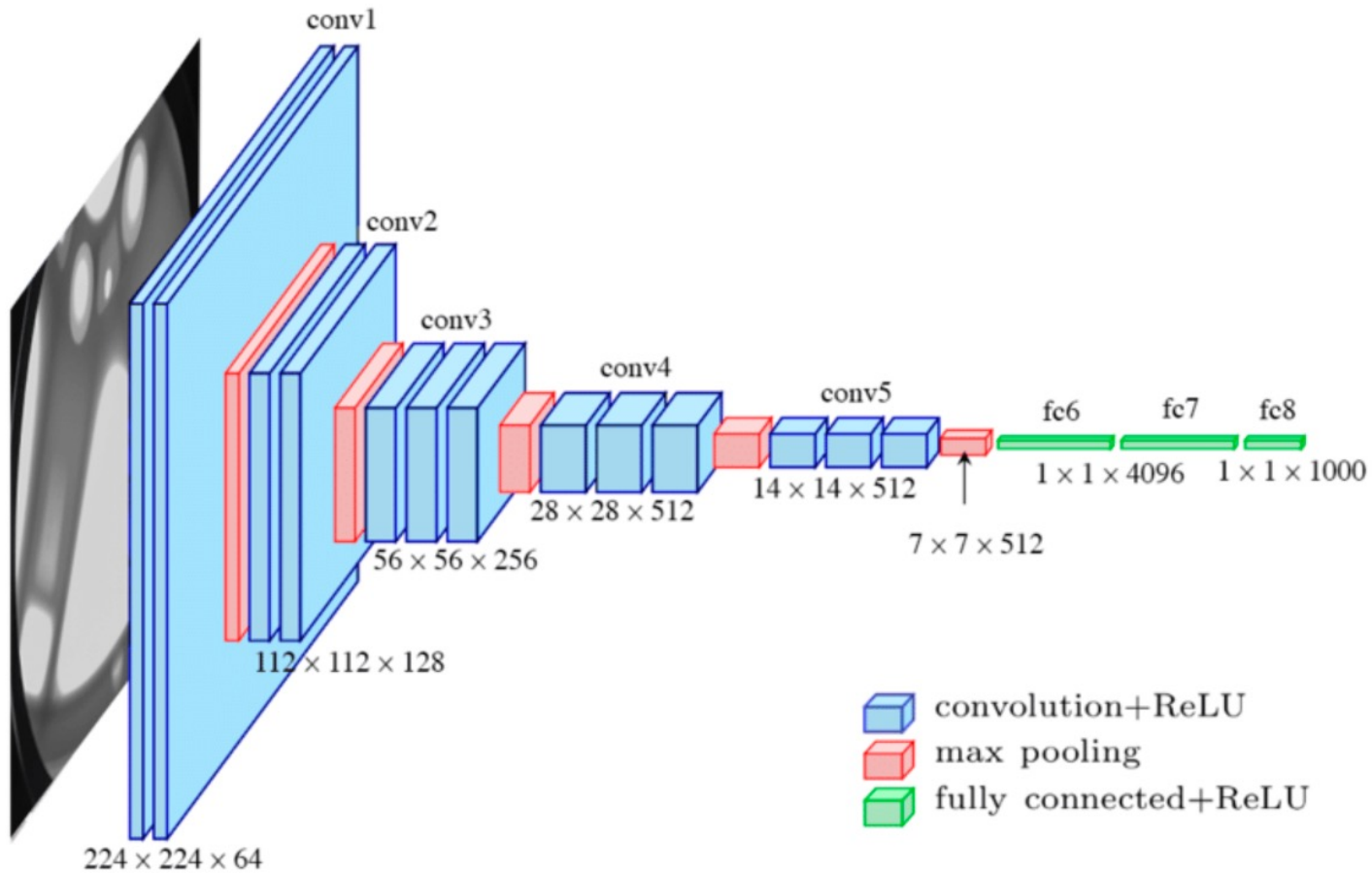


A tour of convolutional architectures



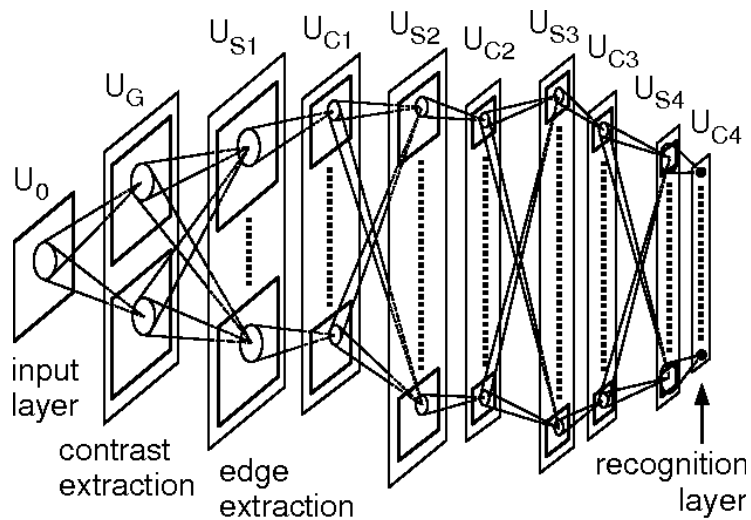
[Image source](#)

Outline

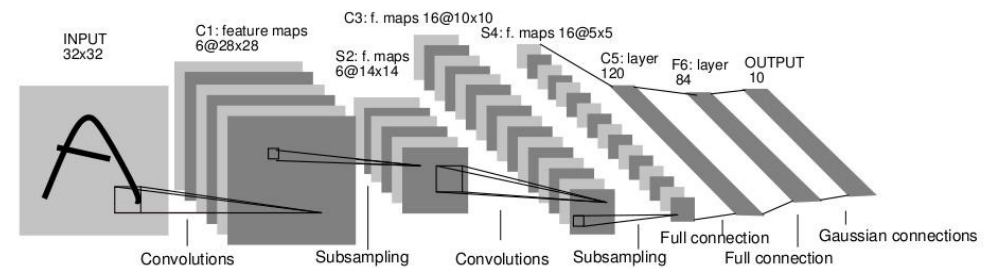
- AlexNet (2012-2013)
- VGGNet (2014)
- GoogLeNet (2014)
- ResNet (2015)
- Beyond ResNet (2016 and onward): Wide ResNet, ResNeXT, DenseNet
- Beyond ImageNet classification

Backstory

Neocognitron



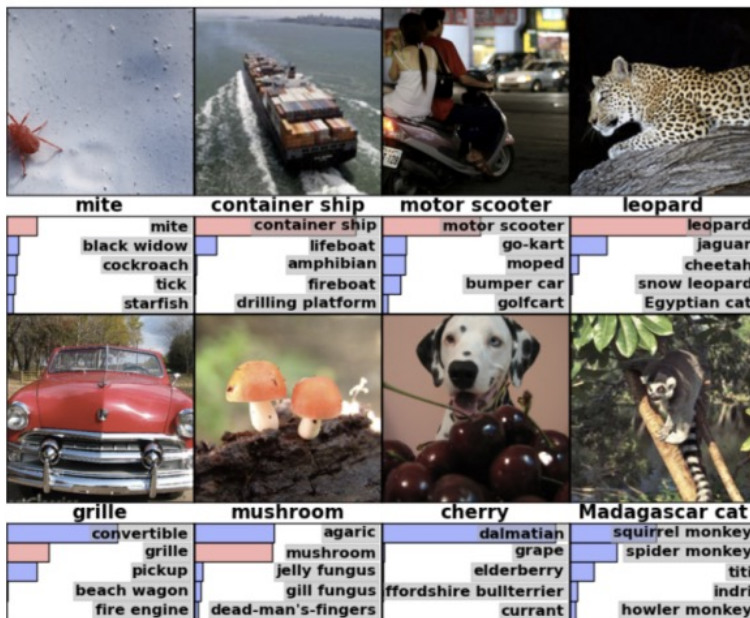
LeNet-5



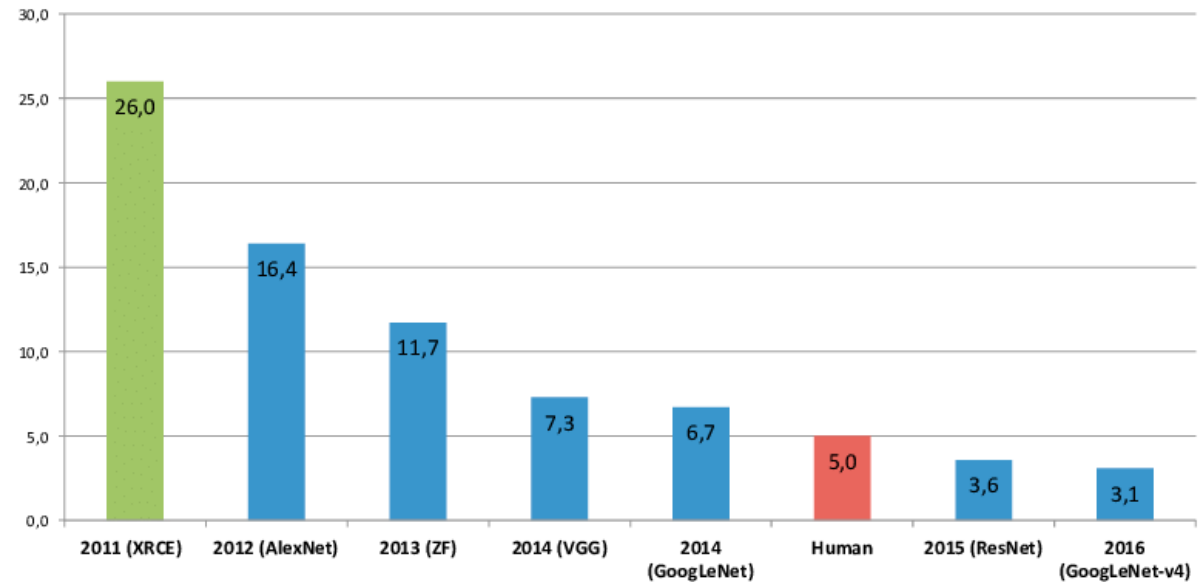
K. Fukushima. [Neocognitron: A Self-organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position](#). Biological Cybernetics, 1980
[Demo video](#)

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, [Gradient-based learning applied to document recognition](#), Proc. IEEE 86(11): 2278–2324, 1998

ImageNet Large Scale Visual Recognition Challenge (ILSVRC)



