

# Lecture 13:

## DNNs in static images

# Outline

- Objectives and Introduction
- LeNet-5
- AlexNet
- VGG
- GoogleNet
- ResNet

# Objectives

- We will examine classic CNN architectures with the goal of:
  - Gaining intuition for building CNNs
  - Reusing CNN architectures

# LeNet-5

- *Gradient Based Learning Applied To Document Recognition* - Y. Lecun, L. Bottou, Y. Bengio, P. Haffner; 1998
- Helped establish how we use CNNs today
- Replaced manual feature extraction

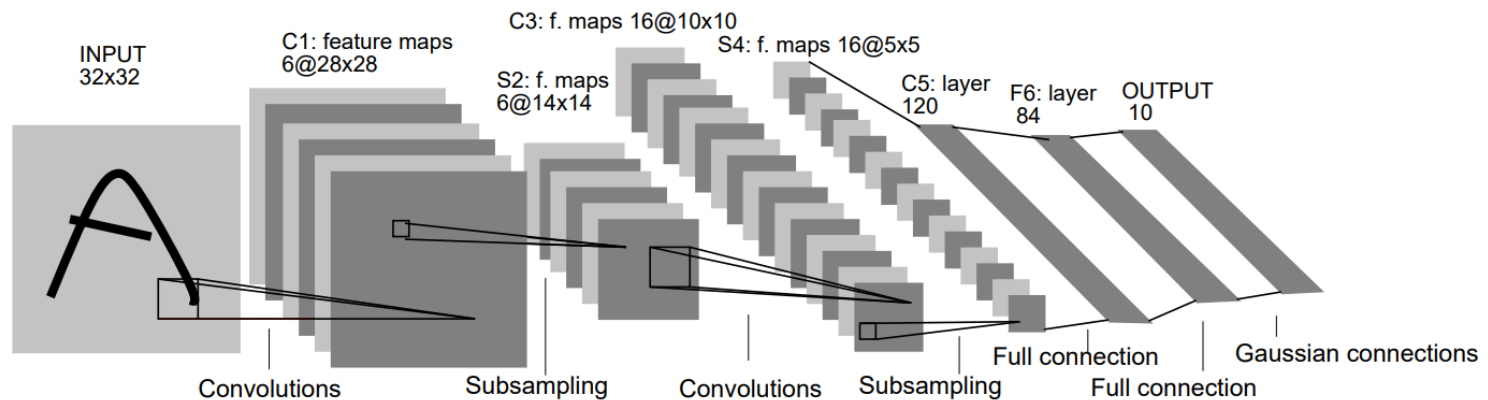
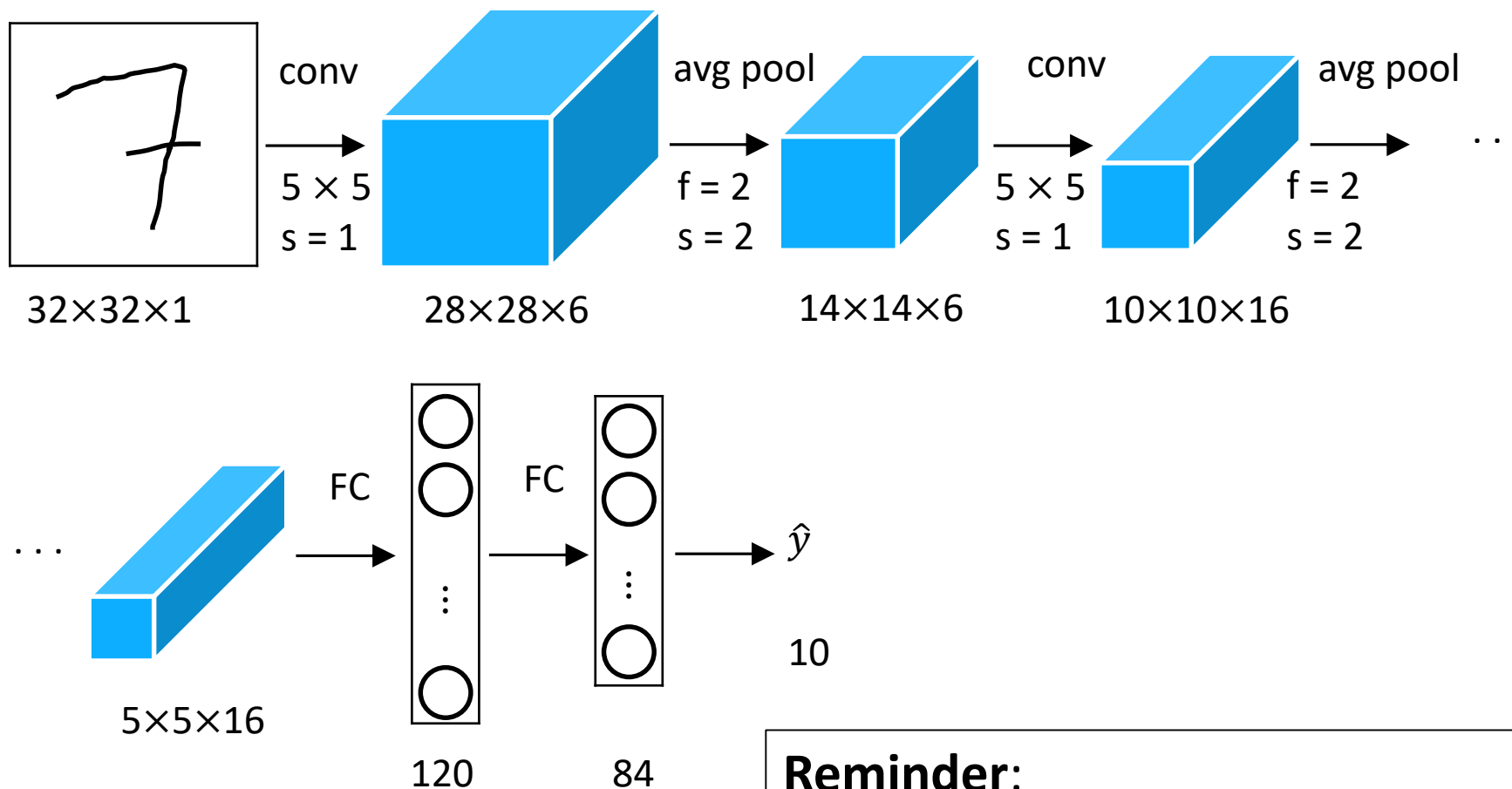


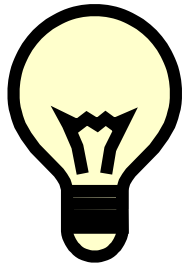
Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

# LeNet-5



**Reminder:**

$$\text{Output size} = (N + 2P - F) / \text{stride} + 1$$



# LeNet-5

- Only 60K parameters
  - As we go deeper in the network:  $N_H \downarrow$ ,  $N_W \downarrow$ ,  $N_C \uparrow$
  - General structure:  
conv->pool->conv->pool->FC->FC->output
- 
- Different filters look at different channels
  - Sigmoid and Tanh nonlinearity

# AlexNet

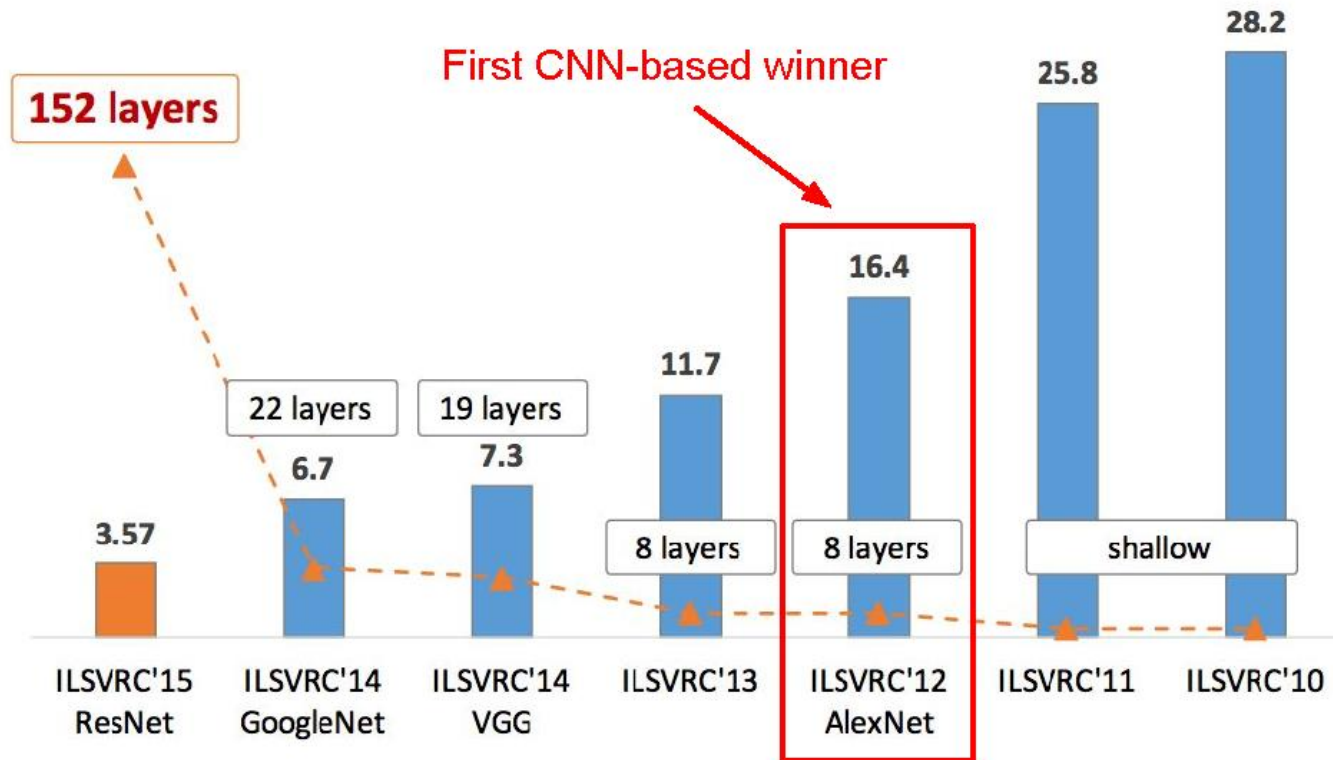
- *ImageNet Classification with Deep Convolutional Neural Networks - Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton; 2012*
- Facilitated by GPUs, highly optimized convolution implementation and large datasets (ImageNet)
- One of the largest CNNs to date
- Has 60 Million parameter compared to 60k parameter of LeNet-5

# ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners

- The annual “Olympics” of computer vision.
- Teams from across the world compete to see who has the best computer vision model for tasks such as classification, localization, detection, and more.
- **2012** marked **the first year where a CNN was used** to achieve a top 5 test error rate of 15.3%.
- The next best entry achieved an error of 26.2%.



# ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners



# AlexNet

## Architecture

CONV1

MAX POOL1

NORM1

CONV2

MAX POOL2

NORM2

CONV3

CONV4

CONV5

Max POOL3

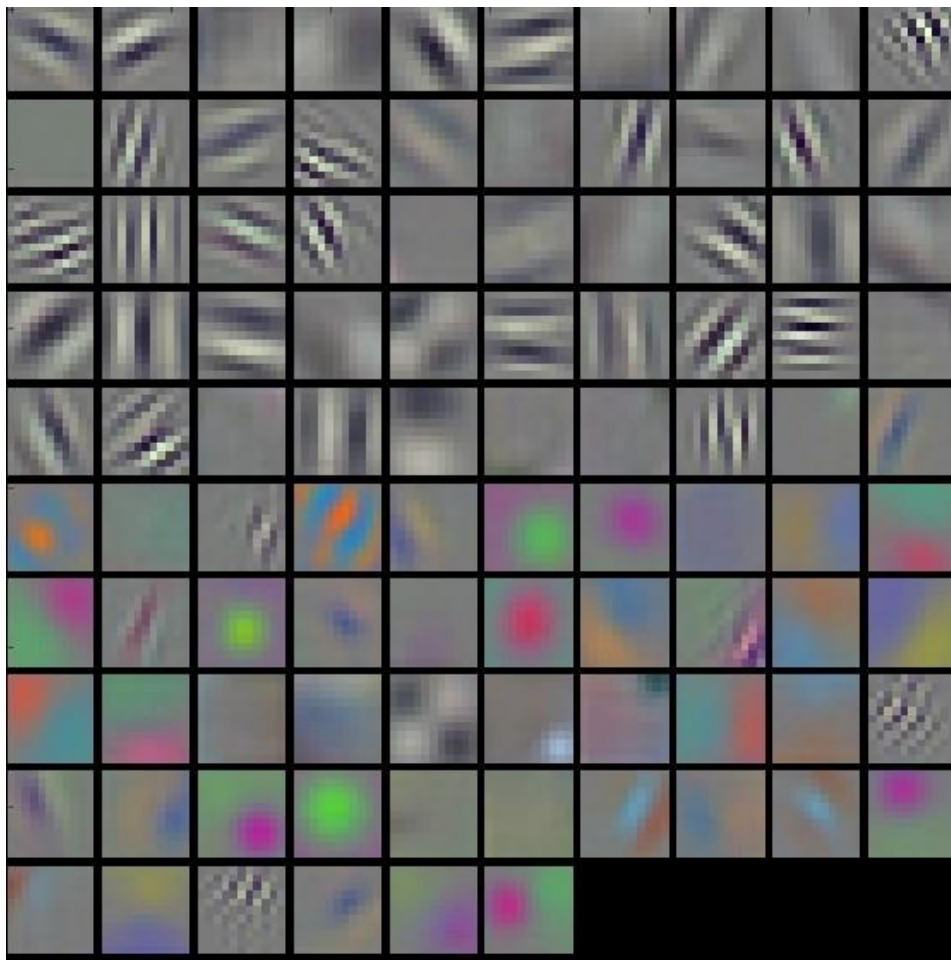
FC6

FC7

FC8

- Input: 227x227x3 images (224x224 before padding)
- First layer: 96 11x11 filters applied at stride 4
- **Output volume size?**  
$$(N-F)/s+1 = (227-11)/4+1 = 55 \rightarrow$$
  
**[55x55x96]**
- **Number of parameters in this layer?**  
$$(11*11*3)*96 = 35K$$

# AlexNet



# AlexNet

## Architecture

CONV1

MAX POOL1

NORM1

CONV2

MAX POOL2

NORM2

CONV3

CONV4

CONV5

Max POOL3

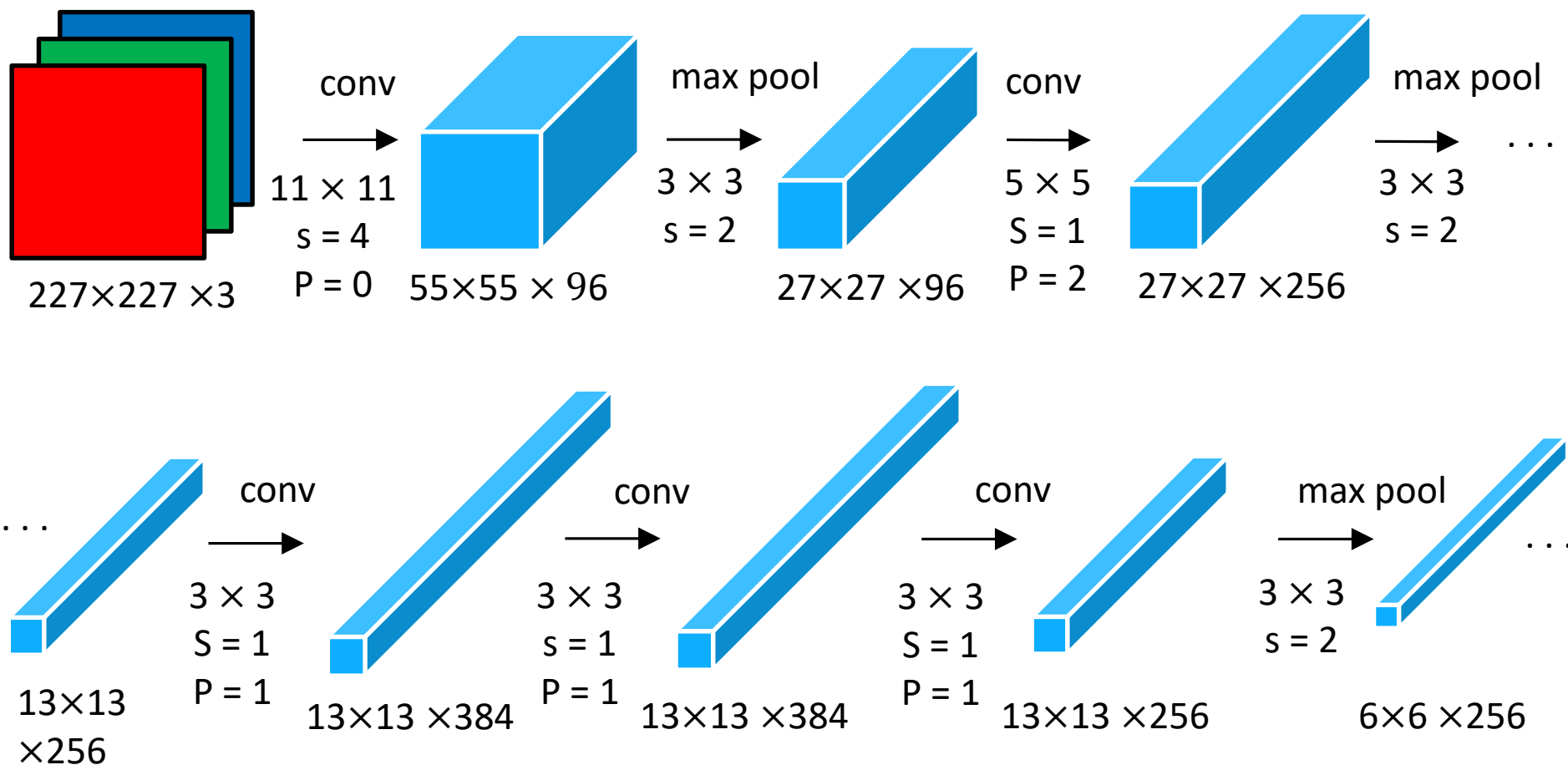
FC6

FC7

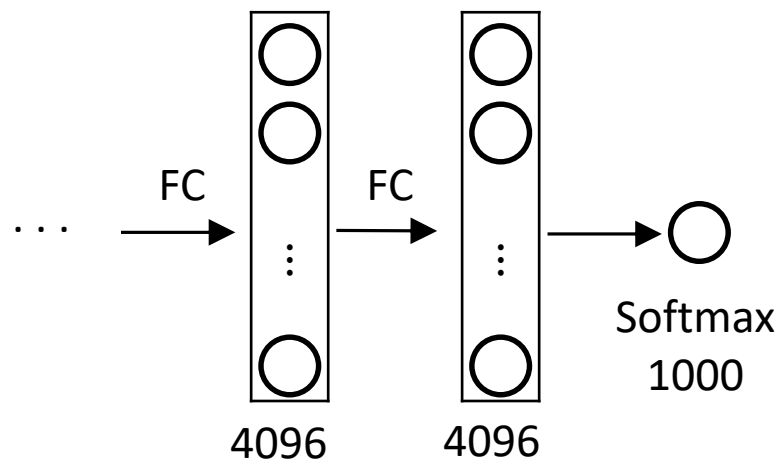
FC8

- Input: 227x227x3 images (224x224 before padding)
- After CONV1: 55x55x96
- Second layer: 3x3 filters applied at stride 2
- **Output volume size?**  
$$(N-F)/s+1 = (55-3)/2+1 = 27 \rightarrow [27 \times 27 \times 96]$$
- **Number of parameters in this layer?**  
**0!**

# AlexNet



# AlexNet



# AlexNet

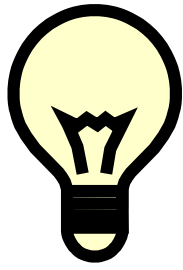
## Details/Retrospectives:

- first use of ReLU
- used Norm layers (not common anymore)
- heavy data augmentation
- dropout 0.5
- batch size 128
- 7 CNN ensemble

# AlexNet

- Trained on GTX 580 GPU with only 3 GB of memory.
- Network spread across 2 GPUs, half the neurons (feature maps) on each GPU.
- CONV1, CONV2, CONV4, CONV5:  
Connections only with feature maps on same GPU.
- CONV3, FC6, FC7, FC8:  
Connections with all feature maps in preceding layer, communication across GPUs.

