Convolutional Neural Networks for Image Classification

Pablo Mesejo

pmesejo@go.ugr.es

Universidad de Granada

Departamento de Ciencias de la Computación e Inteligencia Artificial

UNIVERSIDAD DE GRANADA

DECSAI

DaSCI

Instituto Andaluz de Investigación en **Data Science** and **Computational Intelligence**

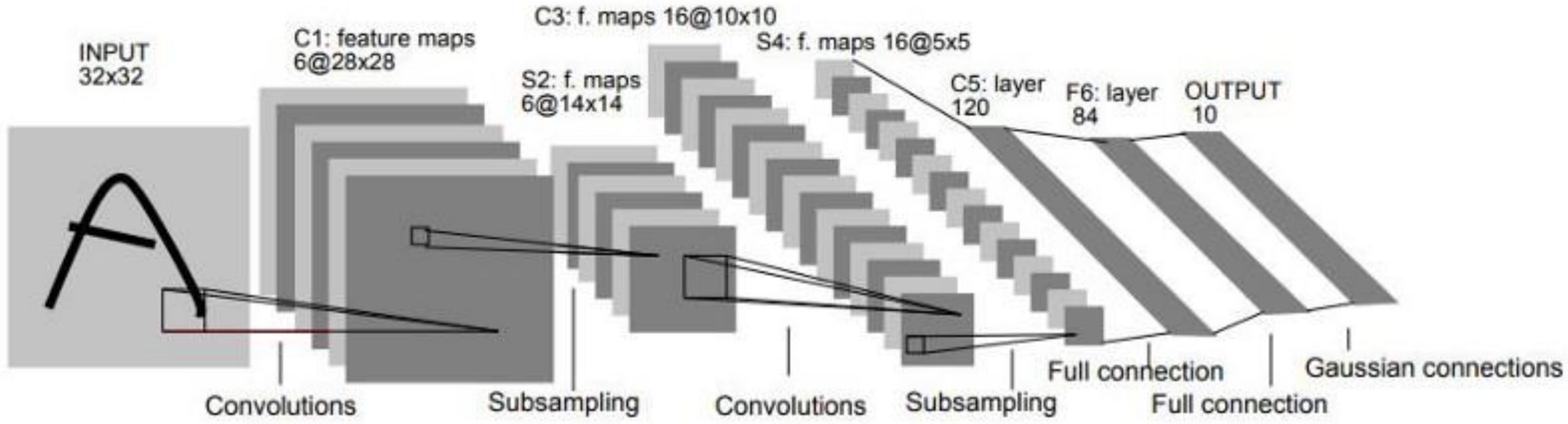
Readings

- Goodfellow, Bengio and Courville (2016), Chapter 9.
- Szeliski (2022), Chapter 5.3, 5.4, and 6.2.
- Zhang, Lipton, Li and Smola (2023), *Dive into Deep Learning*, Chapter 7 and 8.
- Stanford University CS231n (2023): Deep Learning for Computer Vision. Lectures 5 and 6.

Examples of Deep Architectures for Image Classification

Next slides are mainly based on those from [cs231n]

LeNet-5



1 2 3 4 5 6 7

Feature extraction phase

1.- 6 5x5 filters → 6@28x28
2.- Avg Pooling 2x2 → 6@14x14
3.- 16 5x5 filters → 16@10x10
4.- Avg Pooling 2x2 → 16@5x5
5.- Full connection 400→120
6.- Full connection 120→84

tanh activation functions.

7.- Output layer: softmax 10 classes (84 → 10)

Trainable parameters:
[(5*5+1)*6] + [(5*5+1)*16] + [400*120+120] + [120*84+84] + [84*10+10] = 59706

Original LeNet:
LeCun et al. (1989). Backpropagation applied to handwritten zip code recognition. Neural Computation, 1(4):541-551.
LeNet-5:
LeCun et al. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE. 86(11): 2278-2324.

ImageNet Dataset and Challenge

www.image-net.org/challenges/LSVRC/

- Dataset:
  - ~14 million labeled images
  - ~20k classes
- Annotation:
  - First, images are gathered from Internet using certain keywords.
  - Then, humans filter out the noise (Amazon Mechanical Turk)
- Since 2010, ImageNet Large-Scale Visual Recognition Challenge (ILSVRC): 1.2 million training images, 1000 classes. No longer held after 2017 → now moved to Kaggle.

These **datasets and challenges**, along with the rise of **GPU-computing**, were **key to the success of deep learning**.

Recommended Reading:
[“What I learned from competing against a ConvNet on ImageNet”]() by A.Karpathy

Russakovsky et al. (2015). ImageNet large scale visual recognition challenge. *International journal of computer vision*, *115*, 211-252.

AlexNet: ILSVRC 2012 winner

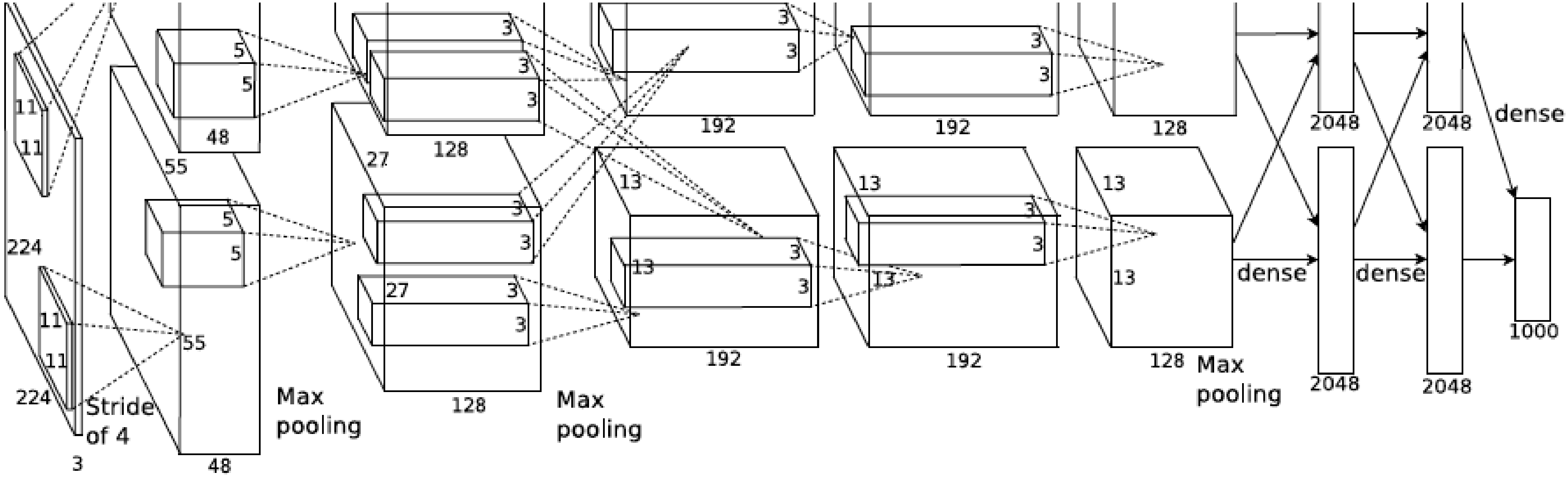


Illustration taken from the original paper, "explicitly showing the delineation of responsibilities between the two GPUs. One GPU runs the layer-parts at the top of the figure while the other runs the layer-parts at the bottom. The GPUs communicate only at certain layers."

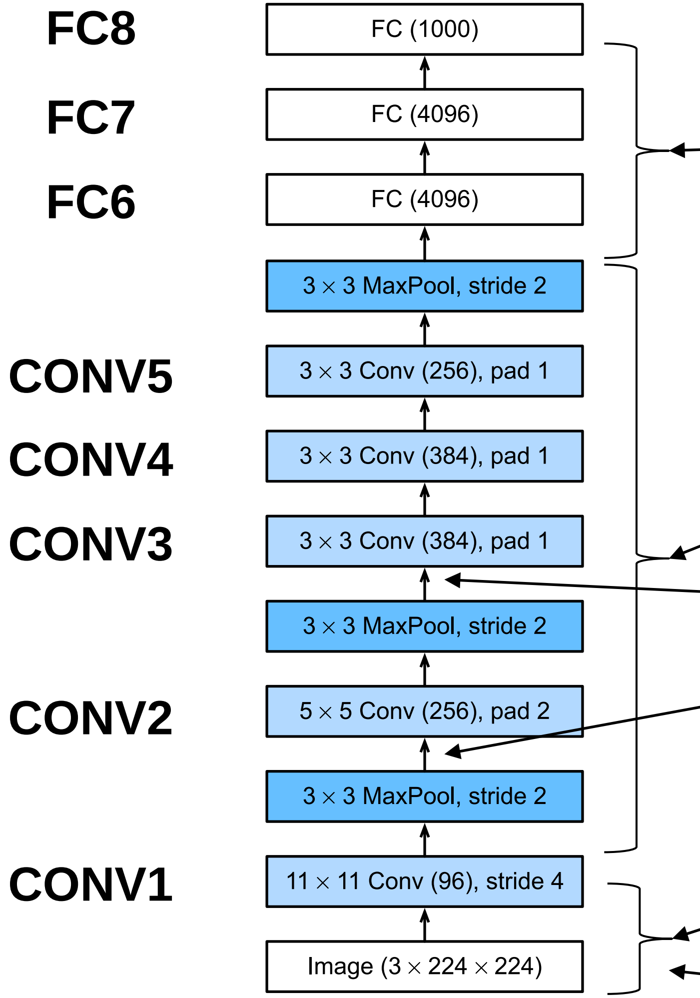
GPU memory is comparatively abundant now, so we rarely need to break up models across GPUs.

- Very similar to LeNet but:
  - **First use of ReLU** nonlinearity (x6 faster training)
  - **Overlapping Max-pooling**
  - More data and **deeper** model (8 layers (5 conv + 3 FC) → **~60M params**)
  - GPU implementation (50x speedup over CPU)
    - Trained on two GPUs for a week
  - Regularization: **Dropout** and **heavy data augmentation** (dataset increased x2048)

One of the top cited papers ever in computer vision and machine learning

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. *NIPS*, *25*. (>120K citations)

AlexNet architecture



Huge fully connected layers. ~96% of all weights are in here… See Roger Grosse's notes and cs231n lecture 6 for more details.

Filter sizes reduce as we move away from the network input: 11x11 → 5x5 → 3x3.

They used a type of normalization here and here that is not used anymore (Local Response Normalization) because it seems it actually has a minimal impact.

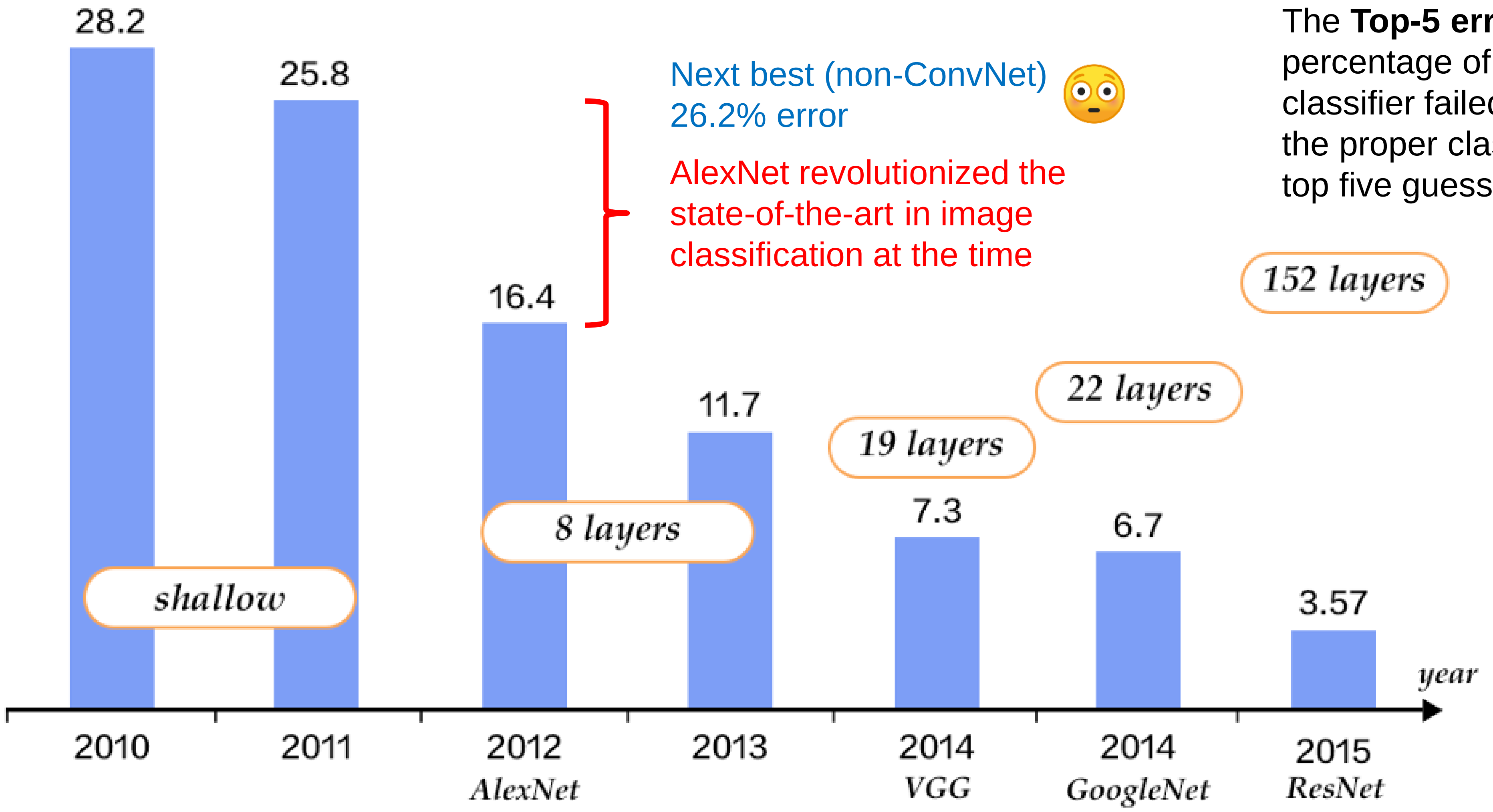
Images in ImageNet are 8 times taller and wider than in MNIST → objects tend to occupy more pixels with more visual detail → larger kernels (11x11 in AlexNet vs 5x5 in LeNet)

The devil is in the details: AlexNet image size should be 227×227×3, instead of 224×224×3, so the math will come out right. See Karpathy's cs231n talk

https://d2l.ai/chapter_convolutional-modern/alexnet.html

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. *NIPS*, *25*.

AlexNet on ILSVRC 2012



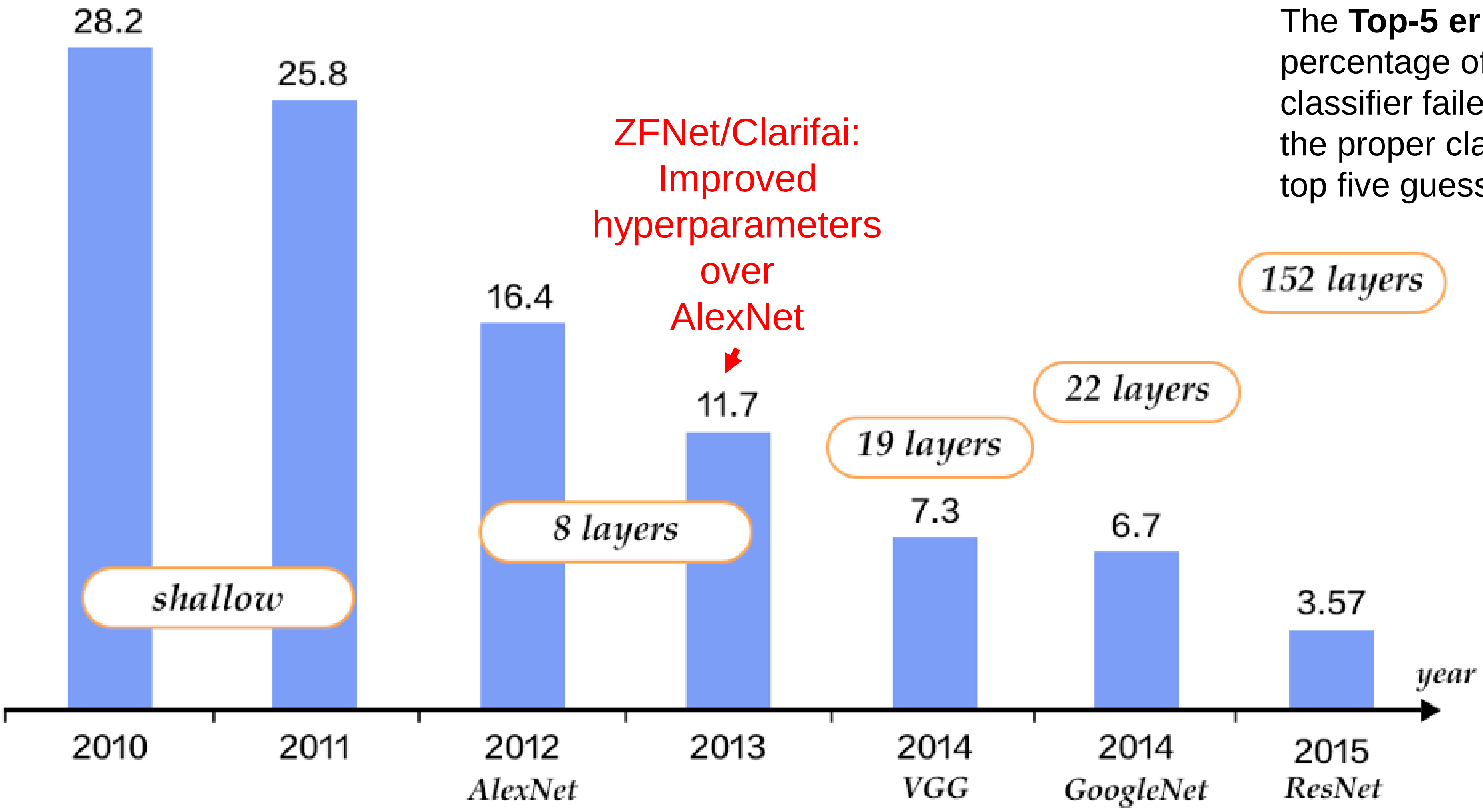
Top-5 error rates on ILSVRC image classification

The **Top-5 error rate** is the percentage of times the classifier failed to include the proper class among its top five guesses.

https://www.cs.toronto.edu/~rgrosse/courses/csc321_2018/tutorials/tut6_slides.pdf

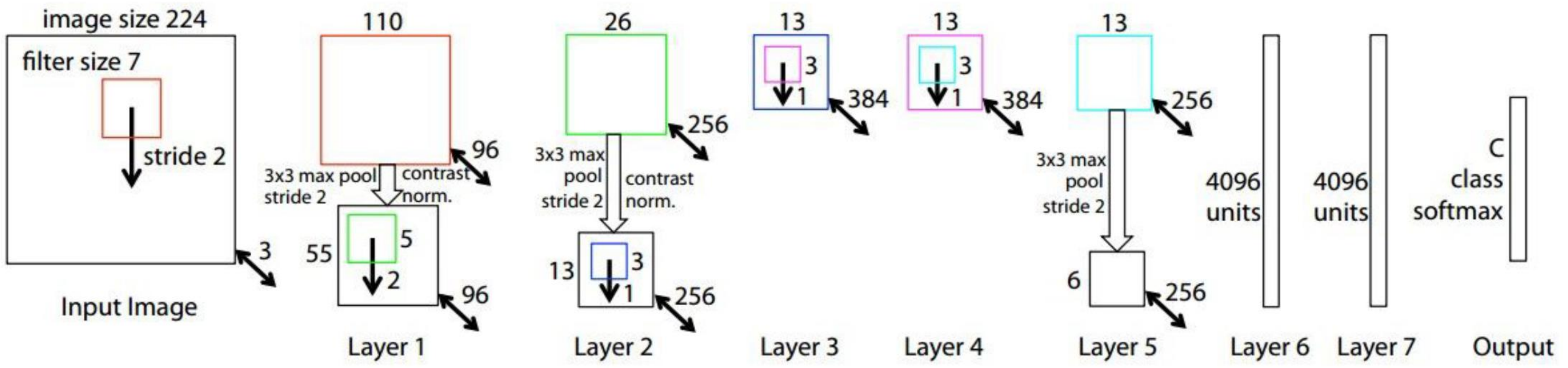
ZFNet: ILSVRC 2013 winner

The **Top-5 error rate** is the percentage of times the classifier failed to include the proper class among its top five guesses.