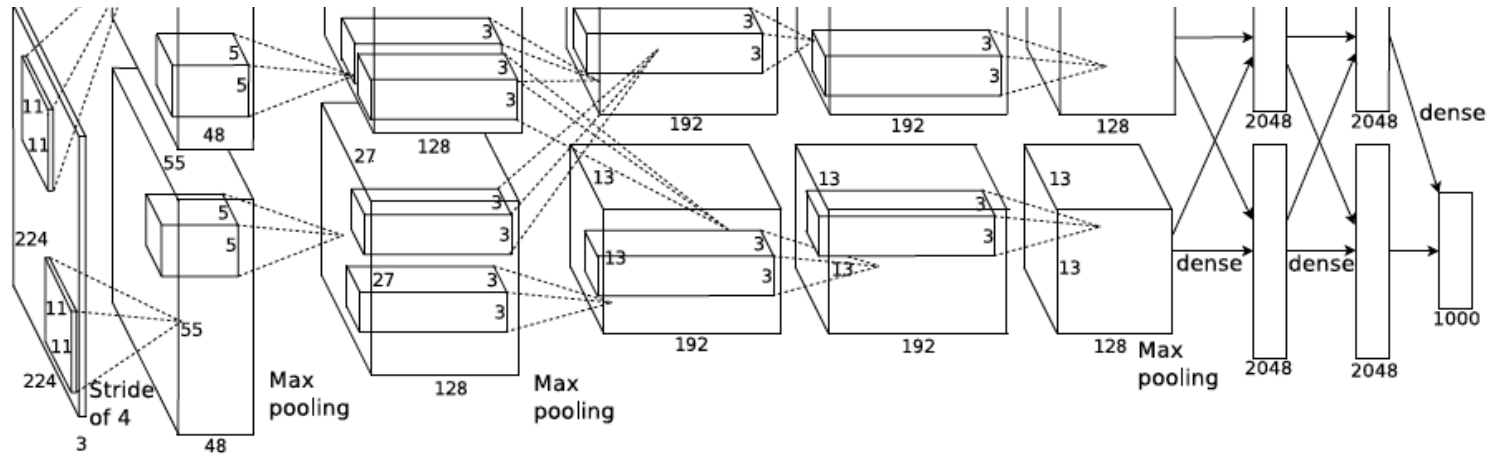
ImageNet Classification Error (Top 5)



[Figure source](#)

[ILSVRC webpage](#)

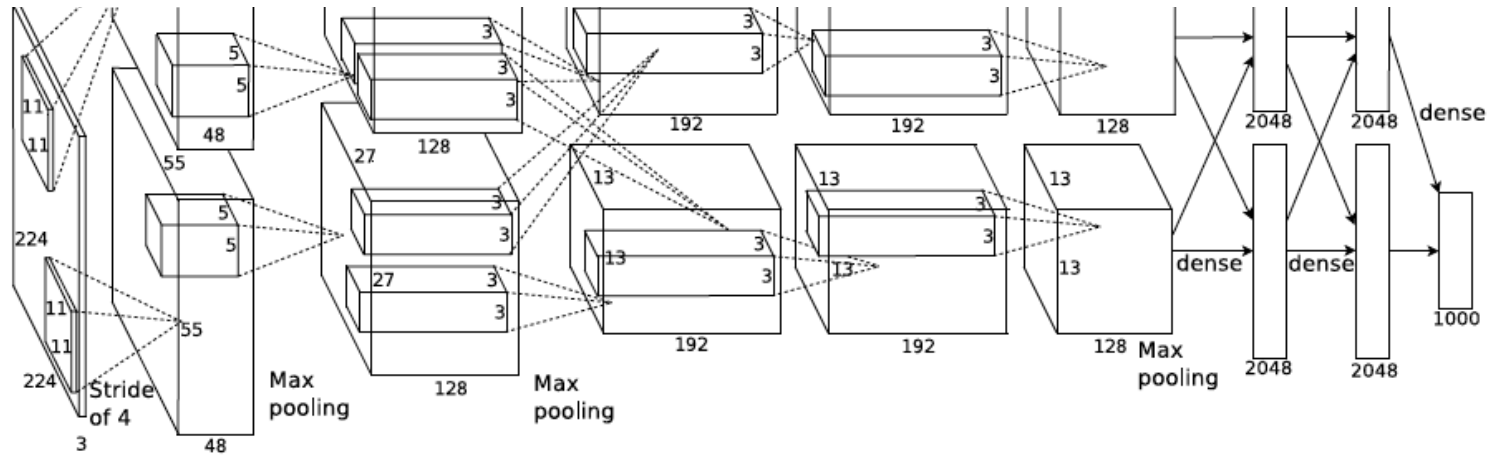
AlexNet: ILSVRC 2012 winner



[Photo source](#)

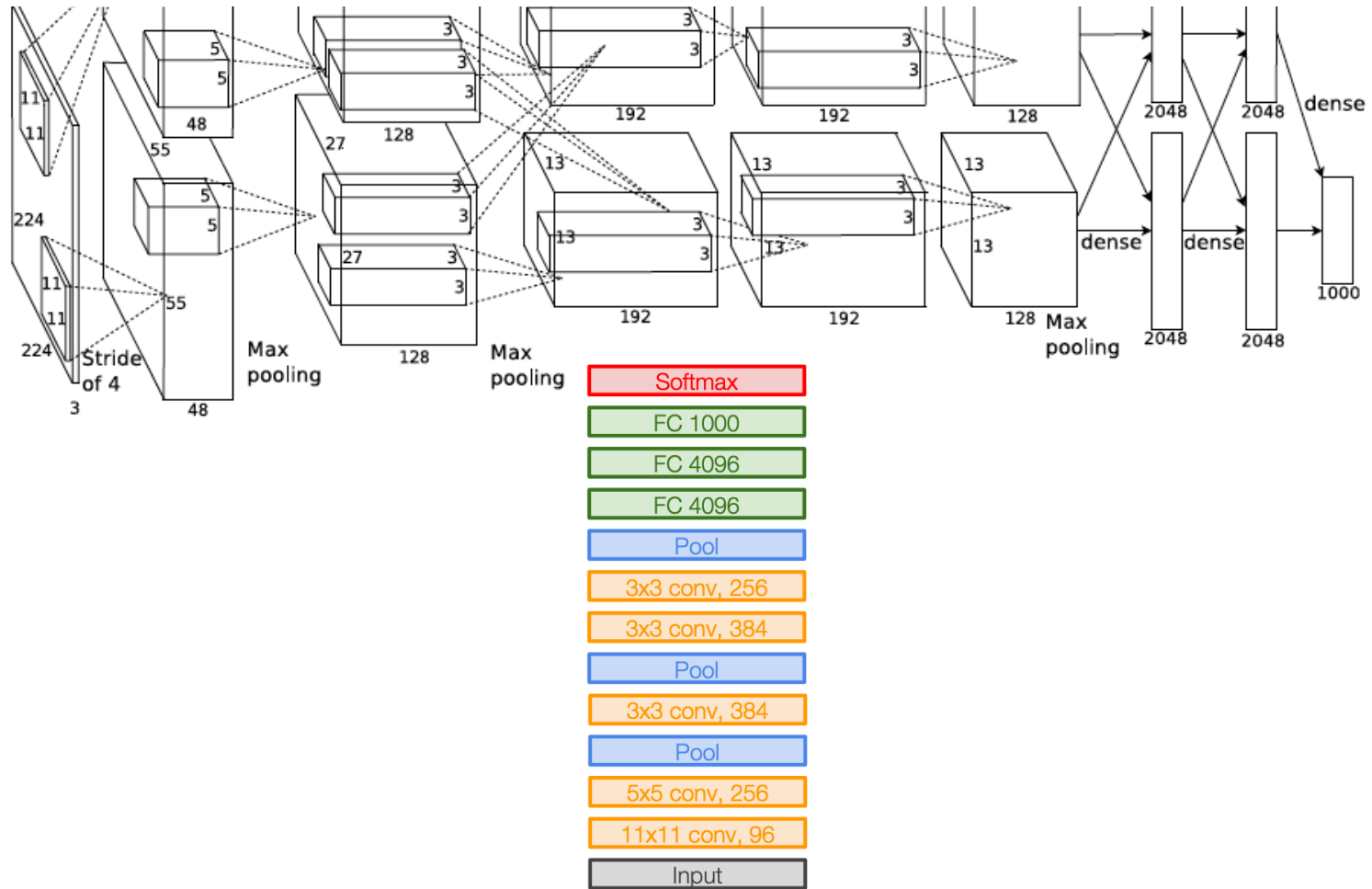
A. Krizhevsky, I. Sutskever, and G. Hinton, [ImageNet Classification with Deep Convolutional Neural Networks](#), NIPS 2012

AlexNet: ILSVRC 2012 winner



- Successor of LeNet-5, but with a few crucial changes
 - Max pooling, ReLU nonlinearity
 - Dropout regularization
 - More data and bigger model (7 hidden layers, 650K units, 60M params)
 - GPU implementation (50x speedup over CPU)
 - Trained on two GPUs for a week

AlexNet: ILSVRC 2012 winner



AlexNet (modified): Table view

Layer	Input size		Layer				Output size		memory (KB)	params (k)	flop (M)
	C	H / W	filters	kernel	stride	pad	C	H / W			
conv1	3	227	64	11	4	2	64	56	784	23	73
pool1	64	56		3	2	0	64	27	182	0	0
conv2	64	27	192	5	1	2	192	27	547	307	224
pool2	192	27		3	2	0	192	13	127	0	0
conv3	192	13	384	3	1	1	384	13	254	664	112
conv4	384	13	256	3	1	1	256	13	169	885	145
conv5	256	13	256	3	1	1	256	13	169	590	100
pool5	256	13		3	2	0	256	6	36	0	0
flatten	256	6					9216		36	0	0
fc6	9216		4096				4096		16	37,749	38
fc7	4096		4096				4096		16	16,777	17
fc8	4096		1000				1000		4	4,096	4

Source: [J. Johnson](#)

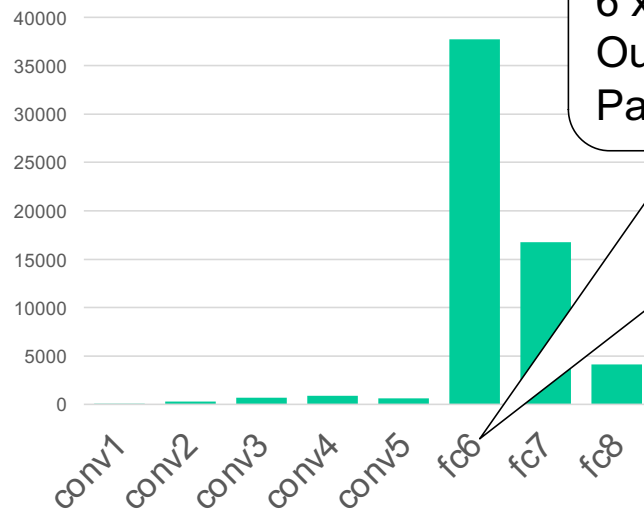
AlexNet (modified): Analysis

Memory (KB)