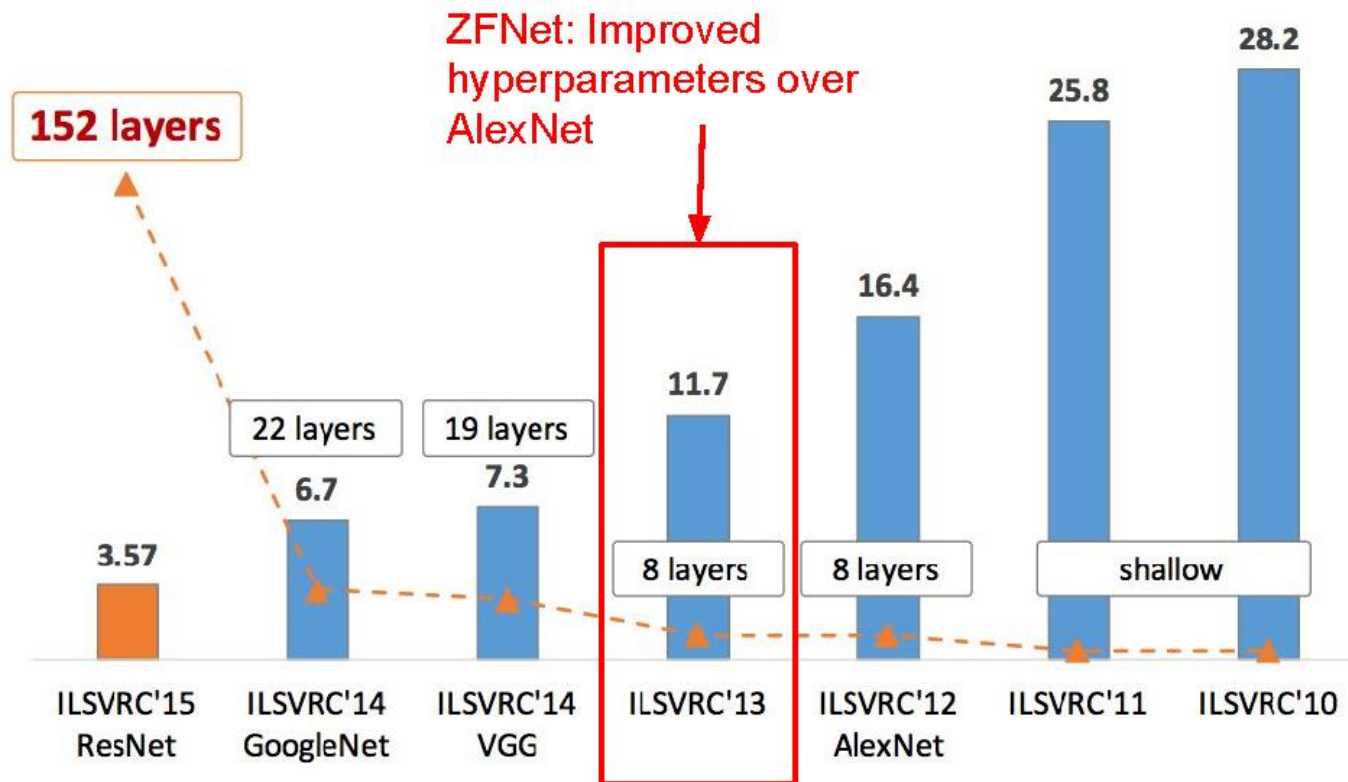




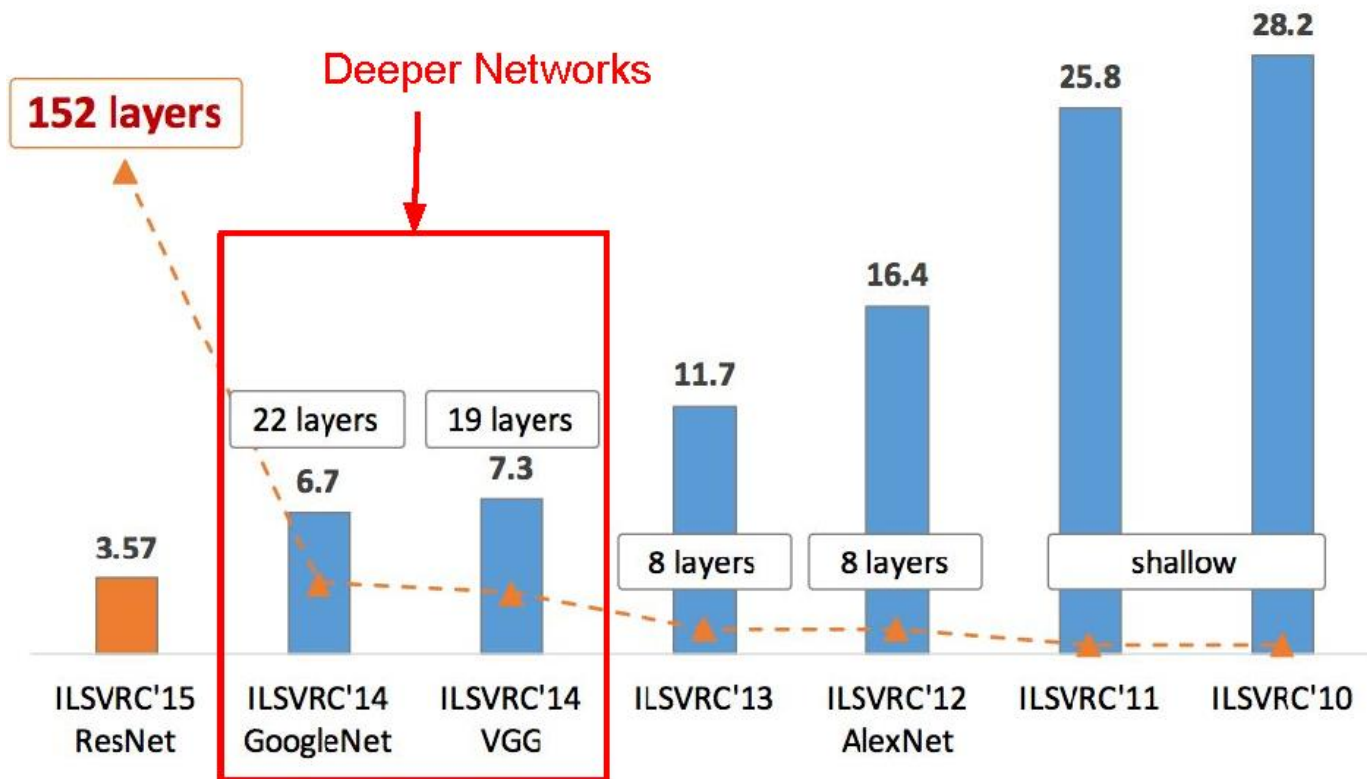
# AlexNet

AlexNet was the coming out party for CNNs in the computer vision community. This was **the first time a model performed so well on a historically difficult ImageNet dataset**. This paper illustrated the benefits of CNNs and backed them up with record breaking performance in the competition.

# ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners



# ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners



# VGGNet

- *Very Deep Convolutional Networks For Large Scale Image Recognition - Karen Simonyan and Andrew Zisserman; 2015*
- The runner-up at the ILSVRC 2014 competition
- Significantly deeper than AlexNet
- 140 million parameters

# VGGNet

Input

3x3 conv, 64

3x3 conv, 64

Pool 1/2

3x3 conv, 128

3x3 conv, 128

Pool 1/2

3x3 conv, 256

3x3 conv, 256

Pool 1/2

3x3 conv, 512

3x3 conv, 512

3x3 conv, 512

Pool 1/2

3x3 conv, 512

3x3 conv, 512

3x3 conv, 512

Pool 1/2

FC 4096

FC 4096

FC 1000

Softmax

- **Smaller filters**  
Only 3x3 CONV filters, stride 1, pad 1  
and 2x2 MAX POOL , stride 2
- **Deeper network**  
AlexNet: 8 layers  
VGGNet: 16 - 19 layers
- ZFNet: 11.7% top 5 error in ILSVRC'13
- VGGNet: 7.3% top 5 error in ILSVRC'14

# VGGNet

- **Why use smaller filters? (3x3 conv)**

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

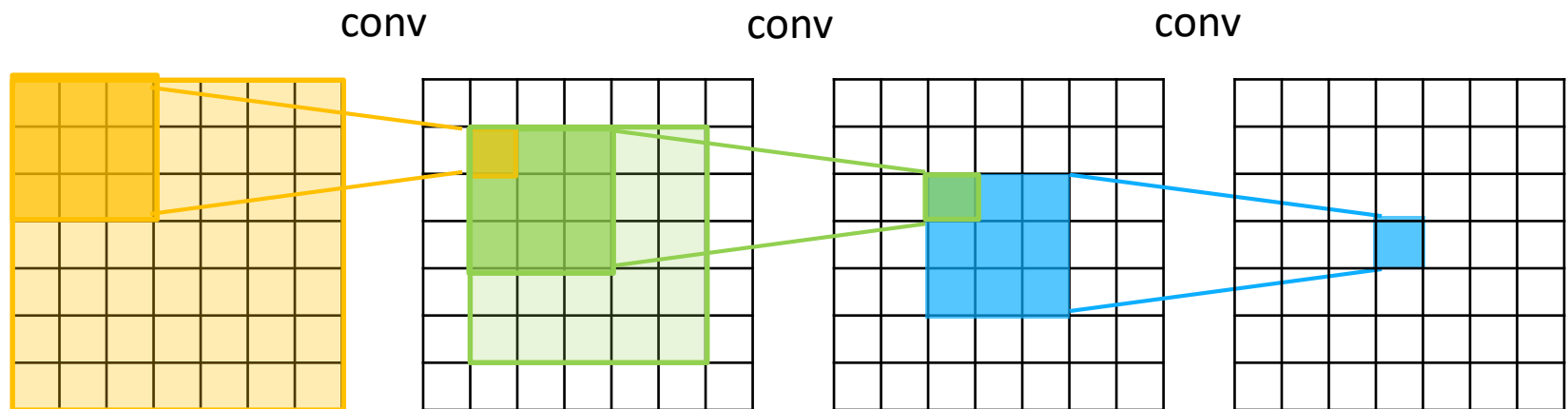
- **What is the effective receptive field of three 3x3 conv (stride 1) layers?**

**7x7**

But deeper, more non-linearities

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

# Reminder: Receptive Field



Input	memory: $224*224*3=150K$	params: 0
3x3 conv, 64	memory: $224*224*64=3.2M$	params: $(3*3*3)*64 = 1,728$
3x3 conv, 64	memory: $224*224*64=3.2M$	params: $(3*3*64)*64 = 36,864$
Pool	memory: $112*112*64=800K$	params: 0
3x3 conv, 128	memory: $112*112*128=1.6M$	params: $(3*3*64)*128 = 73,728$
3x3 conv, 128	memory: $112*112*128=1.6M$	params: $(3*3*128)*128 = 147,456$
Pool	memory: $56*56*128=400K$	params: 0
3x3 conv, 256	memory: $56*56*256=800K$	params: $(3*3*128)*256 = 294,912$
3x3 conv, 256	memory: $56*56*256=800K$	params: $(3*3*256)*256 = 589,824$
3x3 conv, 256	memory: $56*56*256=800K$	params: $(3*3*256)*256 = 589,824$
Pool	memory: $28*28*256=200K$	params: 0
3x3 conv, 512	memory: $28*28*512=400K$	params: $(3*3*256)*512 = 1,179,648$
3x3 conv, 512	memory: $28*28*512=400K$	params: $(3*3*512)*512 = 2,359,296$
3x3 conv, 512	memory: $28*28*512=400K$	params: $(3*3*512)*512 = 2,359,296$
Pool	memory: $14*14*512=100K$	params: 0
3x3 conv, 512	memory: $14*14*512=100K$	params: $(3*3*512)*512 = 2,359,296$
3x3 conv, 512	memory: $14*14*512=100K$	params: $(3*3*512)*512 = 2,359,296$
3x3 conv, 512	memory: $14*14*512=100K$	params: $(3*3*512)*512 = 2,359,296$
Pool	memory: $7*7*512=25K$	params: 0
FC 4096	memory: 4096	params: $7*7*512*4096 = 102,760,448$
FC 4096	memory: 4096	params: $4096*4096 = 16,777,216$
FC 1000	memory: 1000	params: $4096*1000 = 4,096,000$



# VGGNet

## VGG16:

TOTAL memory:  $24M * 4 \text{ bytes} \approx 96MB$  / image

TOTAL params: 138M parameters

Input

3x3 conv, 64

3x3 conv, 64

Pool

3x3 conv, 128

3x3 conv, 128

Pool

3x3 conv, 256

3x3 conv, 256

3x3 conv, 256

Pool

3x3 conv, 512

3x3 conv, 512

3x3 conv, 512

Pool

3x3 conv, 512

3x3 conv, 512

3x3 conv, 512

Pool

FC 4096

FC 4096

FC 1000

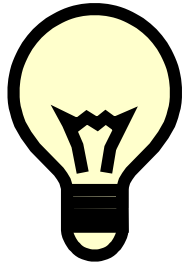
Softmax

Input	memory: $224*224*3=150K$	params: 0
3x3 conv, 64	memory: $224*224*64=3.2M$	params: $(3*3*3)*64 = 1,728$
3x3 conv, 64	memory: $224*224*64=3.2M$	params: $(3*3*64)*64 = 36,864$
Pool	memory: $112*112*64=800K$	params: 0
3x3 conv, 128	memory: $112*112*128=1.6M$	params: $(3*3*64)*128 = 73,728$
3x3 conv, 128	memory: $112*112*128=1.6M$	params: $(3*3*128)*128 = 147,456$
Pool	memory: $56*56*128=400K$	params: 0
3x3 conv, 256	memory: $56*56*256=800K$	params: $(3*3*128)*256 = 294,912$
3x3 conv, 256	memory: $56*56*256=800K$	params: $(3*3*256)*256 = 589,824$
3x3 conv, 256	memory: $56*56*256=800K$	params: $(3*3*256)*256 = 589,824$
Pool	memory: $28*28*256=200K$	params: 0
3x3 conv, 512	memory: $28*28*512=400K$	params: $(3*3*256)*512 = 1,179,648$
3x3 conv, 512	memory: $28*28*512=400K$	params: $(3*3*512)*512 = 2,359,296$
3x3 conv, 512	memory: $28*28*512=400K$	params: $(3*3*512)*512 = 2,359,296$
Pool	memory: $14*14*512=100K$	params: 0
3x3 conv, 512	memory: $14*14*512=100K$	params: $(3*3*512)*512 = 2,359,296$
3x3 conv, 512	memory: $14*14*512=100K$	params: $(3*3*512)*512 = 2,359,296$
3x3 conv, 512	memory: $14*14*512=100K$	params: $(3*3*512)*512 = 2,359,296$
Pool	memory: $7*7*512=25K$	params: 0
FC 4096	memory: 4096	params: $7*7*512*4096 = 102,760,448$
FC 4096	memory: 4096	params: $4096*4096 = 16,777,216$
FC 1000	memory: 1000	params: $4096*1000 = 4,096,000$

# VGGNet

## Details/Retrospectives :

- ILSVRC'14 2nd in classification, 1st in localization
- Similar training procedure as AlexNet
- 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
- Trained on 4 Nvidia Titan Black GPUs for **two to three weeks**.



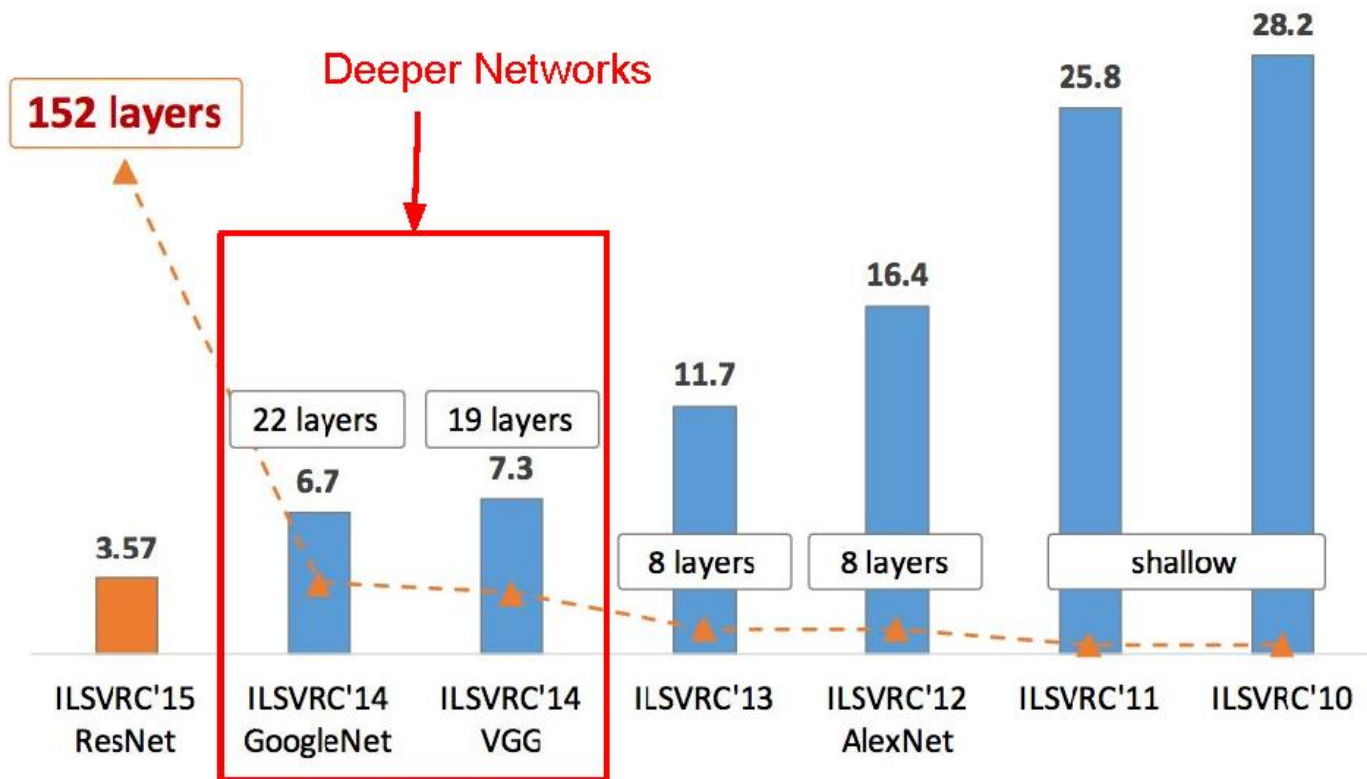
# VGGNet

VGG Net reinforced the notion that **convolutional neural networks have to have a deep network of layers in order for this hierarchical representation of visual data to work.**

Keep it deep.