Top-5 error rates on ILSVRC image classification

ZFNet



They realized that 11x11 filters with stride 4 was too drastic. They opted for performing a denser computation at the beginning.

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

Larger number of filters: increase expressivity

ImageNet top 5 error: 16.4% -> 11.7%

Zeiler, M. D., & Fergus, R. (2014). Visualizing and understanding convolutional networks. In *ECCV* (pp. 818-833)

VGGNet

Small filters, Deeper networks

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

Instead of going crazy with architectural choices (trying many different filters in size and number), they used **only 3x3 CONV stride 1, pad 1** and **2x2 MAX POOL stride 2** (increasing the number of filters as we go deeper; as **spatial size is decreasing, the number of filters is increasing**)

11.7% top 5 error in ILSVRC'13 (ZFNet)

→ 7.3% top 5 error in ILSVRC'14 (VGGNet)

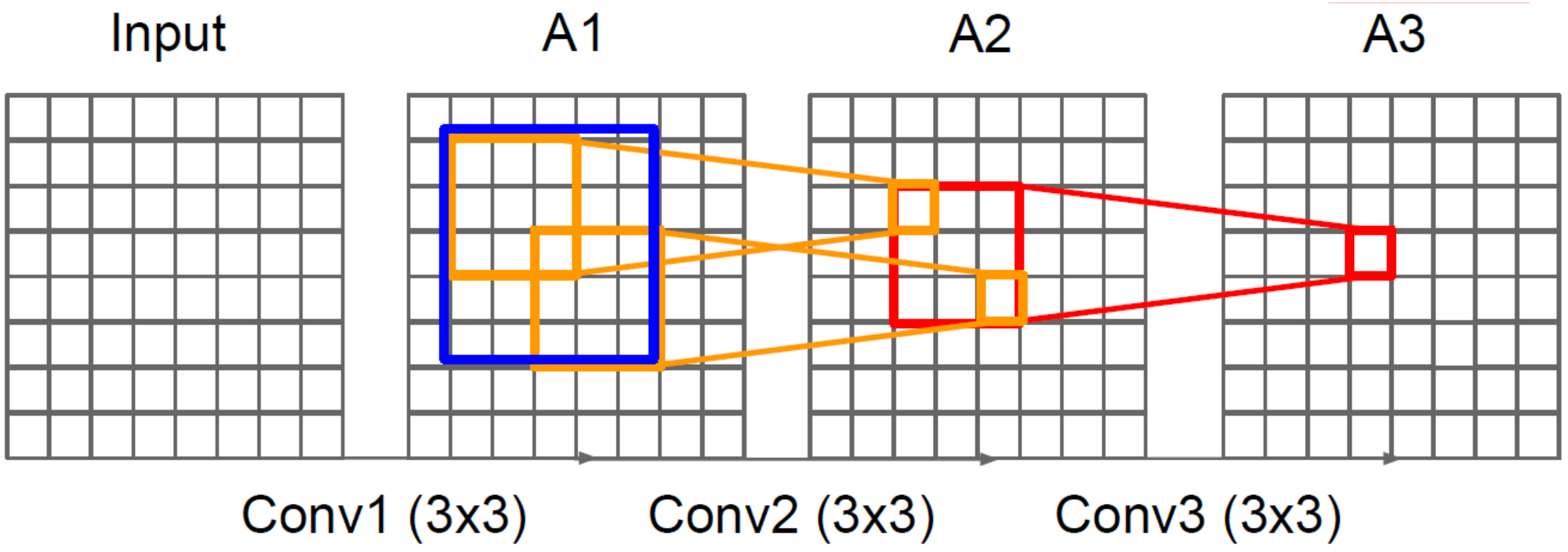
Simonyan, K., & Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. In *ICLR* (pp. 1-14) (>110K citations) ← Top cited paper!

VGGNet

Why use smaller filters? (3x3 conv)

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

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



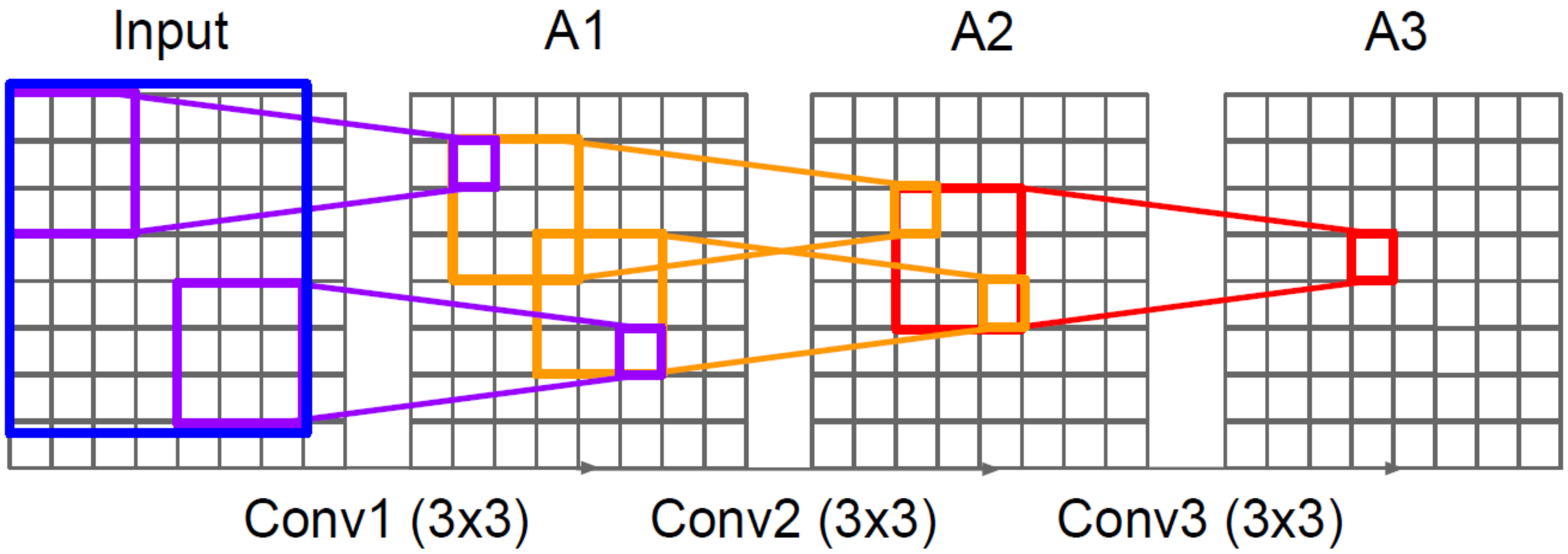
Simonyan, K., & Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. In *ICLR* (pp. 1-14)

VGGNet

Why use smaller filters? (3x3 conv)

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

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



Simonyan, K., & Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. In *ICLR* (pp. 1-14)

VGGNet

Why use smaller filters? (3x3 conv)

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

Deeper (**more non-linearities**)

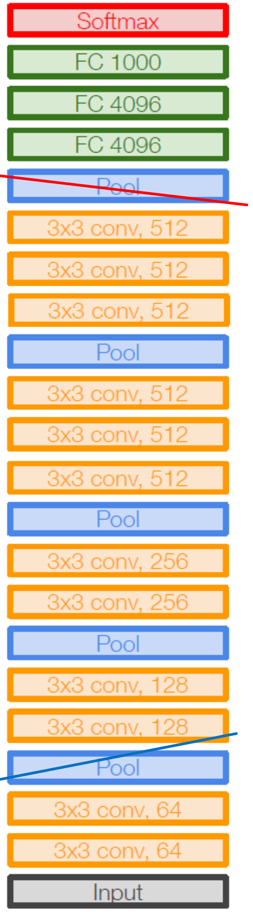
Fewer parameters: $3 \cdot (3^2C^2)$ vs 7^2C^2 for C channels per layer

Simonyan, K., & Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. In *ICLR* (pp. 1-14)

VGGNet

(not counting biases)

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



VGG16

Most memory is in early CONV

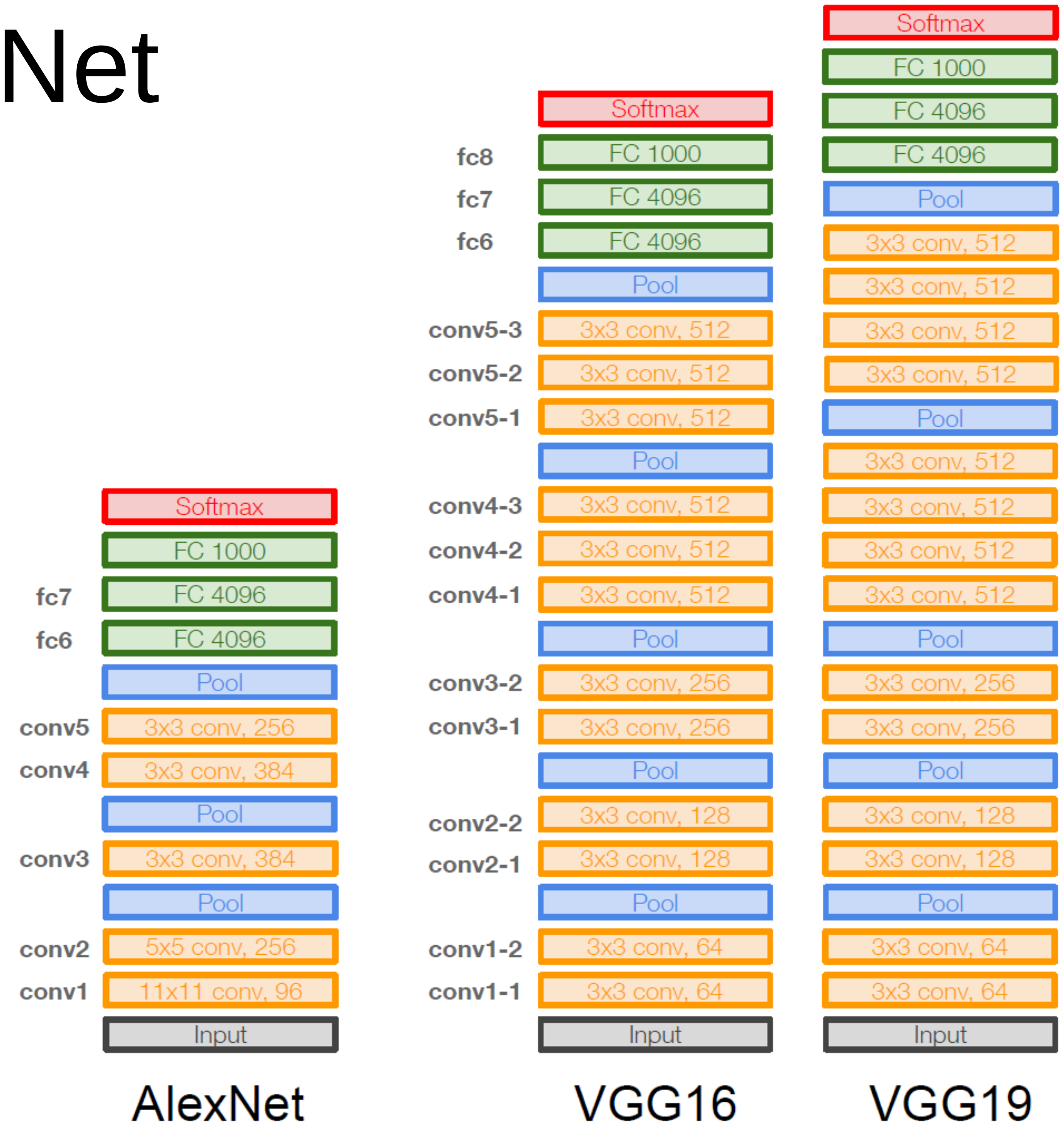
Most params are in late FCs

TOTAL memory: 24M * 4 bytes ~= 96MB / image (for a forward pass)

TOTAL params: 138M parameters vs ~60M (AlexNet) vs ~60K (LeNet-5)

VGGNet

- ILSVRC'14 2nd in classification (GoogLeNet was the winner; see next slides), 1st in localization
- Use VGG16 or VGG19
  - The authors also tested VGG11 and VGG13
- FC7 features generalize well to other tasks



GoogLeNet

This meme was referenced in the original paper: https://arxiv.org/pdf/1409.4842v1.pdf

Szegedy et al. (2015). Going deeper with convolutions. In *CVPR* (pp. 1-9).

GoogLeNet

Deeper networks, with computational efficiency

ILSVRC'14 classification winner (6.7% top 5 error)

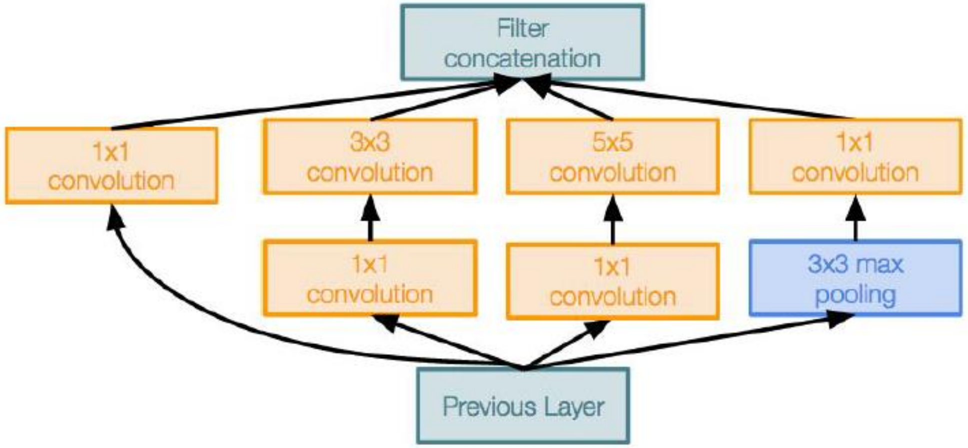
22 layers but… **only 5 million parameters**!

12x less than AlexNet (16.4% top 5 error)

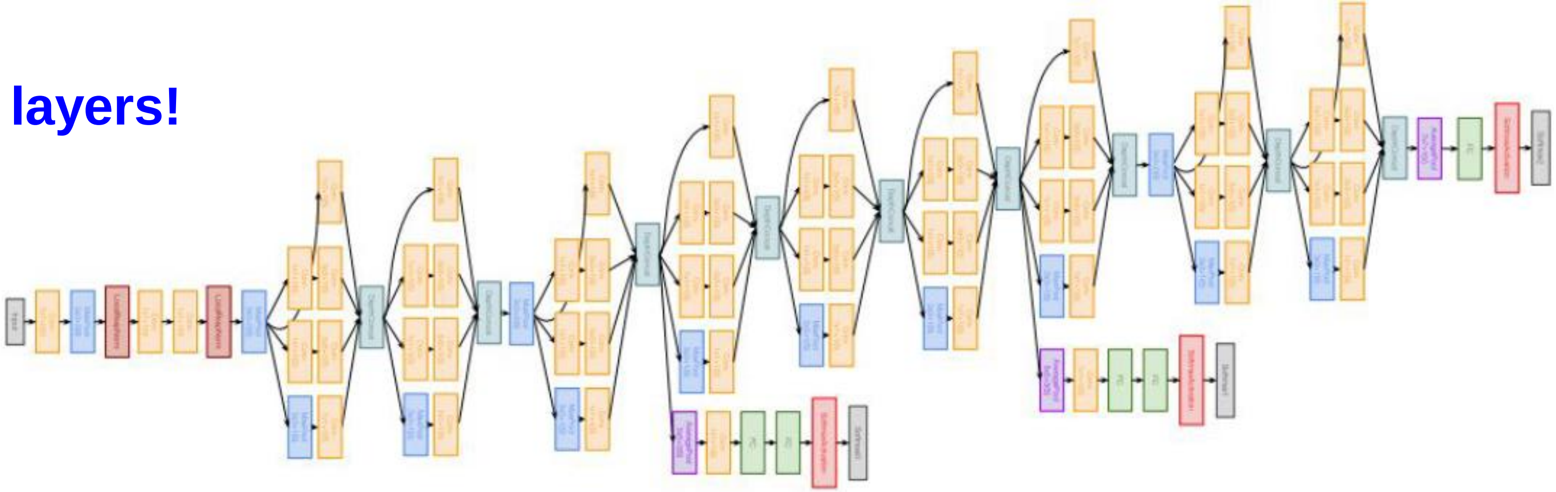
27x less than VGG-16 (7.3% top 5 error)

Efficient "Inception" module

Removes multiple FC layers!