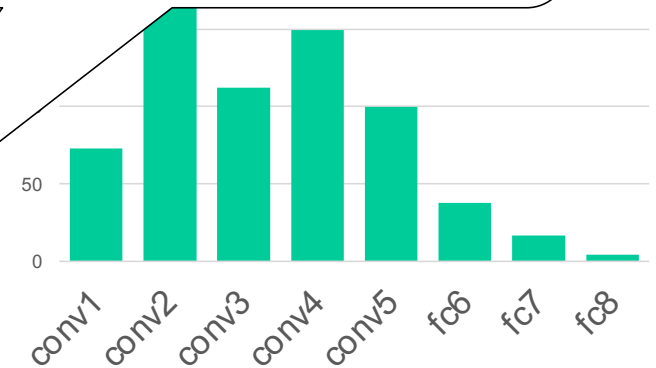
Most of the memory usage is in the early convolution layers

Params (K)



Nearly all parameters are in the fully-connected layers

FC6 input size:
 $6 \times 6 \times 256 = 9216$
Output size: 4096
Params: $9216 \times 4096 = 37,749K$



Most floating-point ops occur in the convolution layers

Clarifai or ZFNet: ILSVRC 2013 winner

- Refinement of AlexNet

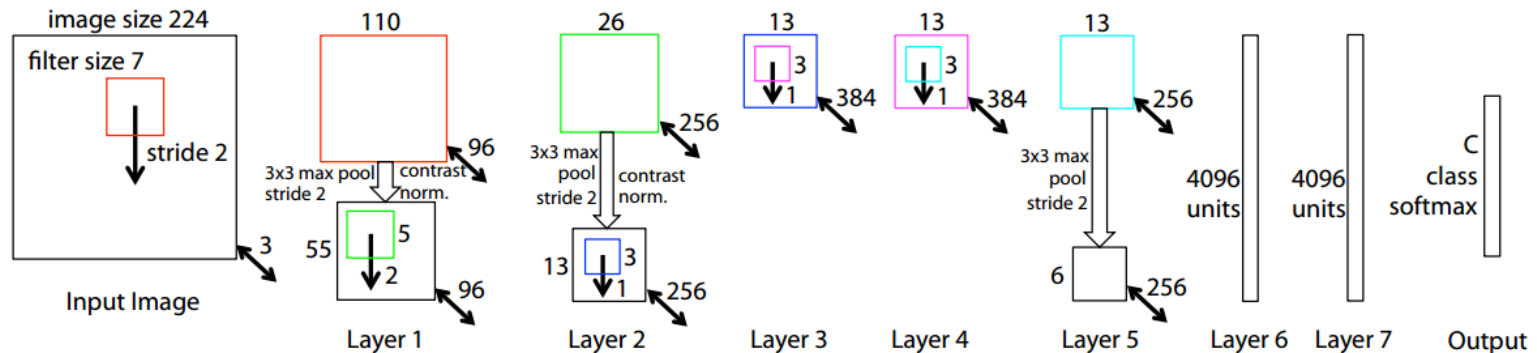
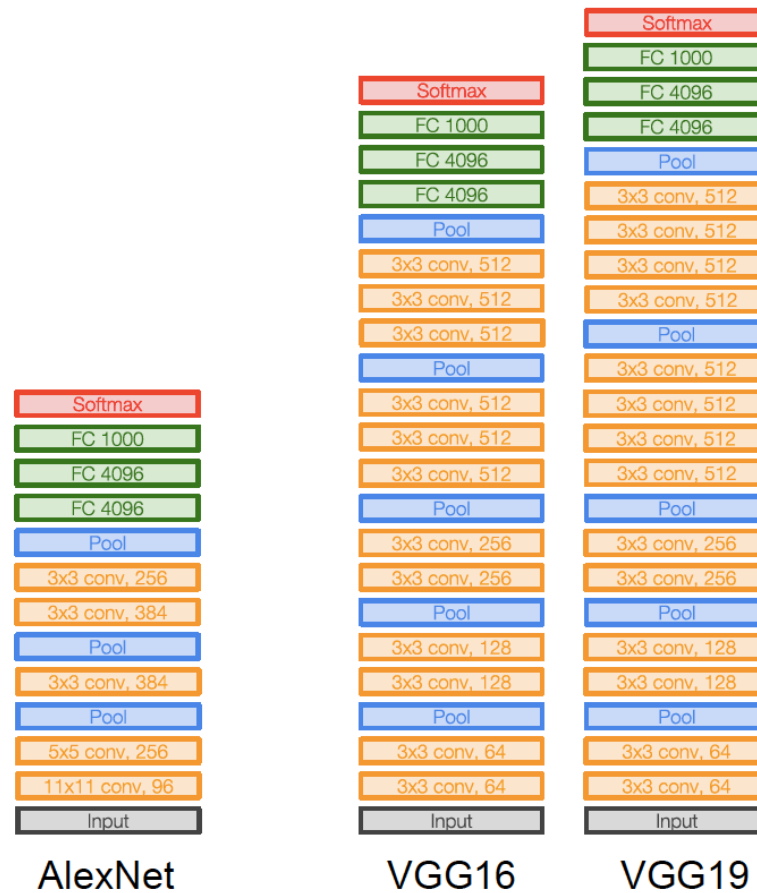


Figure 3. Architecture of our 8 layer convnet model. A 224 by 224 crop of an image (with 3 color planes) is presented as the input. This is convolved with 96 different 1st layer filters (red), each of size 7 by 7, using a stride of 2 in both x and y. The resulting feature maps are then: (i) passed through a rectified linear function (not shown), (ii) pooled (max within 3x3 regions, using stride 2) and (iii) contrast normalized across feature maps to give 96 different 55 by 55 element feature maps. Similar operations are repeated in layers 2,3,4,5. The last two layers are fully connected, taking features from the top convolutional layer as input in vector form ($6 \cdot 6 \cdot 256 = 9216$ dimensions). The final layer is a C -way softmax function, C being the number of classes. All filters and feature maps are square in shape.

M. Zeiler and R. Fergus, [Visualizing and Understanding Convolutional Networks](#),
ECCV 2014 (Best Paper Award winner)

VGGNet: ILSVRC 2014 2nd place



[Image source](#)

K. Simonyan and A. Zisserman, [Very Deep Convolutional Networks for Large-Scale Image Recognition](#), ICLR 2015

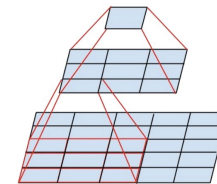
VGGNet: ILSVRC 2014 2nd place

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 LRN	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 conv1-256	conv3-256 conv3-256	conv3-256 conv3-256 conv3-256
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 conv1-512	conv3-512 conv3-512	conv3-512 conv3-512 conv3-512
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 conv1-512	conv3-512 conv3-512	conv3-512 conv3-512 conv3-512
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Table 2: Number of parameters (in millions).

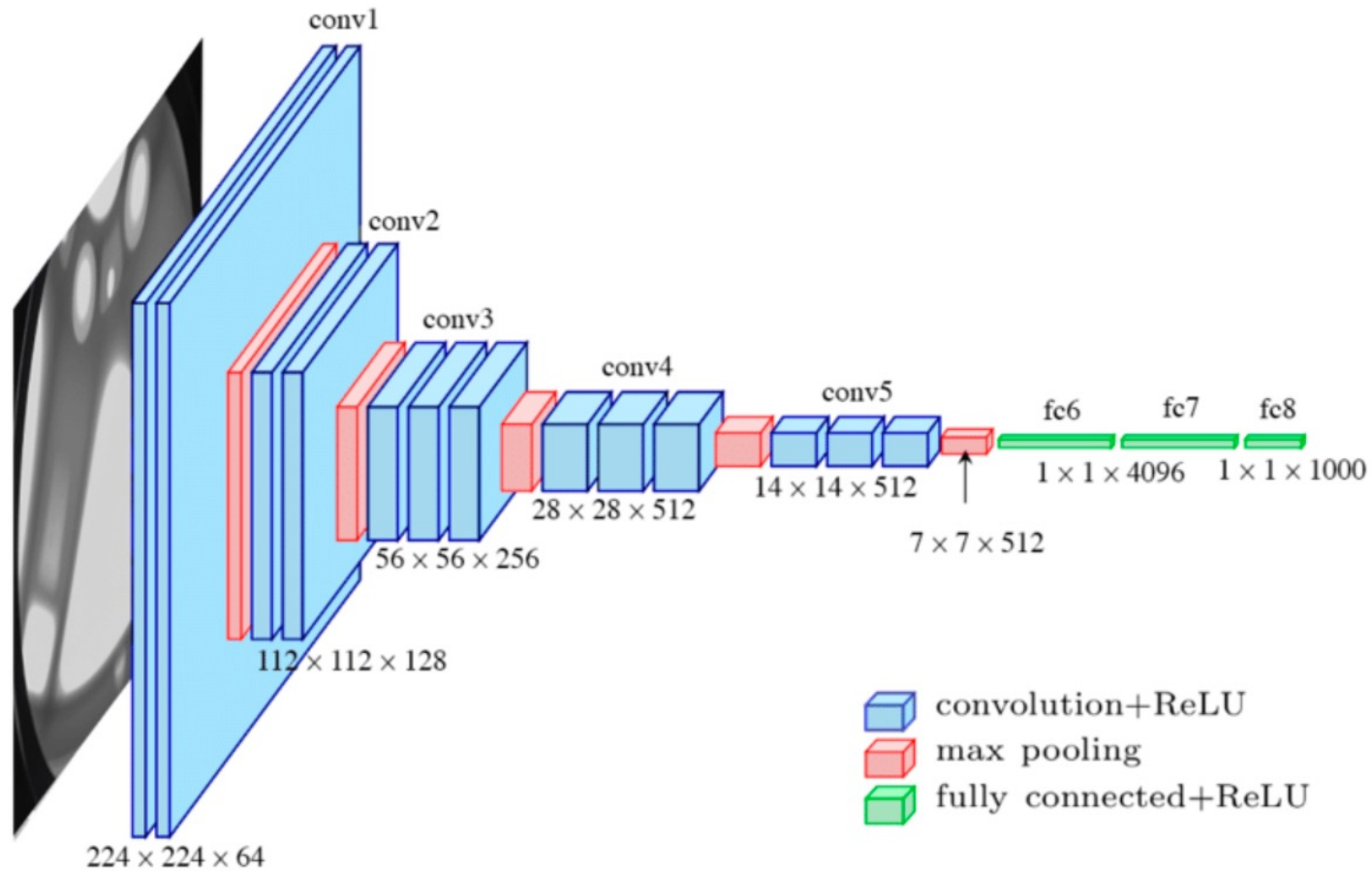
Network	A,A-LRN	B	C	D	E
Number of parameters	133	133	134	138	144

