Idea Factory Intensive Program #2

# 딥러닝 홀로서기

# #23

**이론강의/PyTorch실습/코드리뷰**

딥러닝(Deep Learning)에 관심이 있는 학생 발굴을 통한
딥러닝의 이론적 배경 강의 및 오픈소스 딥러닝 라이브러리 PyTorch를 활용한 실습

Topics to learn today

# 1. Review from last lecture

Assignment: CIFAR-10 classification with CNN

Lecture: Basic of Convolutional Neural Network

# 2. Advanced Architectures of CNN

AlexNet, VGGNet, GoogleNet, ResNet

# 3. Implementing ResNet with Pytorch

# Hierarchical organization

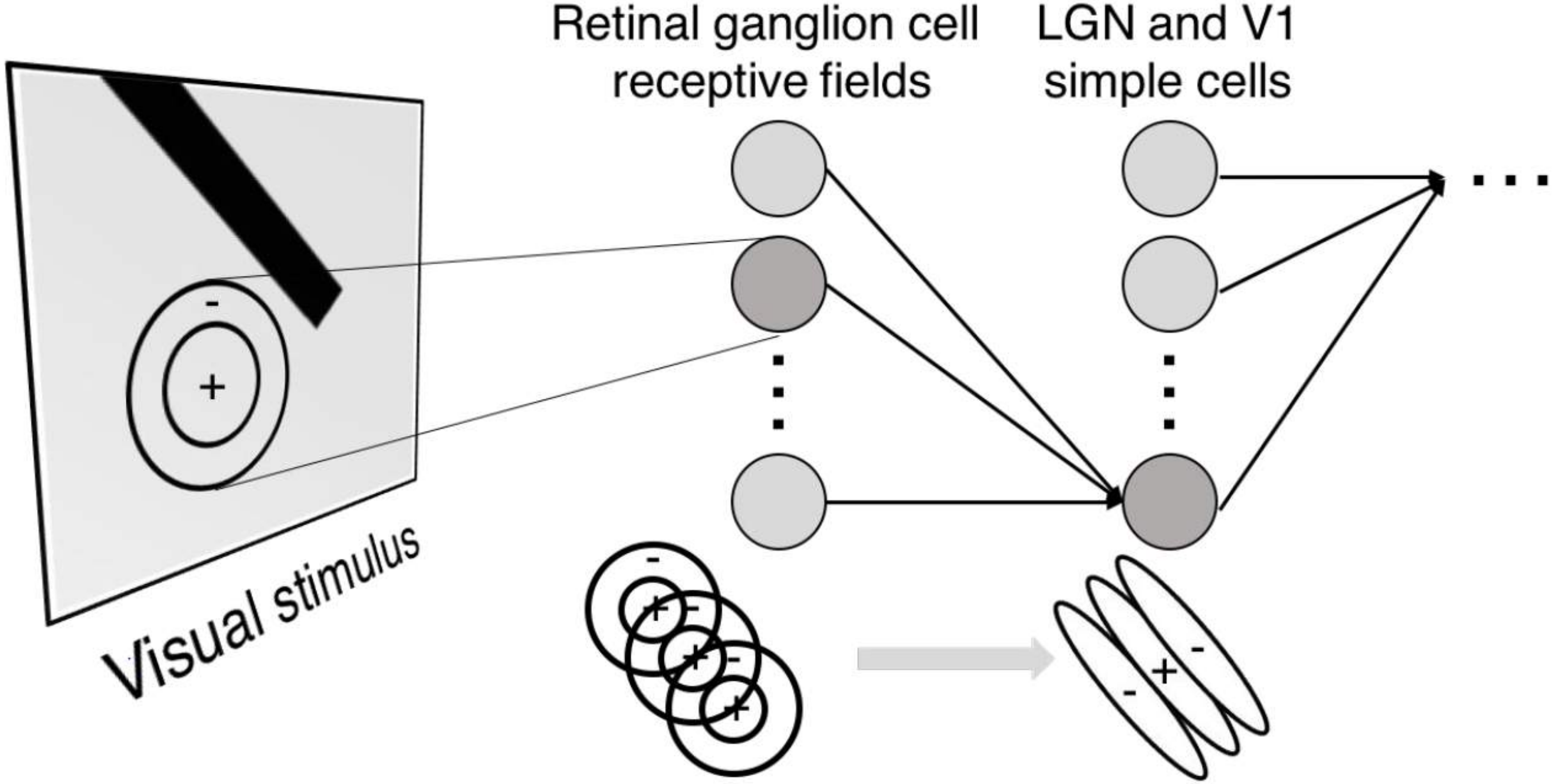
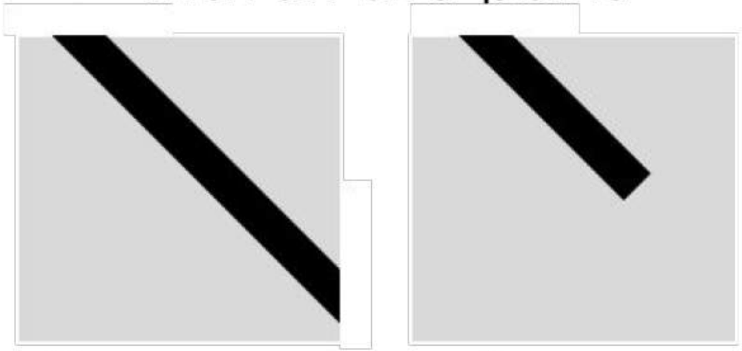



Illustration of hierarchical organization in early visual pathways by Lane McIntosh, copyright CS231n 2017

**Simple cells:**
Response to light orientation

**Complex cells:**
Response to light orientation and movement

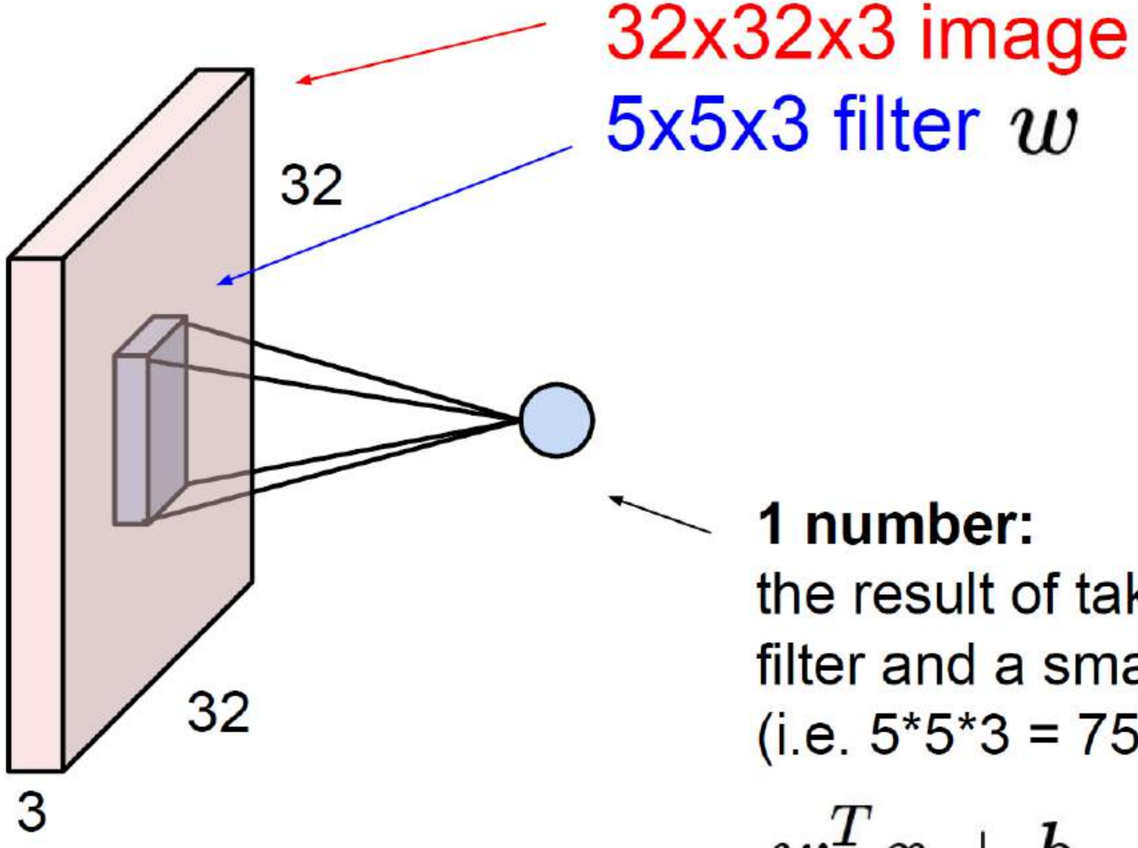
**Hypercomplex cells:**
response to movement with an end point



No response

Response (end point)

# Convolution Layer
: Preserve the spatial structure

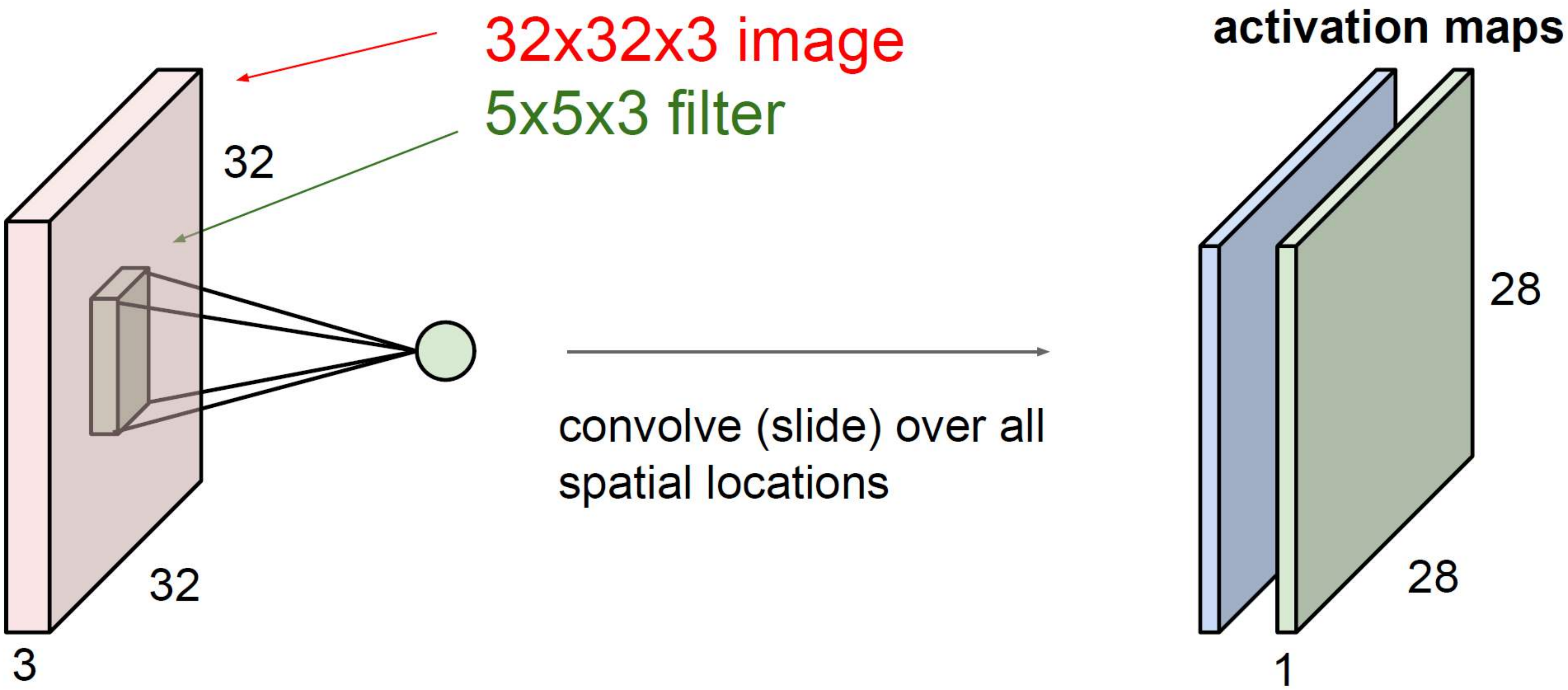



**1 number:**
the result of taking a dot product between the filter and a small 5x5x3 chunk of the image
(i.e. 5*5*3 = 75-dimensional dot product + bias)