Keep it simple.

# ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners

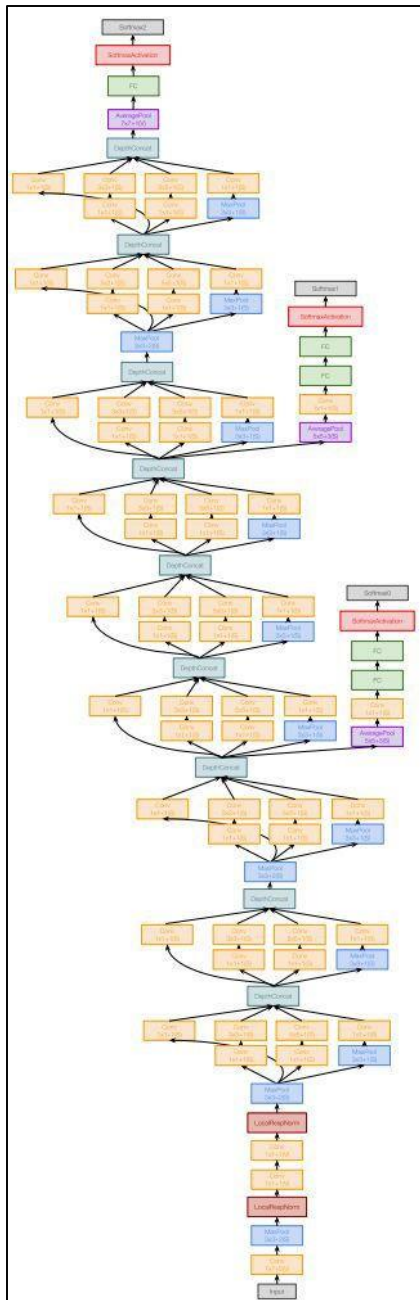


# GoogleNet

- *Going Deeper with Convolutions - Christian Szegedy et al.; 2015*
- ILSVRC 2014 competition winner
- Also significantly deeper than AlexNet
- x12 less parameters than AlexNet
- Focused on computational efficiency

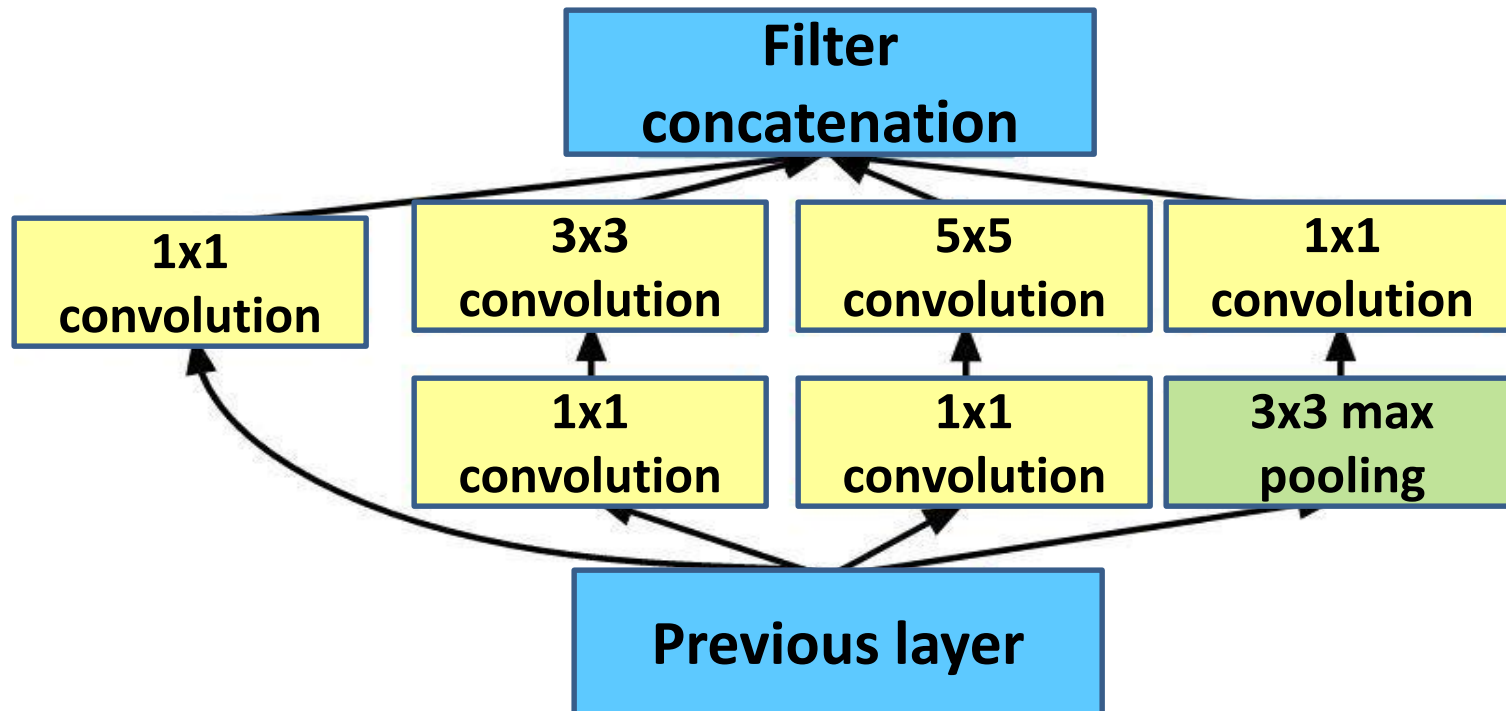
# GoogleNet

- 22 layers
- Efficient **“Inception” module** - strayed from the general approach of simply stacking conv and pooling layers on top of each other in a sequential structure
- No FC layers
- Only 5 million parameters!
- ILSVRC’14 classification winner (6.7% top 5 error)



# GoogleNet

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

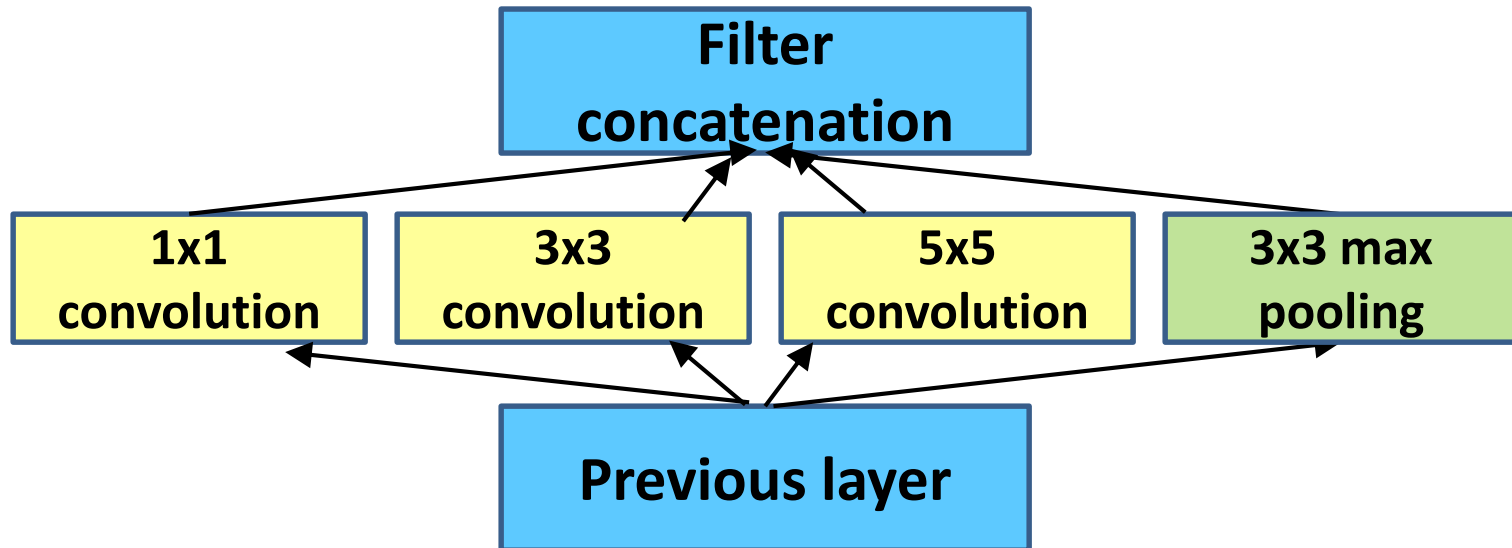




# GoogleNet

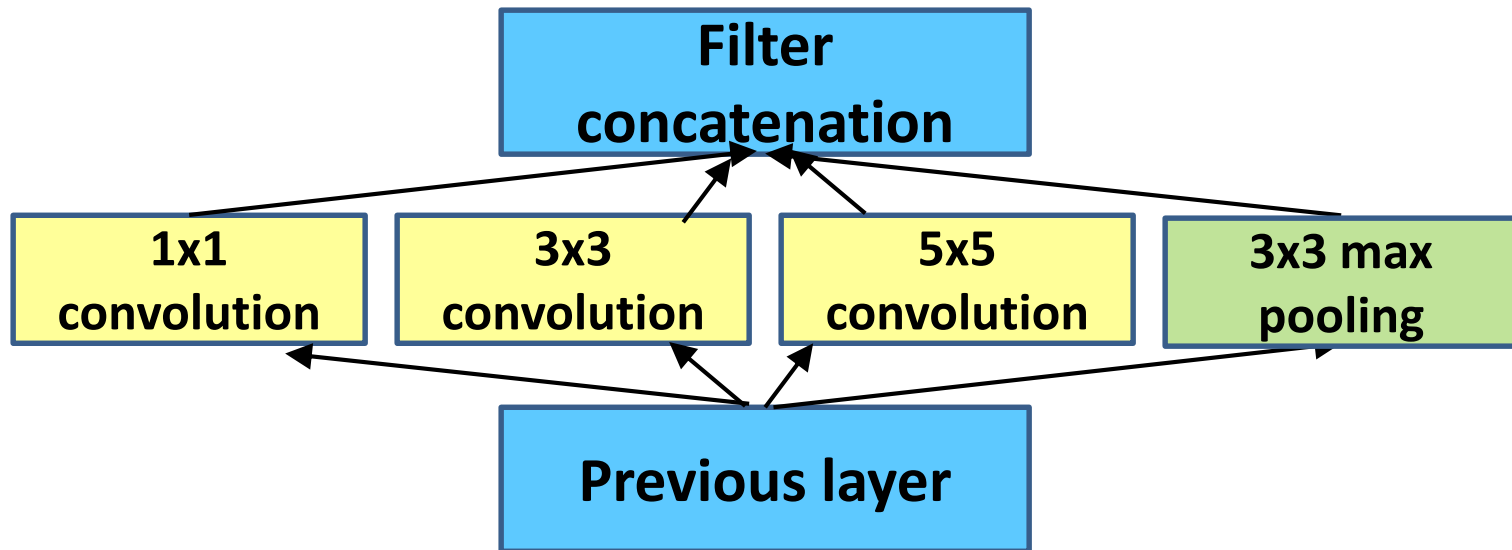
## Naïve Inception Model

- Apply parallel filter operations on the input :
  - Multiple receptive field sizes for convolution (1x1, 3x3, 5x5)
  - Pooling operation (3x3)
- Concatenate all filter outputs together depth-wise