Inception module

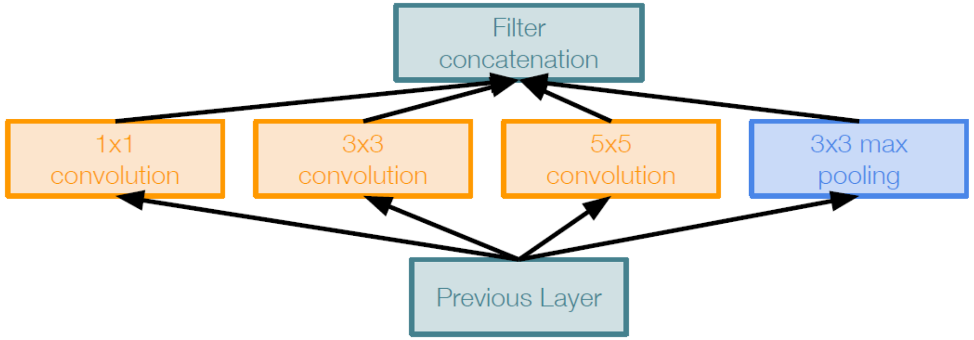


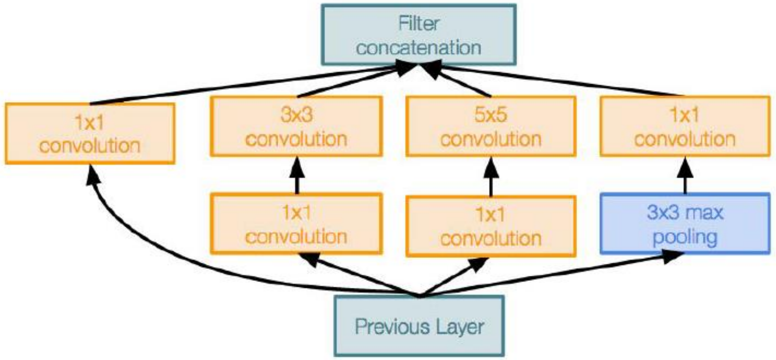
Szegedy et al. (2015). Going deeper with convolutions. In *CVPR* (pp. 1-9).

GoogLeNet

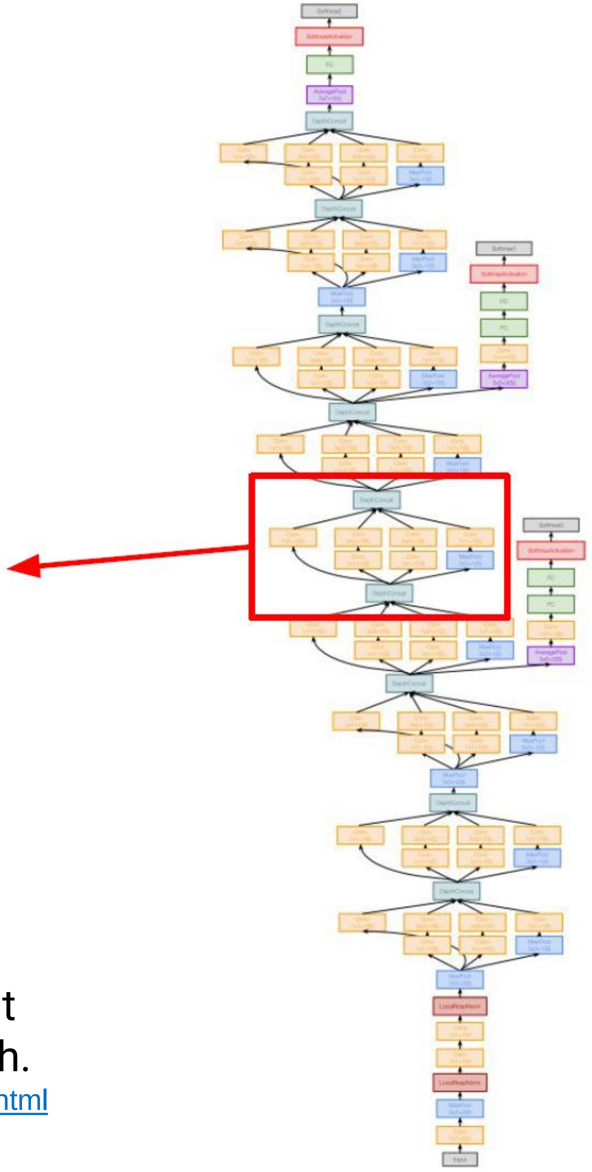
“Inception module”: **design a good local network topology** (network within a network) **and then stack these modules on top of each other.**

Ok. But why so complicated? 🤔
Why not just using something like…?





Inception module



Note: The four branches use appropriate **padding** to give the input and output the same height and width.
https://d2l.ai/chapter_convolutional-modern/googlenet.html

Szegedy et al. (2015). Going deeper with convolutions. In *CVPR* (pp. 1-9).

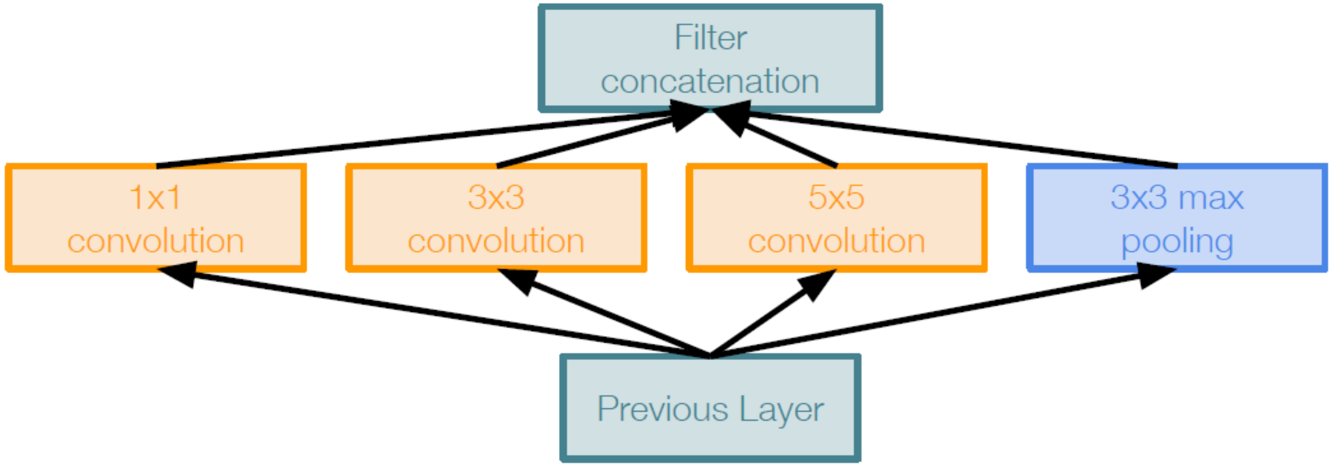
GoogLeNet

Apply parallel filter operations on the input from previous layer:

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

Naive proposal: **concatenate all filter outputs together channel-wise**

What is the problem with this? (As usual, computational complexity)

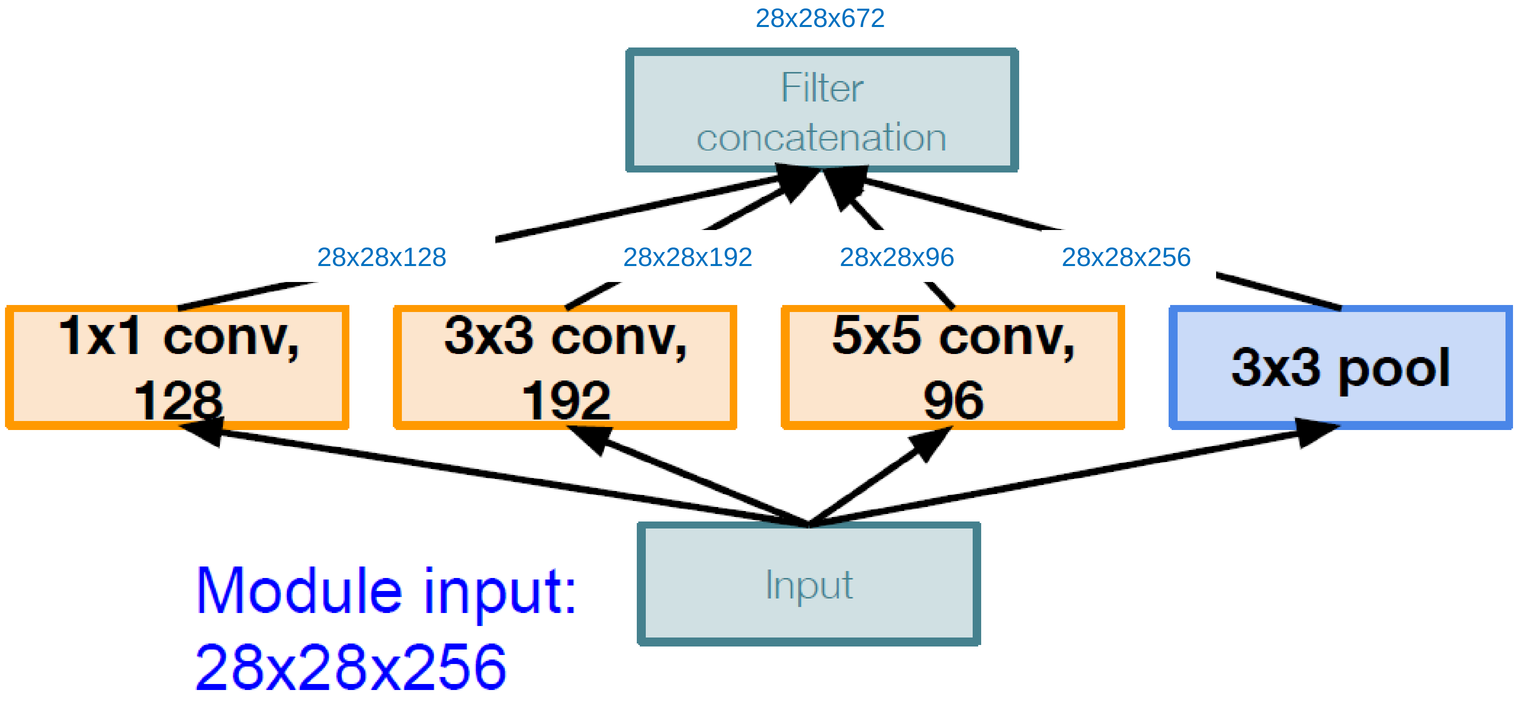


Naive Inception module

Szegedy et al. (2015). Going deeper with convolutions. In *CVPR* (pp. 1-9).

GoogLeNet

Example:



What is output size after filter concatenation?

28x28x(128+192+96+256) =
28x28x672

How many multiplications are we performing?

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

Very expensive to compute

Szegedy et al. (2015). Going deeper with convolutions. In *CVPR* (pp. 1-9).

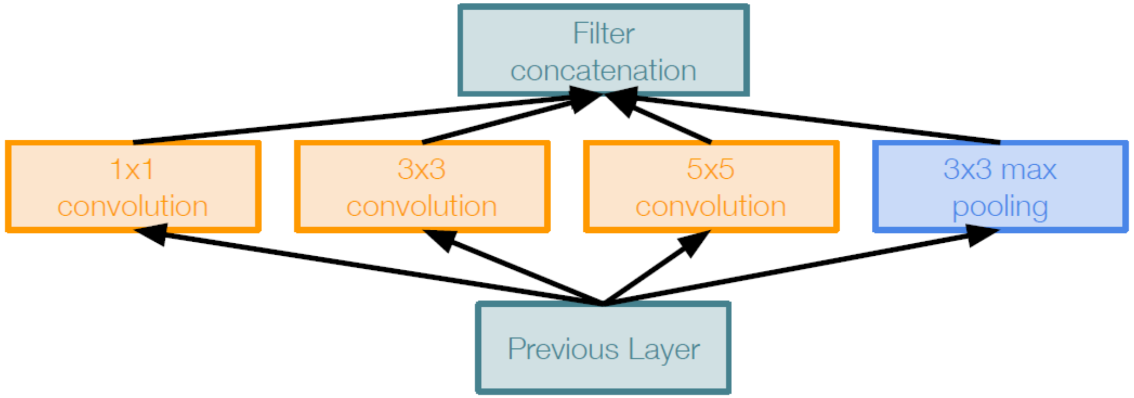
GoogLeNet

Solution:

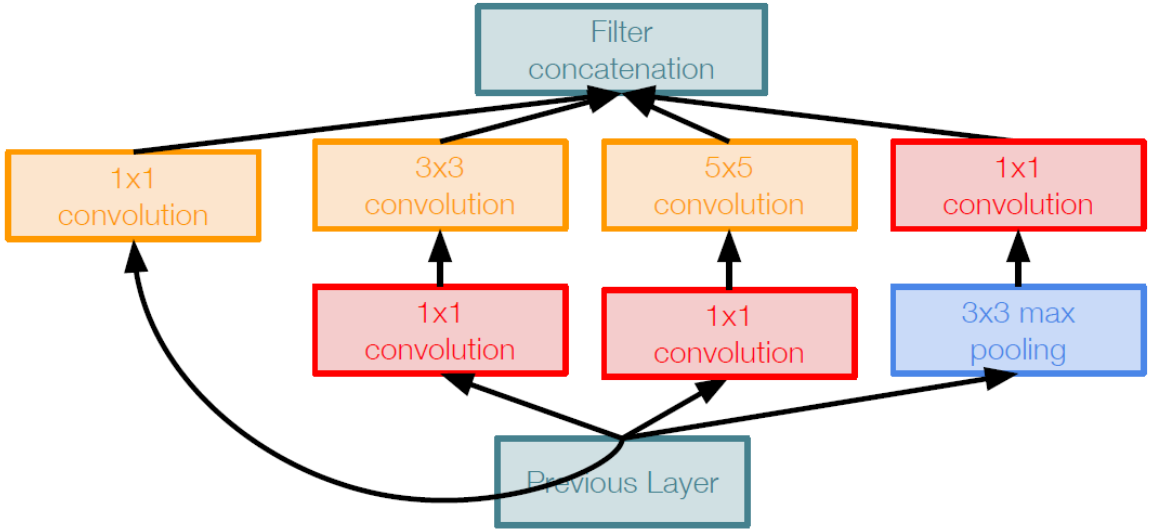
Use **1x1 convolutions** → preserve spatial dimensions, reduce depth!

1x1 convolutions project depth to lower dimension (combination of feature maps)

1x1 conv “bottleneck” layers



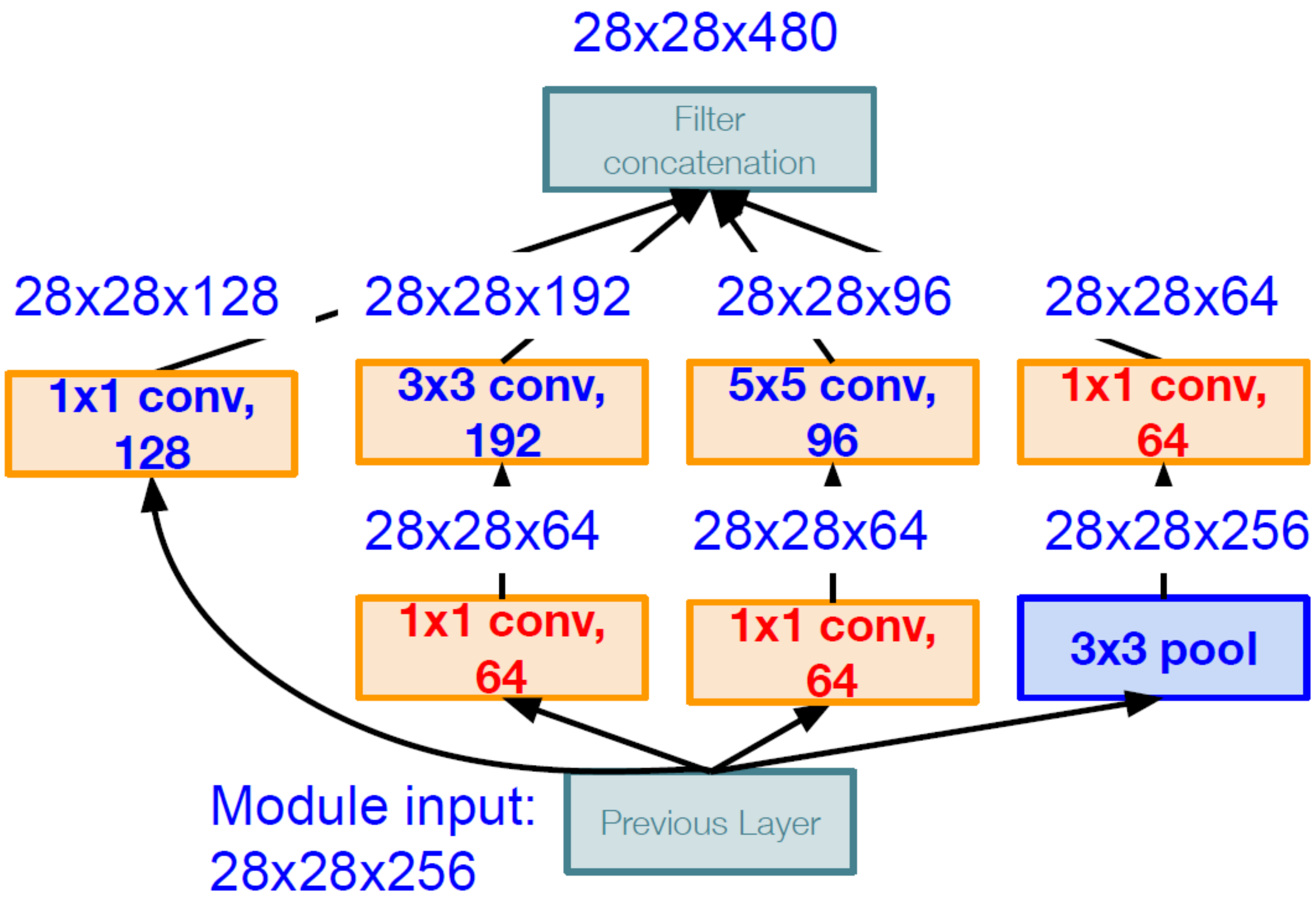
Naive Inception module



Inception module with dimension reduction

Szegedy et al. (2015). Going deeper with convolutions. In *CVPR* (pp. 1-9).