$$w^T x + b$$

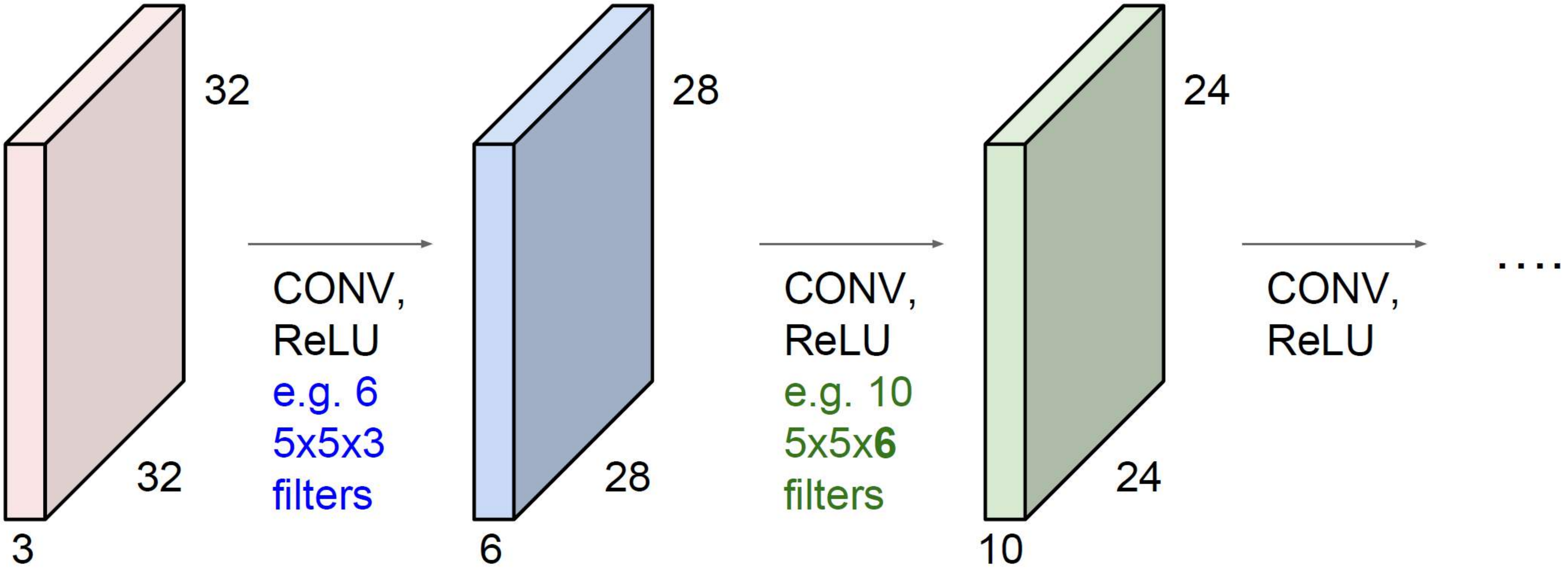
Stanford, cs231n 2017

# Convolution Layer
: Preserve the spatial structure

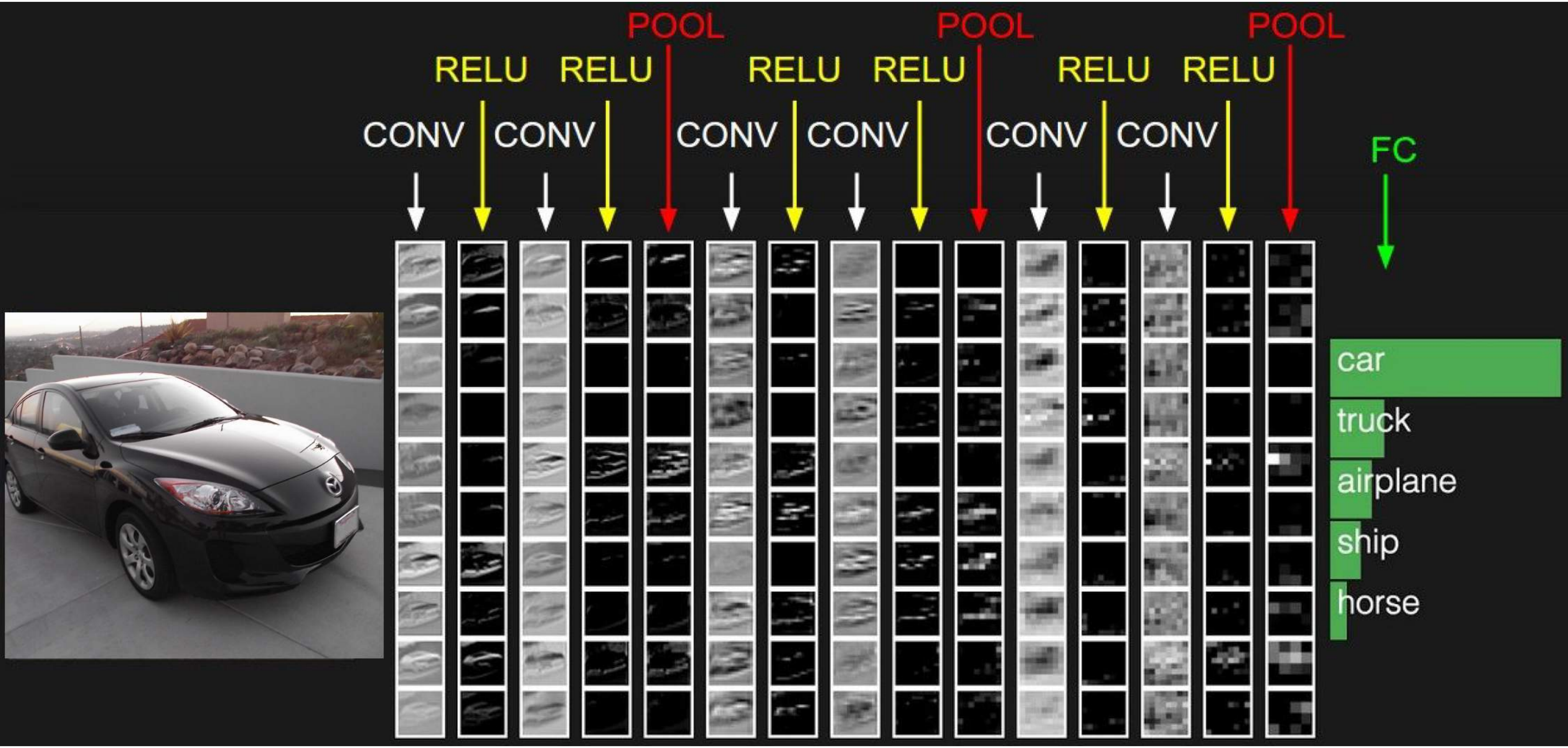



Stanford, cs231n 2017

# Convolutional Net
: Sequence of Convolutional Layers, interspersed with activation functions




Stanford, cs231n 2017




Stanford, cs231n 2017

# Advanced CNN Architectrues
# AlexNet

# ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners

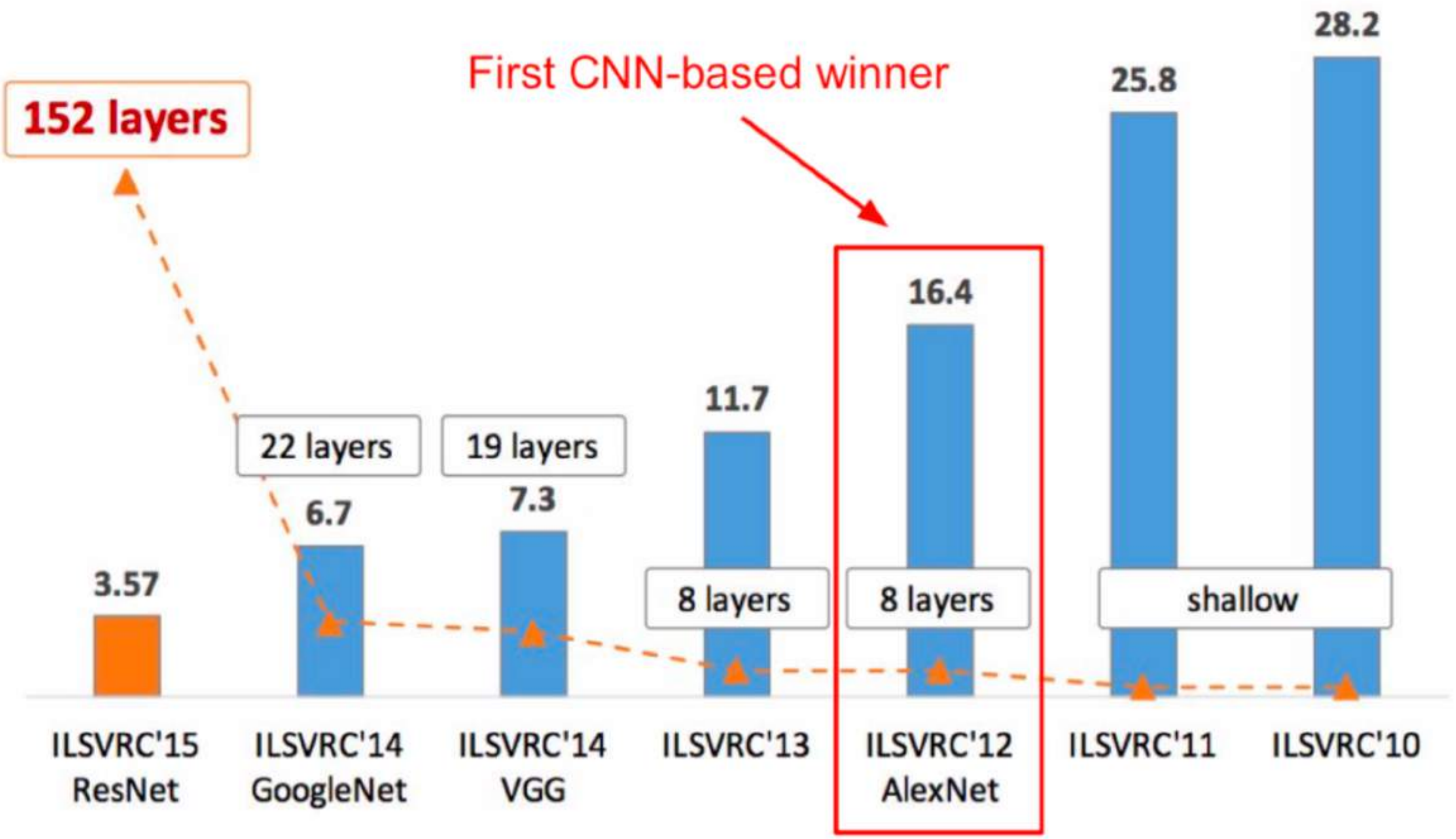
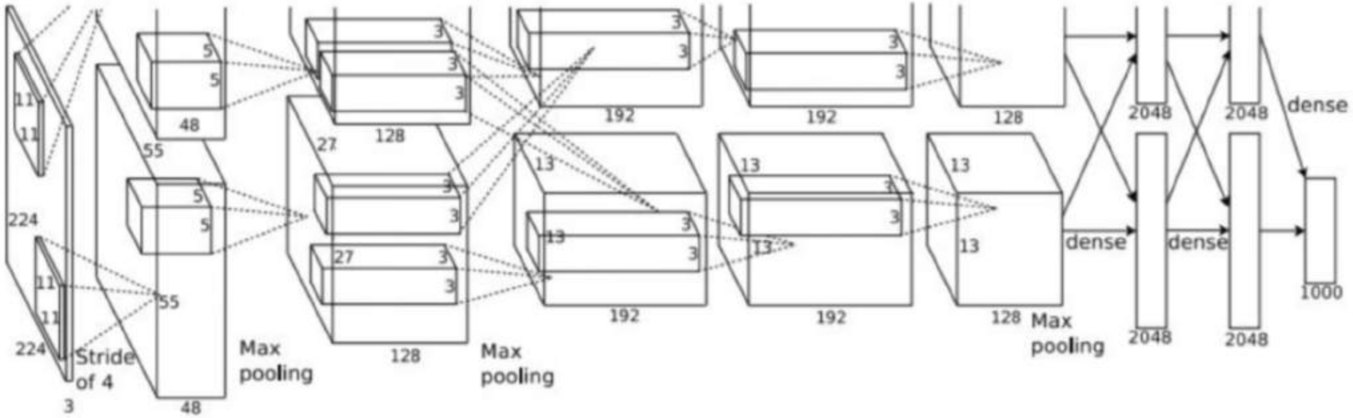



