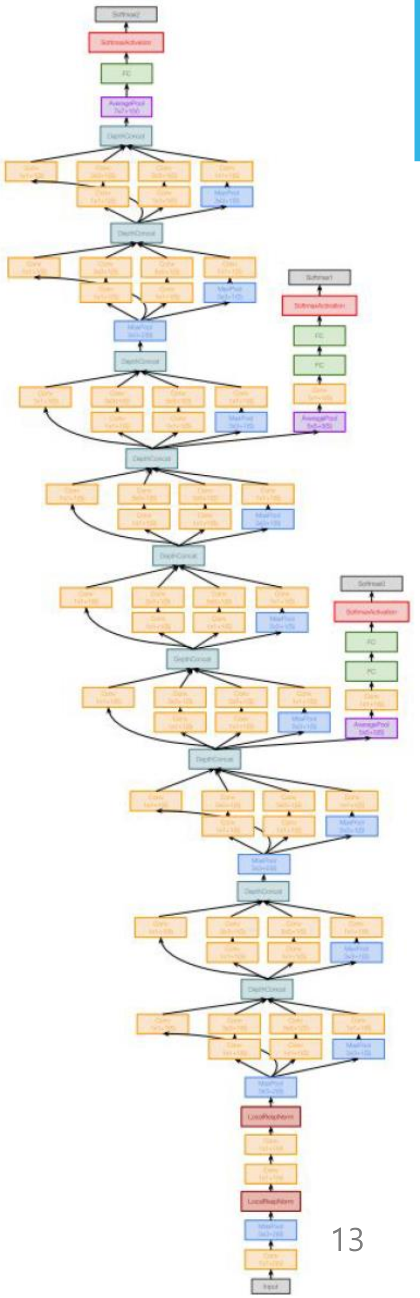


Case Study: GoogLeNet [Szegedy et al., 2014]

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



Inception module



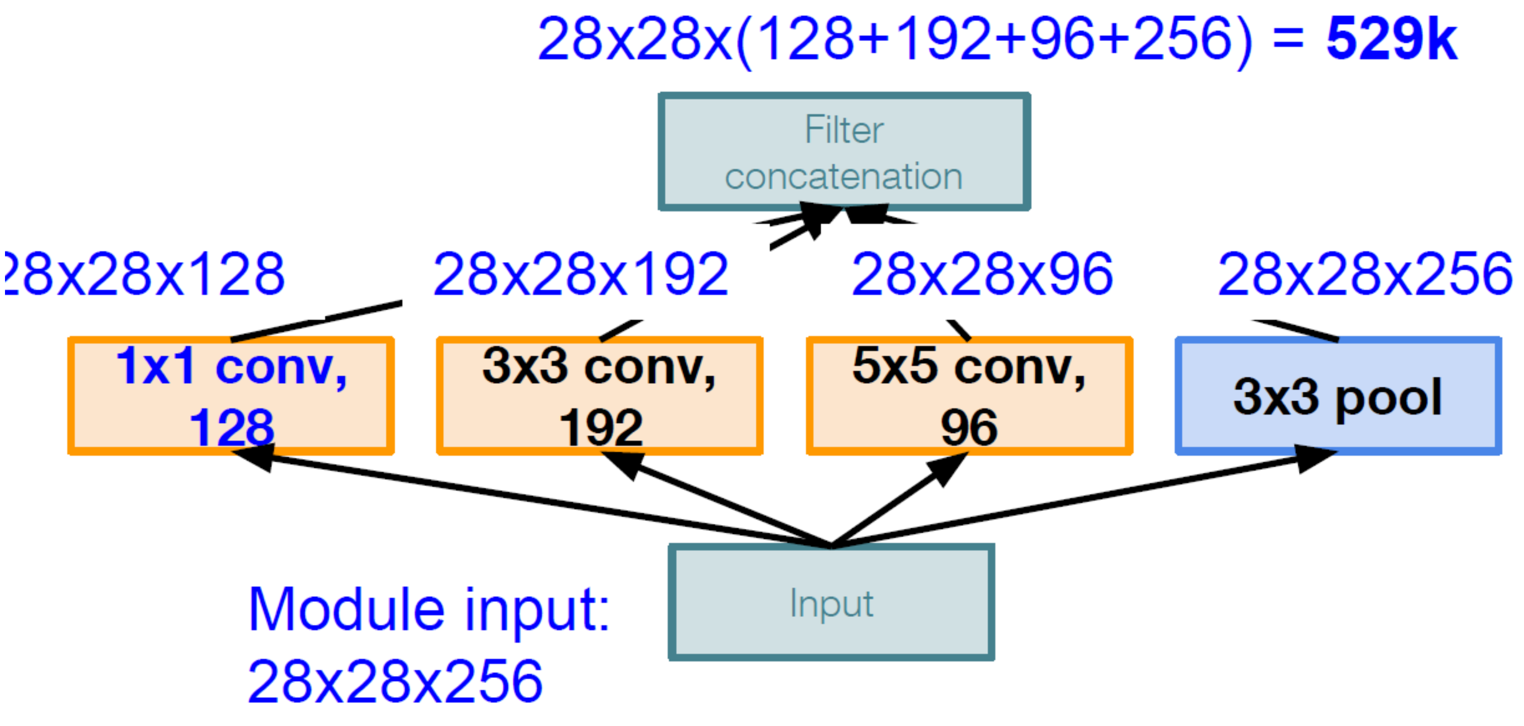
Case Study: GoogLeNet [Szegedy et al., 2014]

Example:

Q3:What is output size after filter concatenation?

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

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



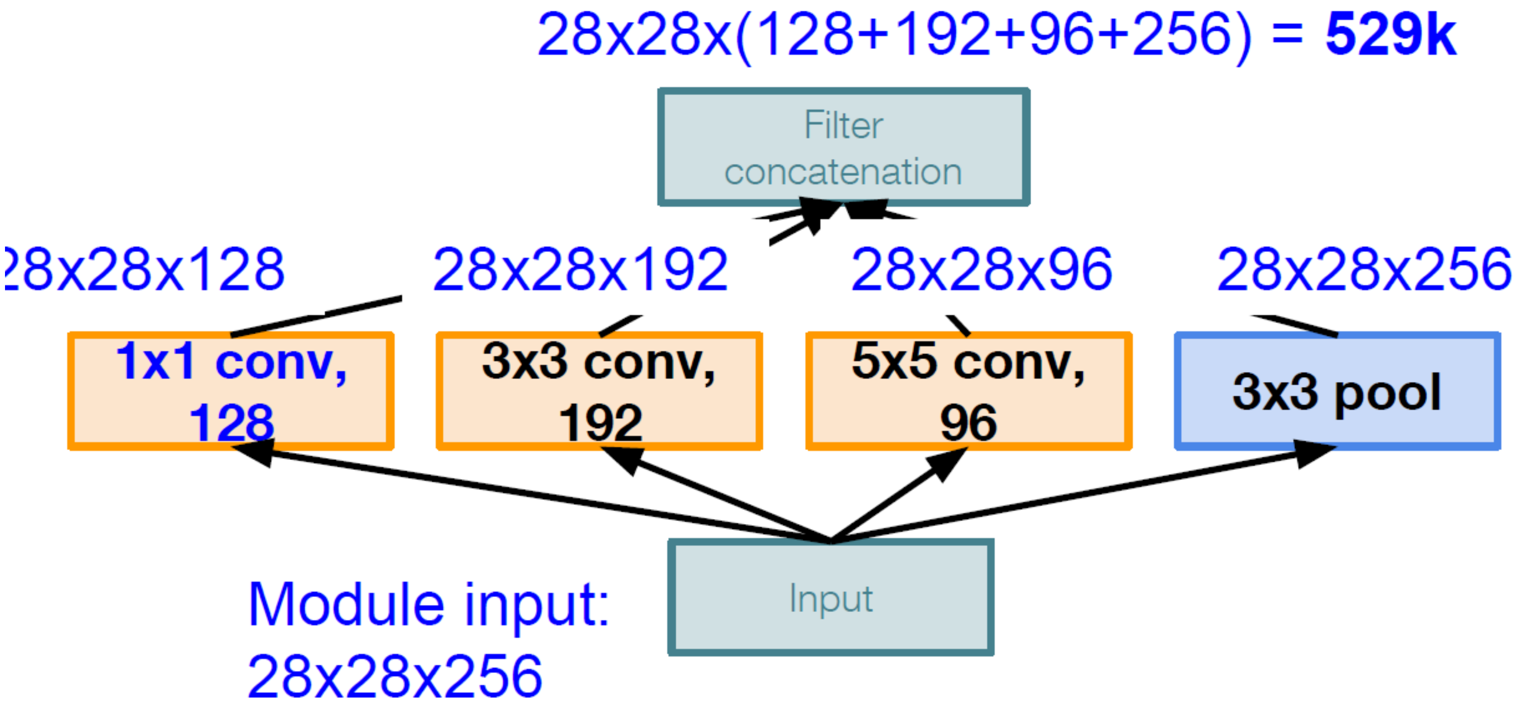
Naive Inception module

Case Study: GoogLeNet [Szegedy et al., 2014]

Example:

Q3:What is output size after filter concatenation?