- Sequence of deeper networks trained progressively
- Large receptive fields replaced by successive layers of 3×3 convolutions (with ReLU in between)



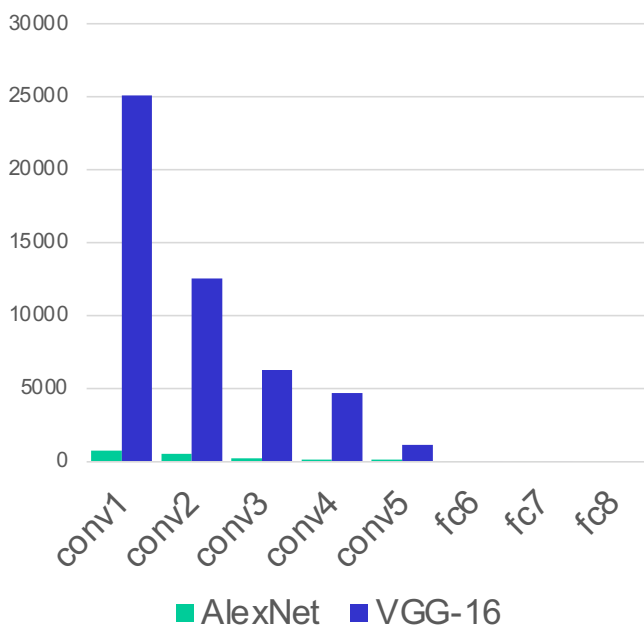
- How many weights does one 7×7 conv layer with K input and output channels have?
 - $49K^2$
- What about three 3×3 conv layers with K input and output channels?
 - $27K^2$
- All maxpool layers are 2×2 stride 2, followed by doubling of the number of channels (keeping cost of convolutions the same)

Another view of VGG-19 architecture (modified)



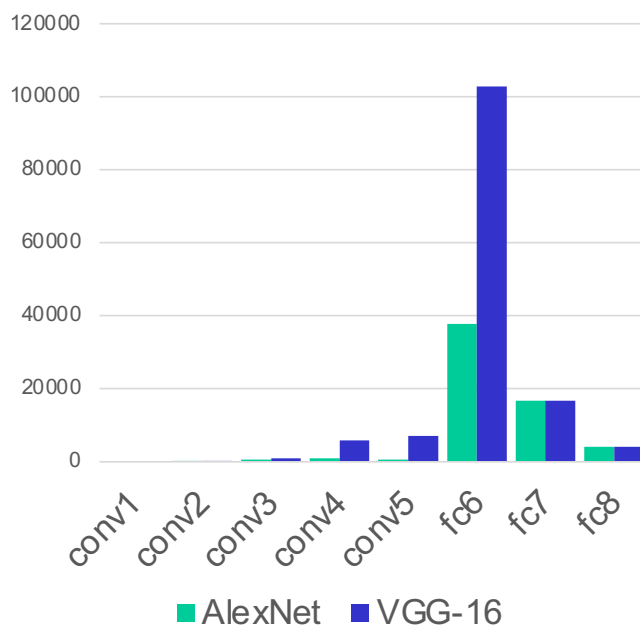
VGGNet vs. AlexNet

AlexNet vs VGG-16
(Memory, KB)



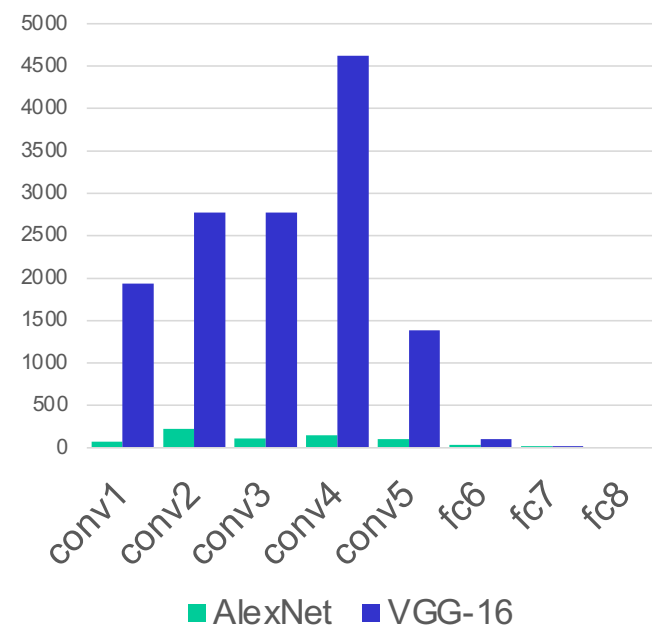
AlexNet total: 1.9 MB
VGG-16 total: 48.6 MB (25x)

AlexNet vs VGG-16
(Params, M)



AlexNet total: 61M
VGG-16 total: 138M (2.3x)

AlexNet vs VGG-16
(MFLOPs)



AlexNet total: 0.7 GFLOP
VGG-16 total: 13.6 GFLOP (19.4x)

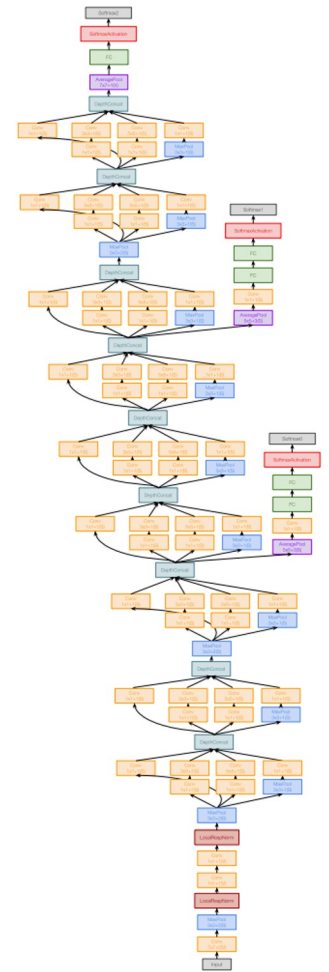
Source: [J. Johnson](#)

GoogLeNet: ILSVRC 2014 winner



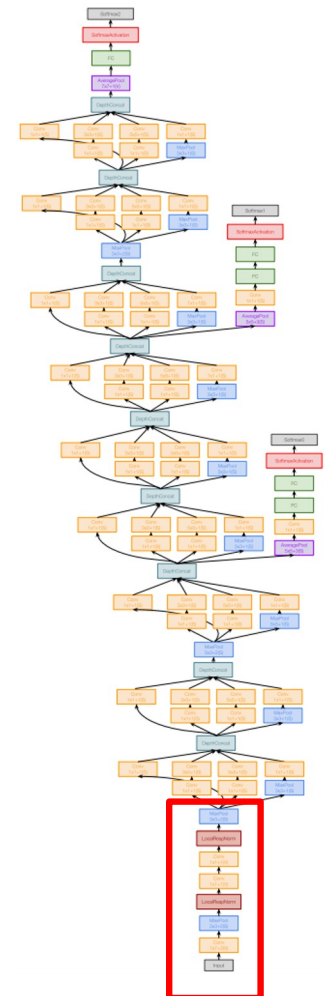
<http://knowyourmeme.com/memes/we-need-to-go-deeper>

C. Szegedy et al., [Going deeper with convolutions](#), CVPR 2015



GoogLeNet: Aggressive stem

- **Stem network** at the start aggressively downsamples input

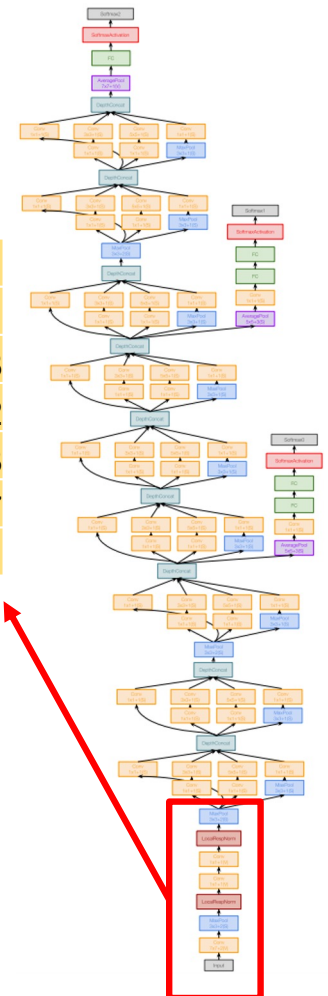


GoogLeNet: Aggressive stem

- **Stem network** at the start aggressively downsamples input

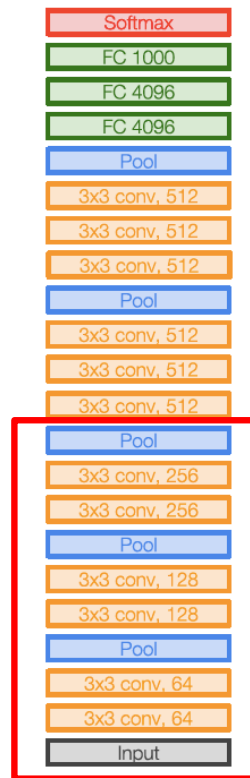
Layer	Input size		Layer				Output size		memory (KB)	params (K)	flop (M)
	C	H / W	filters	kernel	stride	pad	C	H/W			
conv	3	224	64	7	2	3	64	112	3136	9	118
max-pool	64	112		3	2	1	64	56	784	0	2
conv	64	56	64	1	1	0	64	56	784	4	13
conv	64	56	192	3	1	1	192	56	2352	111	347
max-pool	192	56		3	2	1	192	28	588	0	1