# GoogleNet

- What's the problem with this?  
High computational complexity



# GoogleNet

- Output volume sizes:**

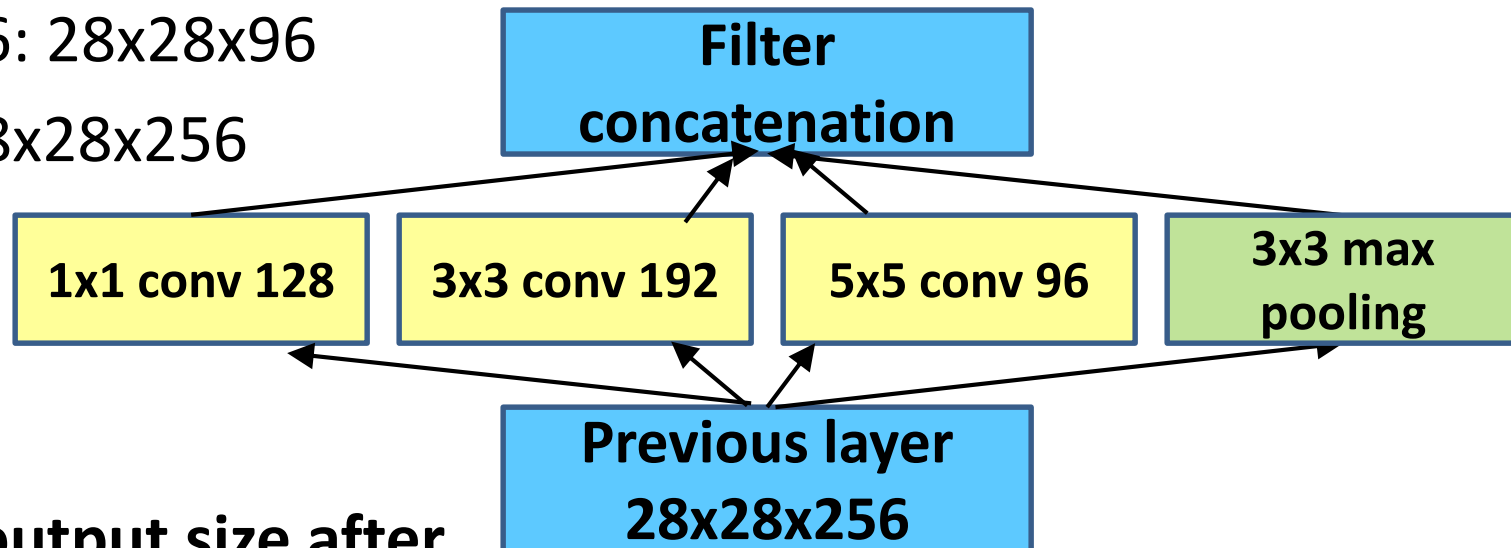
1x1 conv, 128: 28x28x128

3x3 conv, 192: 28x28x192

5x5 conv, 96: 28x28x96

3x3 pool: 28x28x256

## Example:



- What is output size after filter concatenation?**

$$28 \times 28 \times (128 + 192 + 96 + 256) = 28 \times 28 \times 672$$

# GoogleNet

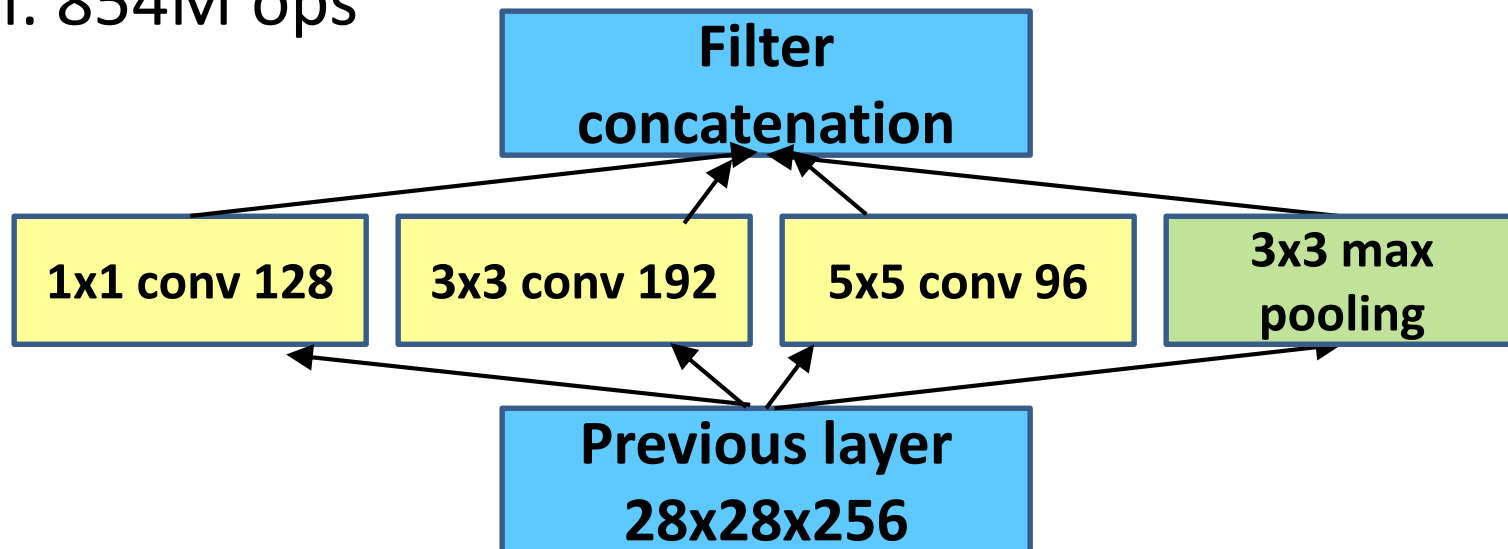
- **Number of convolution operations:**

1x1 conv, 128:  $28 \times 28 \times 128 \times 1 \times 1 \times 256$

3x3 conv, 192:  $28 \times 28 \times 192 \times 3 \times 3 \times 256$

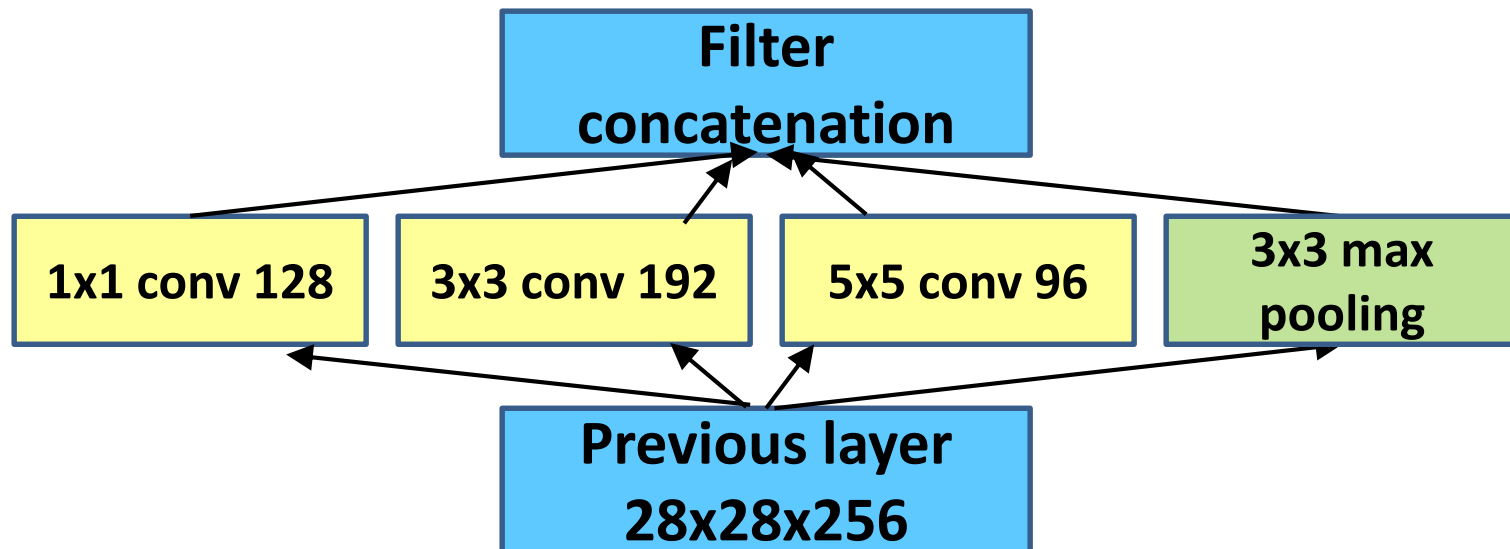
5x5 conv, 96:  $28 \times 28 \times 96 \times 5 \times 5 \times 256$

Total: 854M ops



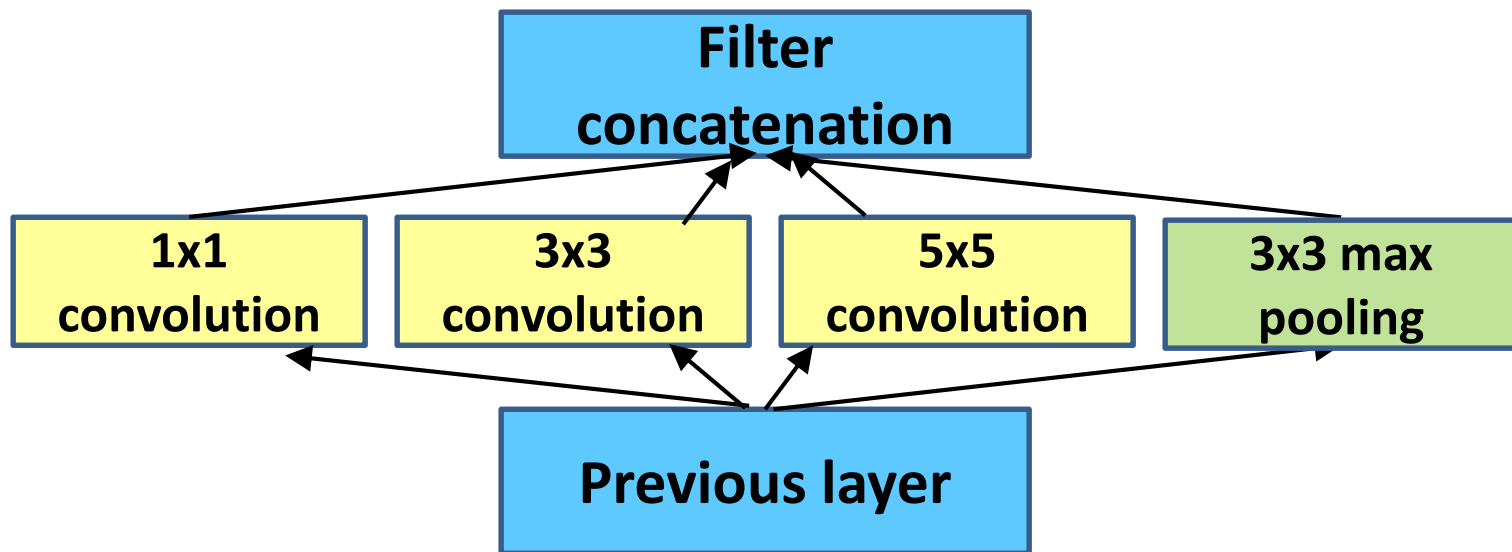
# GoogleNet

- **Very expensive compute!**
- Pooling layer also preserves feature depth, which means total depth after concatenation can only grow at every layer.