GoogLeNet



Inception module with dimension reduction

Using same parallel layers as naive example, and adding "1x1 conv, 64 filter" bottlenecks:

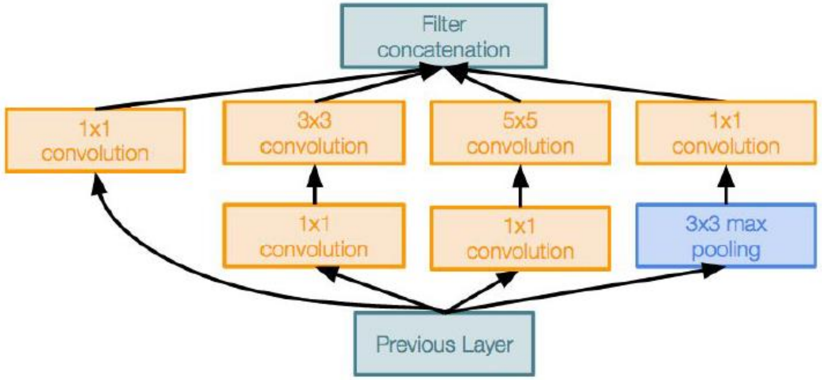
Conv Ops:
[1x1 conv, 64] 28x28x64x1x1x256
[1x1 conv, 64] 28x28x64x1x1x256
[1x1 conv, 128] 28x28x128x1x1x256
[3x3 conv, 192] 28x28x192x3x3x64
[5x5 conv, 96] 28x28x96x5x5x64
[1x1 conv, 64] 28x28x64x1x1x256

Total: 358M ops (vs 854M ops for naive version)

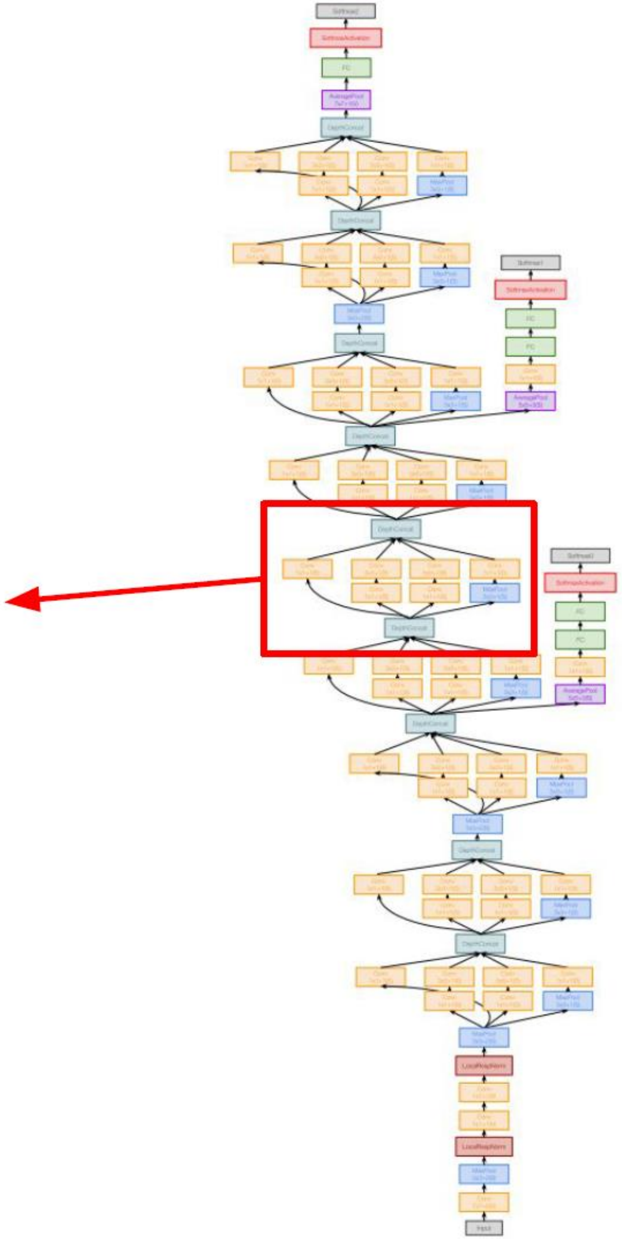
Szegedy et al. (2015). Going deeper with convolutions. In *CVPR* (pp. 1-9).

GoogLeNet

Stack Inception modules
with dimension reduction
on top of each other



Inception module



Szegedy et al. (2015). Going deeper with convolutions. In *CVPR* (pp. 1-9).

GoogLeNet

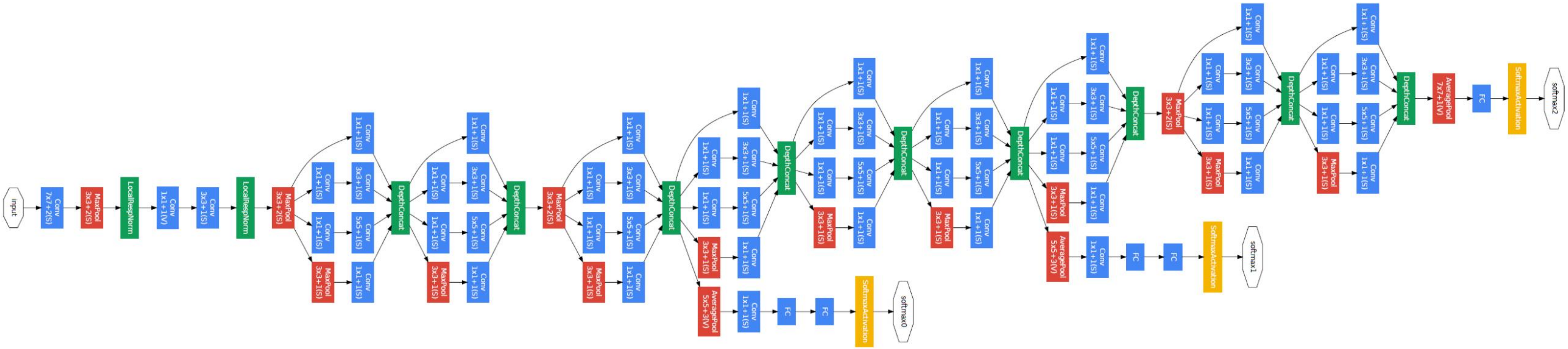
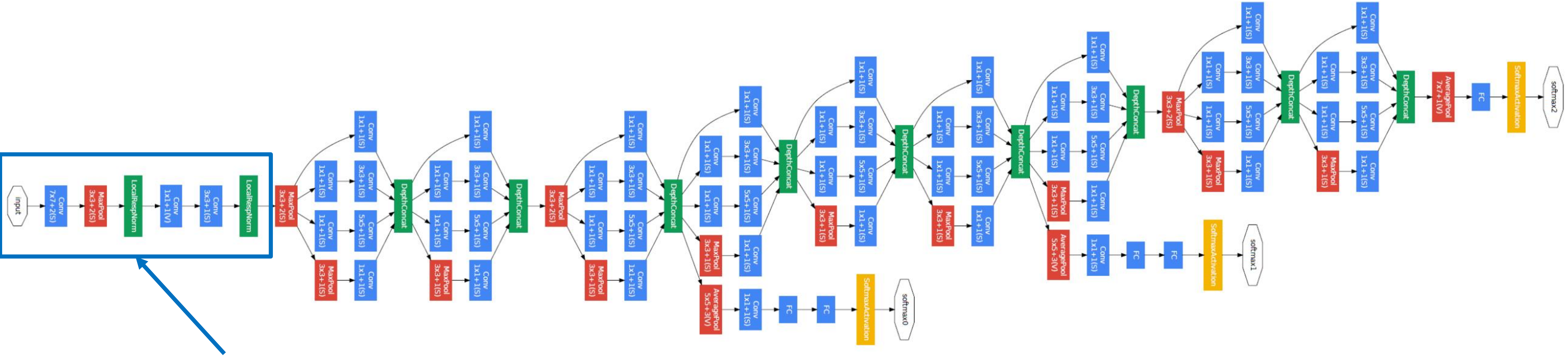


Image source

Notation:
7x7+2(S) → convolutional layer with 7x7 filters, stride=2, padding='same' (i.e., padding=3)
1x1+1(V) → convolutional layer with 1x1 filters, stride=1, padding='valid' (i.e., no padding)

Szegedy et al. (2015). Going deeper with convolutions. In *CVPR* (pp. 1-9).

GoogLeNet



Stem Network:
Conv-Pool-Conv-Conv-Pool

Image source

One of the first networks that exhibited a clear distinction among the stem (data ingest), body (data processing), and head (prediction) in a ConvNet

Szegedy et al. (2015). Going deeper with convolutions. In *CVPR* (pp. 1-9).

GoogLeNet

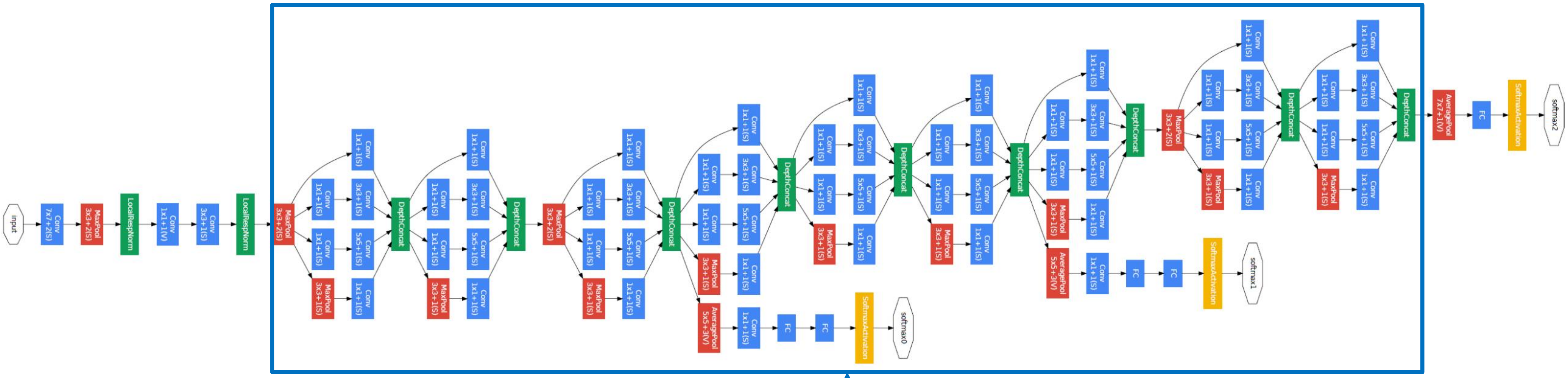


Image source

Stacked Inception Modules

One of the first networks that exhibited a clear distinction among the stem (data ingest), body (data processing), and head (prediction) in a ConvNet

Szegedy et al. (2015). Going deeper with convolutions. In *CVPR* (pp. 1-9).

GoogLeNet

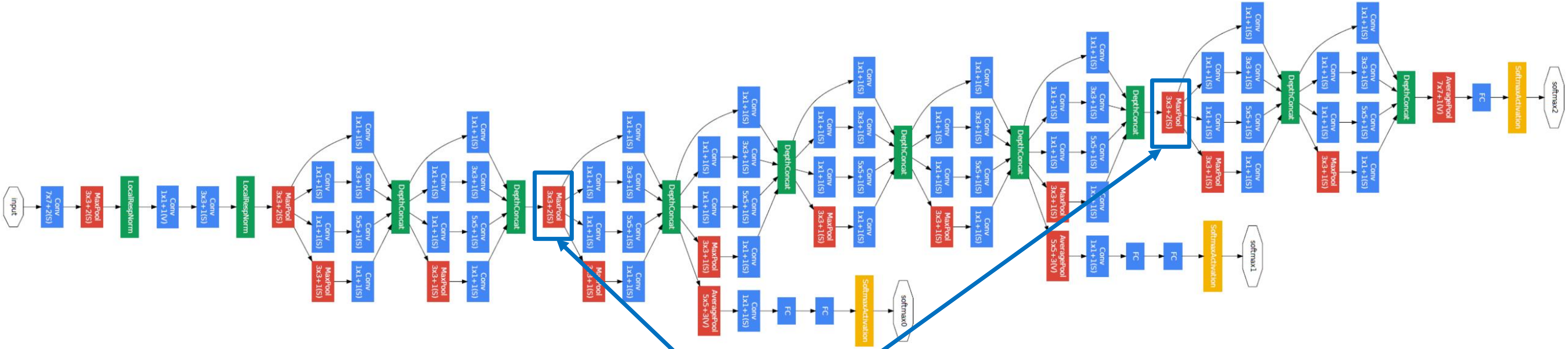


Image source

Max-pooling between Inception blocks reduces the dimensionality

One of the first networks that exhibited a clear distinction among the stem (data ingest), body (data processing), and head (prediction) in a ConvNet

Szegedy et al. (2015). Going deeper with convolutions. In *CVPR* (pp. 1-9).

GoogLeNet

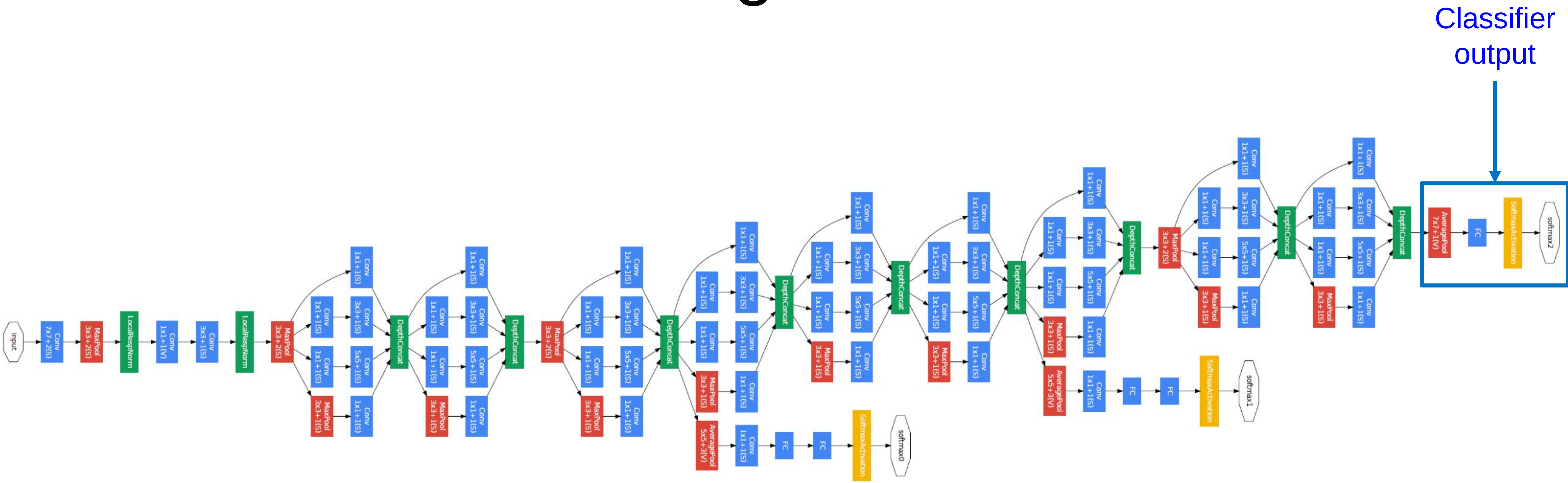


Image source

One of the first networks that exhibited a clear distinction among the stem (data ingest), body (data processing), and head (prediction) in a ConvNet

Szegedy et al. (2015). Going deeper with convolutions. In *CVPR* (pp. 1-9).

GoogLeNet

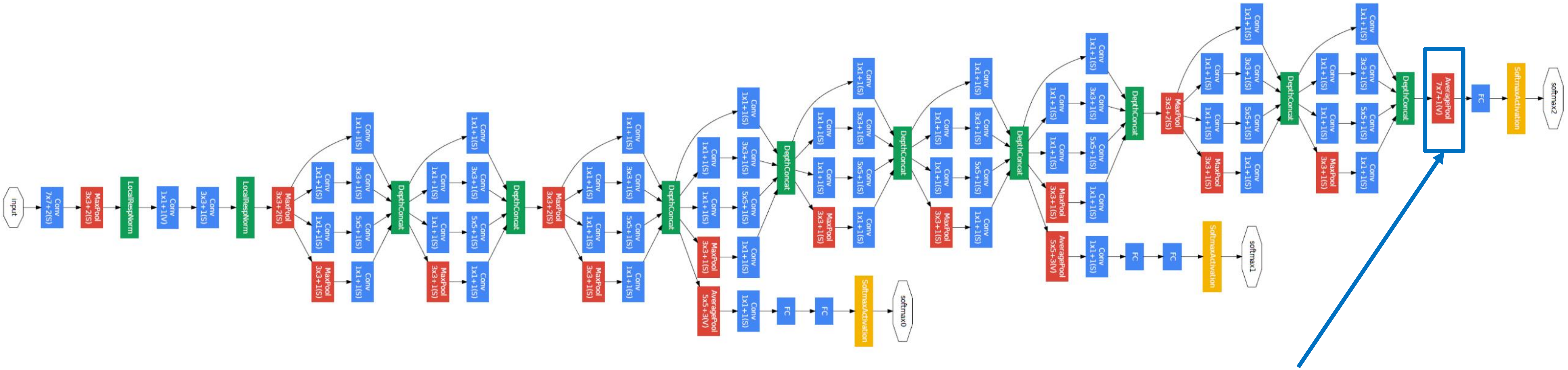
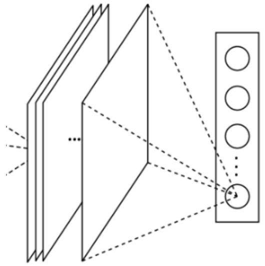


Image source



After the last convolutional layer, a **global average pooling** layer (introduced by Lin et al. in "Network In Network" (2013)) is used to spatially average across each feature maps. **No longer multiple expensive FC layers!**

Szegedy et al. (2015). Going deeper with convolutions. In *CVPR* (pp. 1-9).

GoogLeNet

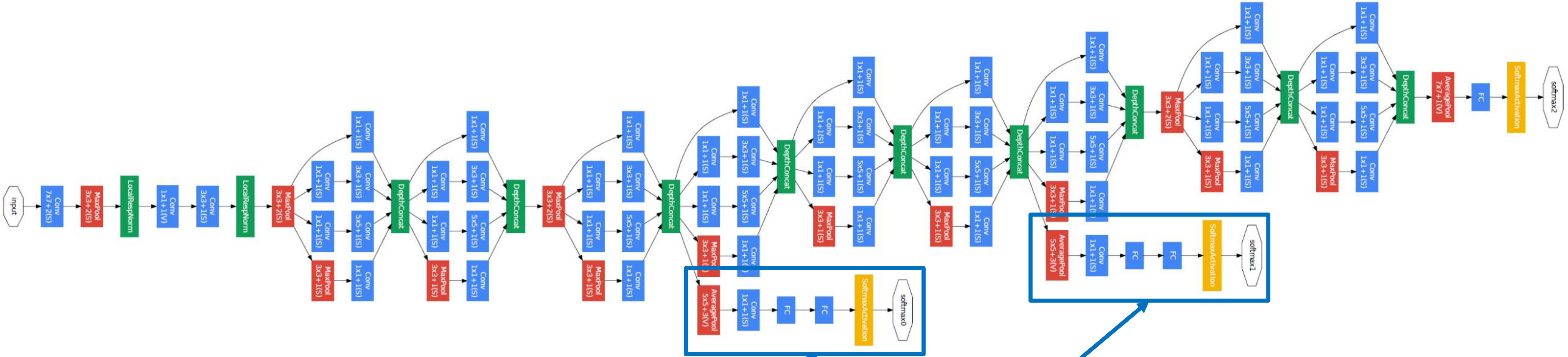


Image source

Motivation: Push gradients to the lower layers to improve convergence during training by combating the vanishing gradient problem in very deep networks.