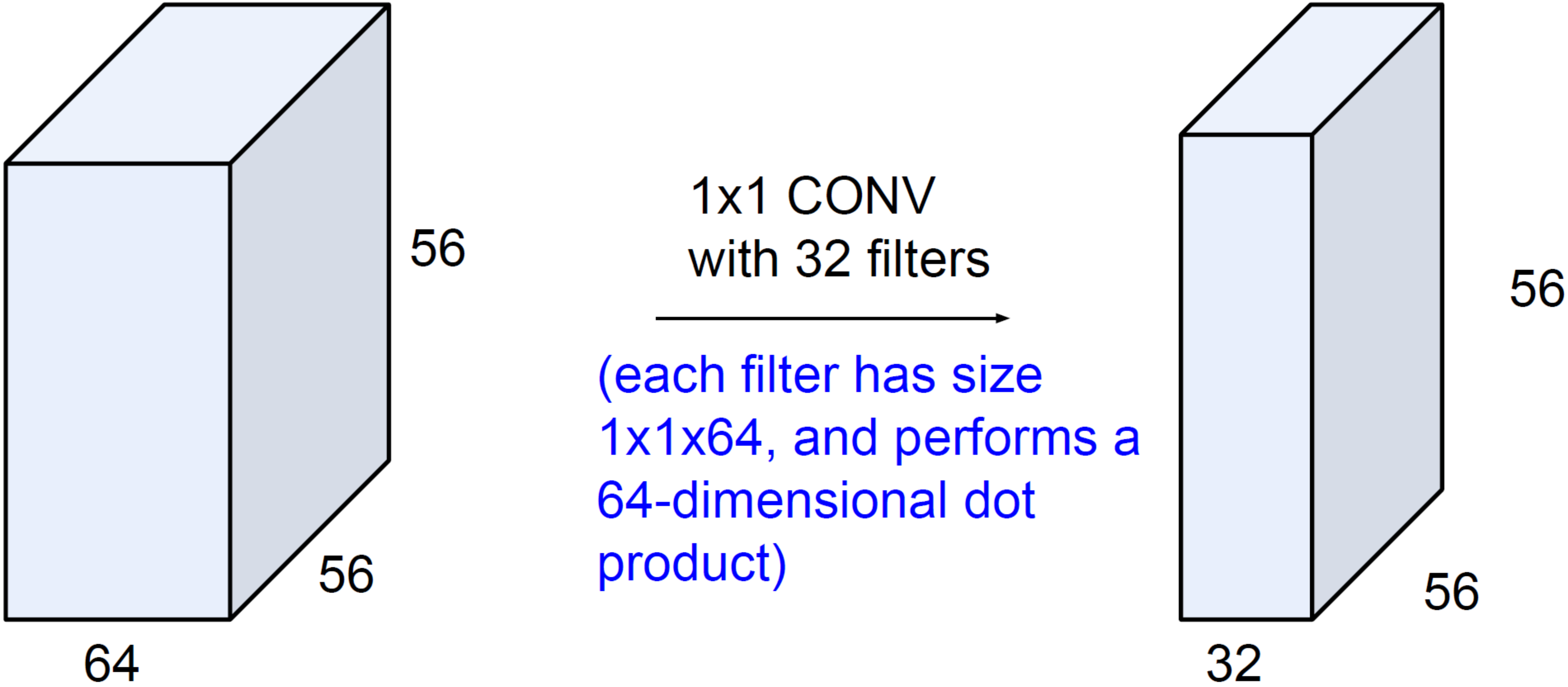
Q: What is the problem with this?
[Hint: Computational complexity]