Figure copyright Kaiming He, 2016. Reproduced with permission.

# 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%

# Advanced CNN Architectrues
# VGGNet

# ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners

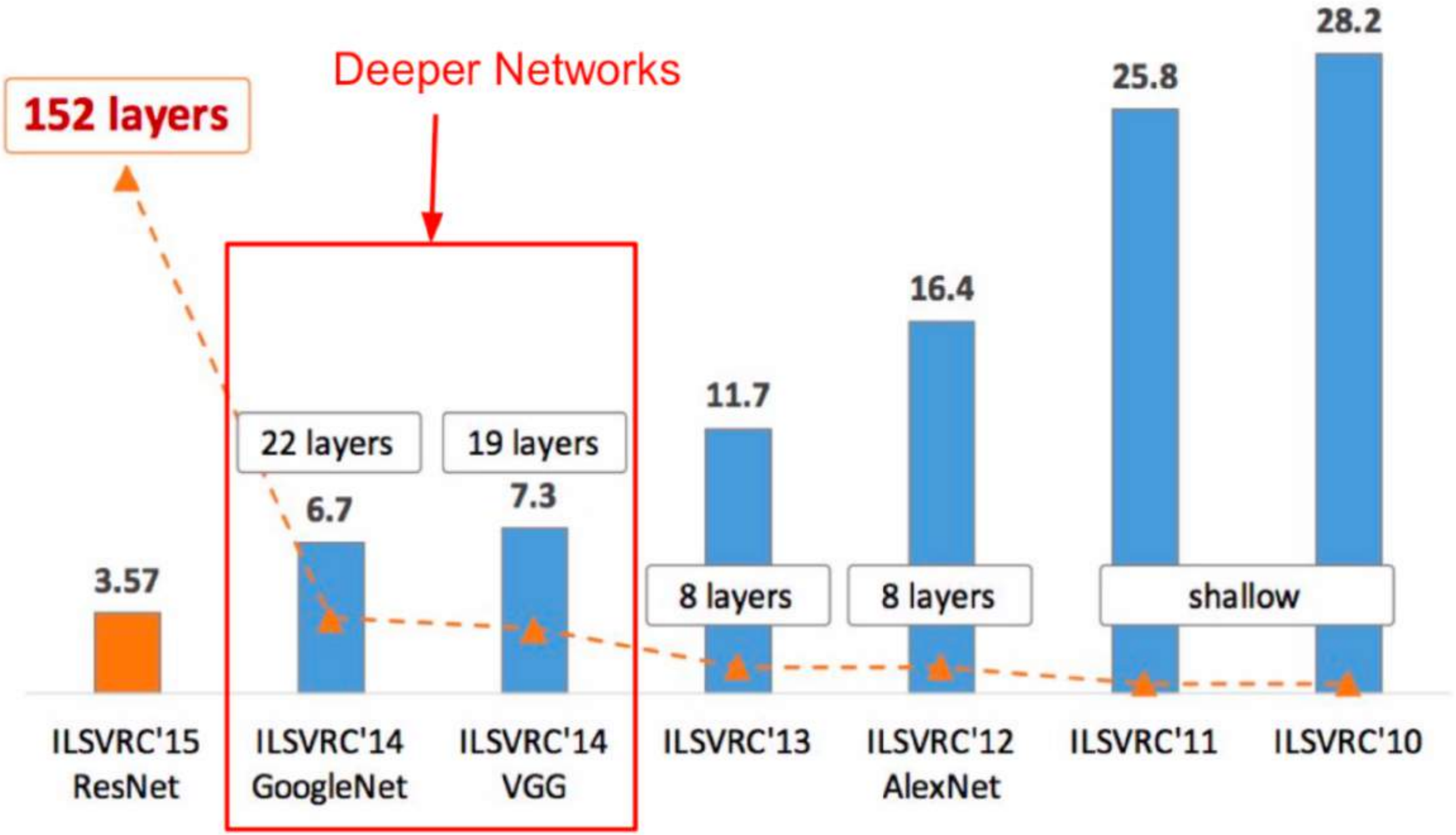



Figure copyright Kaiming He, 2016. Reproduced with permission.

# Case Study: VGGNet

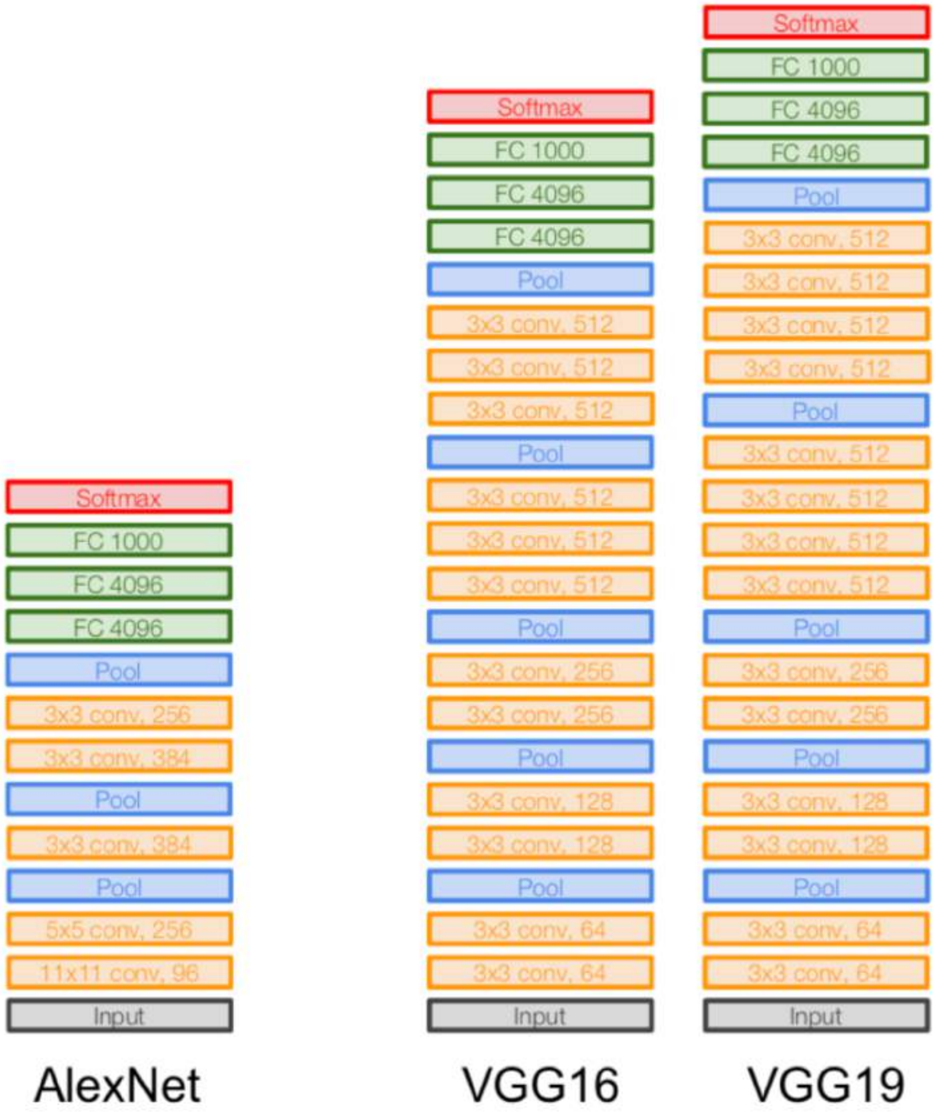
*[Simonyan and Zisserman, 2014]*

Small filters, Deeper networks

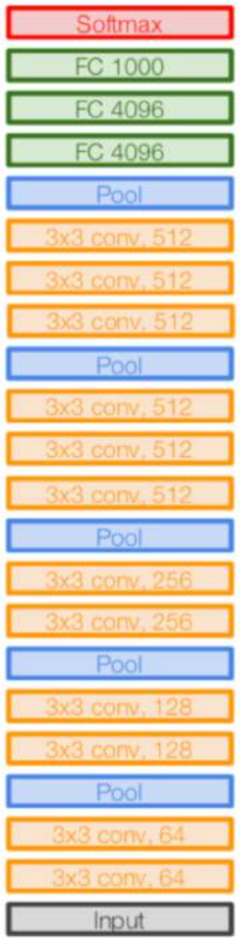
8 layers (AlexNet)
-> 16 - 19 layers (VGG16Net)