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

Szegedy et al. (2015). Going deeper with convolutions. In *CVPR* (pp. 1-9).

GoogLeNet

GoogLeNet combines:

- **1x1 convolutions** and **global average pooling** (like in Network in Network)
- **repeated blocks** (like in VGG)
- a **cocktail of convolution kernels** (instead of manually selecting the best kernel size)
- **auxiliary classifiers** to increase gradient signal that gets propagated back

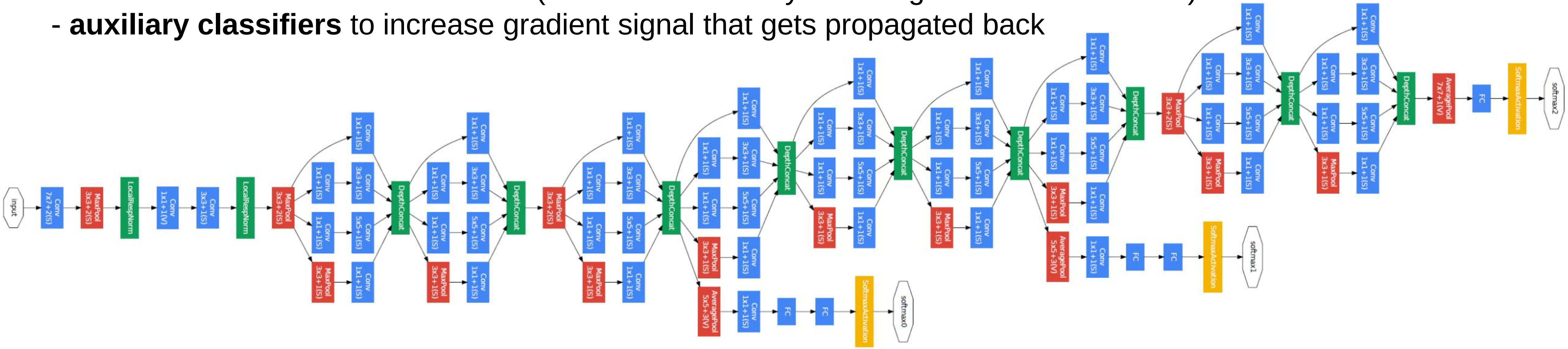


Image source

22 total layers with weights

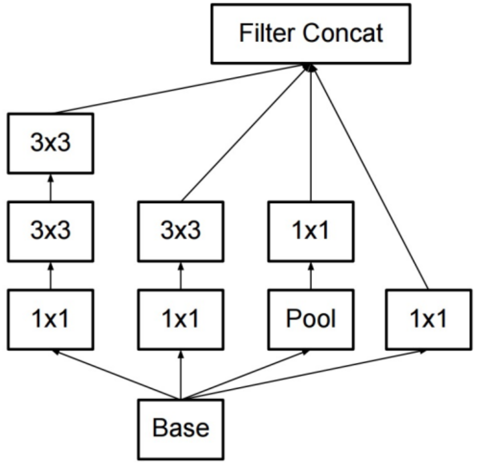
(parallel layers count as 1 layer → 2 layers per Inception module. Don’t count auxiliary output layers)

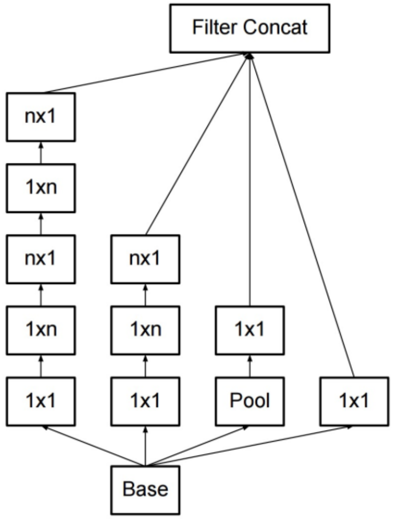
Szegedy et al. (2015). Going deeper with convolutions. In *CVPR* (pp. 1-9).

Inception-v2 and Inception-v3

- The paper where **batch normalization** was presented (Ioffe&Szegedy, 2015) **used Inception** (without Local Response Normalization ("we found that with Batch Normalization it is not necessary") and using other tricks).

- Szegedy et al. (2016) presented two refined versions: **Inception-v2 and Inception-v3**.
  - Factorization of filters
    - They factorize $5x5$ convolution into two stacked $3x3$ convolution
    - They factorize nxn convolution into a combination of $1xn$ convolution and $nx1$ convolution. They call it "asymmetric convolution".
      - Note: these are not like the separable 2D kernels from image processing; they directly learn 1D kernels (i.e. they do not factorize, using e.g. SVD, a 2D kernel as two 1D kernels).
  - More details can be found in https://hackmd.io/@machine-learning/SkD5Xd4DL





Szegedy et al. (2016). Rethinking the Inception architecture for computer vision. In *CVPR* (pp. 2818-2826).

Xception

- Residual connections (that we'll see later)

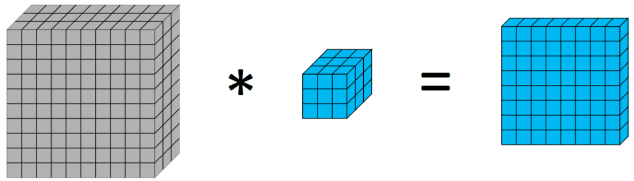
- **Depthwise separable convolutions**

"The mapping of cross-channels correlations and spatial correlations in the feature maps of convolutional neural networks can be *entirely* decoupled."

  - In fact, the paper tries to **replace Inception modules with depthwise separable convolutions** (i.e. by building models that would be stacks of depthwise separable convolutions)

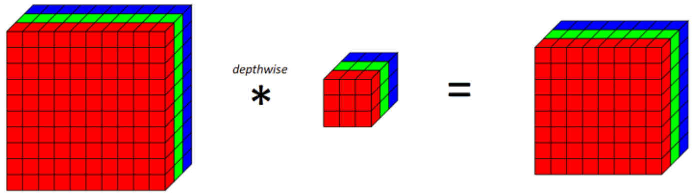
	Top-1 accuracy	Top-5 accuracy
VGG-16	0.715	0.901
ResNet-152	0.770	0.933
Inception V3	0.782	0.941
Xception	**0.790**	**0.945**

We want to apply 64 convolutional filters to our input 10x10 RGB image:

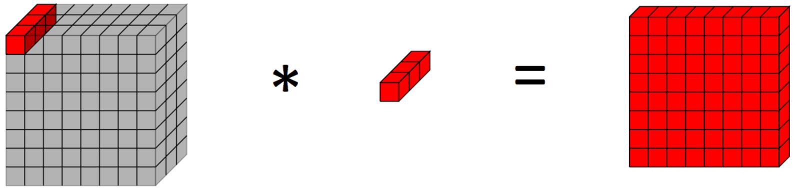


We apply a kernel (3x3x3) over the whole input volume (10x10x3). 3x3x3x64 + 64 = **1792 parameters**

VS



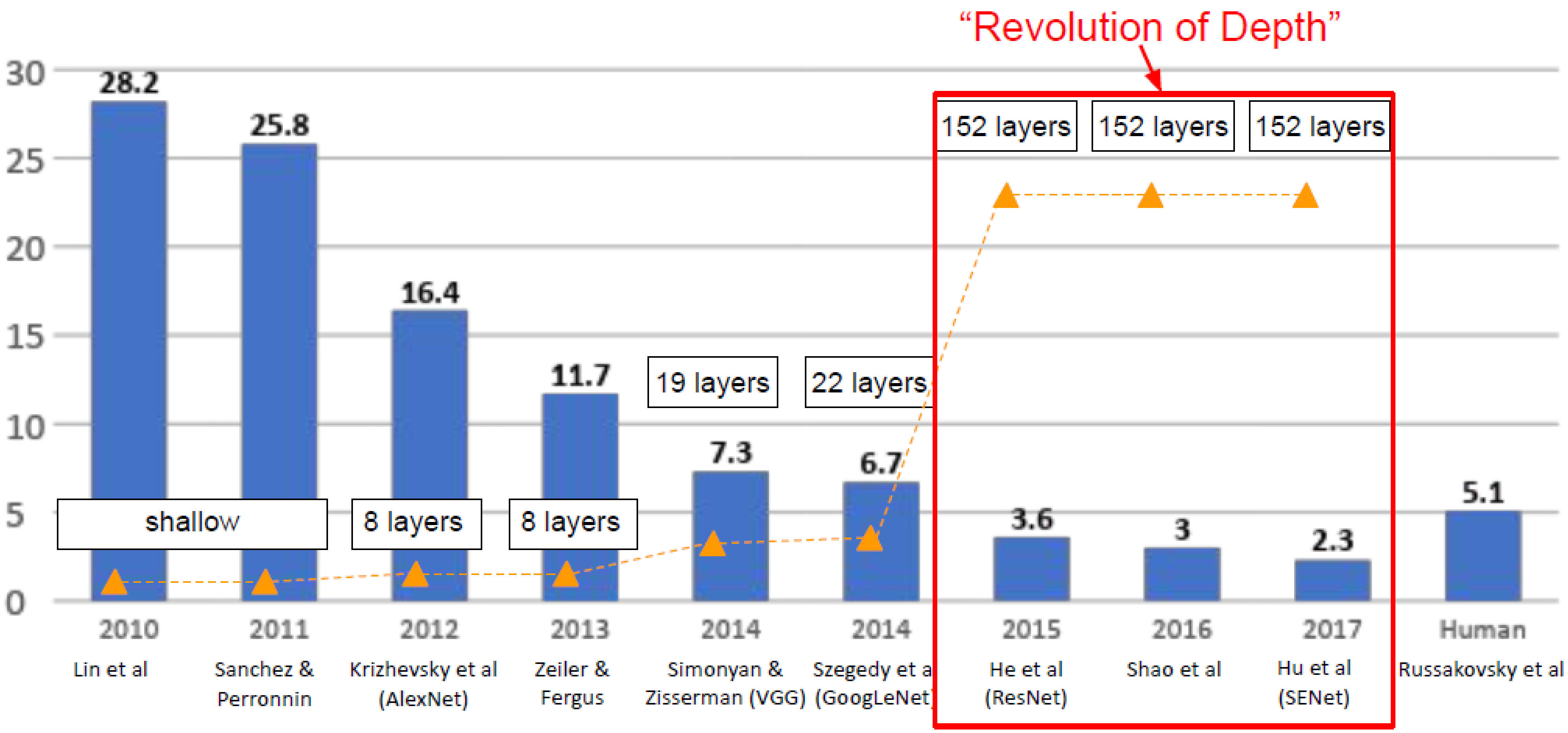
We apply one kernel (3x3x1) per channel in the input volume (10x10x3). We then apply a pointwise convolutional layer (1x1x3) on the resulting volume (8x8x3). (3x3x1x3 + 3) + (1x1x3x64 + 64) = **286 parameters**



https://machinelearningmastery.com/using-depthwise-separable-convolutions-in-tensorflow/

Chollet, F. (2017). Xception: Deep learning with depthwise separable convolutions. In *CVPR* (pp. 1251-1258).

ResNet and "the revolution of depth"

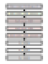


Network **depth** is of **crucial** importance, but

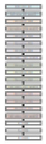
is learning better networks as easy as stacking more layers?

ResNet: ILSVRC 2015 winner

AlexNet, 8 layers
(ILSVRC 2012)



VGG, 19 layers
(ILSVRC 2014)



ResNet, 152 layers
(ILSVRC 2015)

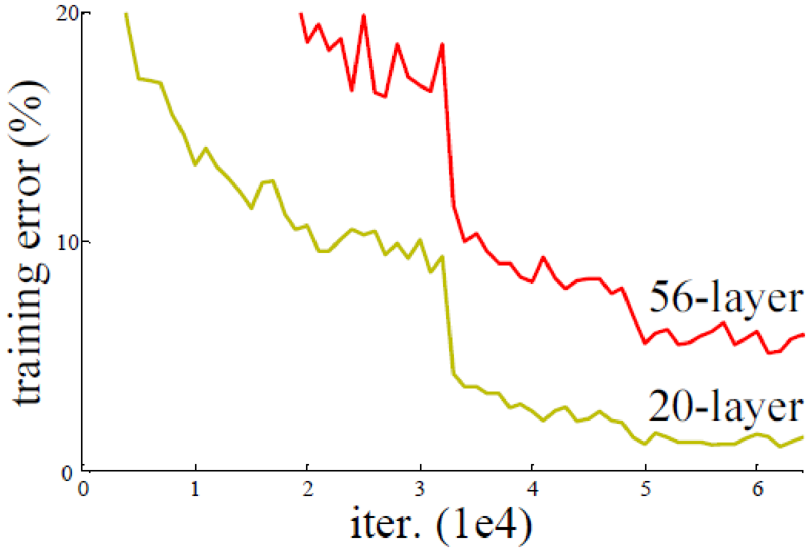
Skip, shortcut or
residual connections

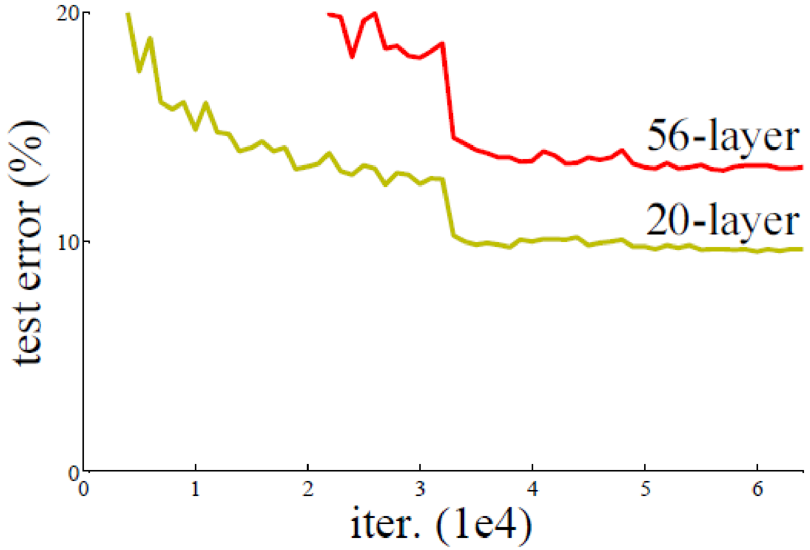
2-3 weeks of training on 8 GPU machine.

He et al. (2016). Deep residual learning for image recognition. In *CVPR* (pp. 770-778).

ResNet

You have to be careful with how do you increase the number of layers!
If you do it just naively, it does not seem to work very well (“degradation problem”).





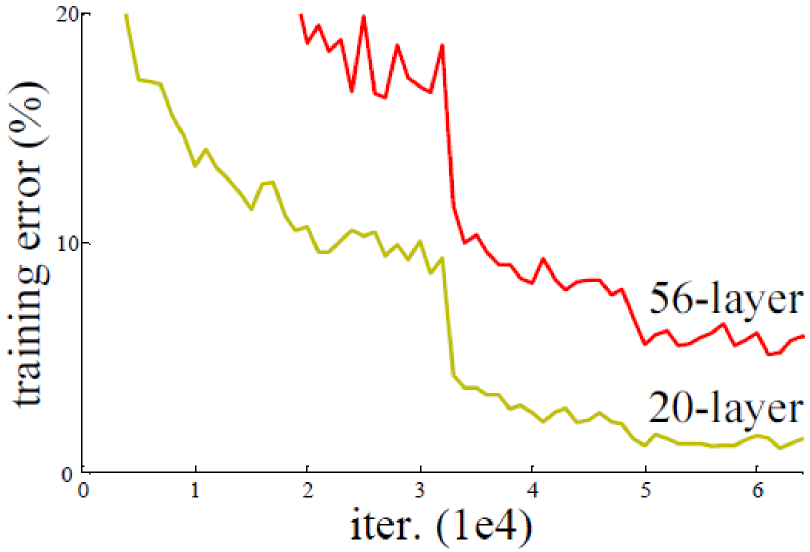
What happens when we continue stacking deeper layers on a “plain” ConvNet?

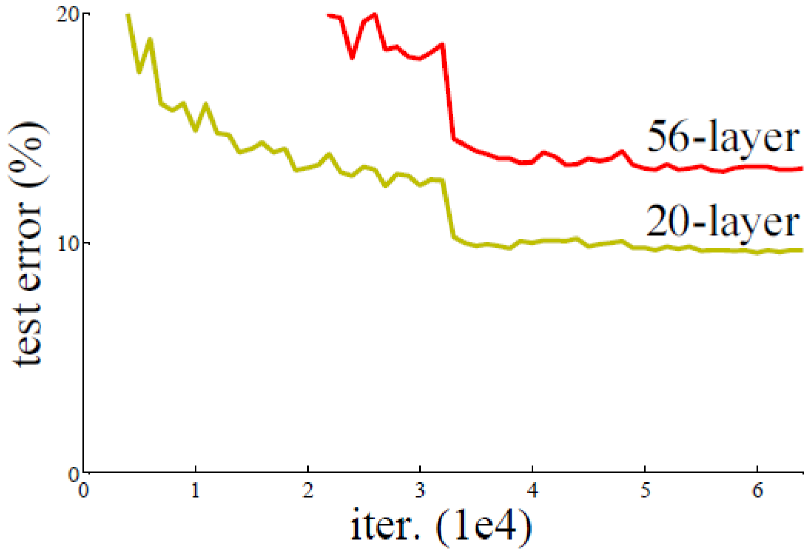
56-layer model performs worse on both test and training error!!!

→ A deeper model performs worse, but it’s not caused by overfitting!

He et al. (2016). Deep residual learning for image recognition. In *CVPR* (pp. 770-778).