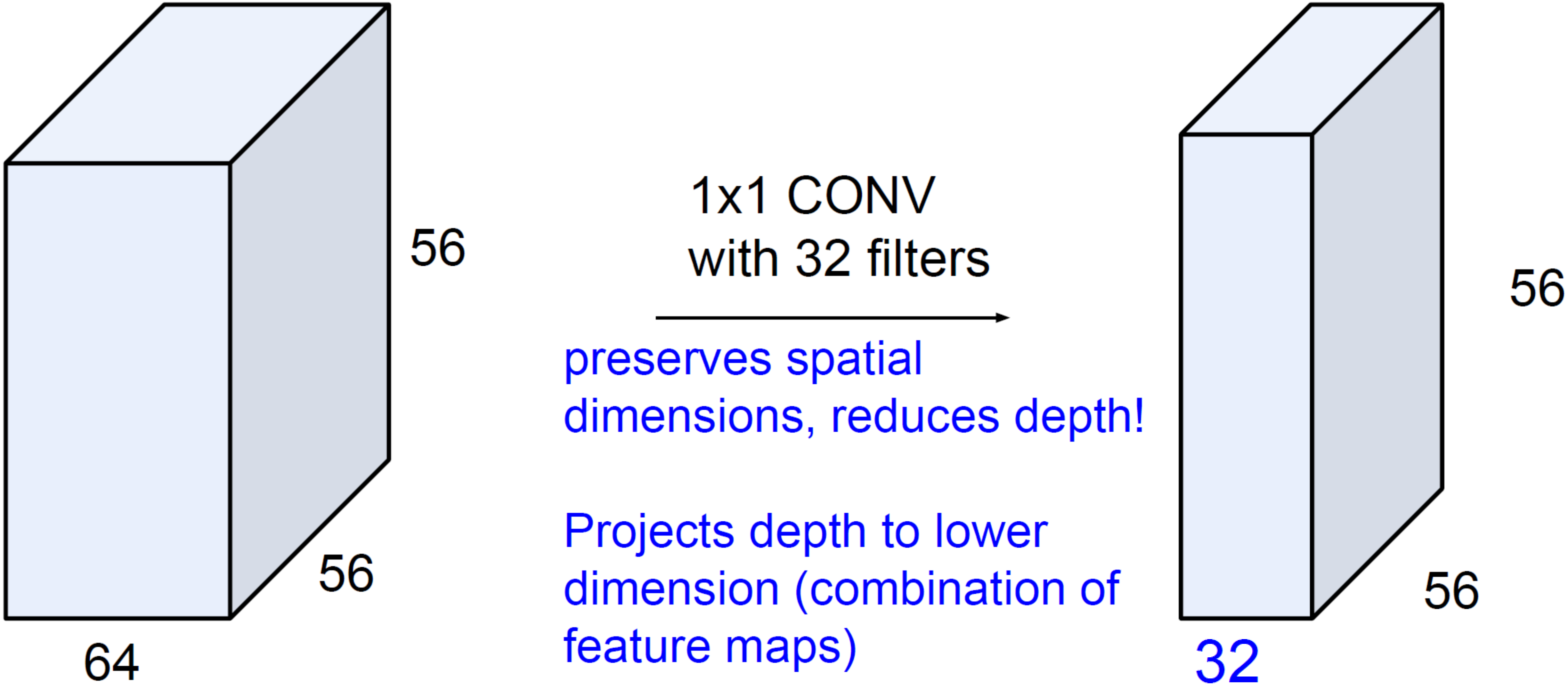
Naive Inception module

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

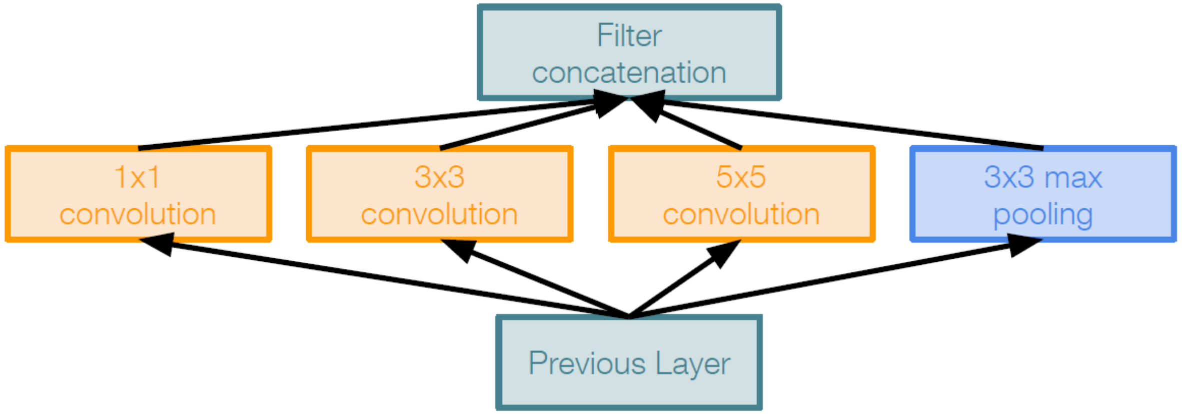
1x1 Convolutions



1x1 Convolutions

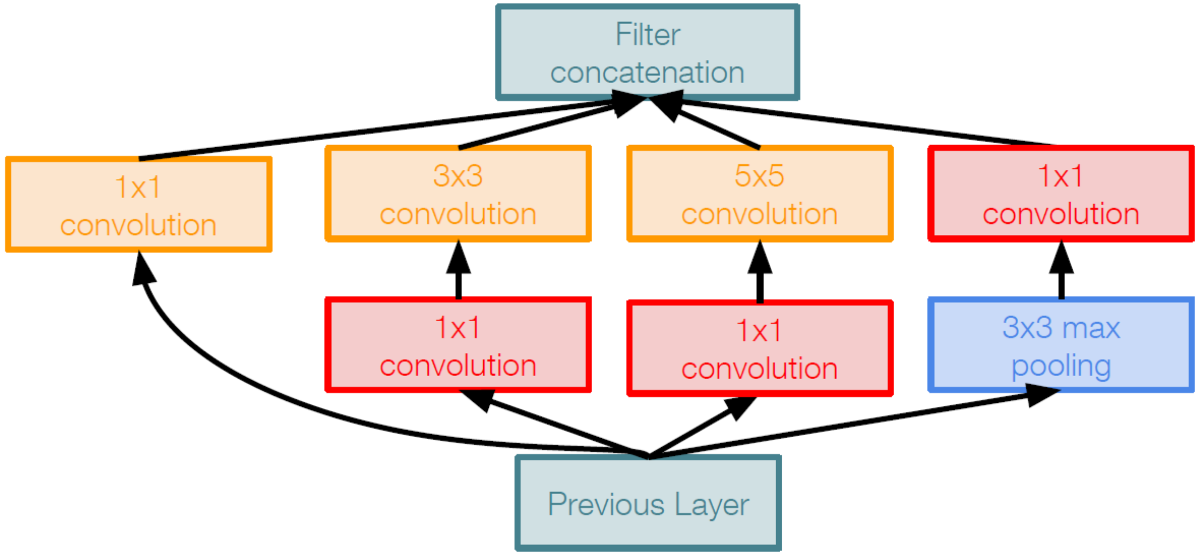


Case Study: GoogLeNet [Szegedy et al., 2014]



Naive Inception module

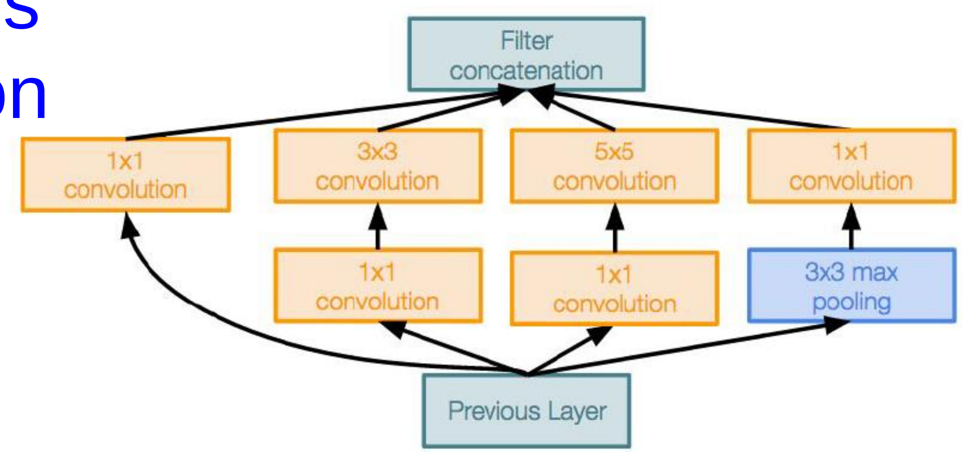
1x1 conv "bottleneck" layers



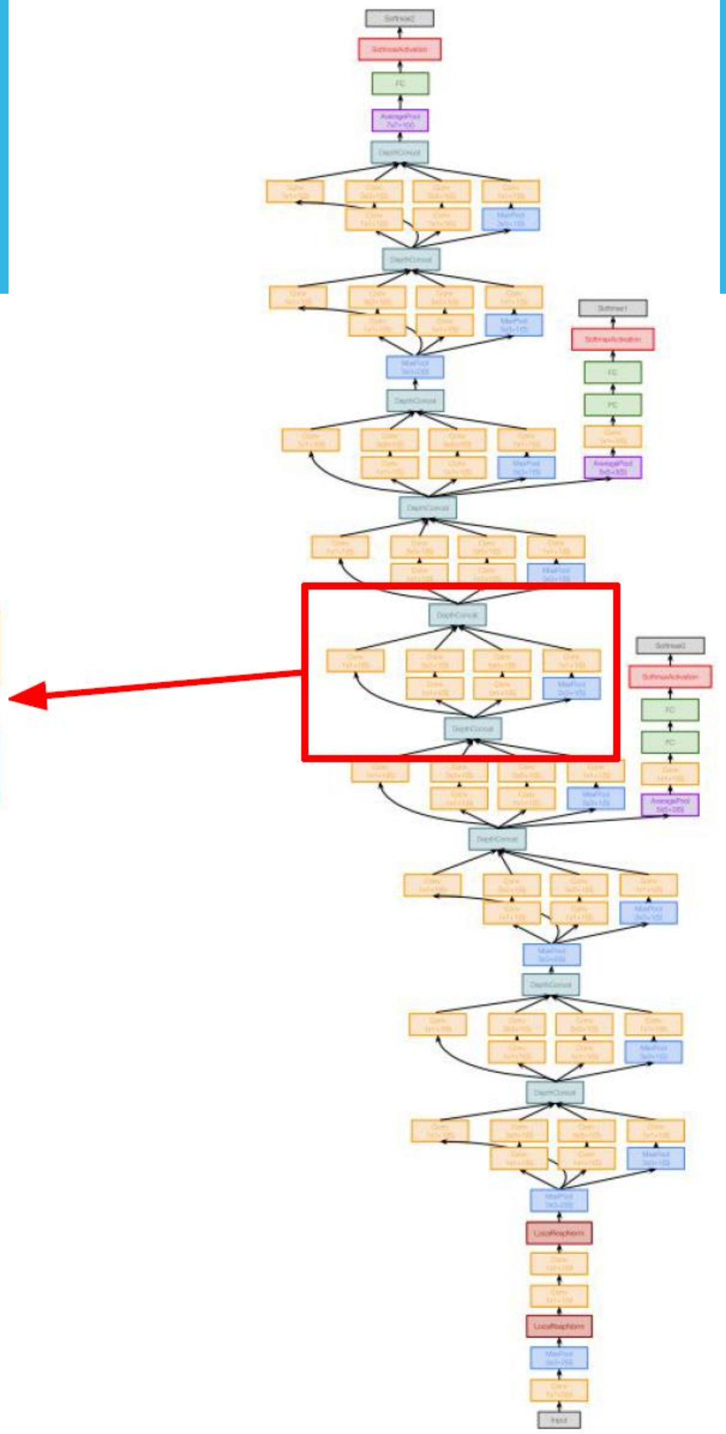
Inception module with dimension reduction