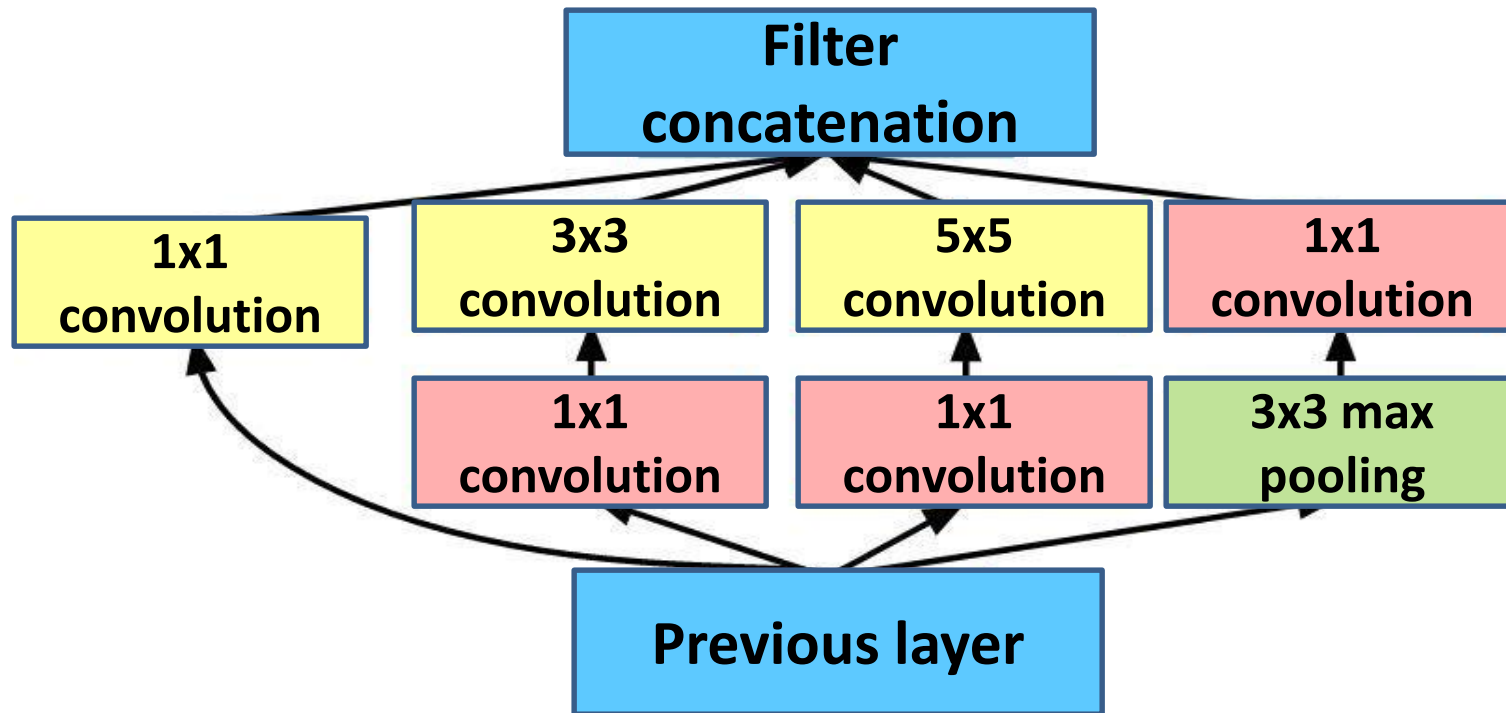
# GoogleNet

- **Solution:** “bottleneck” layers that use 1x1 convolutions to reduce feature depth (from previous hour).



# GoogleNet

- **Solution:** “bottleneck” layers that use 1x1 convolutions to reduce feature depth (from previous hour).



- Number of convolution operations:**

1x1 conv, 64:  $28 \times 28 \times 64 \times 1 \times 1 \times 256$

1x1 conv, 64:  $28 \times 28 \times 64 \times 1 \times 1 \times 256$

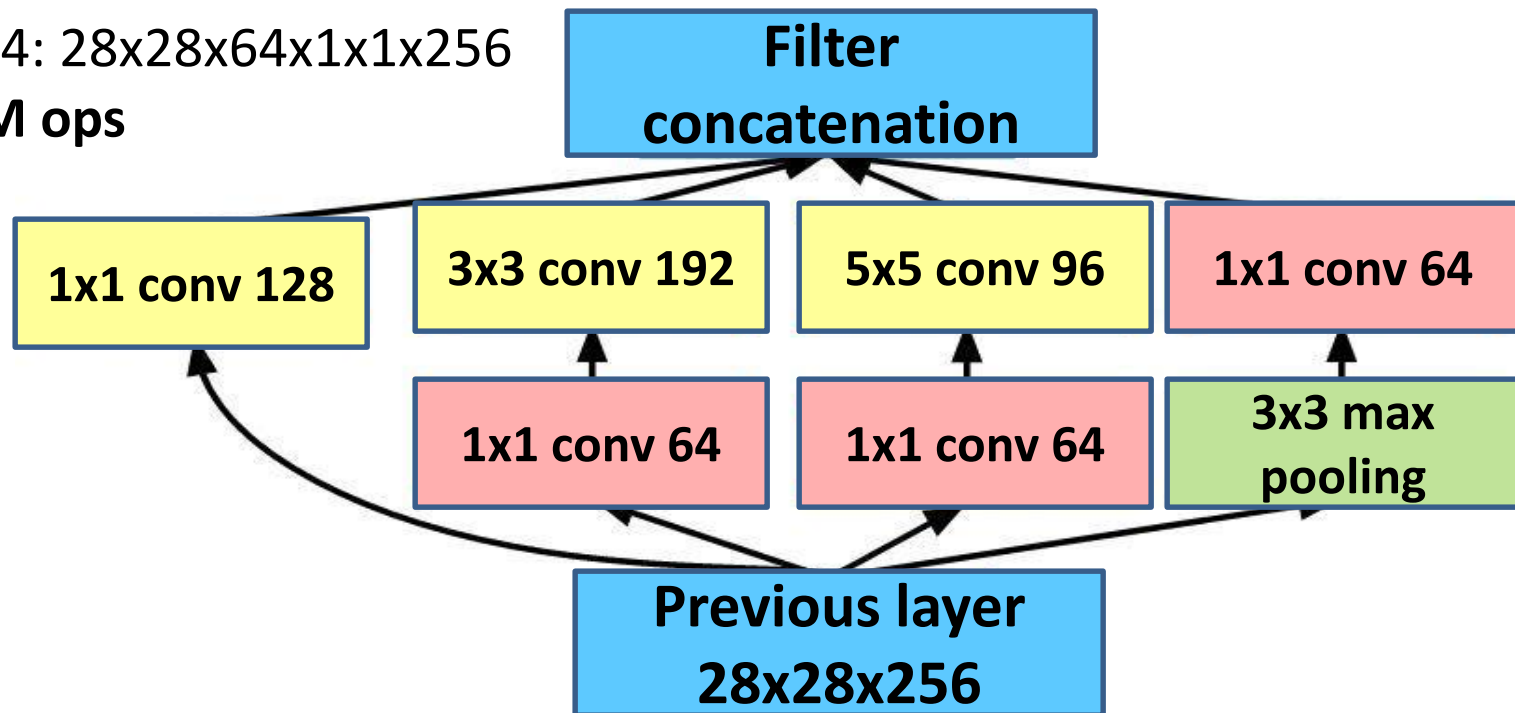
1x1 conv, 128:  $28 \times 28 \times 128 \times 1 \times 1 \times 256$

3x3 conv, 192:  $28 \times 28 \times 192 \times 3 \times 3 \times 64$

5x5 conv, 96:  $28 \times 28 \times 96 \times 5 \times 5 \times 264$

1x1 conv, 64:  $28 \times 28 \times 64 \times 1 \times 1 \times 256$

**Total: 353M ops**



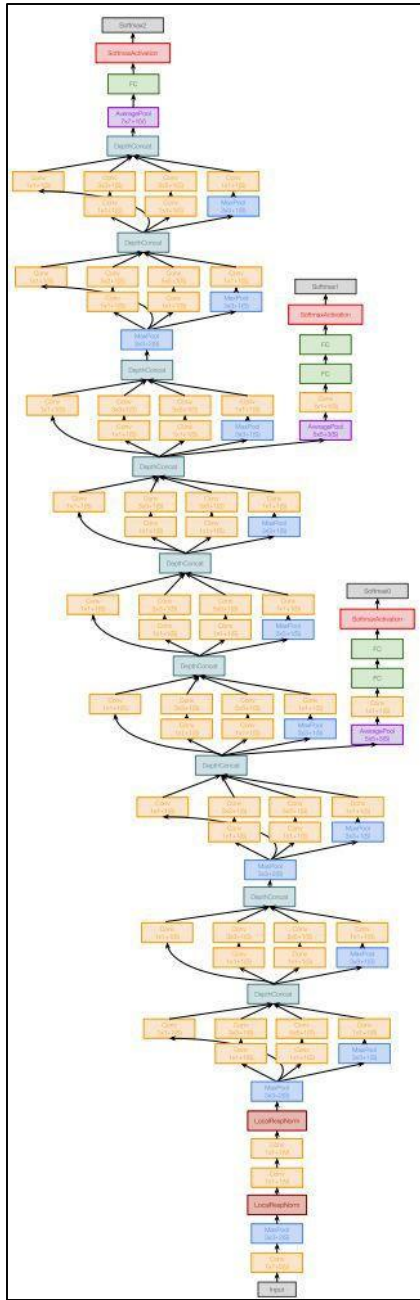
- Compared to 854M ops for naive version

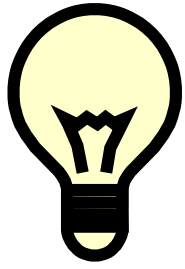


# GoogleNet

## Details/Retrospectives :

- 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)

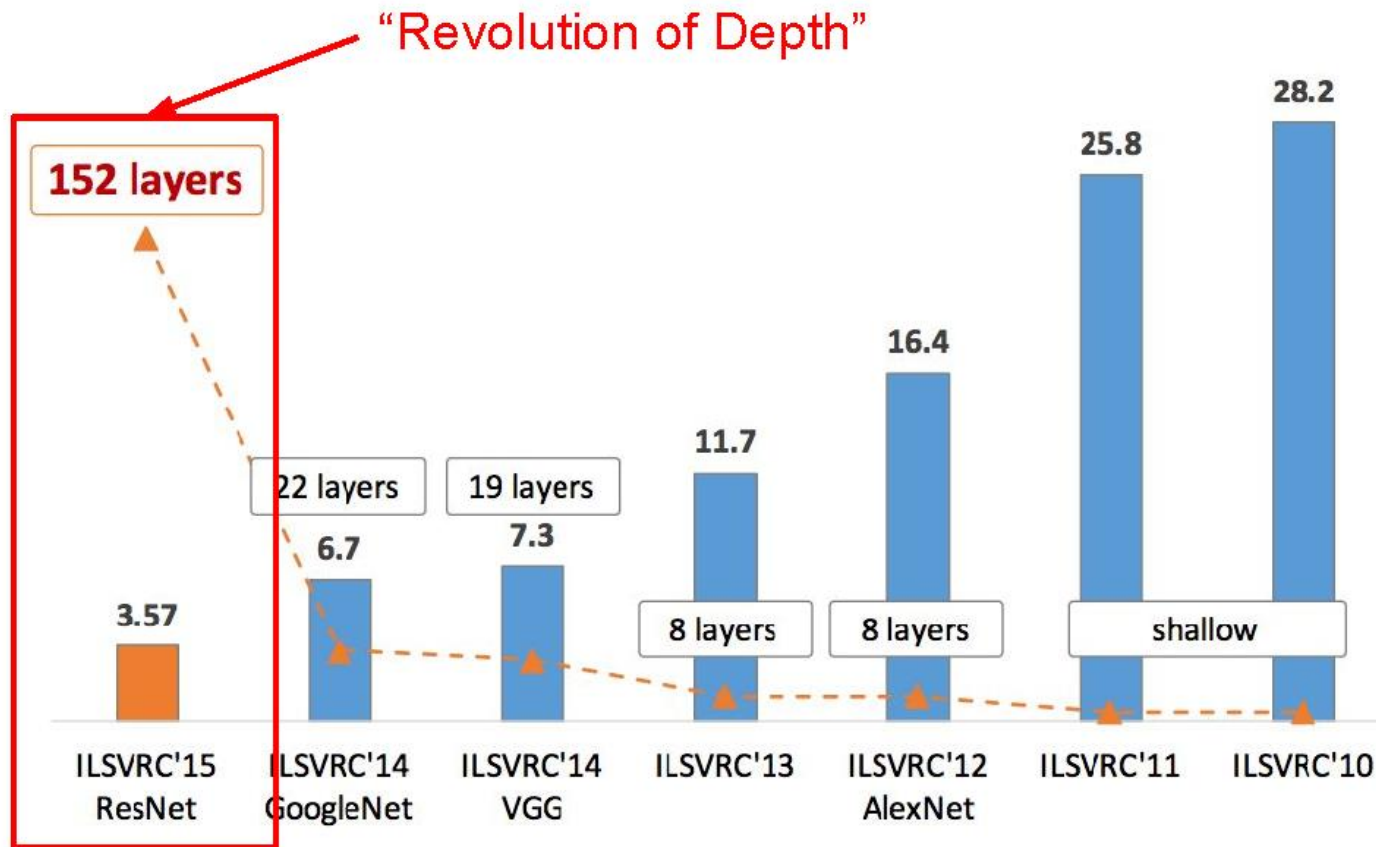




# GoogleNet

Introduced the idea that CNN layers **didn't always have to be stacked up sequentially**. Coming up with the Inception module, the authors showed that a creative structuring of layers can lead to improved performance and **computationally efficiency**.

# ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners

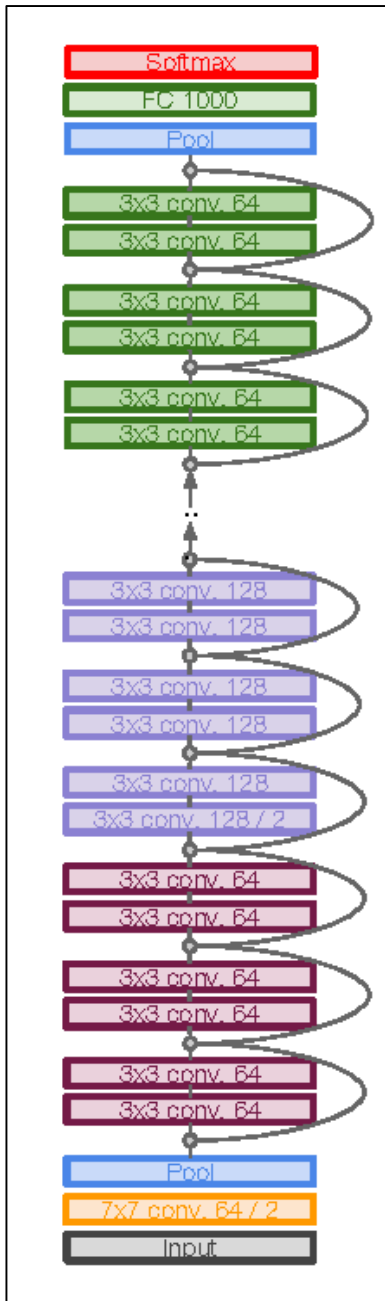


# ResNet

- *Deep Residual Learning for Image Recognition - Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun; 2015*
- Extremely deep network – 152 layers
- Deeper neural networks are more difficult to train.
- Deep networks suffer from vanishing and exploding gradients.
- Present a residual learning framework to ease the training of networks that are substantially deeper than those used previously.

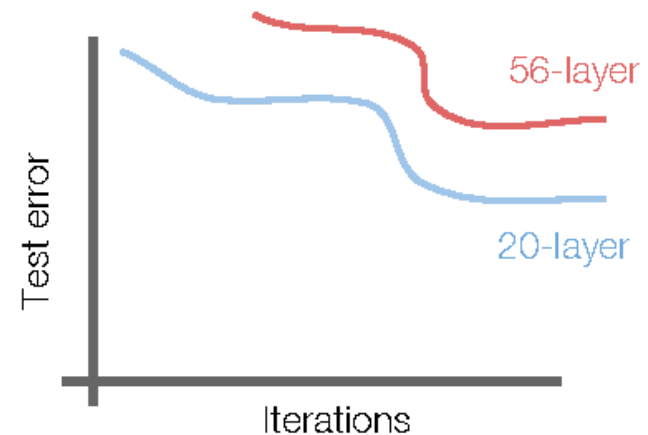
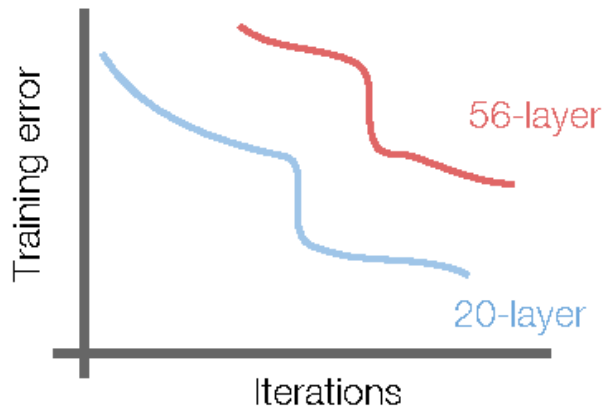
# ResNet

- ILSVRC'15 classification winner (3.57% top 5 error, humans generally hover around a 5-10% error rate)  
Swept all classification and detection competitions in ILSVRC'15 and COCO'15!



# ResNet

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



- 56-layer model performs worse on both training and test error  
-> The deeper model performs worse (not caused by overfitting)!

# ResNet

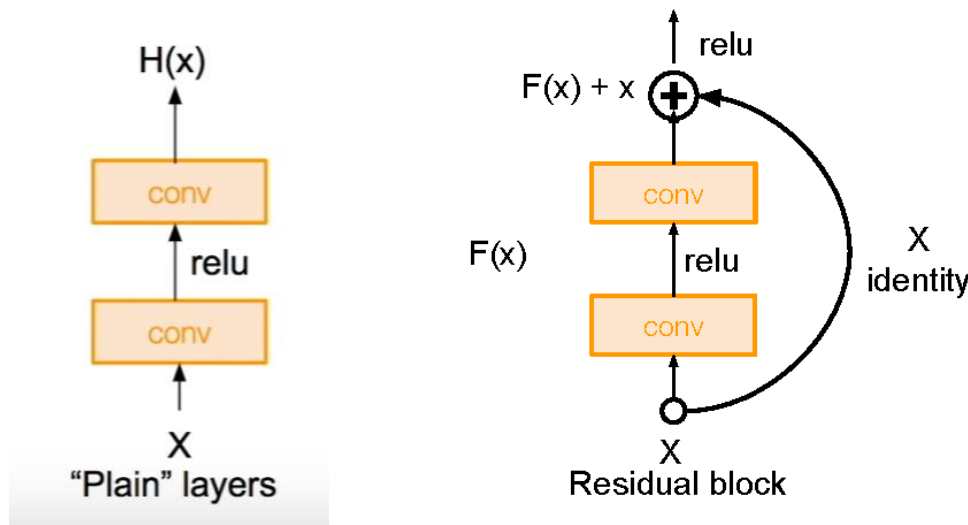
- **Hypothesis:** The problem is an optimization problem. Very deep networks are harder to optimize.
- **Solution:** Use network layers to fit residual mapping instead of directly trying to fit a desired underlying mapping.
- We will use **skip connections** allowing us to take the activation from one layer and feed it into another layer, much deeper into the network.
- Use layers to fit residual  $F(x) = H(x) - x$  instead of  $H(x)$  directly

# ResNet

## Residual Block

Input  $x$  goes through conv-relu-conv series and gives us  $F(x)$ . That result is then added to the original input  $x$ . Let's call that  $H(x) = F(x) + x$ .

In traditional CNNs,  $H(x)$  would just be equal to  $F(x)$ . So, instead of just computing that transformation (straight from  $x$  to  $F(x)$ ), we're computing the term that we have to *add*,  $F(x)$ , to the input,  $x$ .





## Full ResNet architecture:

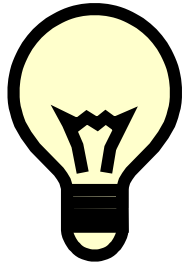
- Stack residual blocks
- Every residual block has two 3x3 conv layers
- Periodically, double # of filters and downsample spatially using stride 2 (in each dimension)
- Additional conv layer at the beginning
- No FC layers at the end (only FC 1000 to output classes)

- Total depths of 34, 50, 101, or 152 layers for ImageNet
- For deeper networks (ResNet-50+), use “bottleneck” layer to improve efficiency (similar to GoogLeNet)

# ResNet

## Experimental Results:

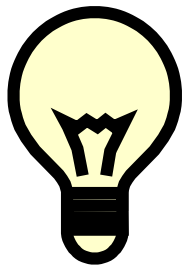
- Able to train very deep networks without degrading
- Deeper networks now achieve lower training errors as expected



# ResNet

The **best** CNN architecture that we currently have and is a great innovation for the idea of residual learning.

Even better than human performance!

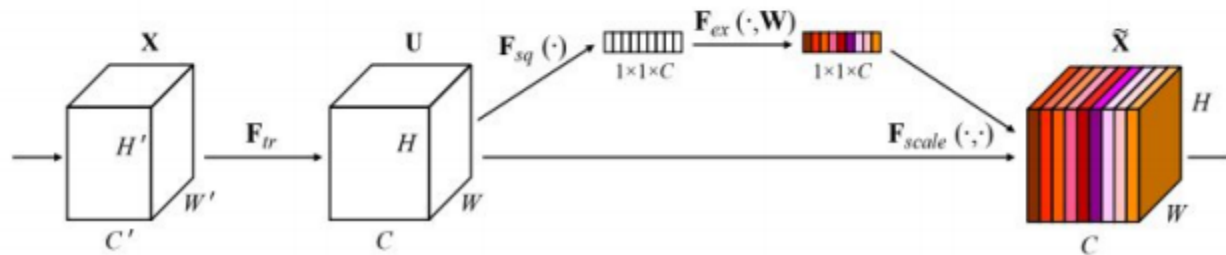
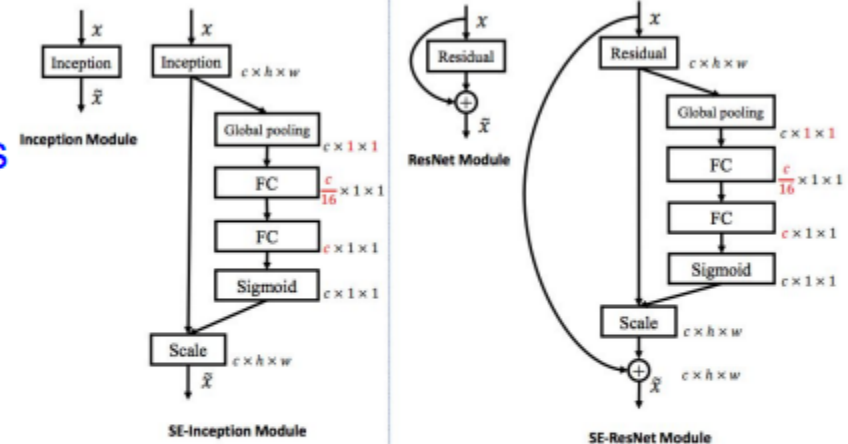


# Improving ResNet

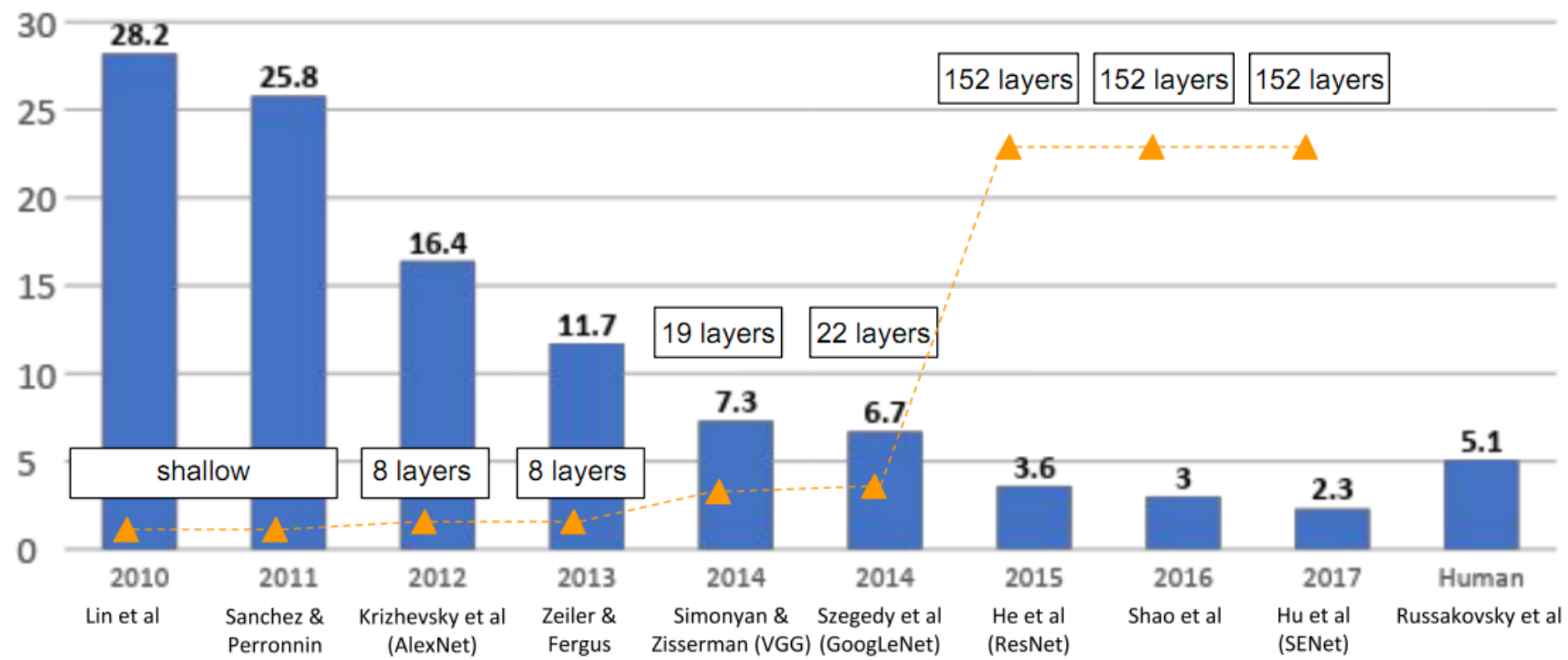
## Squeeze-and-Excitation Networks (SENet)