ResNet



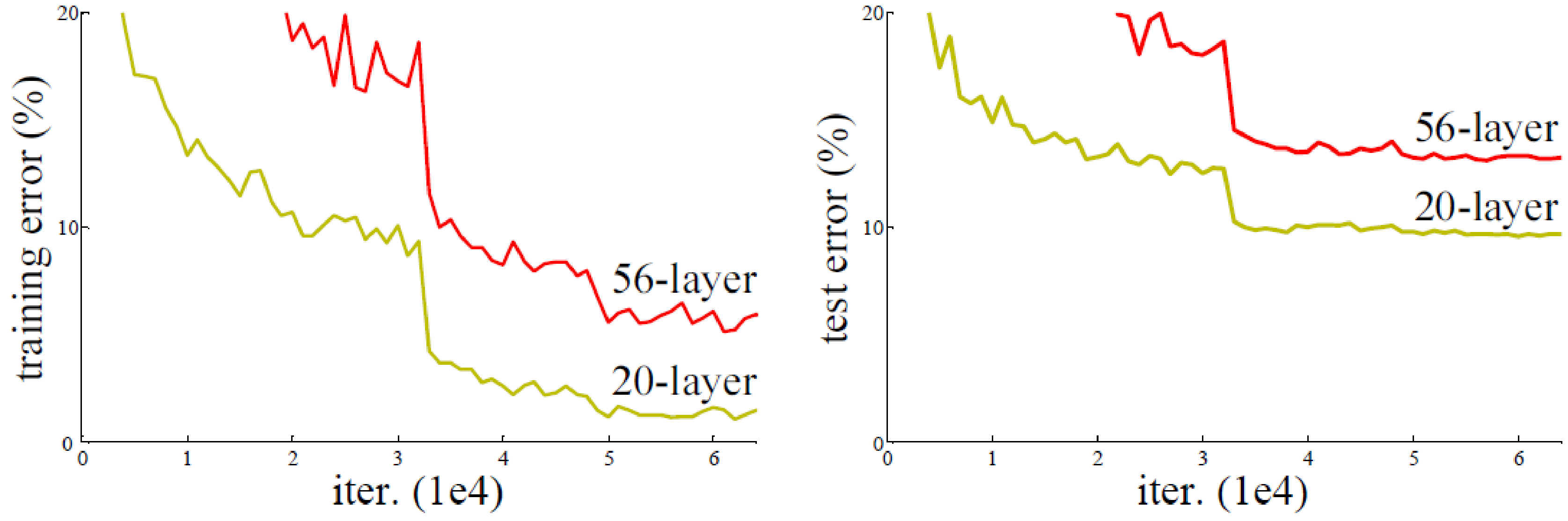


Deep models have **more representation power** (more parameters) than shallower models.

Hypothesis: **not all networks are similarly easy to optimize**

He et al. (2016). Deep residual learning for image recognition. In *CVPR* (pp. 770-778).

ResNet



If deeper models have **more representation power** than shallower models, they should be, at least, as good as them.

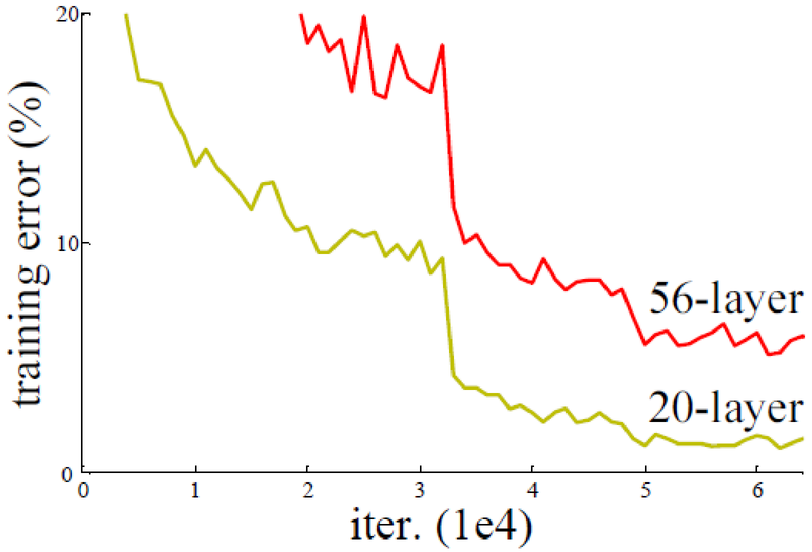
We can choose padding and a specific convolutional filter to “embed” shallower networks.

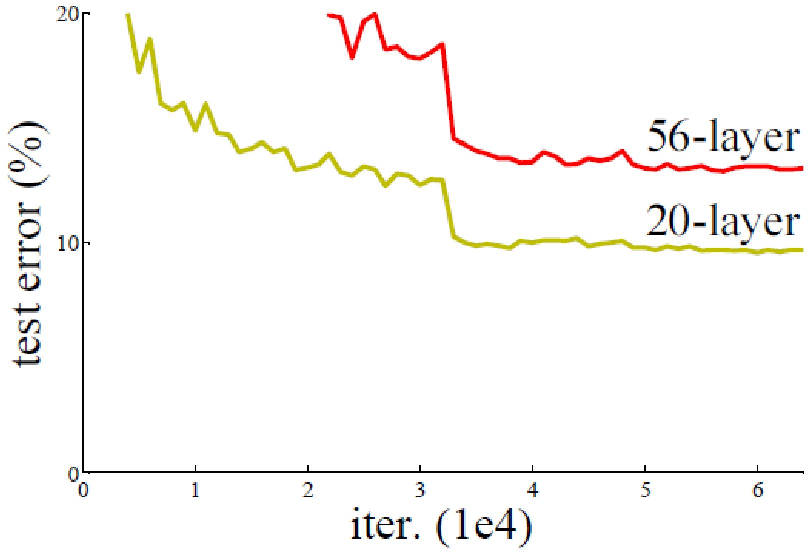
0	0	0
0	1	0
0	0	0

With ‘SAME’ padding, this will output the same feature map it receives as input

He et al. (2016). Deep residual learning for image recognition. In *CVPR* (pp. 770-778).

ResNet





Although identity is representable, learning it may be difficult for optimization methods.

Intuition: **Tweak the network so it doesn't have to learn identity connections → RESIDUAL BLOCKS!**

0	0	0
0	1	0
0	0	0

With 'SAME' padding, this will output the same feature map it receives as input

He et al. (2016). Deep residual learning for image recognition. In *CVPR* (pp. 770-778).

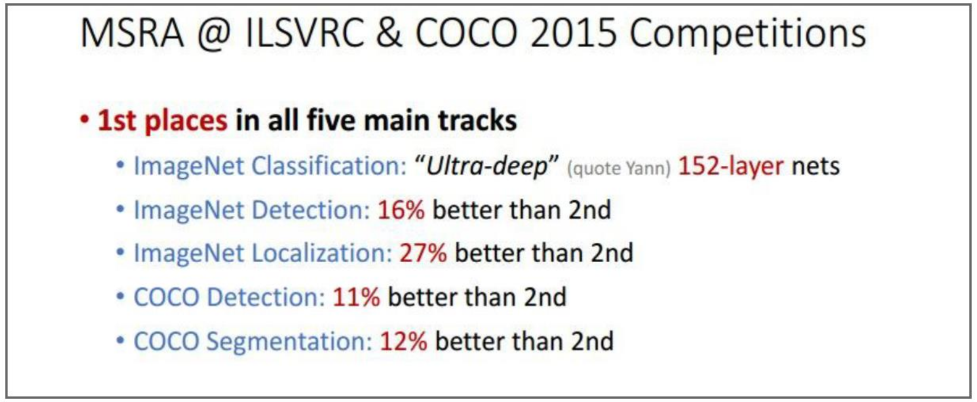
ResNet

Very deep networks using residual connections

- 152-layer model for ImageNet, 1202 layers on CIFAR-10

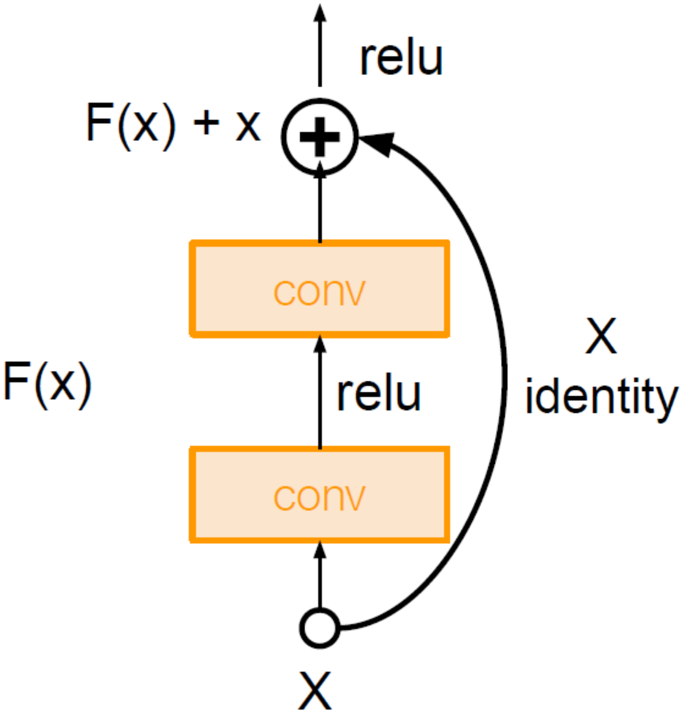
- ILSVRC’15 classification winner (3.57% top 5 error)

- Swept all classification and detection competitions in ILSVRC’15 and COCO’15!


MSRA @ ILSVRC & COCO 2015 Competitions

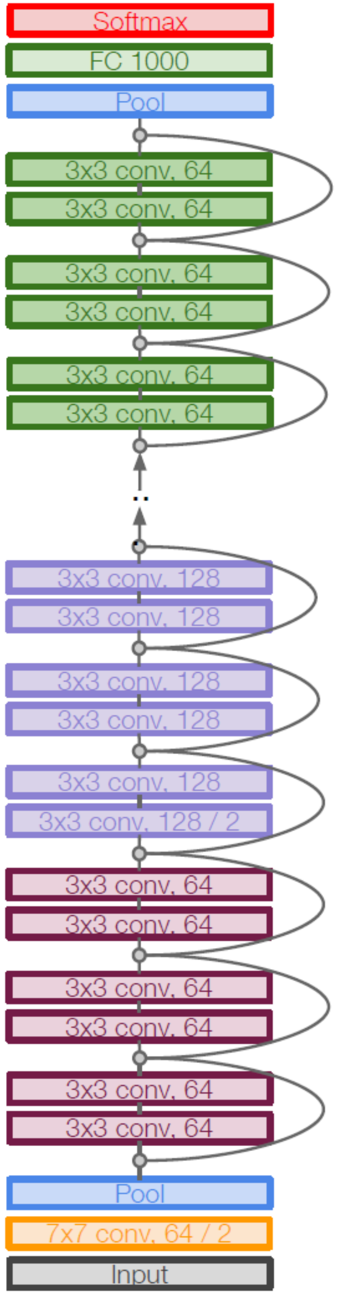
- **1st places in all five main tracks**
  - ImageNet Classification: *“Ultra-deep”* (quote Yann) 152-layer nets
  - ImageNet Detection: 16% better than 2nd
  - ImageNet Localization: 27% better than 2nd
  - COCO Detection: 11% better than 2nd
  - COCO Segmentation: 12% better than 2nd

- GoogLeNet uses modules made up of Inception blocks.
ResNet uses modules made up of residual blocks



Residual block

One of the top cited papers ever in computer vision and machine learning

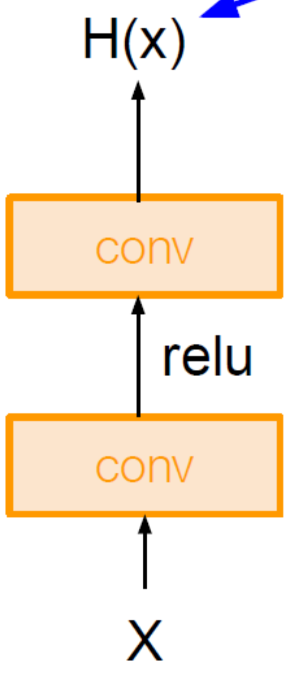


He et al. (2016). Deep residual learning for image recognition. In *CVPR* (pp. 770-778). (>190K citations)

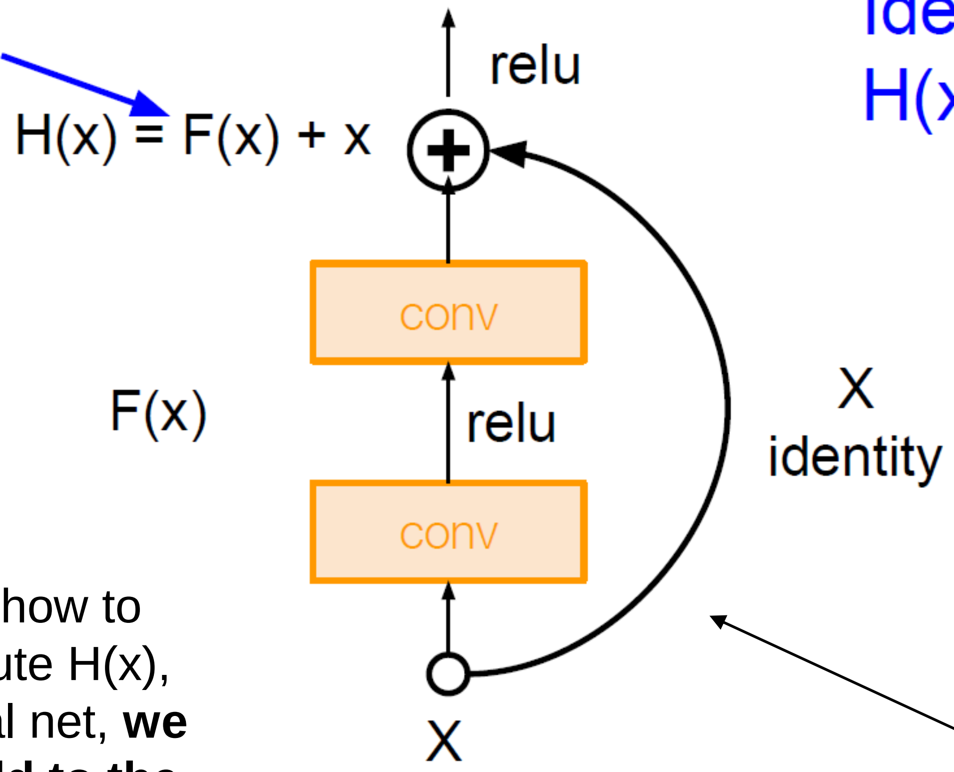
"We hypothesize that it is easier to optimize the residual mapping than to optimize the original, unreferenced mapping"

We want to compute this function H(x).

$H(x) = F(x) + x$



"Plain" layers



Residual block

Identity mapping:
$H(x) = x$ if $F(x) = 0$

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

Instead of learning how to transform x to compute H(x), like in a normal neural net, **we compute what to add to the input x to compute H(x)**. **We learn the residual mapping F(x)=H(x)-x**

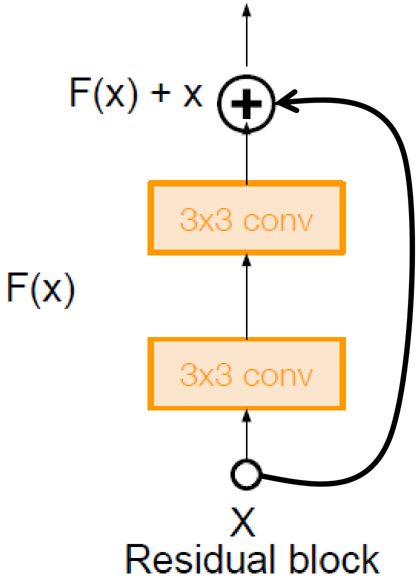
This allows the **gradient to flow from very deep layers to the input**.

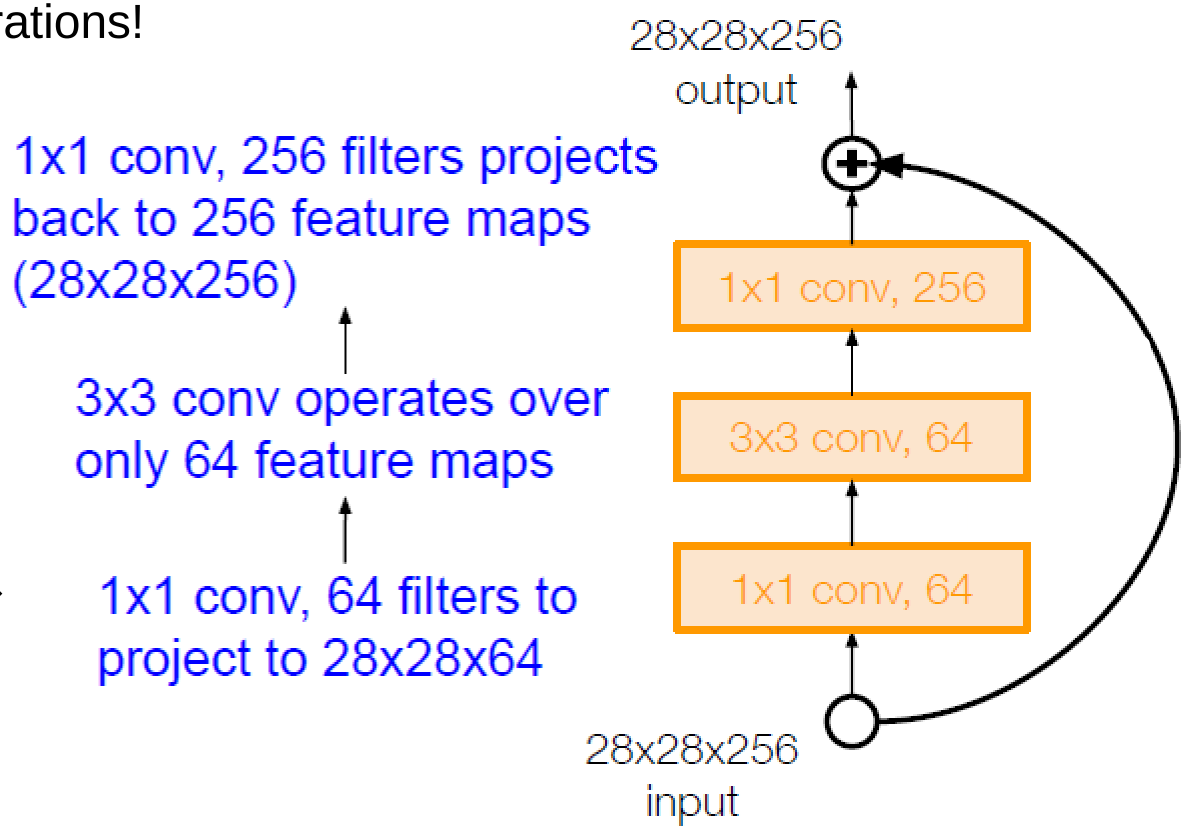
He et al. (2016). Deep residual learning for image recognition. In *CVPR* (pp. 770-778).