Total from 224 to 28 resolution:
 Memory: 7.5 MB
 Params: 124K
 MFLOP: 418



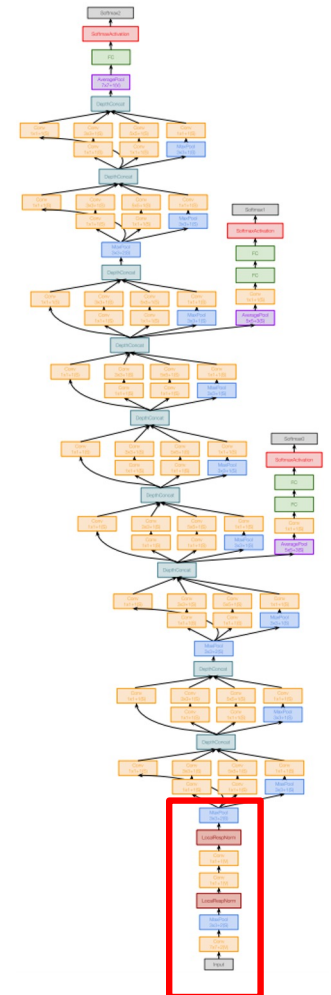
GoogLeNet: Aggressive stem

Compare VGG-16:
 Memory: 42.9 MB (5.7x)
 Params: 1.1M (8.9x)
 MFLOP: 7485 (17.8x)



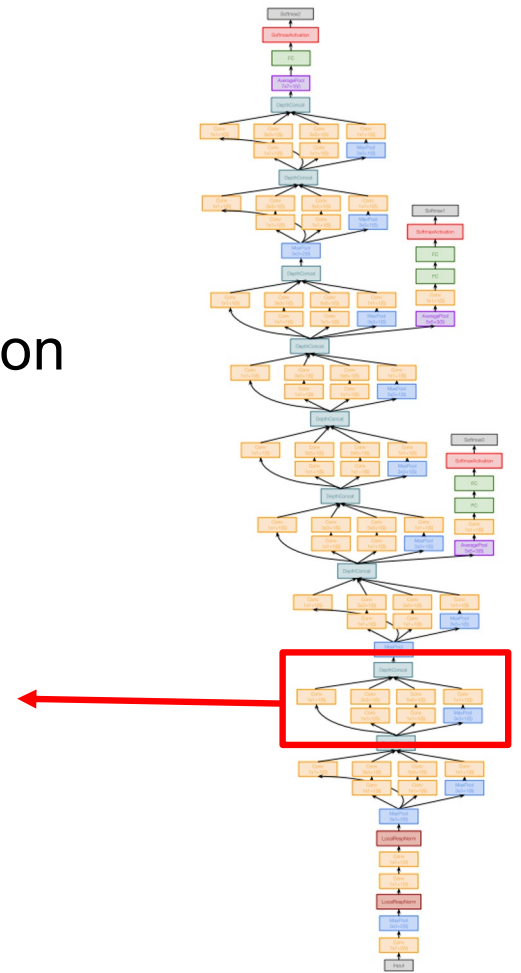
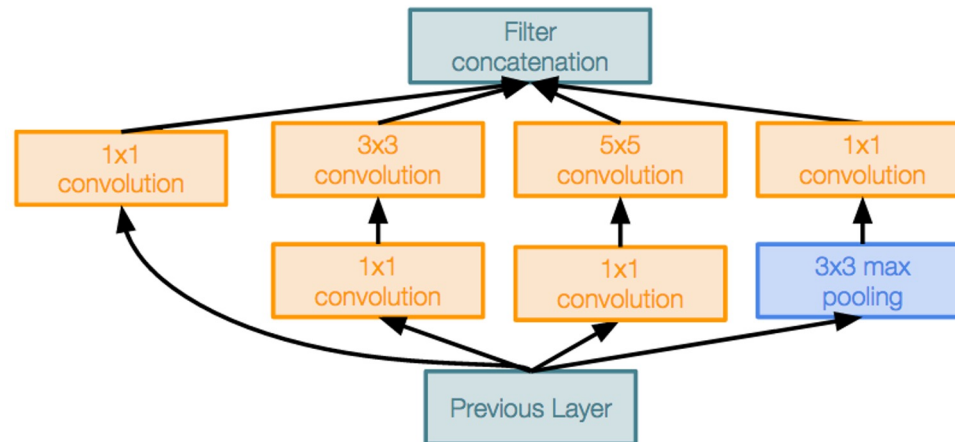
VGG16

Total from 224 to 28 resolution:
 Memory: 7.5 MB
 Params: 124K
 MFLOP: 418



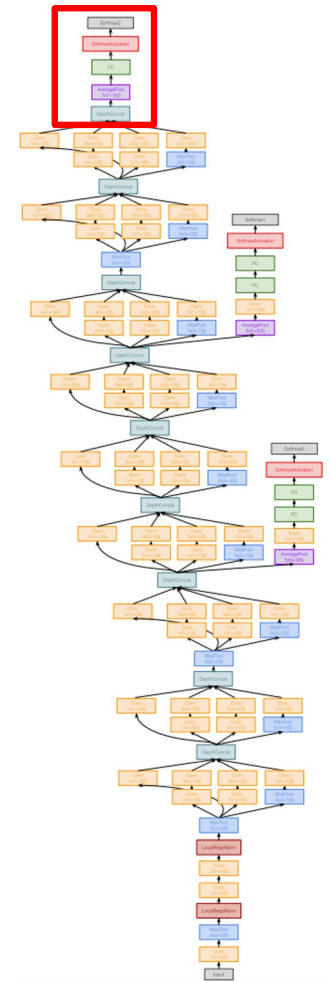
GoogLeNet: Inception module

- Parallel paths with different receptive field sizes and operations are meant to capture sparse patterns of correlations in the stack of feature maps
- Use 1×1 convolutions for dimensionality reduction before expensive convolutions



GoogLeNet: Global average pooling

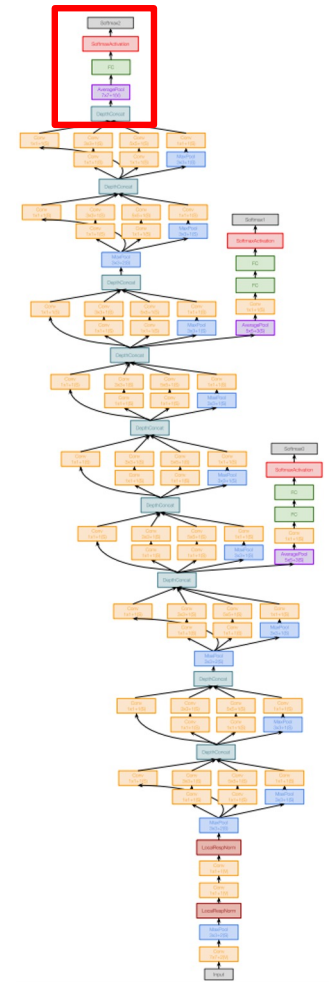
- Instead of large FC layers at the end, use *global average pooling* to collapse spatial dimensions, followed by linear layer to produce class scores
 - Recall VGG-16: Most parameters were in the FC layers!



GoogLeNet: Global average pooling

GoogLeNet head:

Layer	Input size		Layer				Output size		memory (KB)	params (k)	flop (M)
	C	H/W	filters	kernel	stride	pad	C	H/W			
avg-pool	1024	7		7	1	0	1024	1	4	0	0
fc	1024		1000				1000		0	1025	1



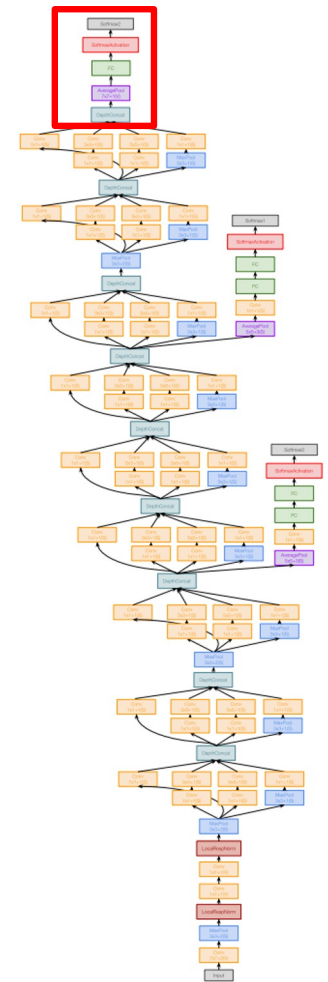
GoogLeNet: Global average pooling

GoogLeNet head:

Layer	Input size		Layer				Output size		memory (KB)	params (k)	flop (M)
	C	H/W	filters	kernel	stride	pad	C	H/W			
avg-pool	1024	7		7	1	0	1024	1	4	0	0
fc	1024		1000				1000		0	1025	1

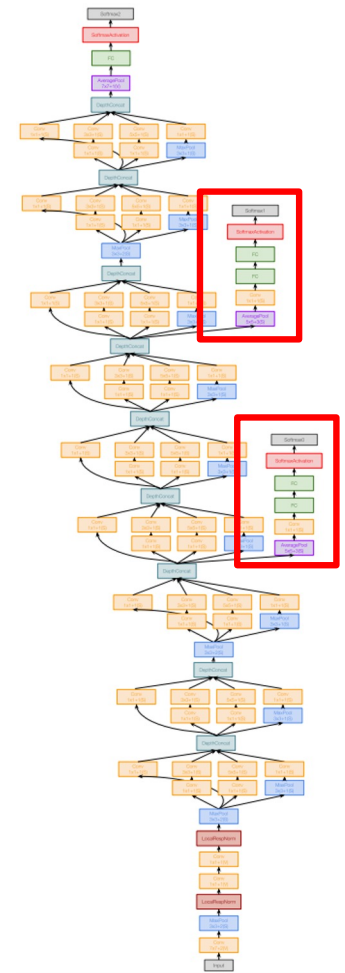
Compare with VGG-16:

Layer	C	H/W	filters	kernel	stride	pad	C	H/W	memory (KB)	params (k)	flop (M)
flatten	512	7					25088		98		
fc6	25088			4096			4096		16	102760	103
fc7	4096			4096			4096		16	16777	17
fc8	4096			1000			1000		4	4096	4



GoogLeNet: Auxiliary classifiers

- Training using loss at the end of the network didn't work well: network is too deep, gradients don't propagate cleanly
- As a hack, attach "auxiliary classifiers" at several intermediate points in the network that also try to classify the image and receive loss
- GoogLeNet was before batch normalization! With BatchNorm no longer need to use this trick



ImageNet Challenge 2012-2014

Team	Year	Place	Error (top-5)	External data
SuperVision – Toronto (7 layers)	2012	-	16.4%	no
SuperVision	2012	1st	15.3%	ImageNet 22k
Clarifai – NYU (7 layers)	2013	-	11.7%	no
Clarifai	2013	1st	11.2%	ImageNet 22k
VGG – Oxford (16 layers)	2014	2nd	7.32%	no
GoogLeNet (19 layers)	2014	1st	6.67%	no
Human expert*			5.1%	