Case Study: GoogLeNet
[Szegedy et al., 2014]

Stack Inception modules
with dimension reduction
on top of each other

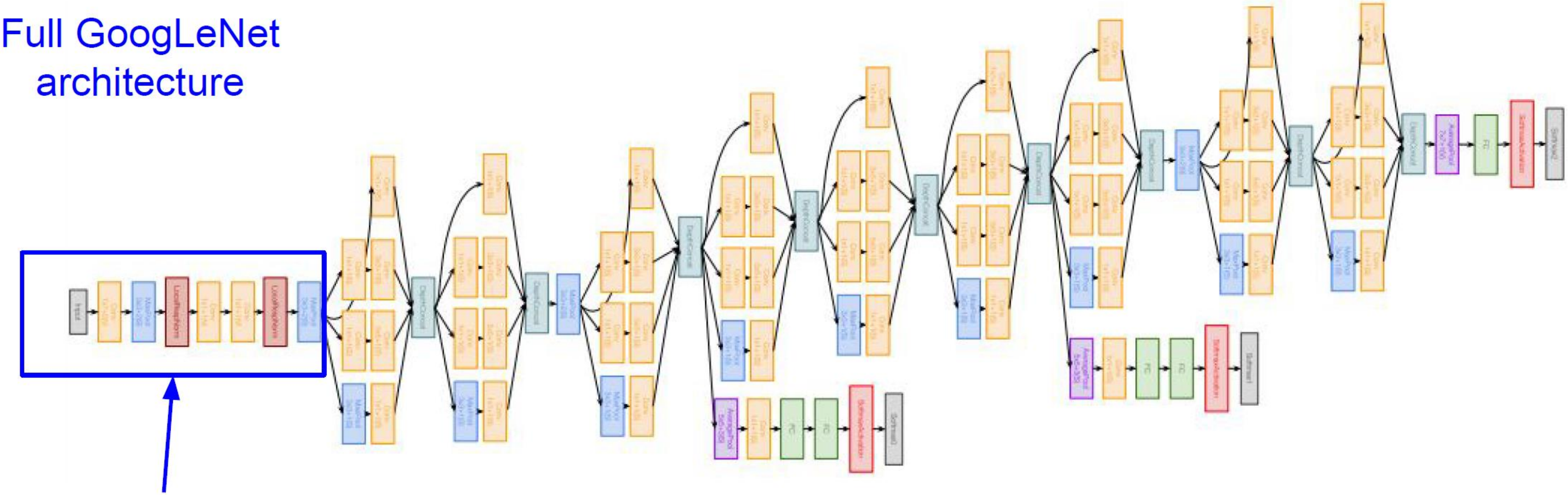


Inception module



Case Study: GoogLeNet [Szegedy et al., 2014]

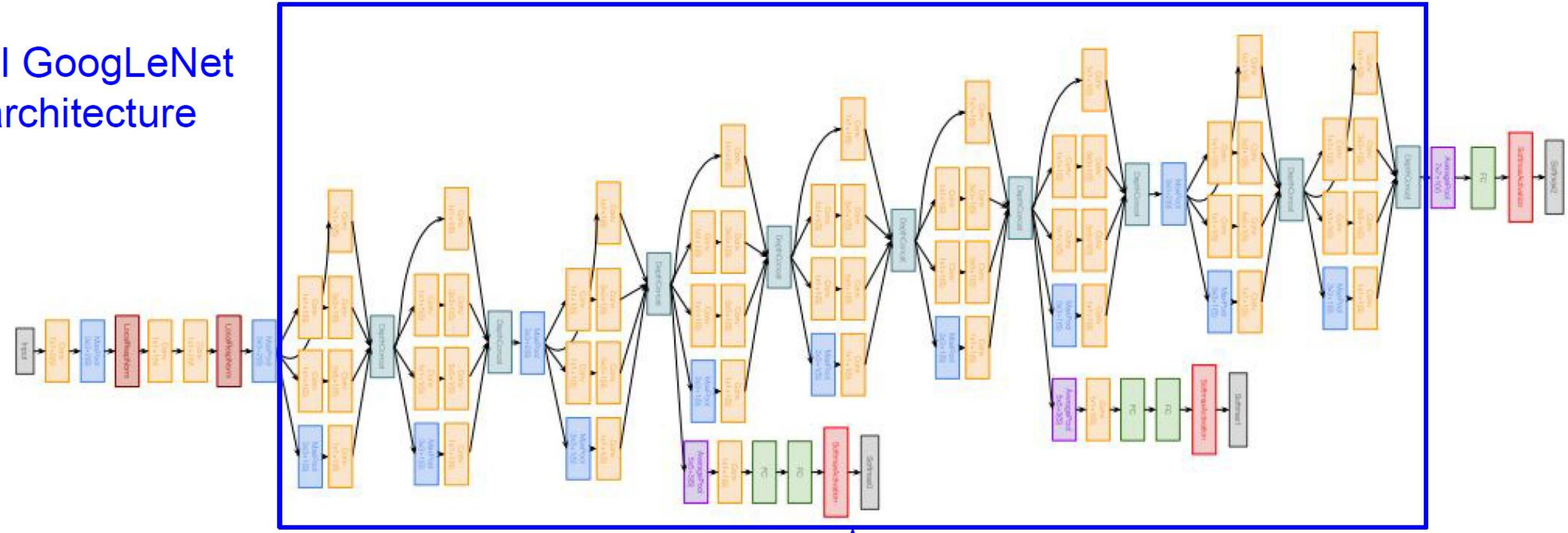
Full GoogLeNet architecture



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

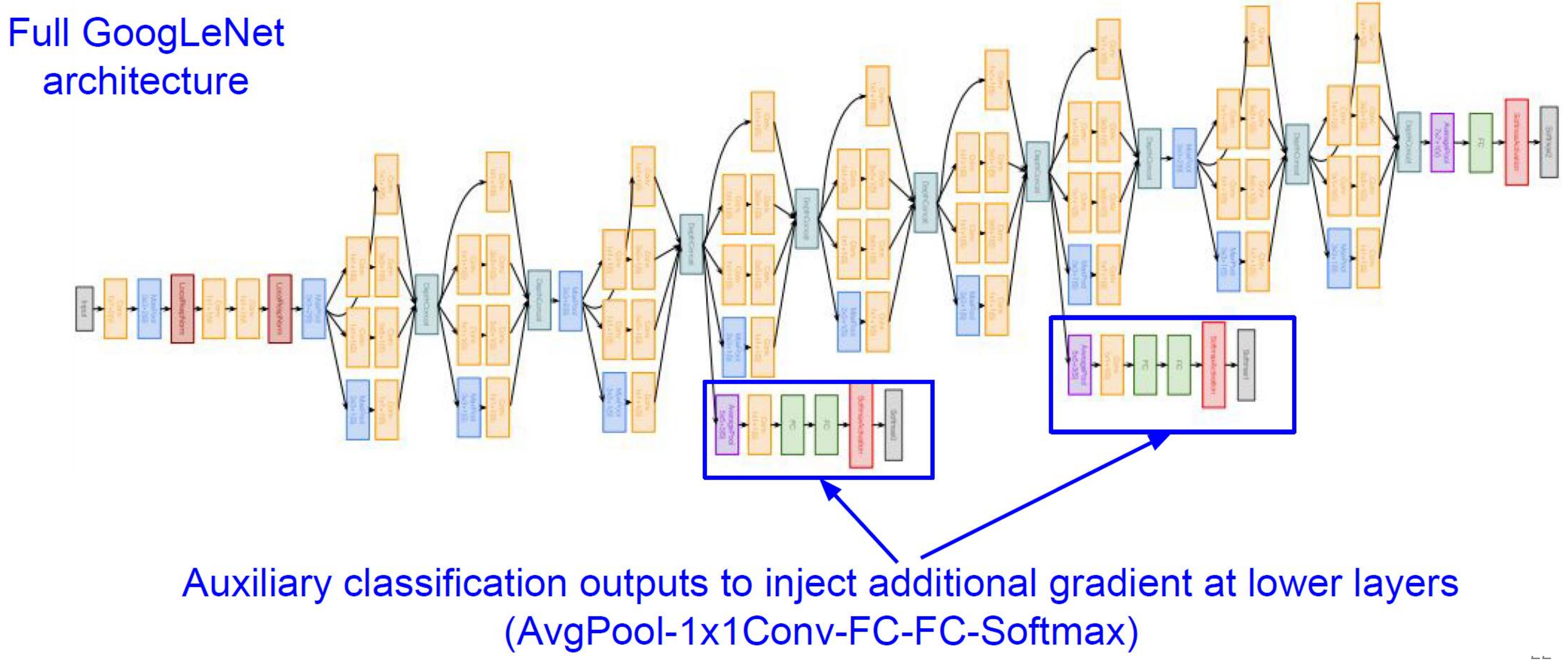
Case Study: GoogLeNet [Szegedy et al., 2014]