ResNet

For deeper networks (ResNet-50+), use “bottleneck” layers (1x1 conv) to improve efficiency (similar to GoogLeNet)

→ Directly performing 3x3 convolutions with 256 feature maps at input and output involves more operations!

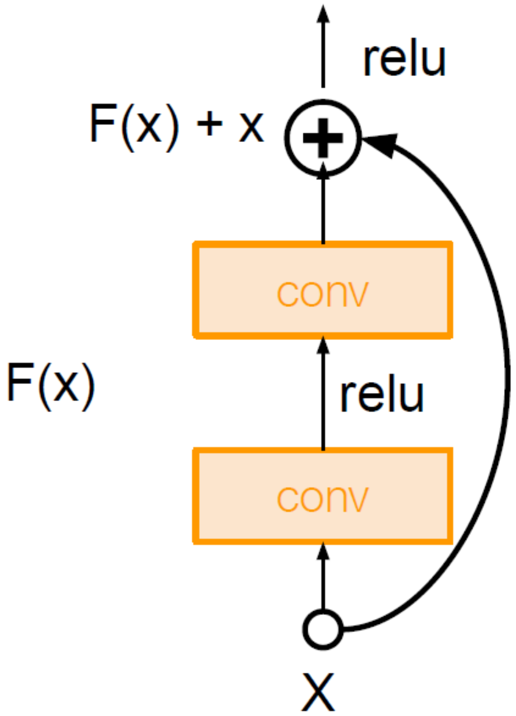




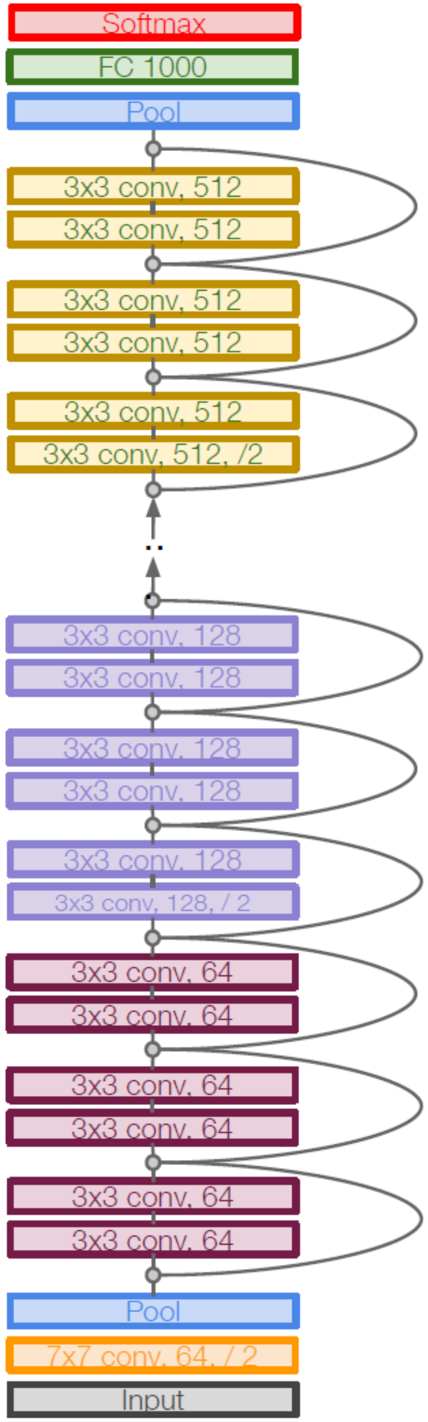
He et al. (2016). Deep residual learning for image recognition. In *CVPR* (pp. 770-778).

ResNet

- Stack residual blocks. Every residual block has two 3x3 conv layers (VGG style)
- Each conv layer is followed by a BN layer (applied before the activation function)
- No dropout used
- Periodically, double # of filters and downsample spatially using stride 2 (/2 in each dimension)
- Additional conv layer at the beginning (stem)
- No FC layers at the end (only FC 1000 to output classes). Global avg pooling after last conv layer



Residual block



He et al. (2016). Deep residual learning for image recognition. In *CVPR* (pp. 770-778).

ResNet

- Architectures for ImageNet:

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

To perform addition with the suitable depth:
(A) Extra zero entries padded for increasing dimensions (no extra parameter)
(B) 1x1 convolutions are used to match dimensions

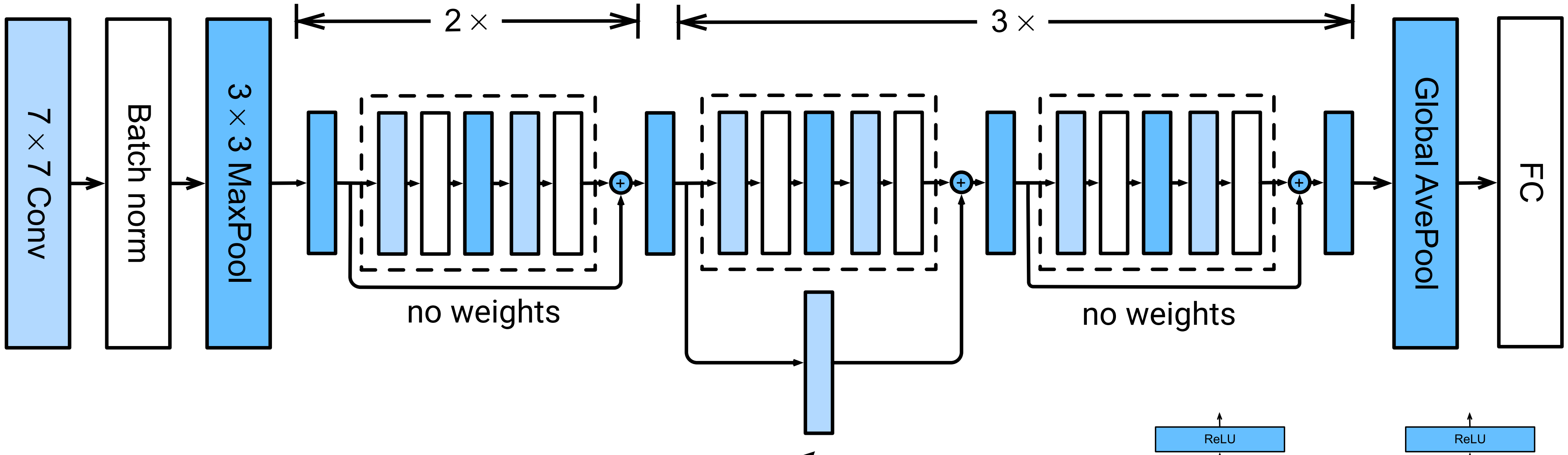
Both are good (and clearly better than a plain network), but (B) is slightly better.

He et al. (2016). Deep residual learning for image recognition. In *CVPR* (pp. 770-778).

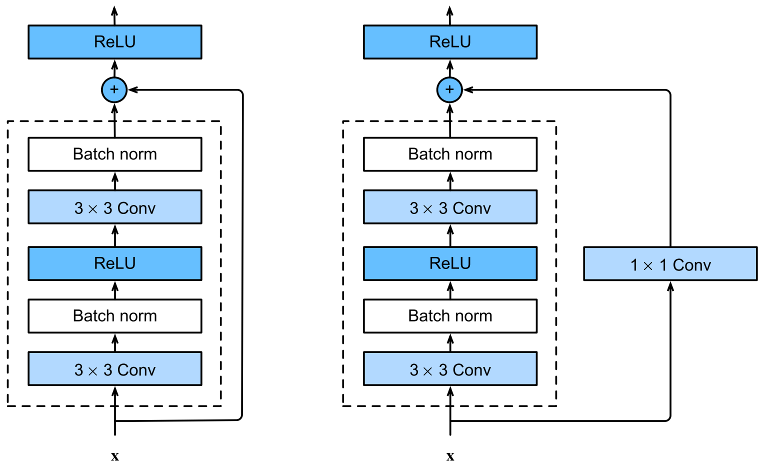
ResNet

ResNet-18 architecture: 17 conv layers + 1 FC layer

https://d2l.ai/chapter_convolutional-modern/resnet.html



We could have an additional 1×1 conv layer to transform the input into the desired shape for the addition operation. See option (B) in previous slide.



He et al. (2016). Deep residual learning for image recognition. In *CVPR* (pp. 770-778).

Deep Learning is Easy - Learn Something Harder by Ferenc Huszár

Deep learning is powerful exactly because it makes hard things easy.

“The research field of deep learning touches on a lot of interesting, very complex topics from machine learning, statistics, optimization, geometry and so on. The slice of deep learning most people are likely to come across - **the lego block building aspect** - however **is relatively simple and straightforward**. […] **it is important to see beyond this simple surface**, and pick some of the harder concepts to master.”

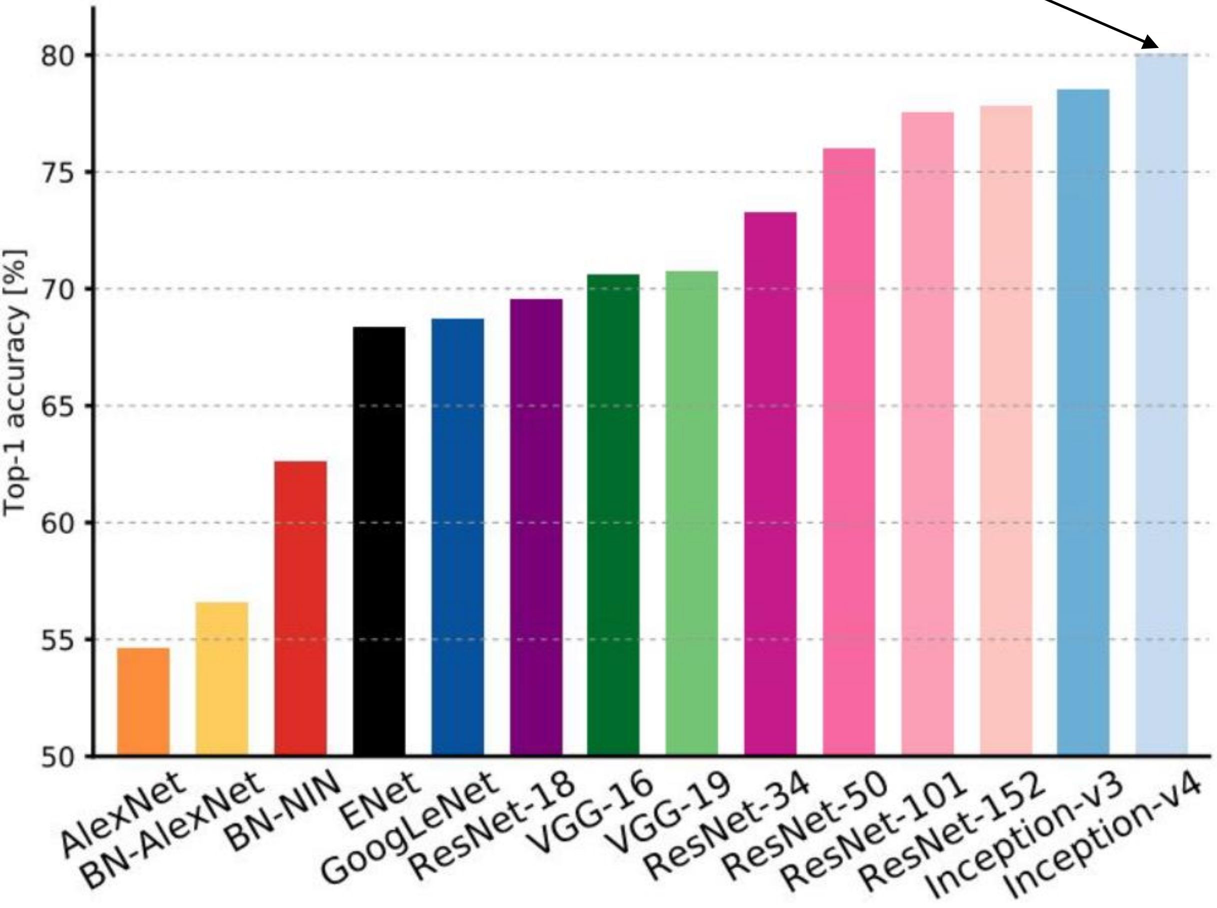
Comparing complexity…

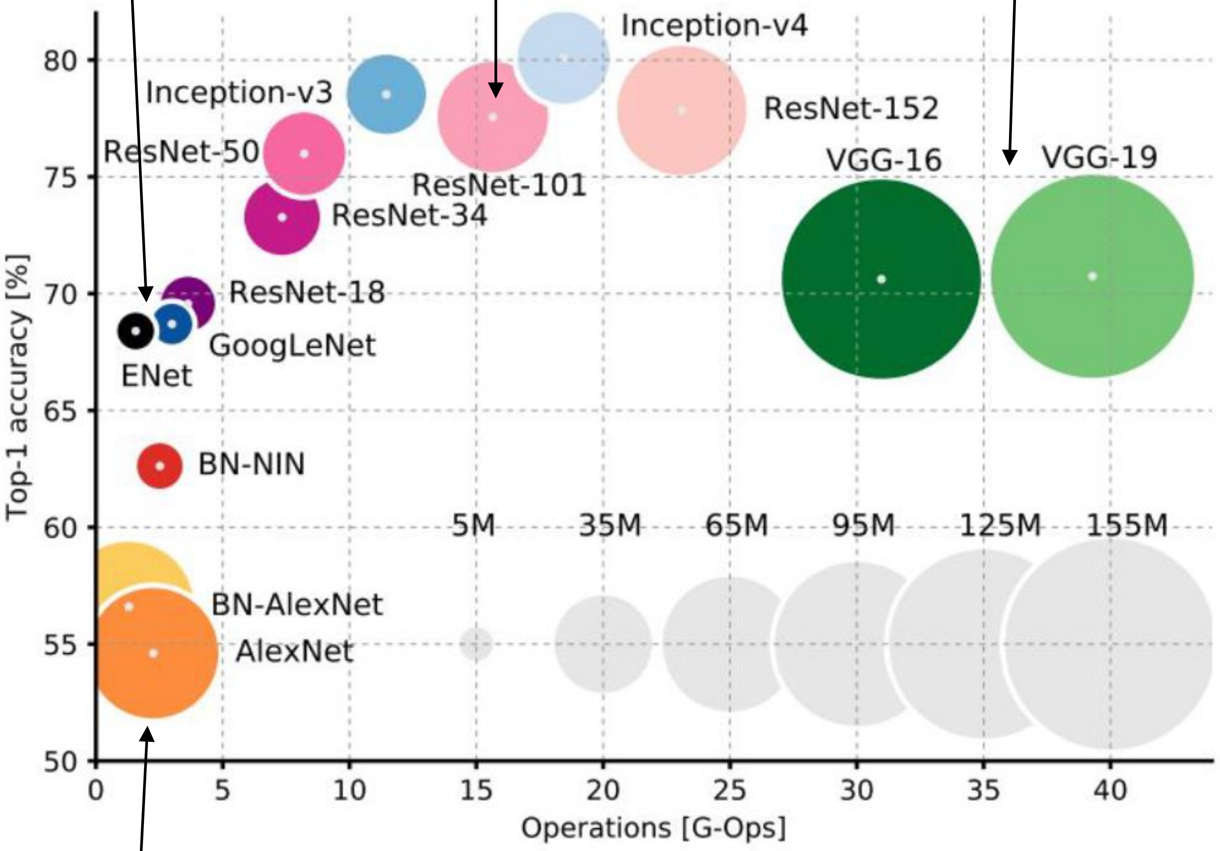
Inception-v4: Inception with residual connections

GoogLeNet: very efficient

ResNet: Moderate efficiency depending on model, high accuracy

VGG: most parameters, most operations

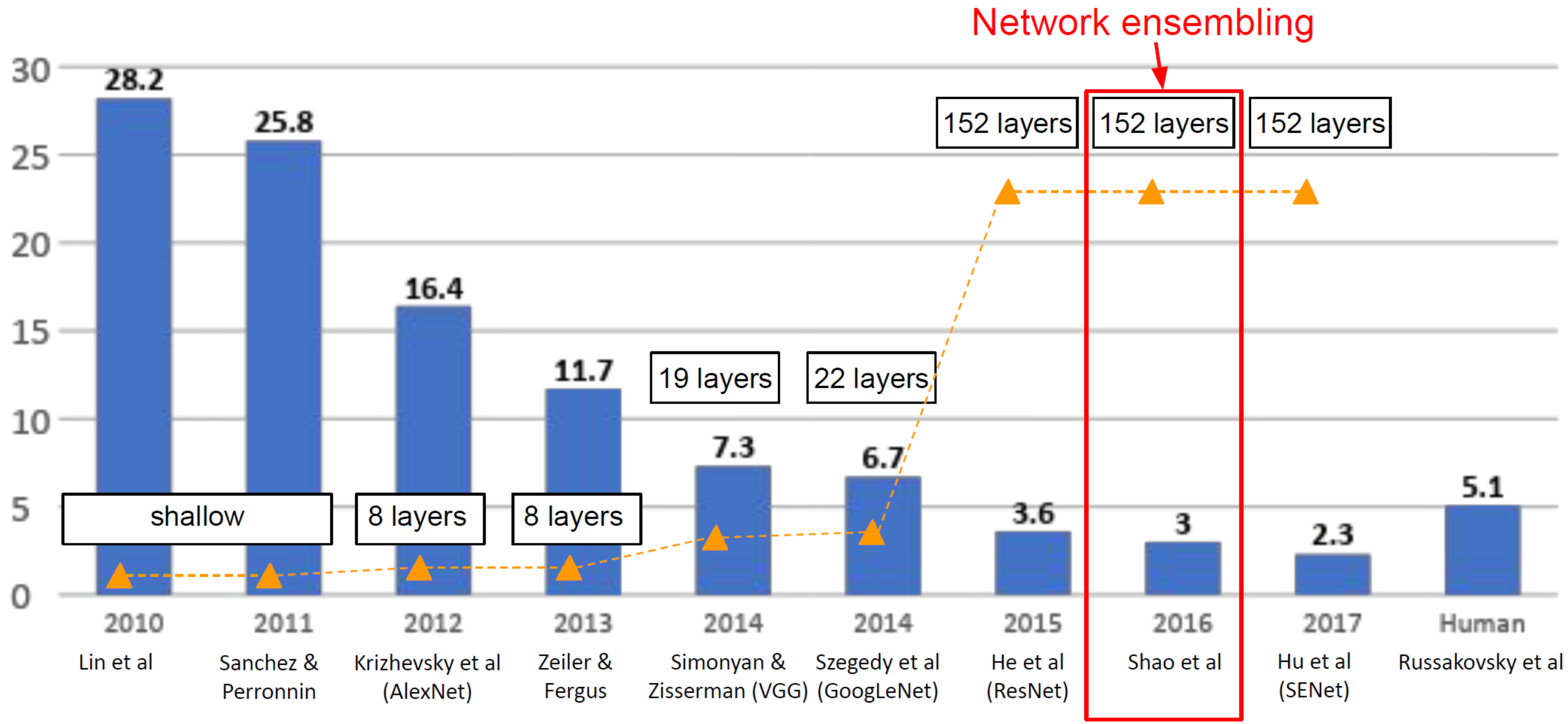




AlexNet: less operations, still memory heavy, lower accuracy

Canziani et al. "An analysis of deep neural network models for practical applications." *arXiv preprint arXiv:1605.07678* (2016).