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



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



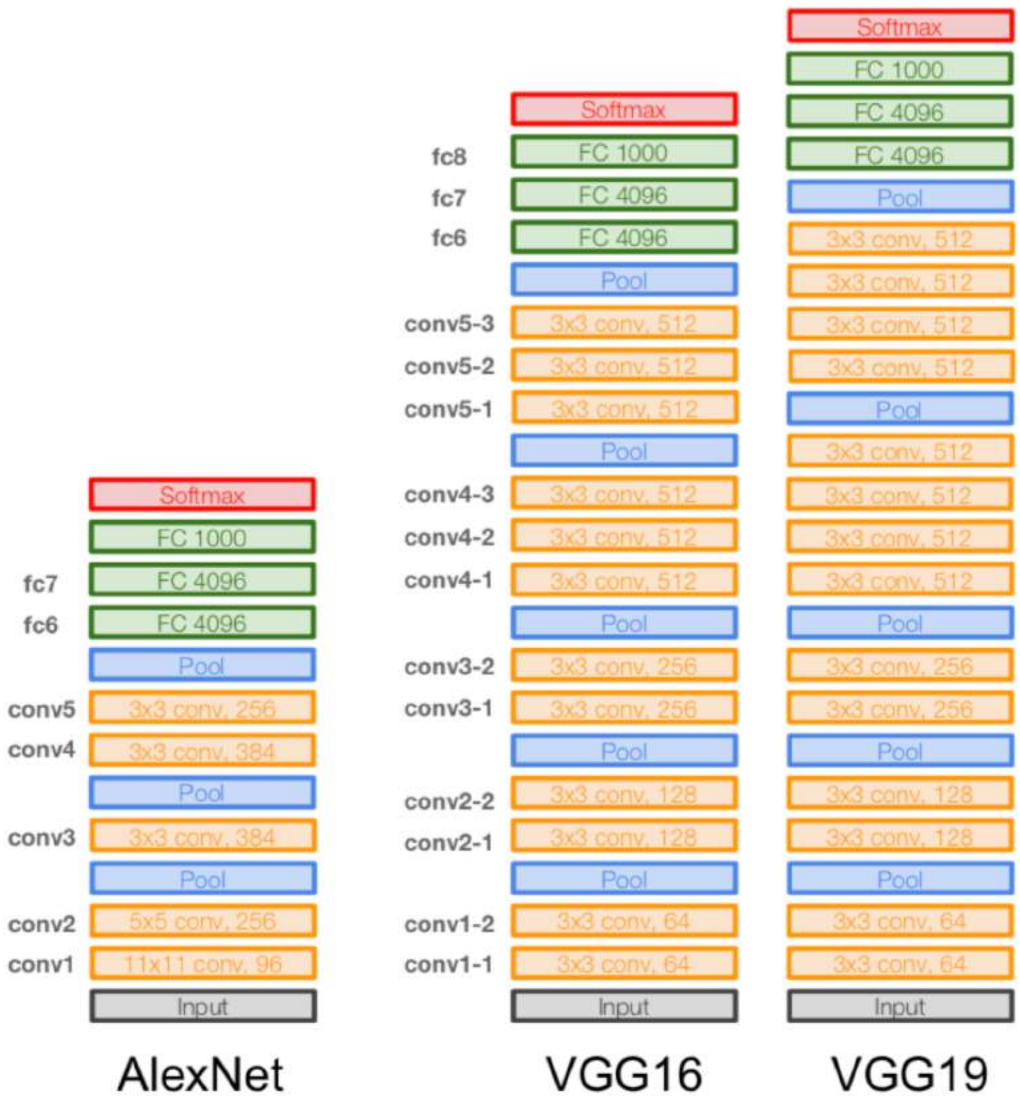

VGG16

# Case Study: VGGNet

*[Simonyan and Zisserman, 2014]*

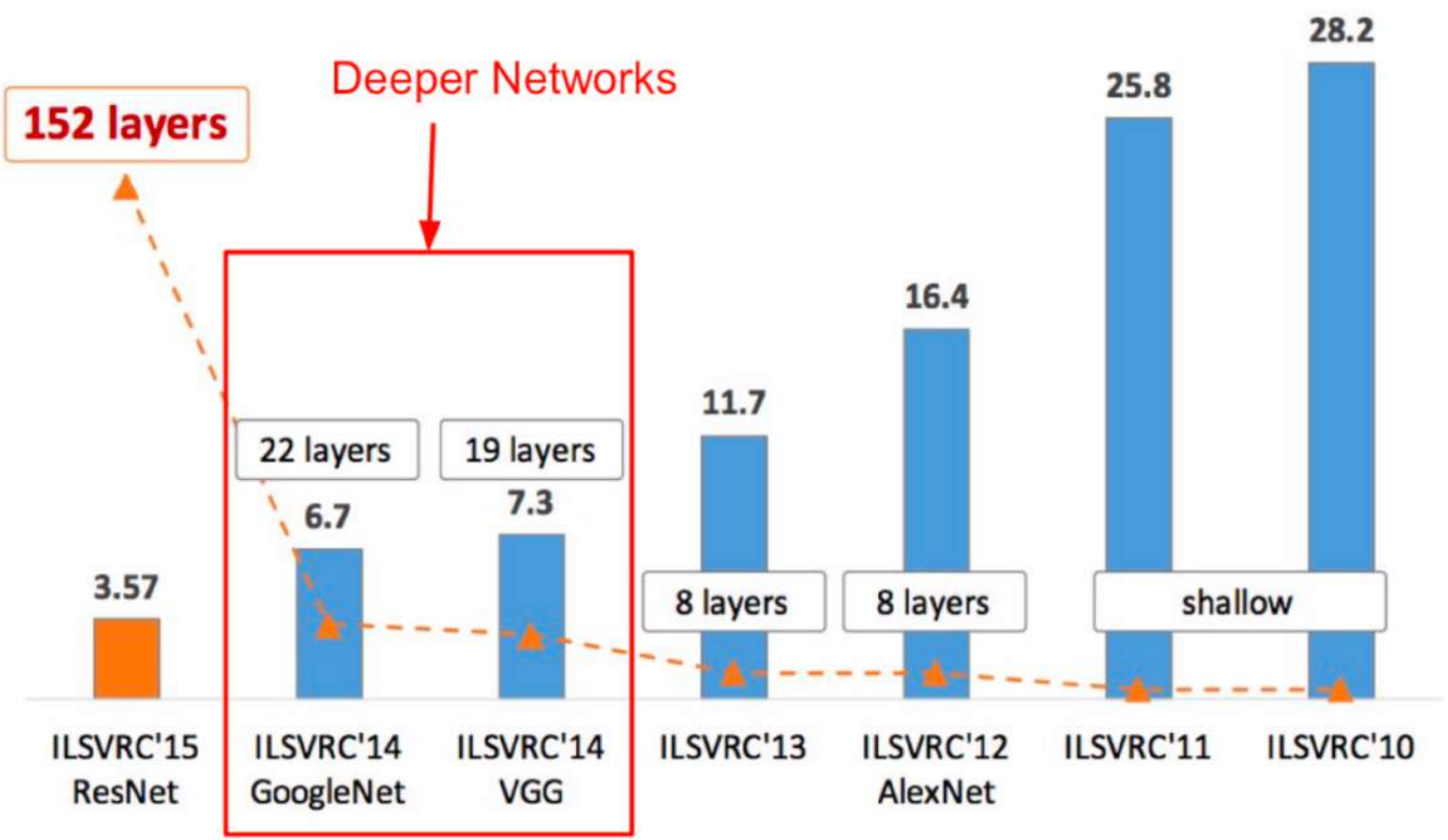
Details:

- 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



# Advanced CNN Architectrues
# GoogleNet

# ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners

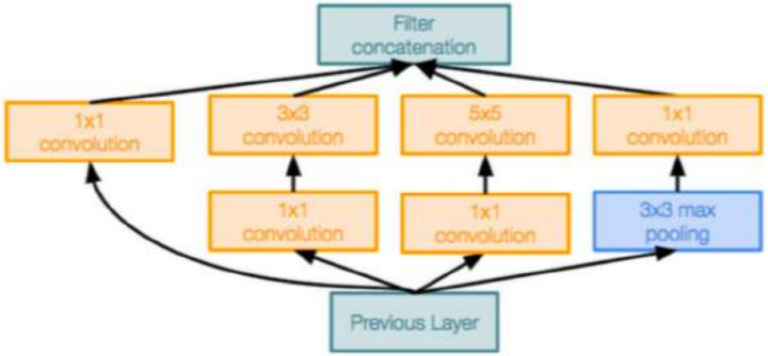


# Case Study: GoogLeNet

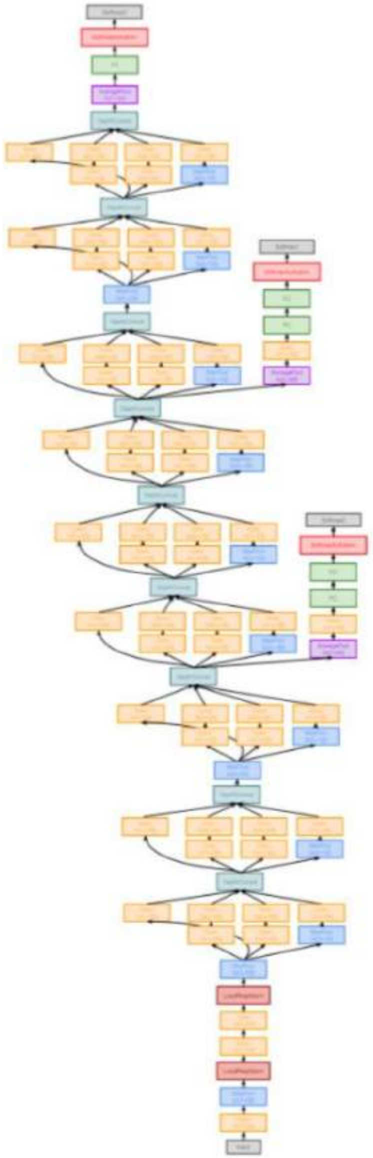
*[Szegedy et al., 2014]*

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)