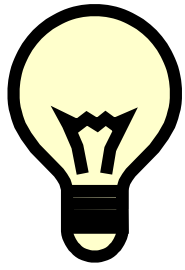
[Hu et al. 2017]

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



# ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners



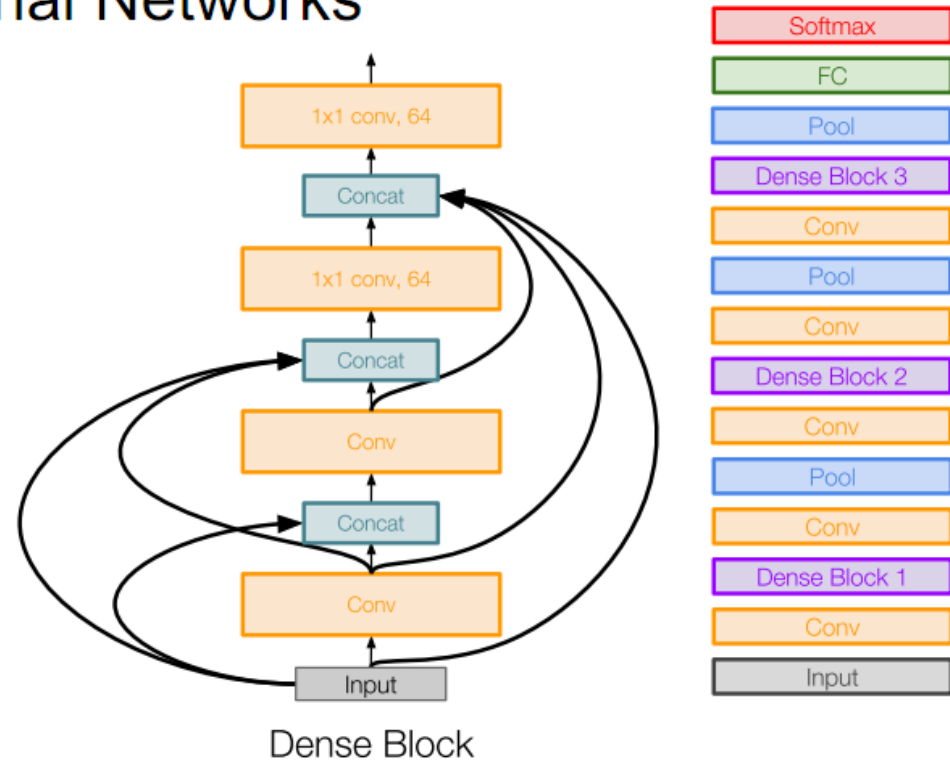


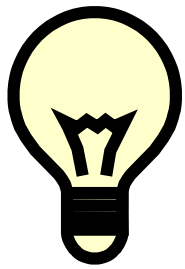
# Beyond ResNet...

## Densely Connected Convolutional Networks

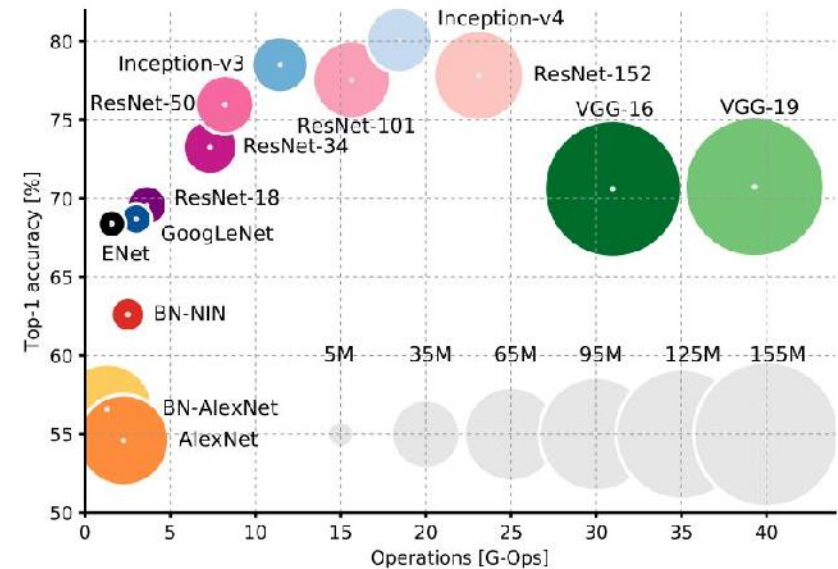
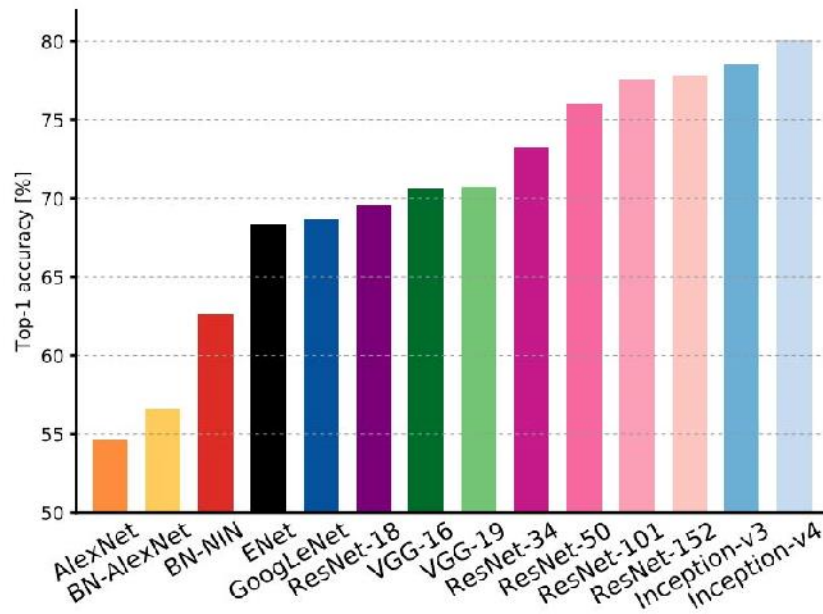
[Huang et al. 2017]

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

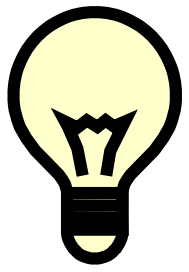




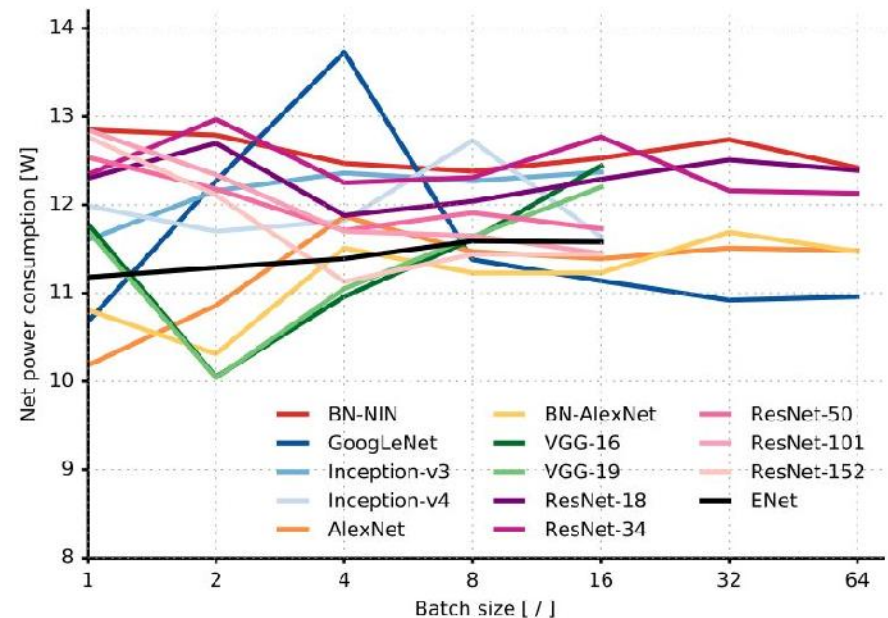
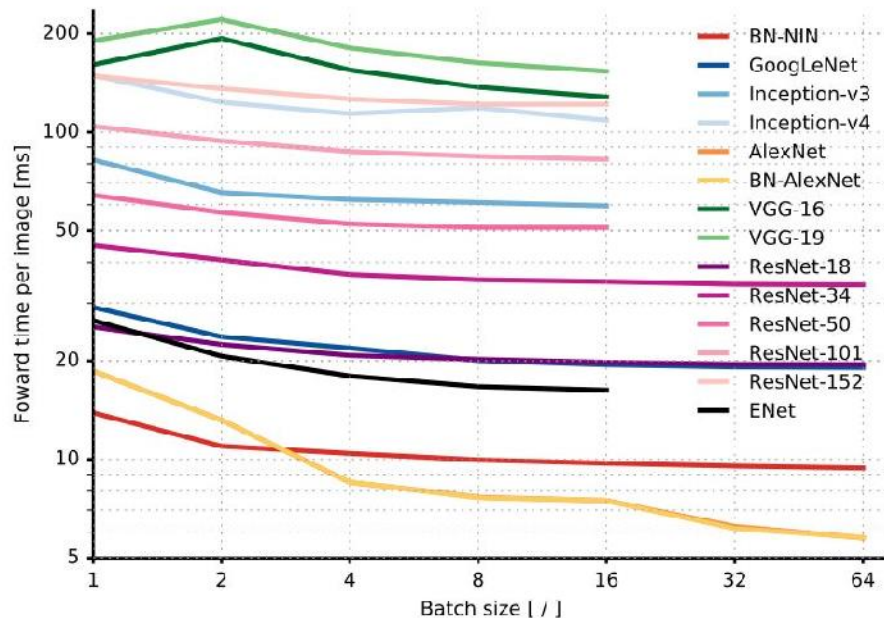
# Accuracy comparison







# Forward pass time and power consumption



# Summary

- LeNet-5
- **AlexNet**
- **VGG**
- **GoogleNet** – Inception module
- **ResNet** – Residual block

# References

- *Gradient-based learning applied to document recognition; ann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner; 1998*
- *ImageNet Classification with Deep Convolutional Neural Networks - Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton; 2012*
- *Very Deep Convolutional Networks For Large Scale Image Recognition - Karen Simonyan and Andrew Zisserman; 2015*
- *Going Deeper with Convolutions - Christian Szegedy et al.; 2015*
- *Deep Residual Learning for Image Recognition - Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun; 2015*
- *Stanford CS231- Fei-Fei & Justin Johnson & Serena Yeung. Lecture 9*
- *Coursera, Machine Learning course by Andrew Ng.*

# References

- *The 9 Deep Learning Papers You Need To Know About (Understanding CNNs Part 3) by Adit Deshpande*  
<https://adeshpande3.github.io/adeshpande3.github.io/The-9-Deep-Learning-Papers-You-Need-To-Know-About.html>
- *CNNs Architectures: LeNet, AlexNet, VGG, GoogLeNet, ResNet and more ... By Siddharth Das*  
[https://medium.com/@siddharthdas\\_32104/cnns-architectures-lenet-alexnet-vgg-googlenet-resnet-and-more-666091488df5](https://medium.com/@siddharthdas_32104/cnns-architectures-lenet-alexnet-vgg-googlenet-resnet-and-more-666091488df5)
- *Slide taken from Forward And Backpropagation in Convolutional Neural Network. – Medium , By Sujit Rai*  
<https://medium.com/@2017csm1006/forward-and-backpropagation-in-convolutional-neural-network-4dfa96d7b37e>