Other approaches

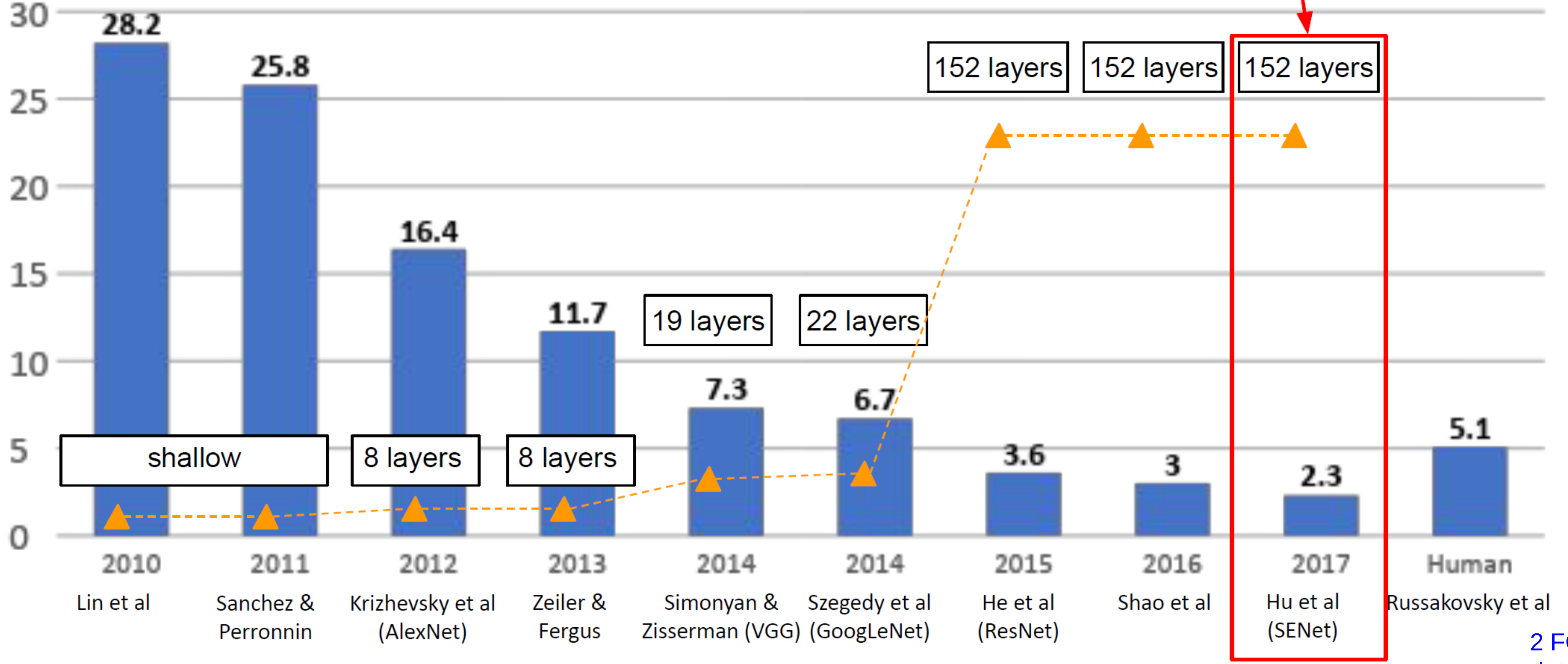


"Good Practices for Deep Feature Fusion" (Jie Shao et al., 2016)

- Multi-scale ensembling of Inception, Inception-Resnet, Resnet, Wide Resnet models
- ILSVRC'16 object classification winner

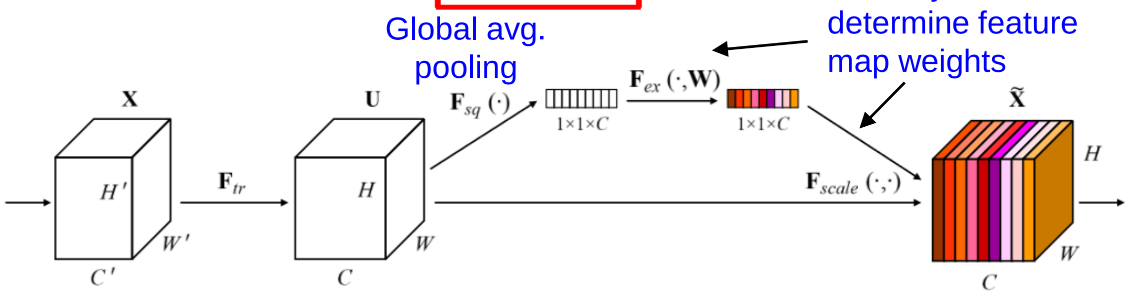
Other approaches

Adaptive feature map reweighting



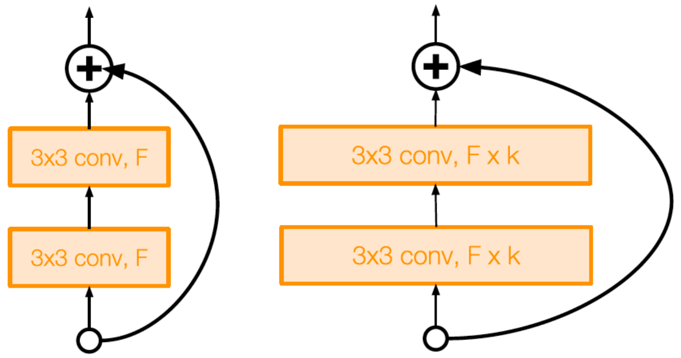
Hu et al. "Squeeze-and-excitation networks." *CVPR* 2018.

- Add a “feature recalibration” module that learns to adaptively reweight feature maps



Research into ConvNets is still flourishing

- Improving ResNets
  - He et al. “Identity mappings in deep residual networks”, ECCV 2016.
  - Zagoruyko and Komodakis. “Wide residual networks”, BMVC 2016.
    - Residuals are the important factor, not depth
    - Use wider residual blocks (F x k filters instead of F filters in each layer)
    - 16-layer WRN outperformed 1000-layer ResNet
  - Xie et al. “Aggregated residual transformations for deep neural networks” (ResNeXt), CVPR 2017.
    - Parallel pathways similar in spirit to Inception module



Basic residual block   Wide residual block

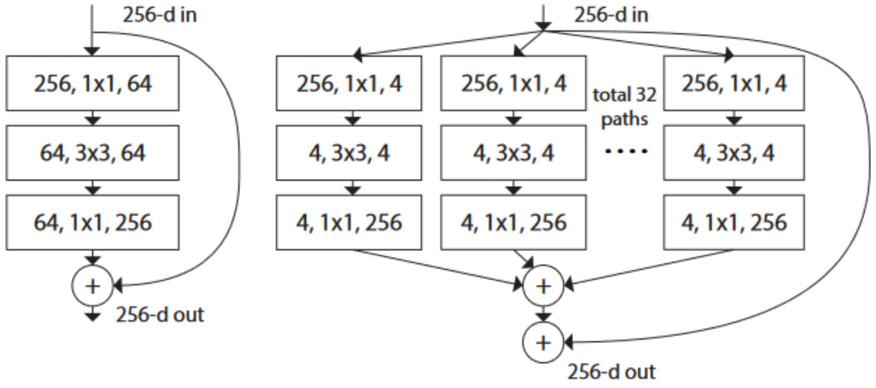
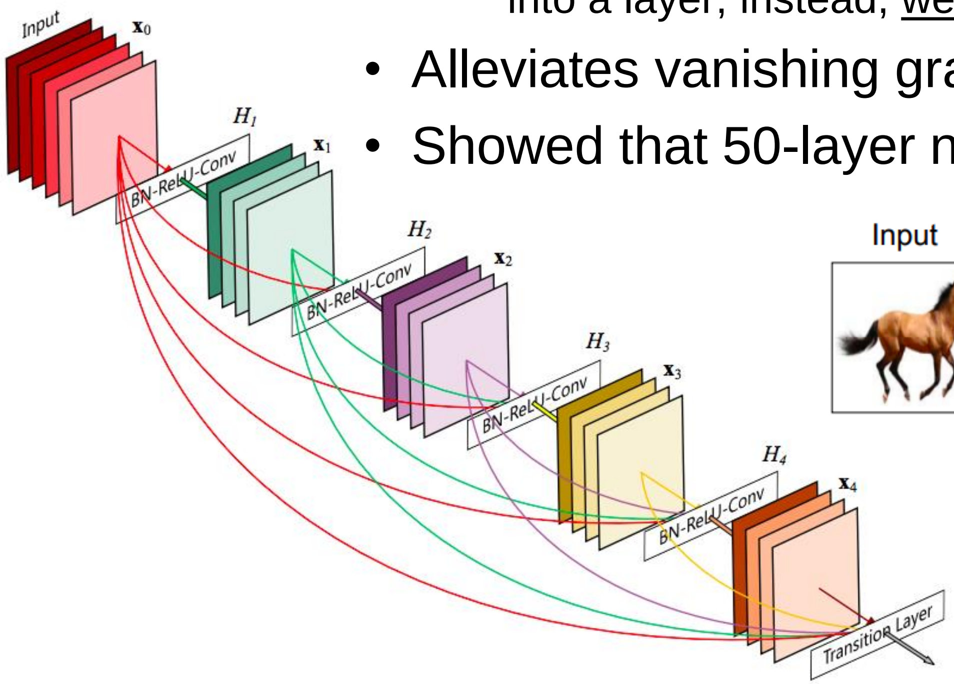
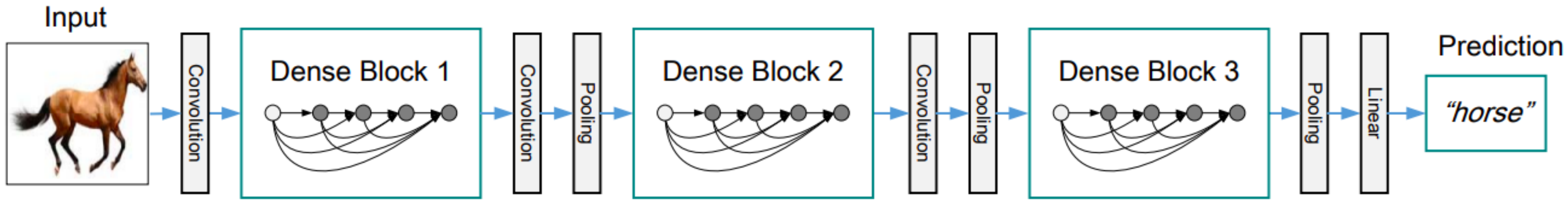


Figure 1. **Left**: A block of ResNet [14]. **Right**: A block of ResNeXt with cardinality = 32, with roughly the same complexity. A layer is shown as (# in channels, filter size, # out channels).

Research into ConvNets is still flourishing

- Improving ResNets
  - Huang et al. "Densely connected convolutional networks" (DenseNet), CVPR 2017.
    - Dense blocks where each layer is connected to every layer in feedforward fashion
      - "In contrast to ResNets, <u>we never combine features through summation</u> before they are passed into a layer; instead, <u>we combine features by concatenating them</u>."
    - Alleviates vanishing gradient, strengthens feature propagation
    - Showed that 50-layer network can outperform deeper ResNet-152





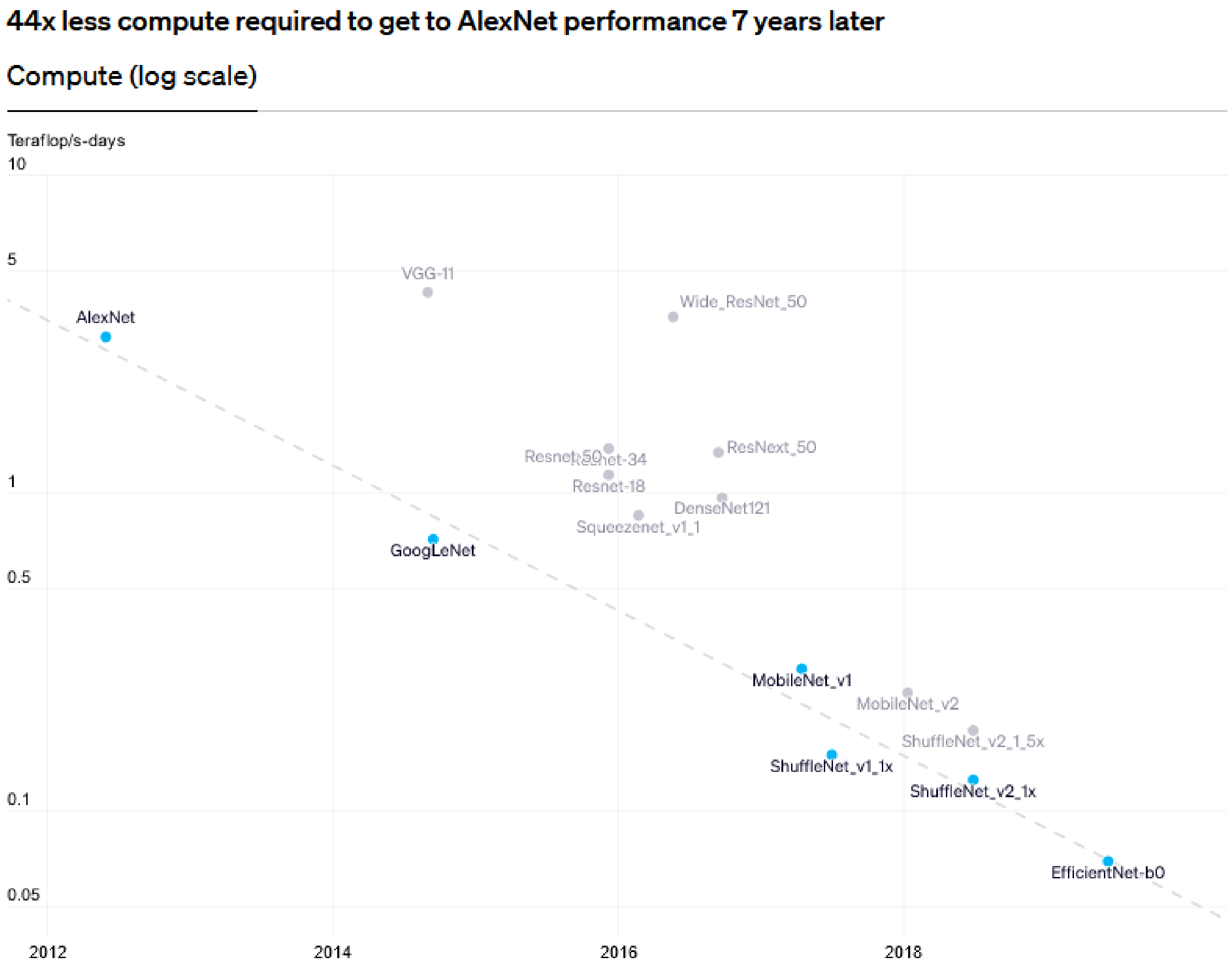
Research into ConvNets is still flourishing

- Efficient networks
  - Iandola et al. “SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and < 0.5 MB model size.” arXiv (2016)
  - Howard et al. “MobileNets: Efficient convolutional neural networks for mobile vision applications.” arXiv (2017).
  - Sandler et al. “MobileNetV2: Inverted residuals and linear bottlenecks”, CVPR 2018.
  - Zhang et al. “ShuffleNet: An extremely efficient convolutional neural network for mobile devices”, CVPR 2018.

Research into ConvNets is still flourishing

- Learning to search for network architectures
  - Zoph and Le, “Neural Architecture Search with Reinforcement Learning”, ICLR 2017
  - Zoph et al. “Learning transferable architectures for scalable image recognition”, CVPR 2018
  - Tan and Le. “EfficientNet: Rethinking model scaling for convolutional neural networks”, ICML 2019.
    - RL-based approach to develop a baseline architecture (Efficient-B0). Multi-objective search that optimizes for both Accuracy and FLOPS.
    - Scale up using smart heuristic rules:
      - Increase network capacity by scaling network depth (#layers) and width (#channels), and image resolution, while balancing accuracy and efficiency.
      - Intuition: if input image is bigger → network needs more layers (to increase the receptive field) and more channels (to capture more fine-grained patterns on the bigger image).

AI and efficiency



Total amount of compute in teraflops/s-days used to train to AlexNet level performance.

https://openai.com/research/ai-and-efficiency

General Design Principles

- Reduce filter sizes (except possibly at the lowest layer), and factorize filters aggressively
- Use 1x1 convolutions to reduce and expand the number of feature maps judiciously
- Use skip connections and/or create multiple paths through the network

Slide credit: Svetlana Lazebnik

Practical advice for using ConvNets

- **Use open-source implementations**: Many times, it is difficult to reproduce DNNs results (many details, technical tricks, etc.)

- **Transfer Learning**: Take a pretrained network and transfer that knowledge to a new task.
  - In transfer learning, we take a complete network, remove a few layers from it, and add custom layers on top of the remaining layers to train our model.

- **Data Augmentation**: DL models perform well when we have a large amount of data. Use data augmentation to generate training data from the available data: mirroring, random cropping, rotating, shearing, color shifting,…

Andrew Ng’s advices from DeepLearning.AI

What else?

- Maaaany training tricks and details:
  - Network initialization
    - Xavier Glorot initialization
    - Kaiming He initialization
    - Yann LeCun initialization
  - Normalization strategies (like batch renormalization)
  - Different regularization and training strategies
    - Mixup (train with weighted averages of input images and labels)
    - Label smoothing (don't encourage the model to predict something overconfidently by injecting noise in the labels)
    - Test-time augmentation (averaging outputs over multiple crops/flips)
    - Progressive resizing (gradually using larger and larger images as you train)
    - Discriminative fine-tuning (tune each layer with different learning rates)

Convolutional Neural Networks for Image Classification

Pablo Mesejo

pmesejo@go.ugr.es

Universidad de Granada

Departamento de Ciencias de la Computación e Inteligencia Artificial

UNIVERSIDAD DE GRANADA

DECSAI

DaSCI

Instituto Andaluz de Investigación en Data Science and Computational Intelligence