<http://karpathy.github.io/2014/09/02/what-i-learned-from-competing-against-a-convnet-on-imagenet/>

Outline

- AlexNet (2012-2013)
- VGGNet (2014)
- GoogLeNet (2014)
- ResNet (2015)

ResNet: ILSVRC 2015 winner

AlexNet, 8 layers
(ILSVRC 2012)



VGG, 19 layers
(ILSVRC 2014)



K. He, X. Zhang, S. Ren, and J. Sun, [Deep Residual Learning for Image Recognition](#), CVPR 2016 (Best Paper)

ResNet: ILSVRC 2015 winner

AlexNet, 8 layers
(ILSVRC 2012)



VGG, 19 layers
(ILSVRC 2014)



ResNet, **152 layers**
(ILSVRC 2015)



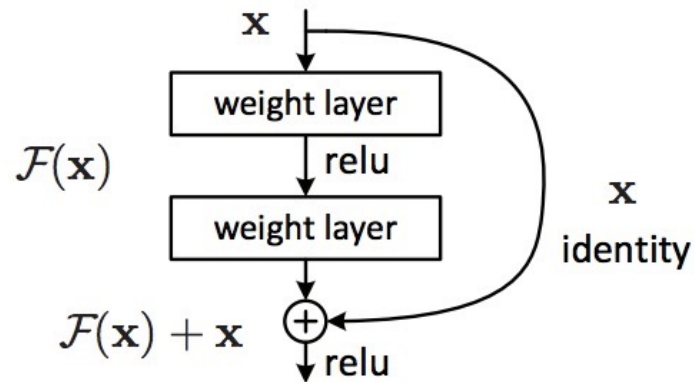
K. He, X. Zhang, S. Ren, and J. Sun, [Deep Residual Learning for Image Recognition](#), CVPR 2016 (Best Paper)



[Source \(?\)](#)

ResNet

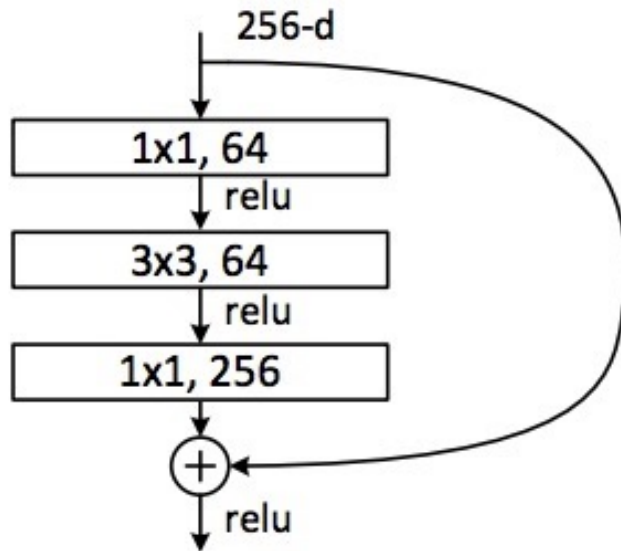
- The residual module
 - Introduce *skip* or *shortcut* connections (existing before in various forms in literature)
 - Make it easy for network layers to represent the identity mapping



K. He, X. Zhang, S. Ren, and J. Sun, [Deep Residual Learning for Image Recognition](#), CVPR 2016 (Best Paper)

ResNet

Deeper residual module (bottleneck)



- Directly performing 3×3 convolutions with 256 feature maps at input and output:
 $256 \times 256 \times 3 \times 3 \approx 600K$ operations
- Using 1×1 convolutions to reduce 256 to 64 feature maps, followed by 3×3 convolutions, followed by 1×1 convolutions to expand back to 256 maps:
 $256 \times 64 \times 1 \times 1 \approx 16K$
 $64 \times 64 \times 3 \times 3 \approx 36K$
 $64 \times 256 \times 1 \times 1 \approx 16K$
Total $\approx 70K$

K. He, X. Zhang, S. Ren, and J. Sun, [Deep Residual Learning for Image Recognition](#), CVPR 2016 (Best Paper)

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				
		3×3 max pool, stride 2				
conv2_x	56×56	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$
conv3_x	28×28	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 8$
conv4_x	14×14	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 23$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 36$
conv5_x	7×7	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$
	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

K. He, X. Zhang, S. Ren, and J. Sun, [Deep Residual Learning for Image Recognition](#), CVPR 2016 (Best Paper)

Why do ResNets work?

- ResNets are collections of many paths of different length, and shorter paths predominantly contribute to training

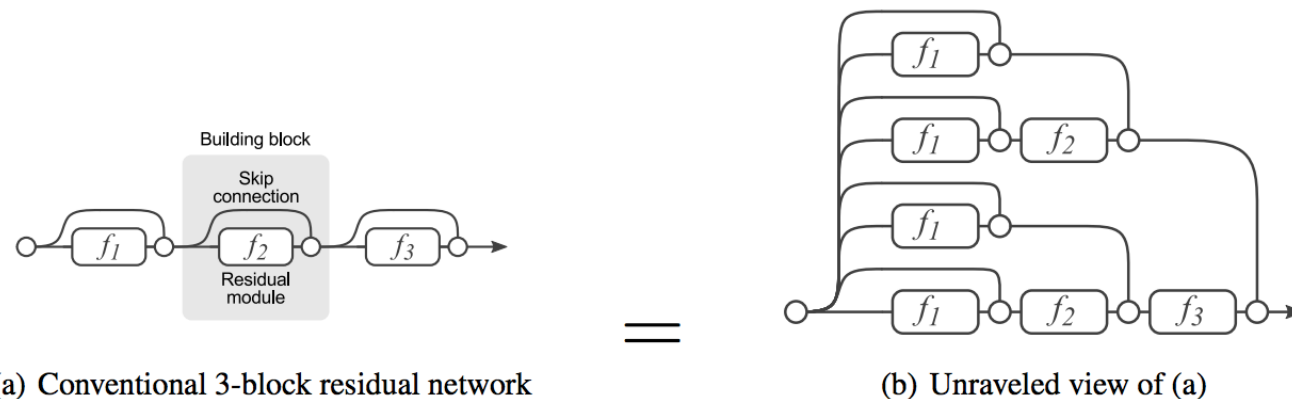


Figure 1: Residual Networks are conventionally shown as (a), which is a natural representation of Equation (1). When we expand this formulation to Equation (6), we obtain an *unraveled view* of a 3-block residual network (b). Circular nodes represent additions. From this view, it is apparent that residual networks have $O(2^n)$ implicit paths connecting input and output and that adding a block doubles the number of paths.

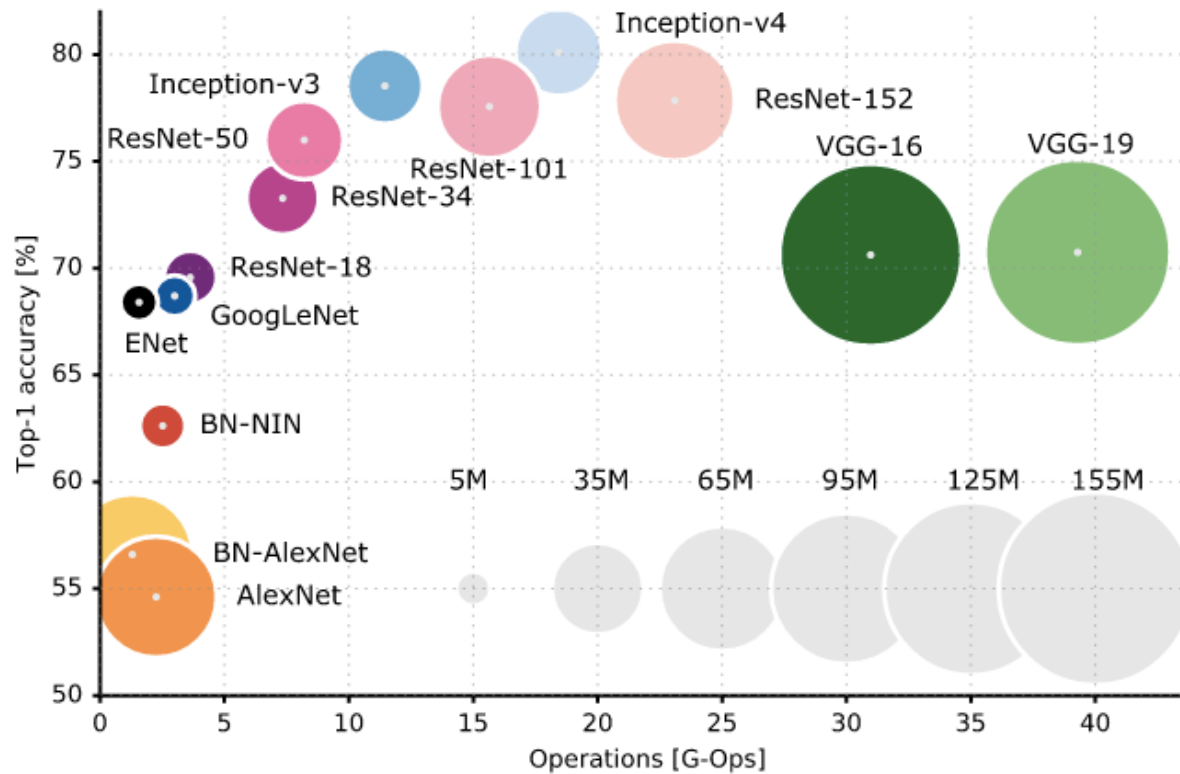
A. Veit, M. Wilber, S. Belongie, [Residual Networks Behave Like Ensembles of Relatively Shallow Networks](#), NIPS 2016

Summary: ILSVRC 2012-2015

Team	Year	Place	Error (top-5)	External data
SuperVision – Toronto (AlexNet, 7 layers)	2012	-	16.4%	no
SuperVision	2012	1st	15.3%	ImageNet 22k
Clarifai – NYU (7 layers)	2013	-	11.7%	no
Clarifai	2013	1st	11.2%	ImageNet 22k
VGG – Oxford (16 layers)	2014	2nd	7.32%	no
GoogLeNet (19 layers)	2014	1st	6.67%	no
ResNet (152 layers)	2015	1st*	3.57%	

*Officially, there was no longer a classification competition and ResNet-based systems won in localization and detection