Full GoogLeNet architecture



Stacked Inception Modules

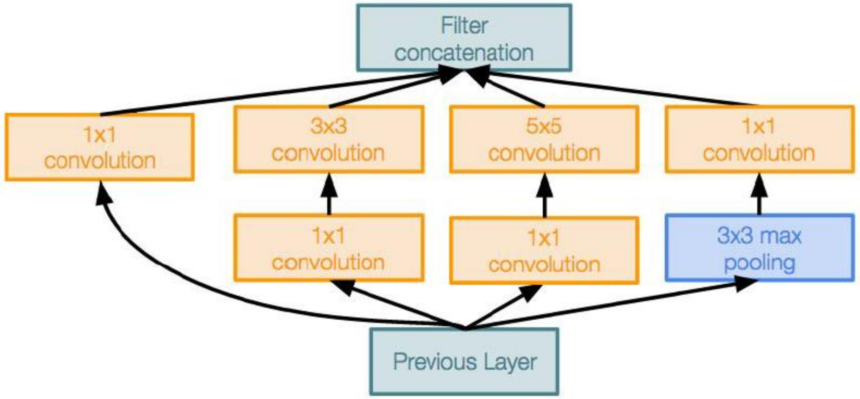
Case Study: GoogLeNet [Szegedy et al., 2014]



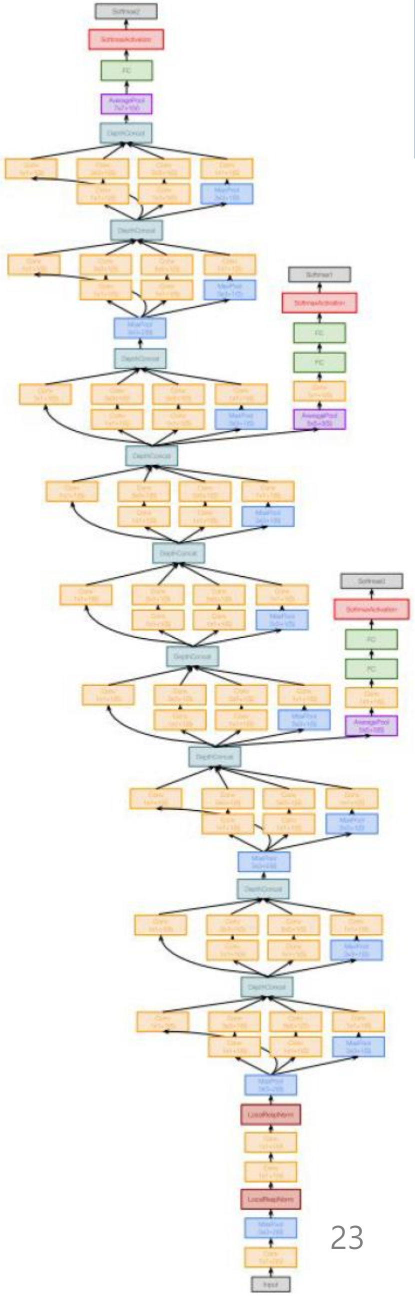
Case Study: GoogLeNet [Szegedy et al., 2014]

Deeper networks, with computational Efficiency

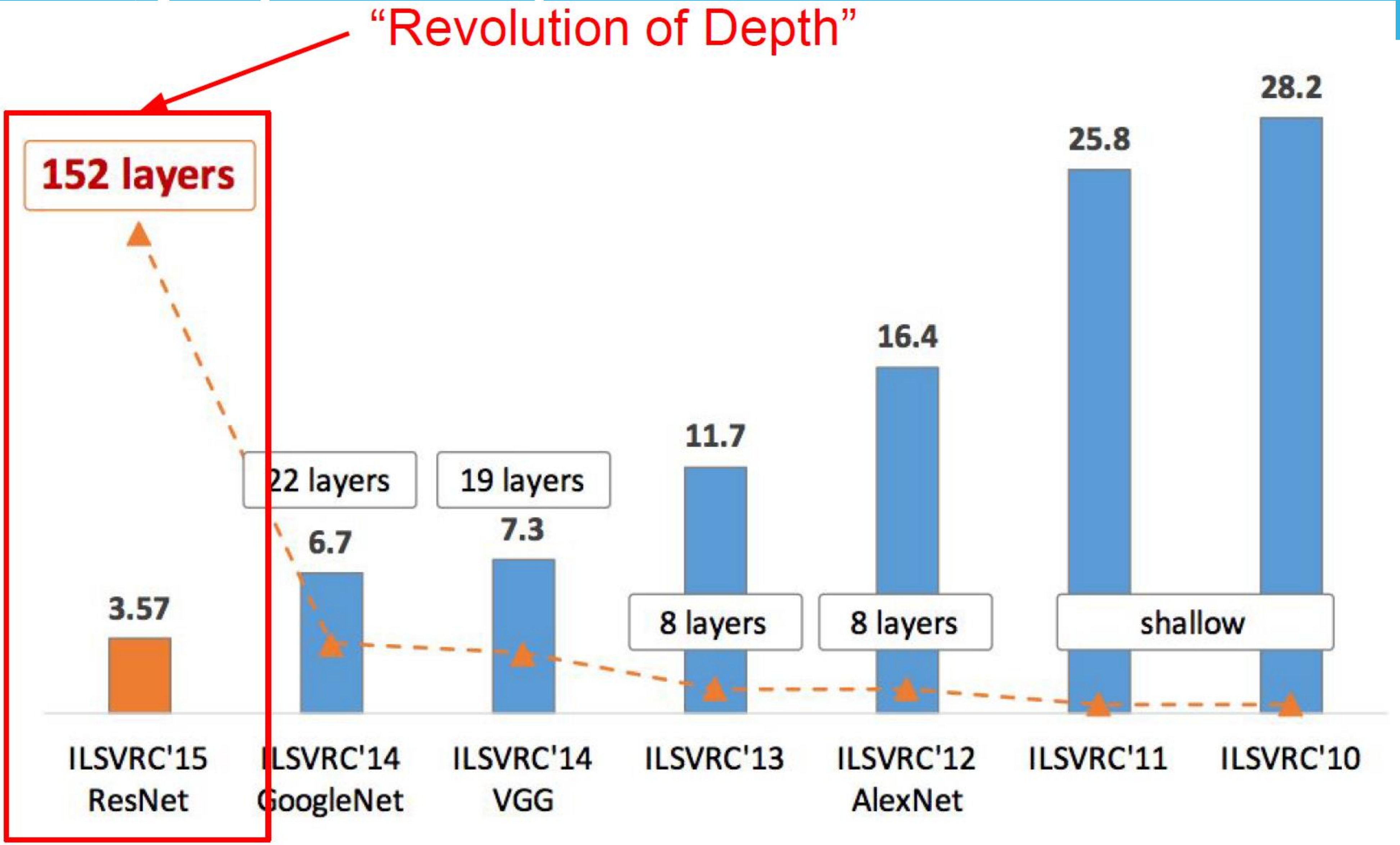
- 22 layers
- Efficient “Inception” module
- No FC layers
- Only 5 million parameters!
12x less than AlexNet
- ILSVRC’14 classification winner
(6.7% top 5 error)