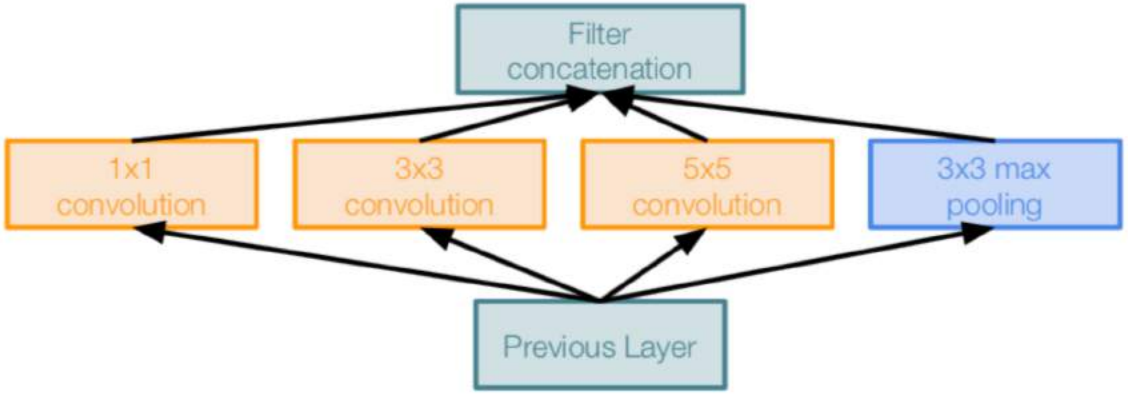



Inception module



# Case Study: GoogLeNet

*[Szegedy et al., 2014]*

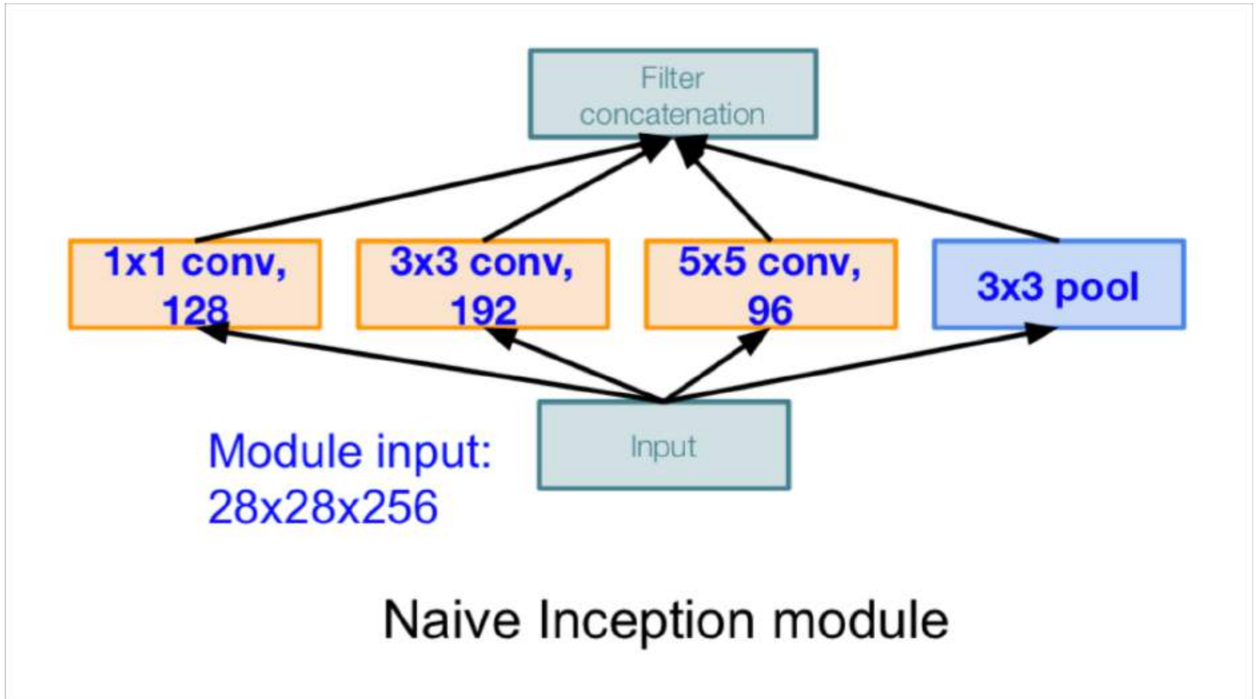



Naive Inception module

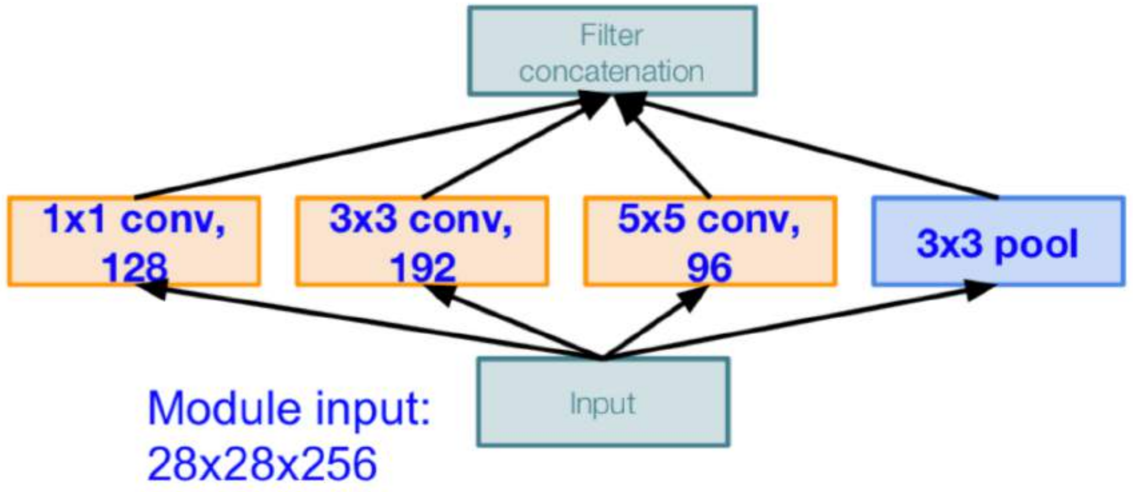
Apply parallel filter operations on the input from previous layer:
- Multiple receptive field sizes for convolution (1x1, 3x3, 5x5)
- Pooling operation (3x3)

Concatenate all filter outputs together depth-wise

# Problem of Naïve Inception Module



# Problem of Naïve Inception Module



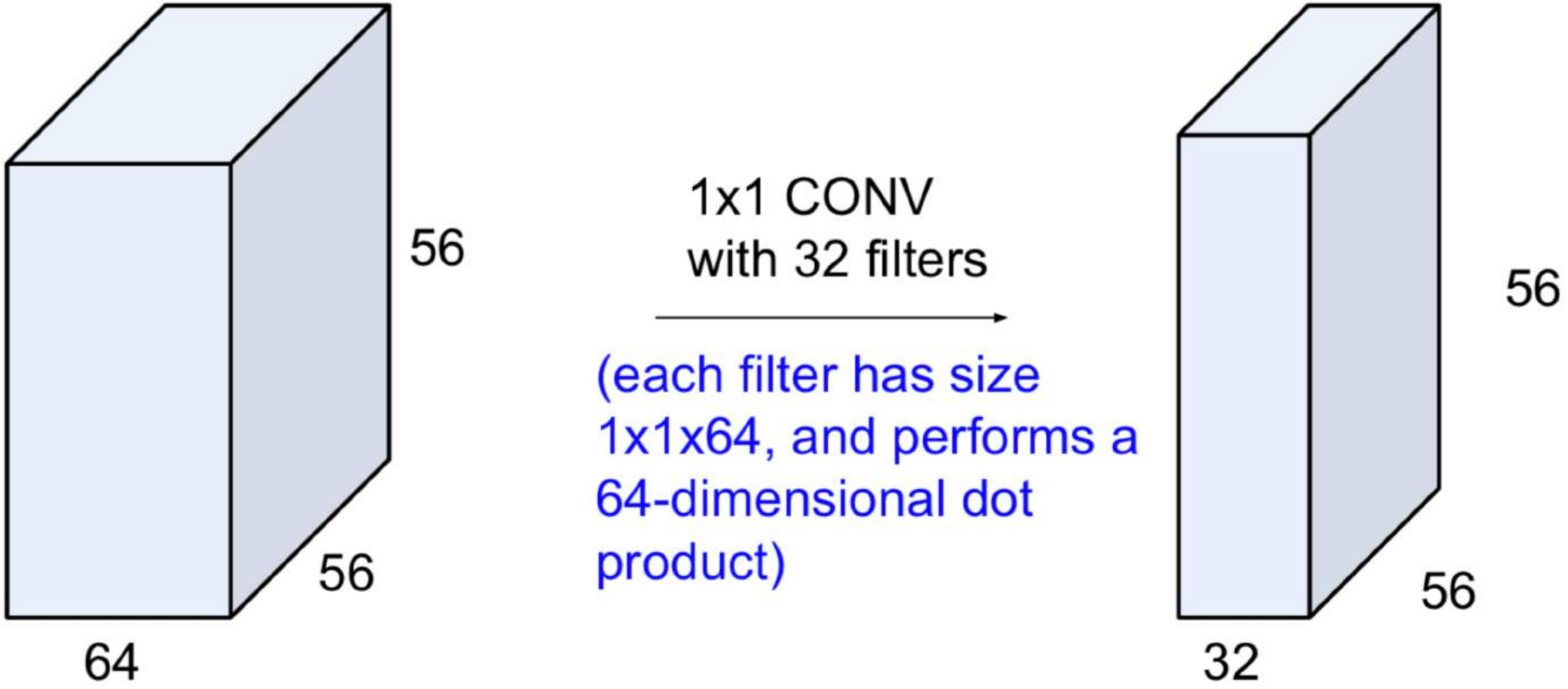
Naive Inception module

**Conv Ops:**
[1x1 conv, 128]  28x28x128x1x1x256
[3x3 conv, 192]  28x28x192x3x3x256
[5x5 conv, 96]  28x28x96x5x5x256
**Total: 854M ops**