Comparing architectures



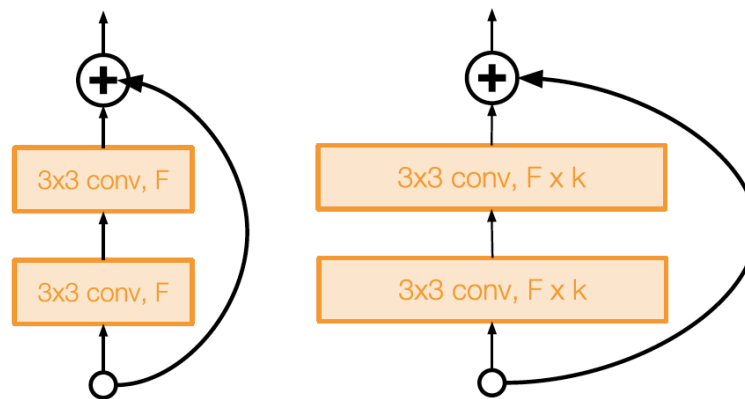
<https://culurciello.github.io/tech/2016/06/04/nets.html>

Outline

- AlexNet (2012-2013)
- VGGNet (2014)
- GoogLeNet (2014)
- ResNet (2015)
- Beyond ResNet (2016 and onward): Wide ResNet, ResNeXT, DenseNet

Beyond ResNet: Wide ResNet

- Reduce number of residual blocks, but increase number of feature maps in each block
 - More parallelizable, better feature reuse
 - 16-layer WRN outperforms 1000-layer ResNets, though with much larger # of parameters



Basic residual block

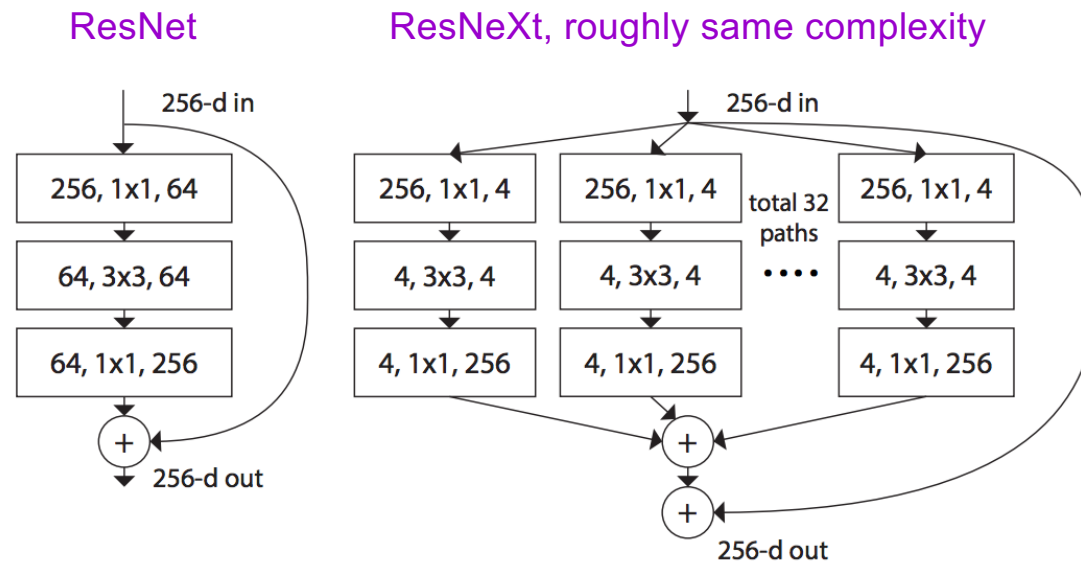
Wide residual block

[Image source](#)

S. Zagoryuko and N. Komodakis, [Wide Residual Networks](#), BMVC 2016

Beyond ResNet: ResNeXt

- Propose “cardinality” as a new factor in network design, apart from depth and width
- Claim that increasing cardinality is a better way to increase capacity than increasing depth or width



S. Xie, R. Girshick, P. Dollar, Z. Tu, and K. He, [Aggregated Residual Transformations for Deep Neural Networks](#), CVPR 2017

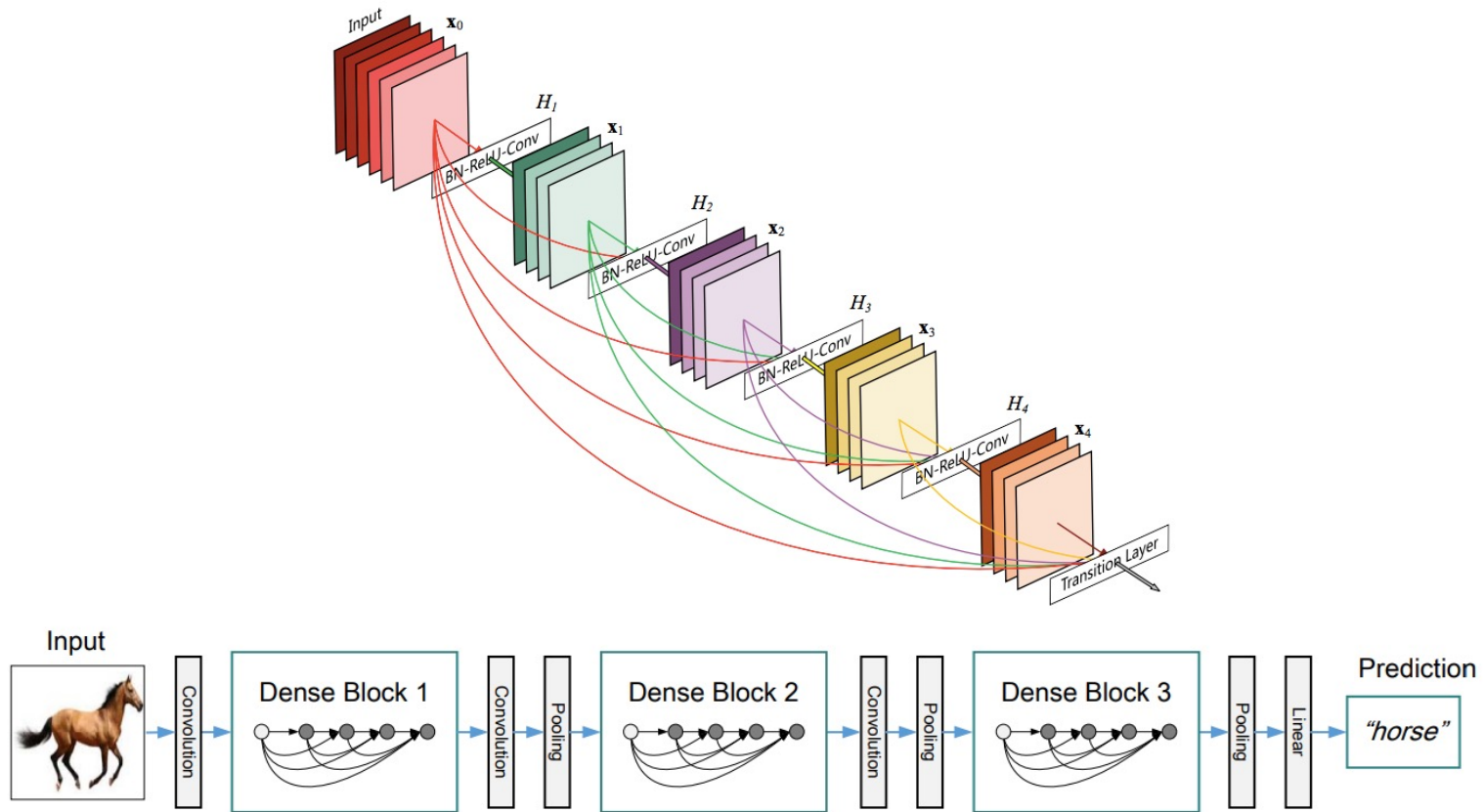
Beyond ResNet: ResNeXt

	setting	top-1 error (%)
ResNet-50	$1 \times 64d$	23.9
ResNeXt-50	$2 \times 40d$	23.0
ResNeXt-50	$4 \times 24d$	22.6
ResNeXt-50	$8 \times 14d$	22.3
ResNeXt-50	$32 \times 4d$	22.2
ResNet-101	$1 \times 64d$	22.0
ResNeXt-101	$2 \times 40d$	21.7
ResNeXt-101	$4 \times 24d$	21.4
ResNeXt-101	$8 \times 14d$	21.3
ResNeXt-101	$32 \times 4d$	21.2

Table 3. Ablation experiments on ImageNet-1K. (**Top**): ResNet-50 with preserved complexity (~ 4.1 billion FLOPs); (**Bottom**): ResNet-101 with preserved complexity (~ 7.8 billion FLOPs). The error rate is evaluated on the single crop of 224×224 pixels.

S. Xie, R. Girshick, P. Dollar, Z. Tu, and K. He, [Aggregated Residual Transformations for Deep Neural Networks](#), CVPR 2017

Beyond ResNet: DenseNets



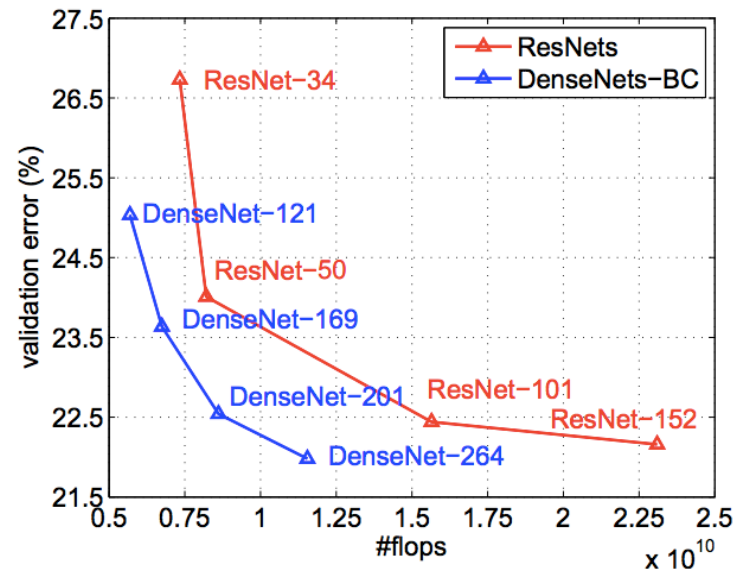
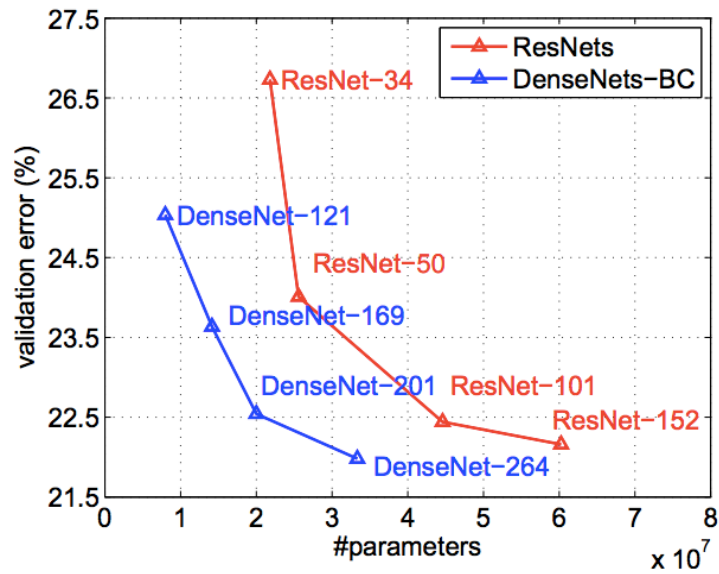
G. Huang, Z. Liu, and L. van der Maaten, [Densely Connected Convolutional Networks](#),
CVPR 2017 (Best Paper Award)