Inception module



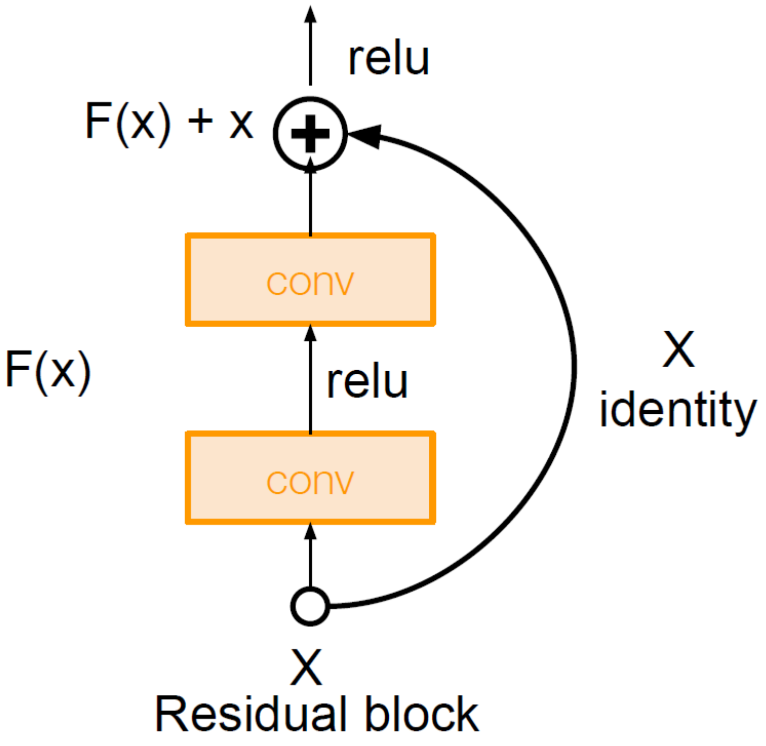
ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners

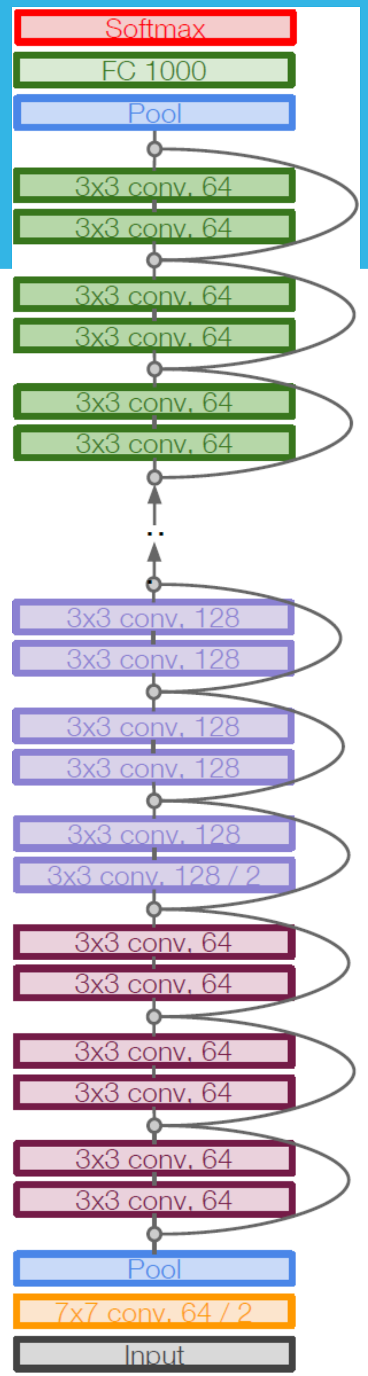


Case Study: ResNet [He et al., 2015]

Very deep networks using residual connections

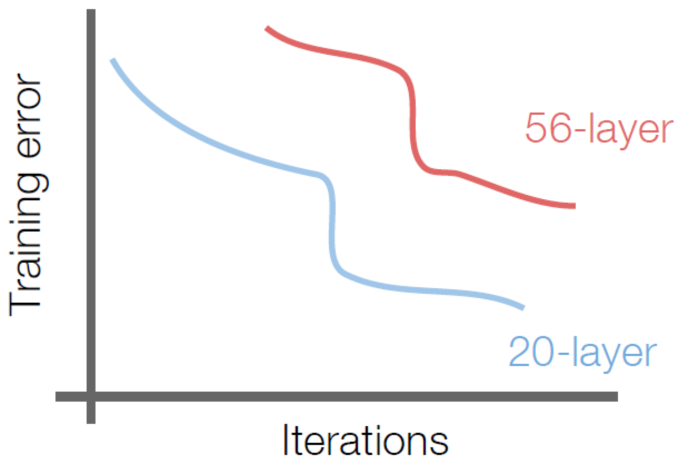
- 152-layer model for ImageNet
- ILSVRC'15 classification winner (3.57% top 5 error)
- Swept all classification and detection competitions in ILSVRC'15 and COCO'15!



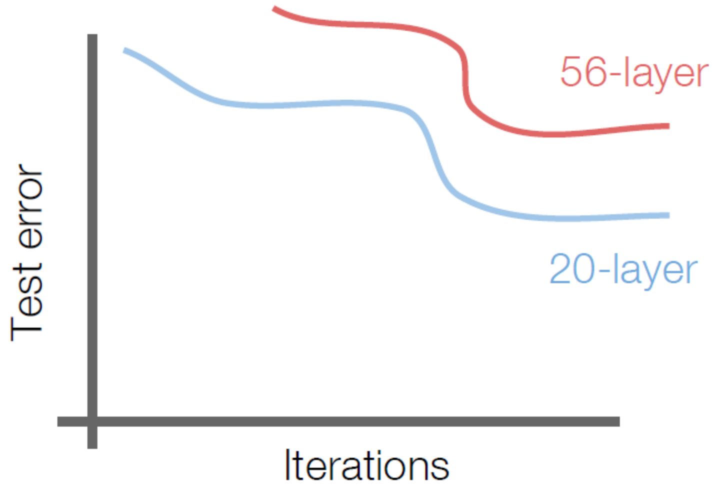


Case Study: ResNet [He et al., 2015]

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



확장



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

Case Study: ResNet [He et al., 2015]

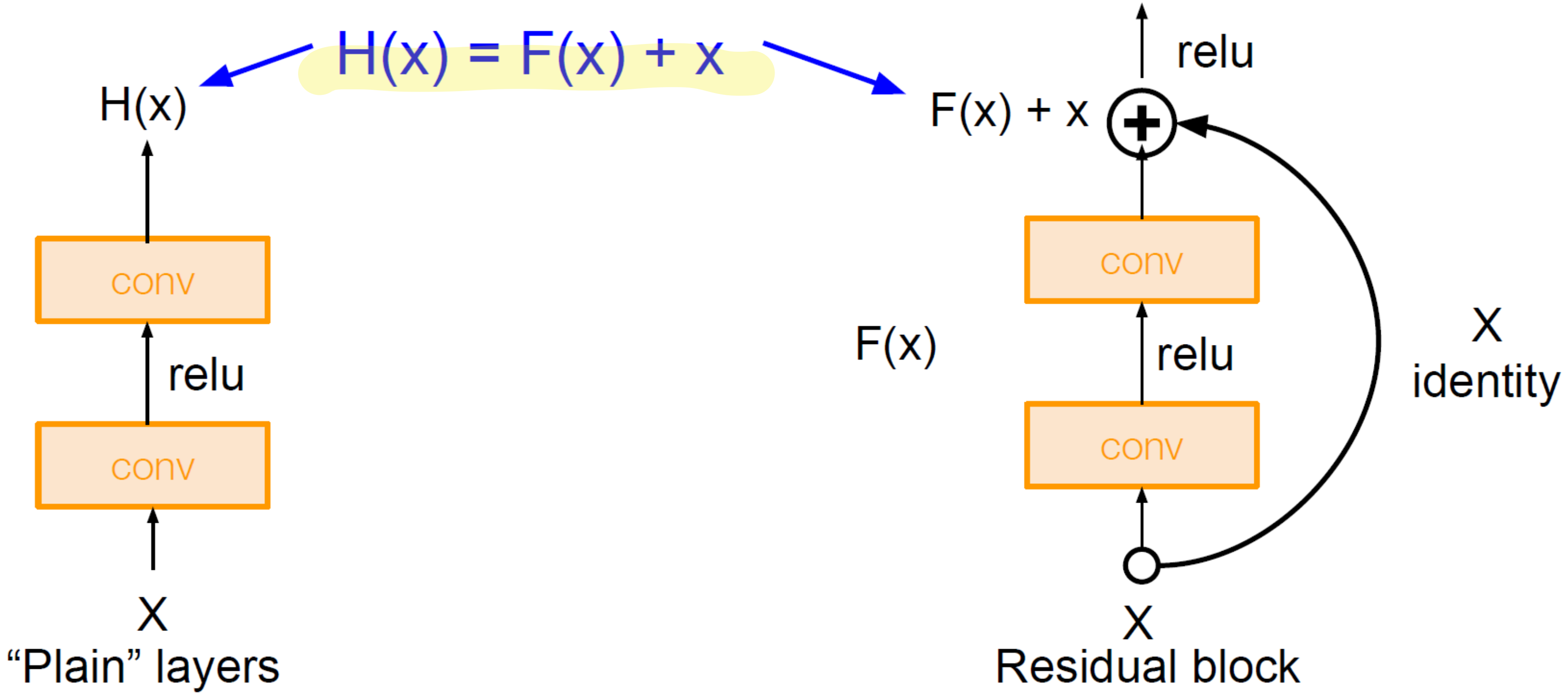
Hypothesis: the problem is an *optimization* problem, deeper models are harder to optimize