Very expensive compute

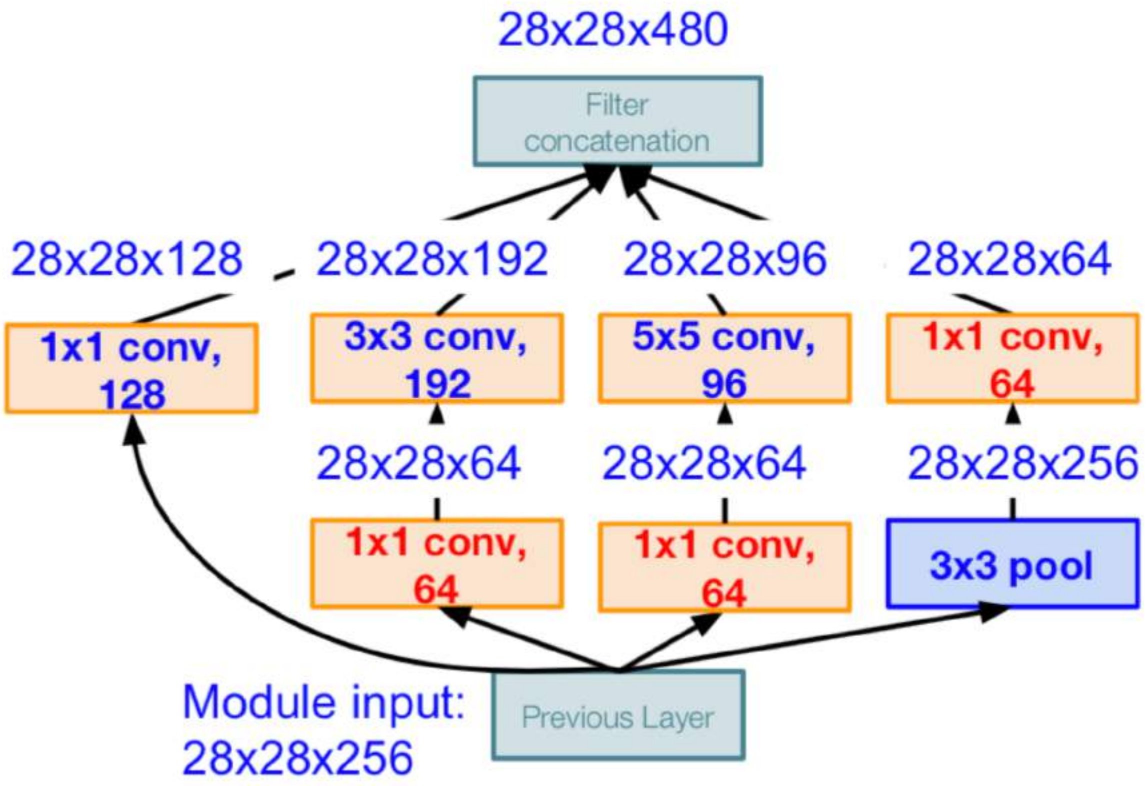
Pooling layer also preserves feature depth, which means total depth after concatenation can only grow at every layer!

# Reminder: 1x1 convolutions



# Case Study: GoogLeNet

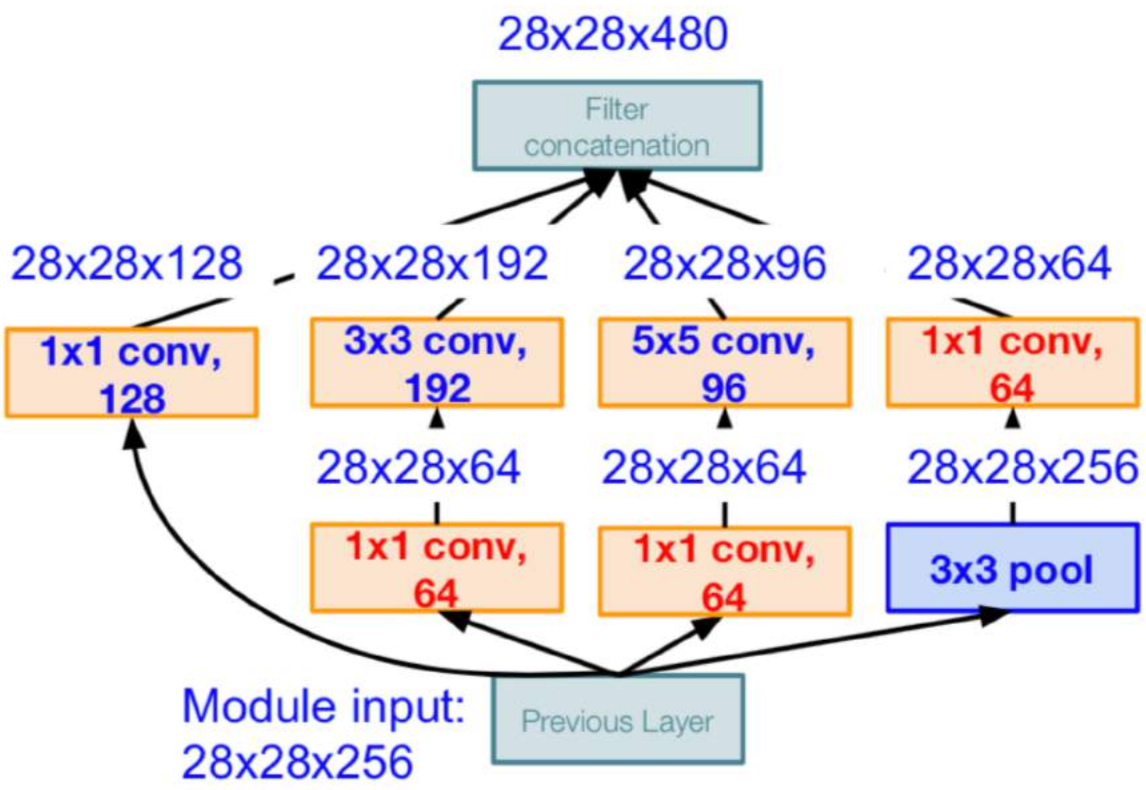
*[Szegedy et al., 2014]*




Inception module with dimension reduction

# 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:**

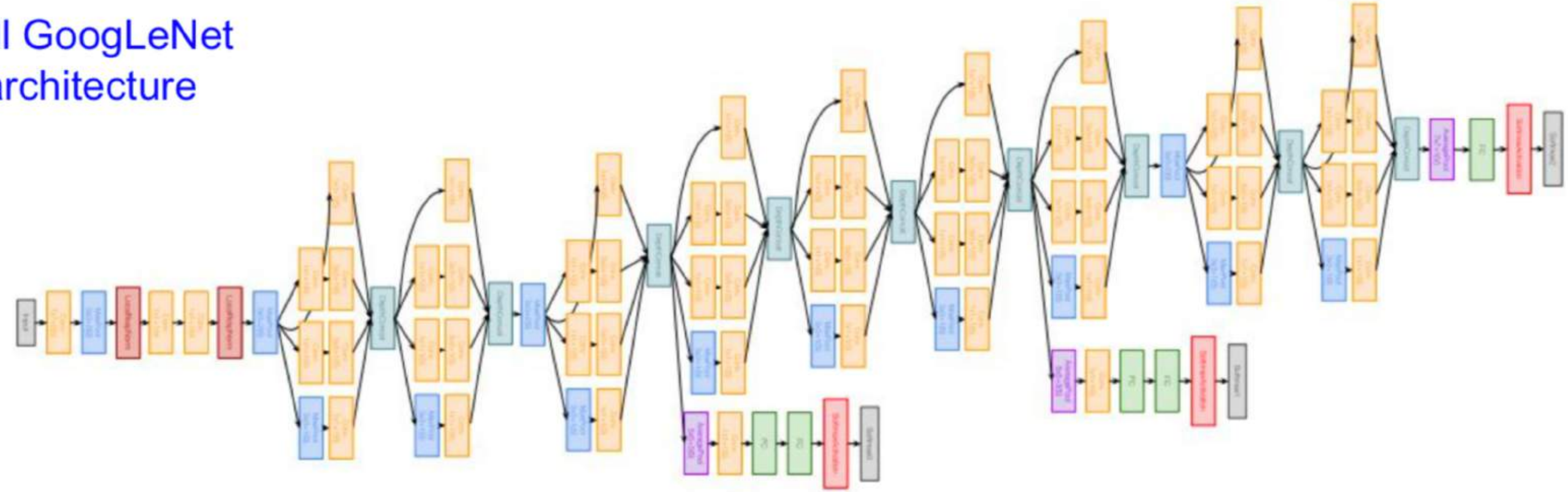
[1x1 conv, 64] 28x28x64x1x1x256
[1x1 conv, 64] 28x28x64x1x1x256
[1x1 conv, 128] 28x28x128x1x1x256
[3x3 conv, 192] 28x28x192x3x3x64
[5x5 conv, 96] 28x28x96x5x5x64
[1x1 conv, 64] 28x28x64x1x1x256
**Total: 358M ops**

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

# Case Study: GoogLeNet

*[Szegedy et al., 2014]*

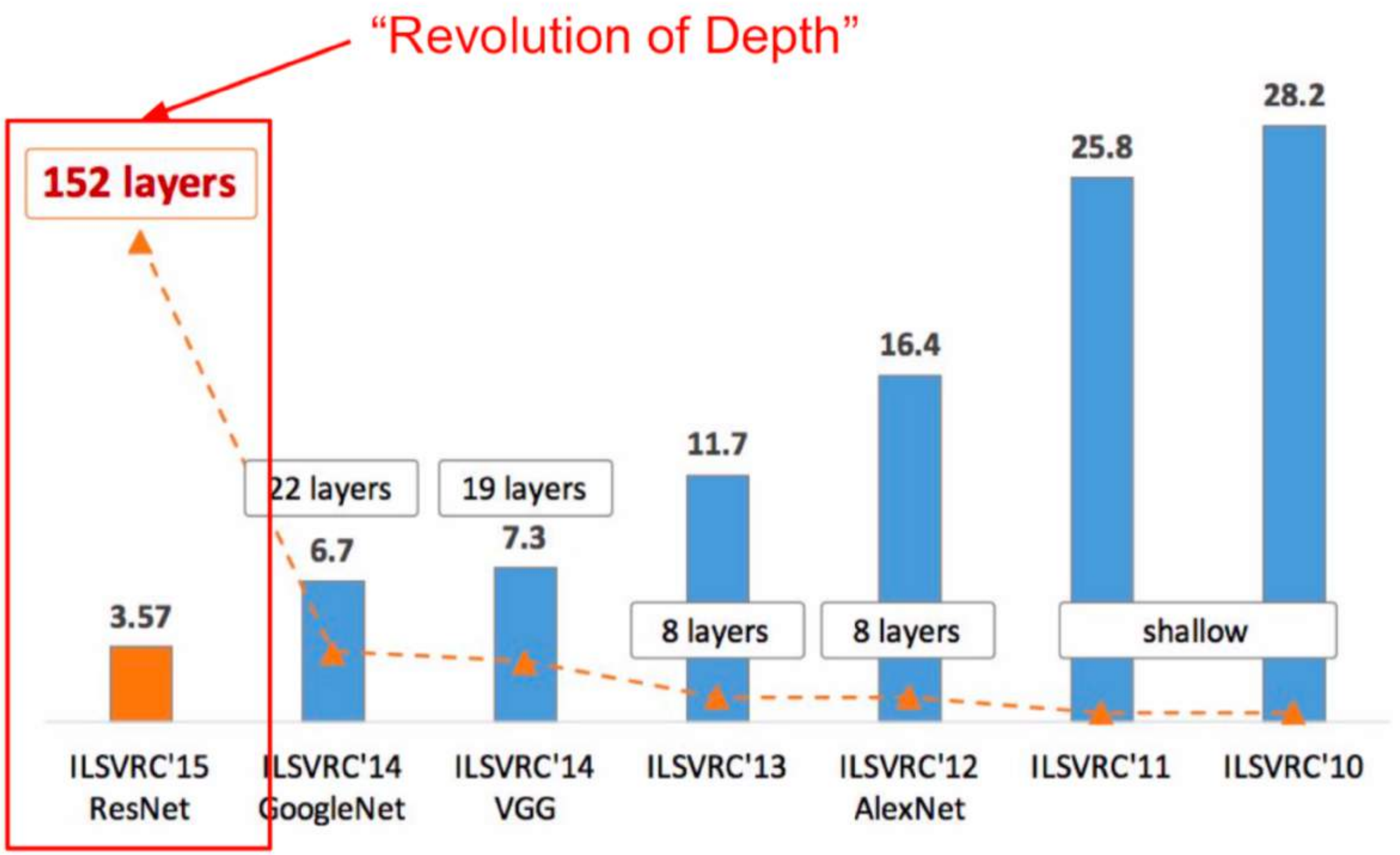
Full GoogLeNet architecture



22 total layers with weights (including each parallel layer in an Inception module)

# Advanced CNN Architectrues
# ResNet

# ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners

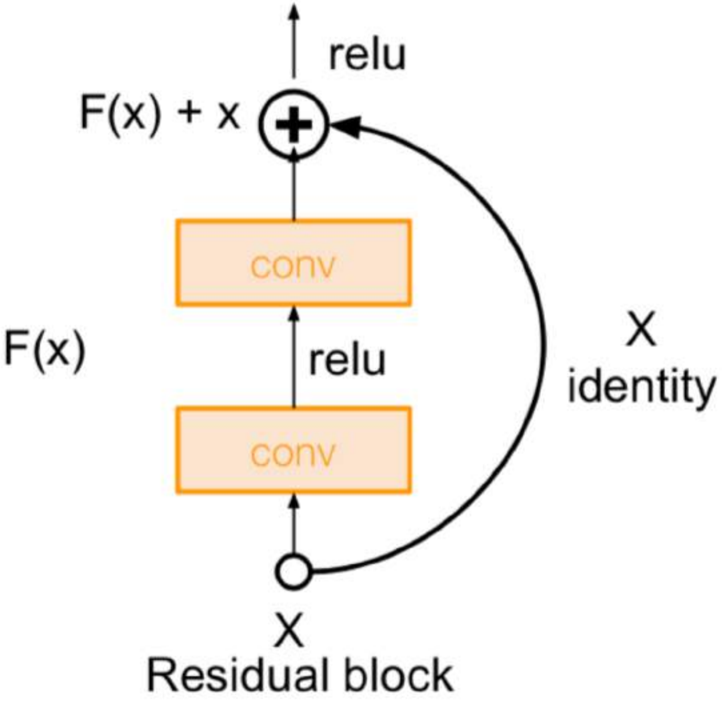


# Case Study: ResNet

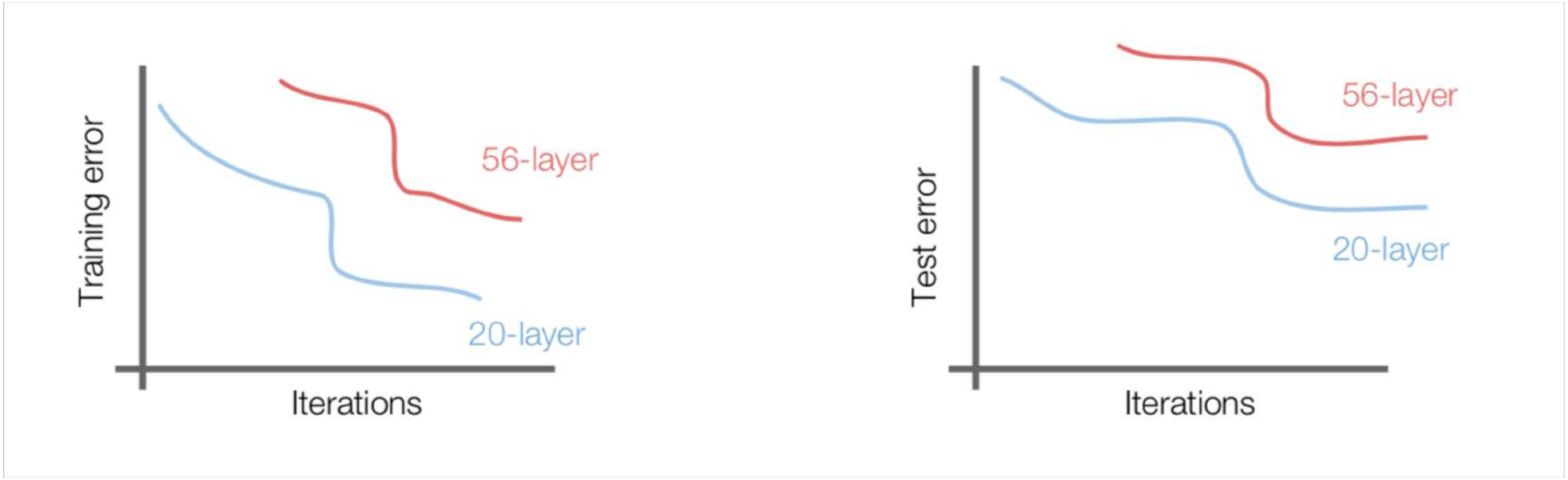
*[He et al., 2015]*

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!