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.

Case Study: ResNet [He et al., 2015]

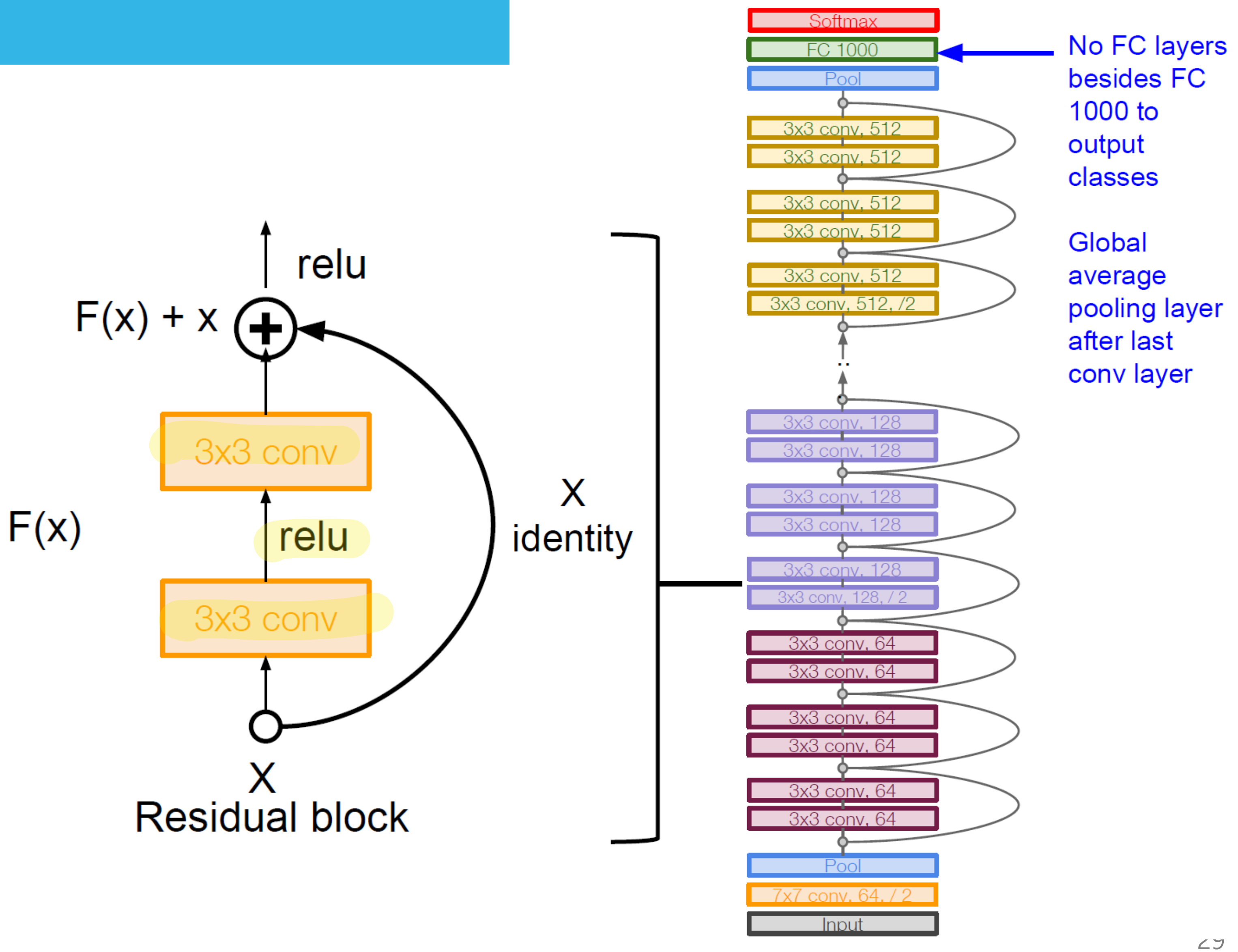
Solution: Use network layers to fit a residual mapping instead of directly trying to fit a desired underlying mapping



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

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)



Case Study: ResNet [He et al., 2015]

$h(x) = f(x) + x$

$\frac{dh}{dx} = f'(x) + 1$

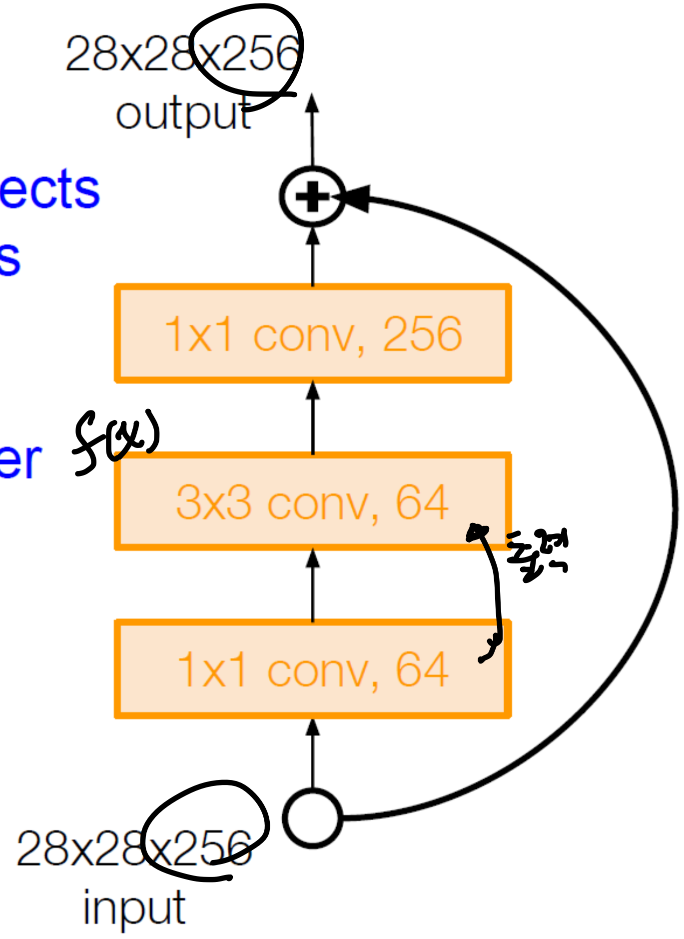
For deeper networks (ResNet-50+), use "bottleneck" layer to improve efficiency (similar to GoogLeNet)

$g'(h(x))$

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

3x3 conv operates over only 64 feature maps

1x1 conv, 64 filters to project to 28x28x64



Case Study: ResNet [He et al., 2015]