# Problem of Deep CNN

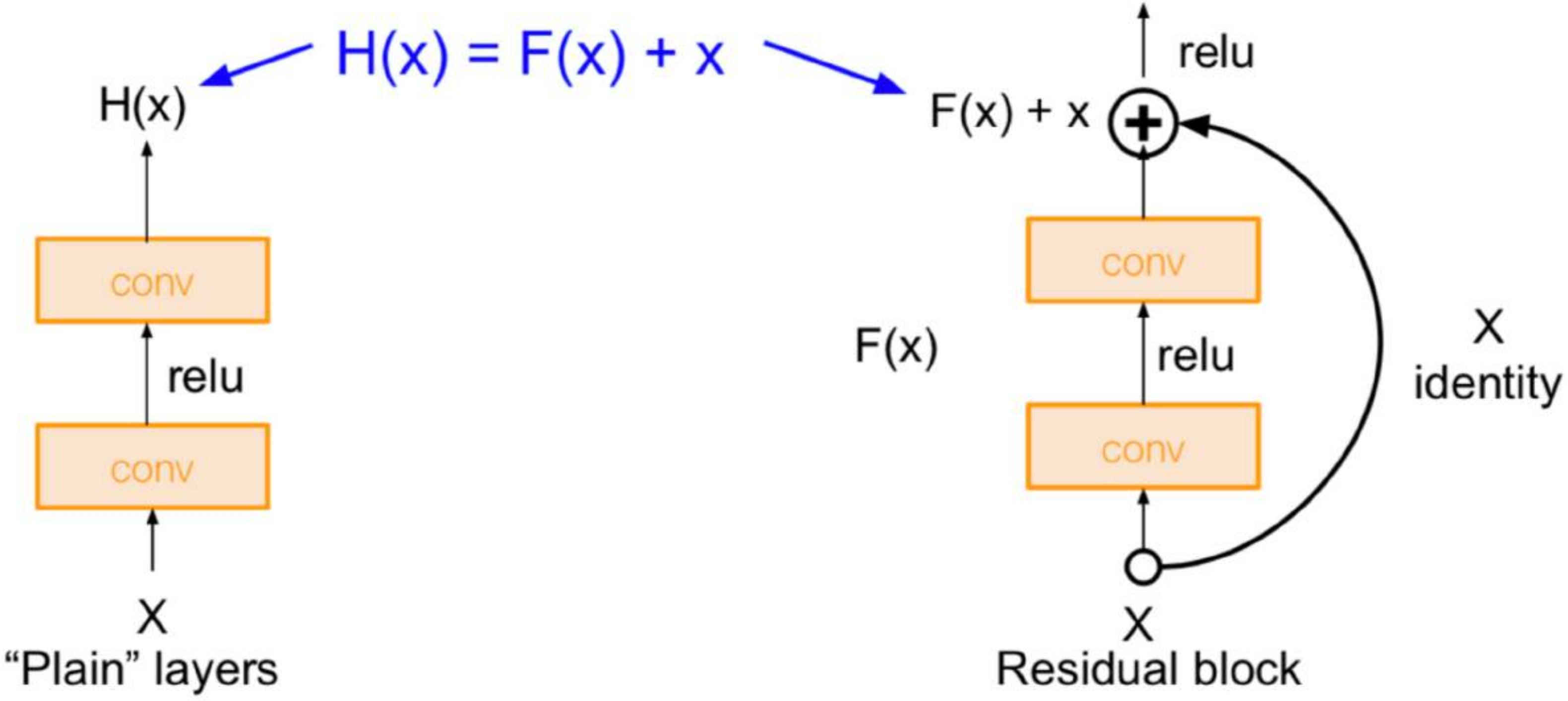



Stanford, cs231n 2017

# 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

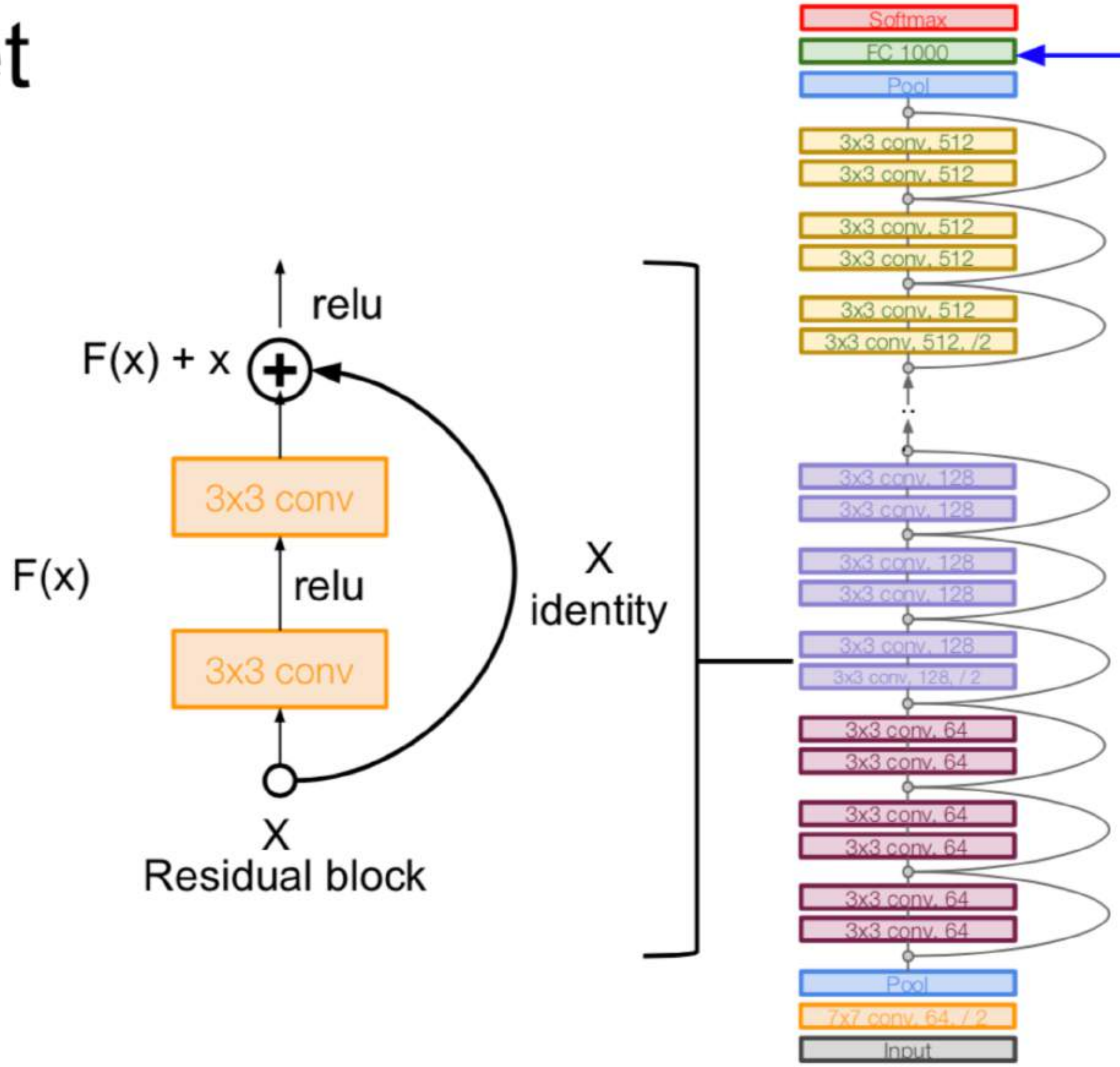



Use layers to fit residual $F(x) = H(x) - x$ instead of $H(x)$ directly

# 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)




No FC layers besides FC 1000 to output classes

Global average pooling layer after last conv layer

# Case Study: ResNet

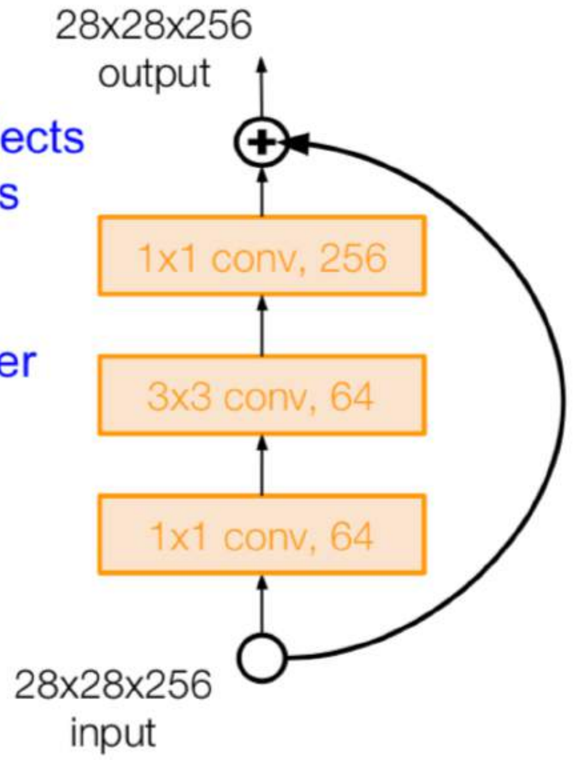
*[He et al., 2015]*

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

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

3x3 conv operates over only 64 feature maps

1x1 conv, 64 filters to project to 28x28x64



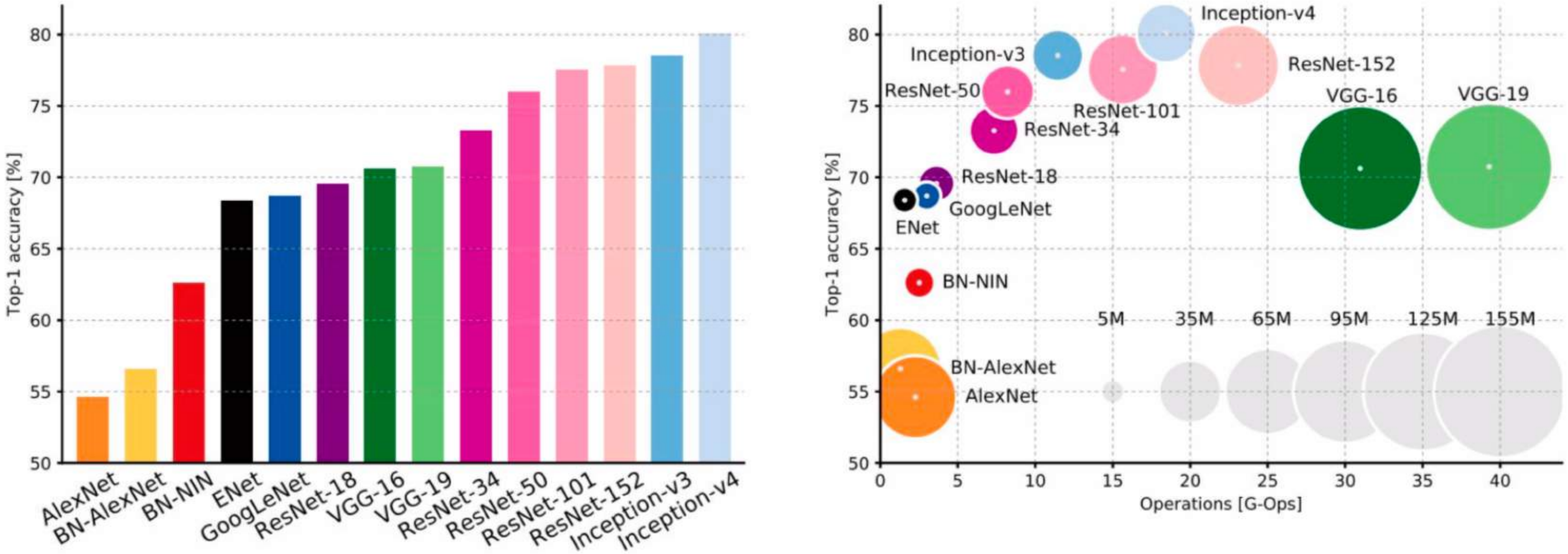
# Case Study: ResNet

*[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

# Comparing complexity...



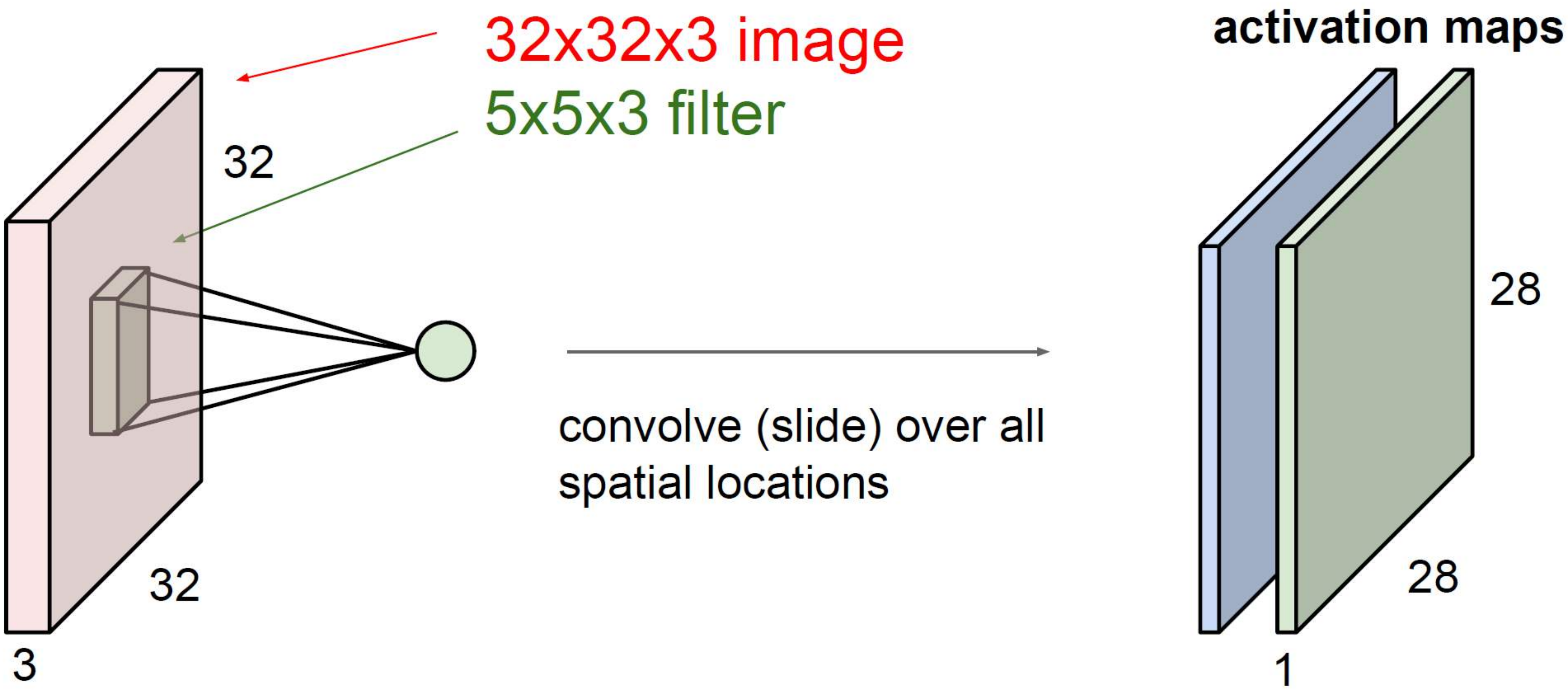

An Analysis of Deep Neural Network Models for Practical Applications, 2017.

# Convolution Layer
# : Preserve the spatial structure

# Convolution Layer
# : Preserve the spatial structure

# Convolution Layer
# : Preserve the spatial structure

# Convolution Layer
: Preserve the spatial structure




Stanford, cs231n 2017

# Advanced CNN Architectrues
# AlexNet

# Advanced CNN Architectrues
# AlexNet

# Advanced CNN Architectrues
# AlexNet