Experimental Results

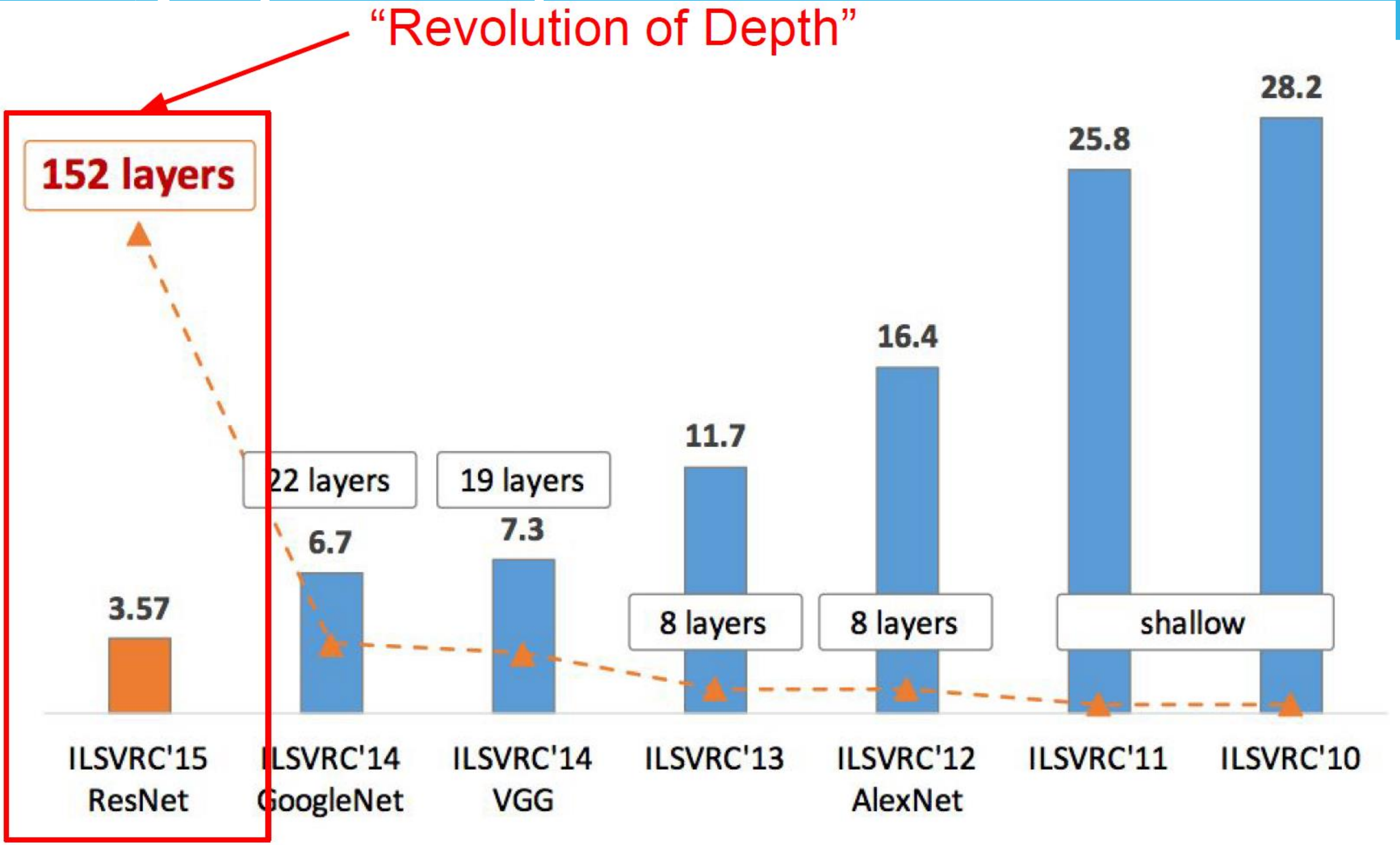
- Able to train very deep networks without degrading (152 layers on ImageNet, 1202 on Cifar)
- Deeper networks now achieve lowing training error as expected
- Swept 1st place in all ILSVRC and COCO 2015 competitions

MSRA @ ILSVRC & COCO 2015 Competitions

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

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

ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners

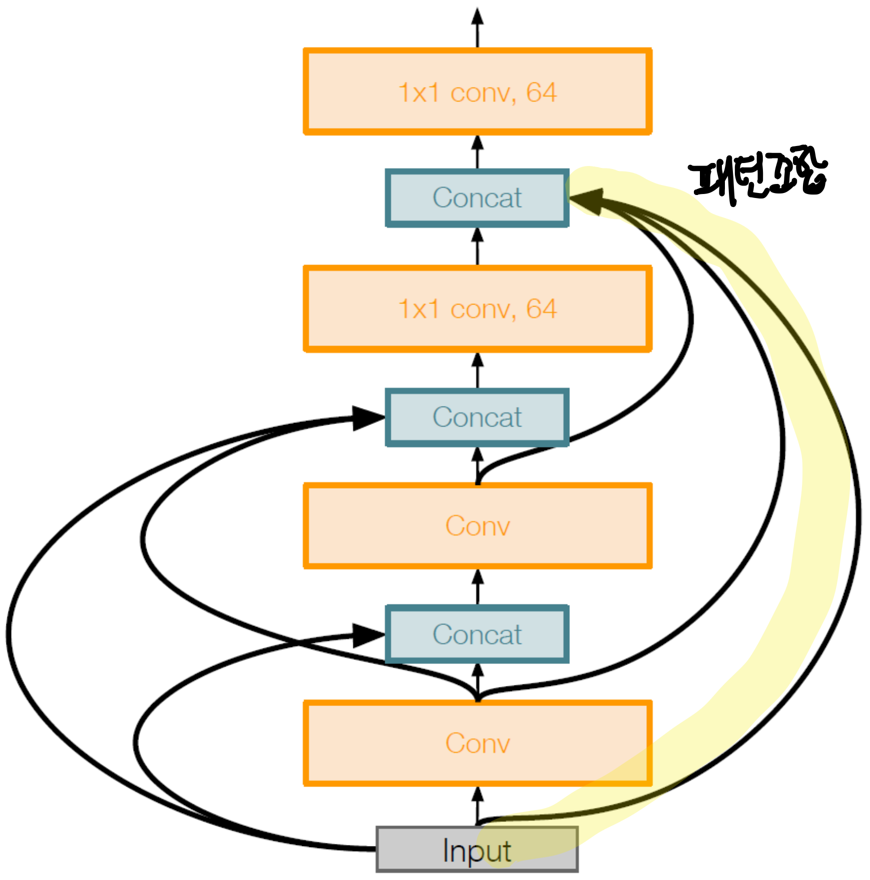


Beyond ResNets

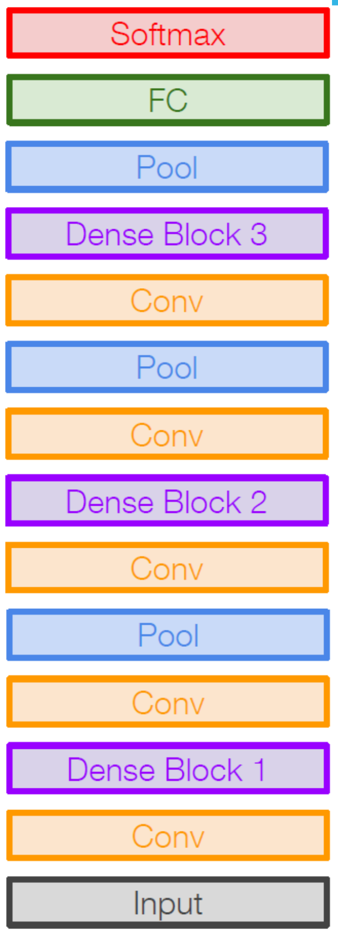
Densely Connected Convolutional Networks

[Huang et al. 2017]

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



Dense Block



Summary: CNN Architectures

- VGG, GoogLeNet, ResNet all in wide use, available in model zoos

- ResNet current best default + DenseNet

- Trend towards extremely deep networks

- Significant research centers around design of layer / skip connections and improving gradient flow

- Even more recent trend towards examining necessity of depth vs. width and residual connections

References

Convolutional Neural Network

Bumsoo Kim, Computer Vision & Deep Learning pdf

https://brohrer.github.io/how_convolutional_neural_networks_work.html

http://cs231n.stanford.edu/syllabus.html

Convolutional Neural Network for Text Classification

http://cs224d.stanford.edu/syllabus.html

http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp/

https://arxiv.org/abs/1510.03820

https://arxiv.org/abs/1408.5882

Q&A

THANK YOU!