DenseNets

Layers	Output Size	DenseNet-121	DenseNet-169	DenseNet-201	DenseNet-264
Convolution	112 × 112	7 × 7 conv, stride 2			
Pooling	56 × 56	3 × 3 max pool, stride 2			
Dense Block (1)	56 × 56	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$
Transition Layer (1)	56 × 56	1 × 1 conv			
	28 × 28	2 × 2 average pool, stride 2			
Dense Block (2)	28 × 28	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$
Transition Layer (2)	28 × 28	1 × 1 conv			
	14 × 14	2 × 2 average pool, stride 2			
Dense Block (3)	14 × 14	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 24$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 48$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 64$
Transition Layer (3)	14 × 14	1 × 1 conv			
	7 × 7	2 × 2 average pool, stride 2			
Dense Block (4)	7 × 7	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 16$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 48$
Classification Layer	1 × 1	7 × 7 global average pool			
		1000D fully-connected, softmax			

G. Huang, Z. Liu, and L. van der Maaten, [Densely Connected Convolutional Networks](#), CVPR 2017 (Best Paper Award)

DenseNets



ImageNet validation error vs. number of parameters and test-time operations

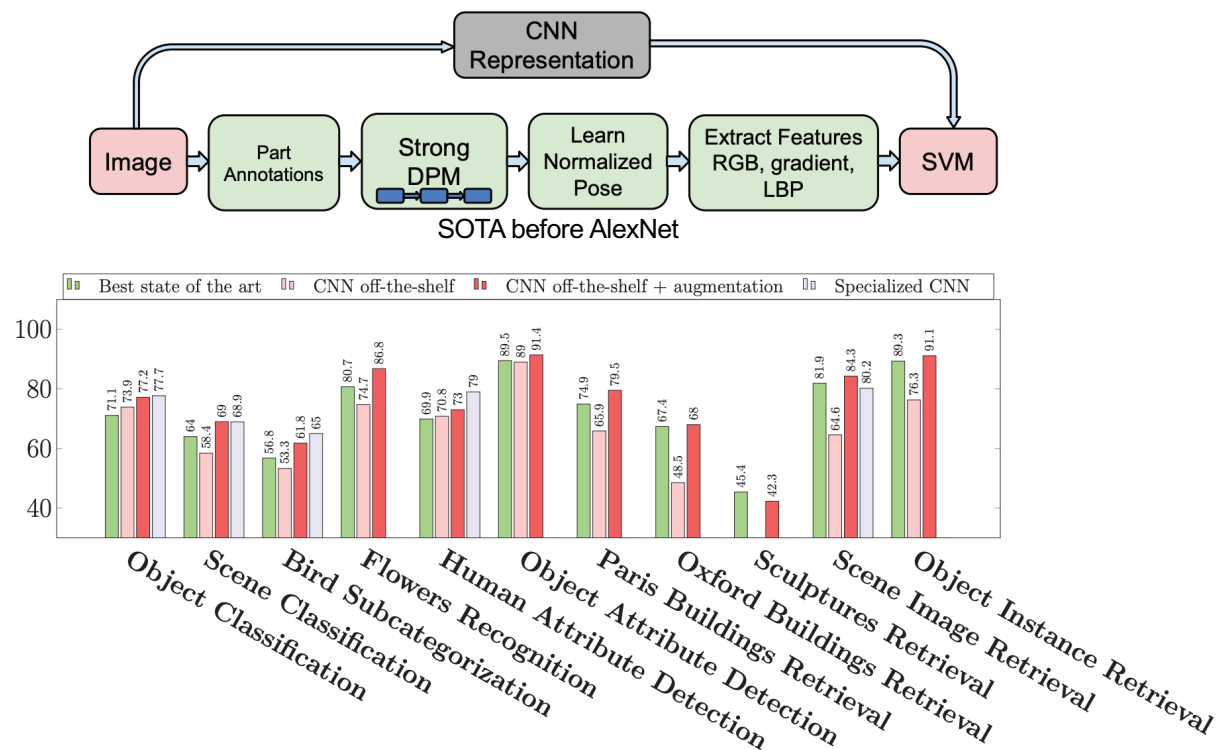
G. Huang, Z. Liu, and L. van der Maaten, [Densely Connected Convolutional Networks](#), CVPR 2017 (Best Paper Award)

Outline

- AlexNet (2012-2013)
- VGGNet (2014)
- GoogLeNet (2014)
- ResNet (2015)
- Beyond ResNet (2016 and onward): Wide ResNet, ResNeXT, DenseNet
- **Beyond ImageNet classification**

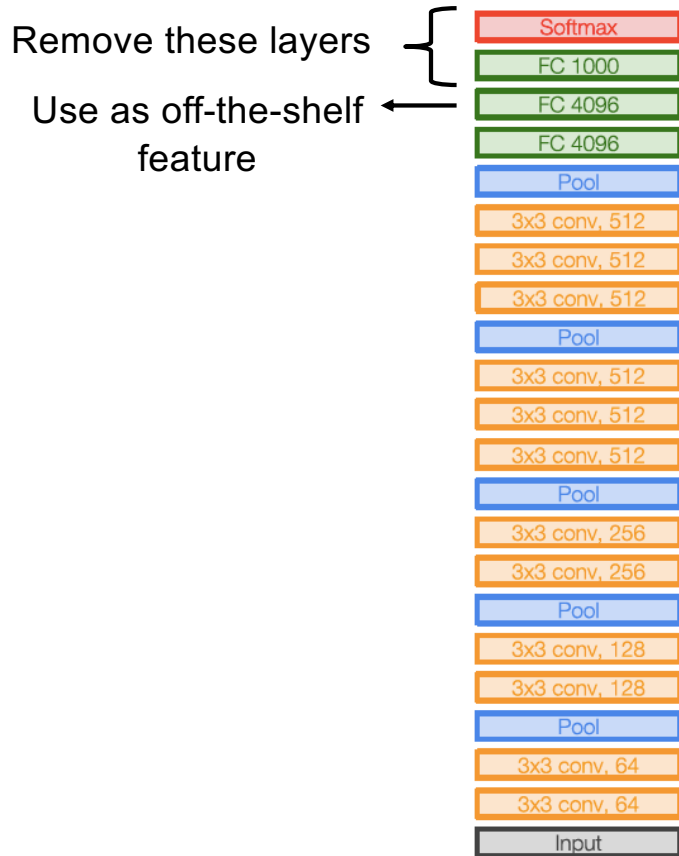
Beyond ILSVRC

- Question (circa 2013): what else are CNNs good for?



A. Razavian et al. [CNN Features off-the-shelf: an Astounding Baseline for Recognition.](#) CVPR 2014 workshops

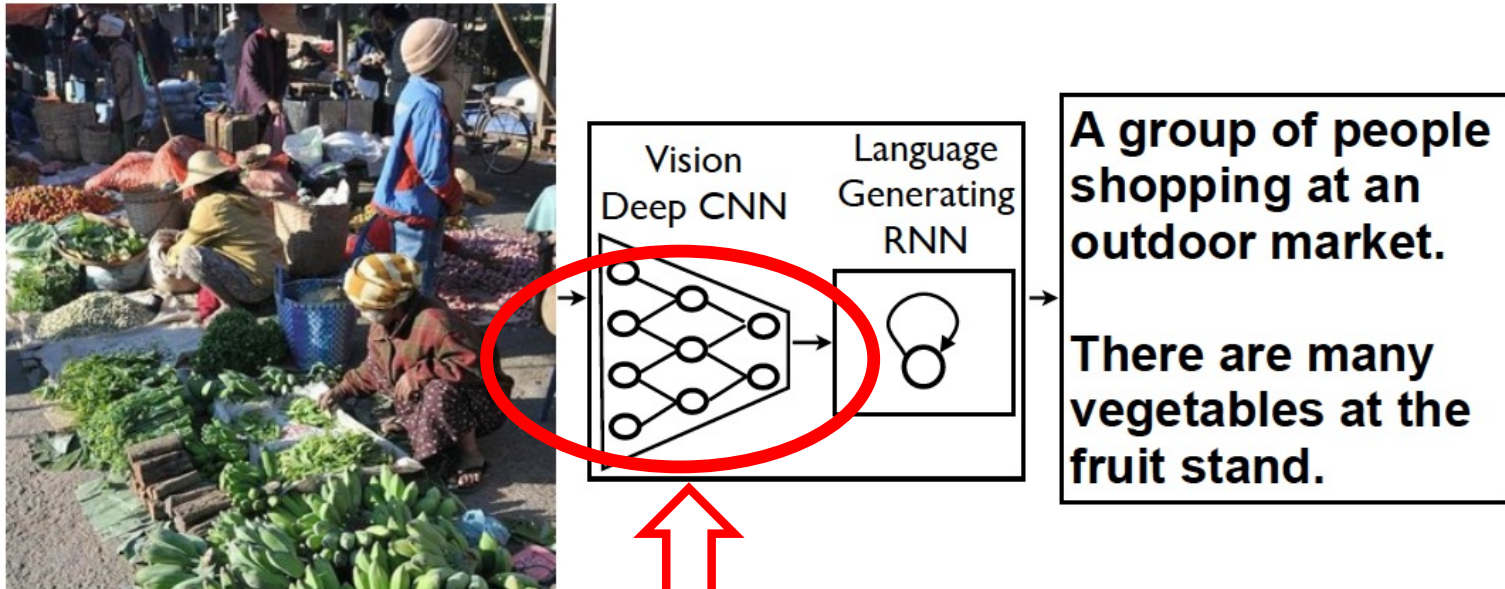
How to use a pre-trained network for a new task?



VGG16

- Strategy 1: Use as feature extractor

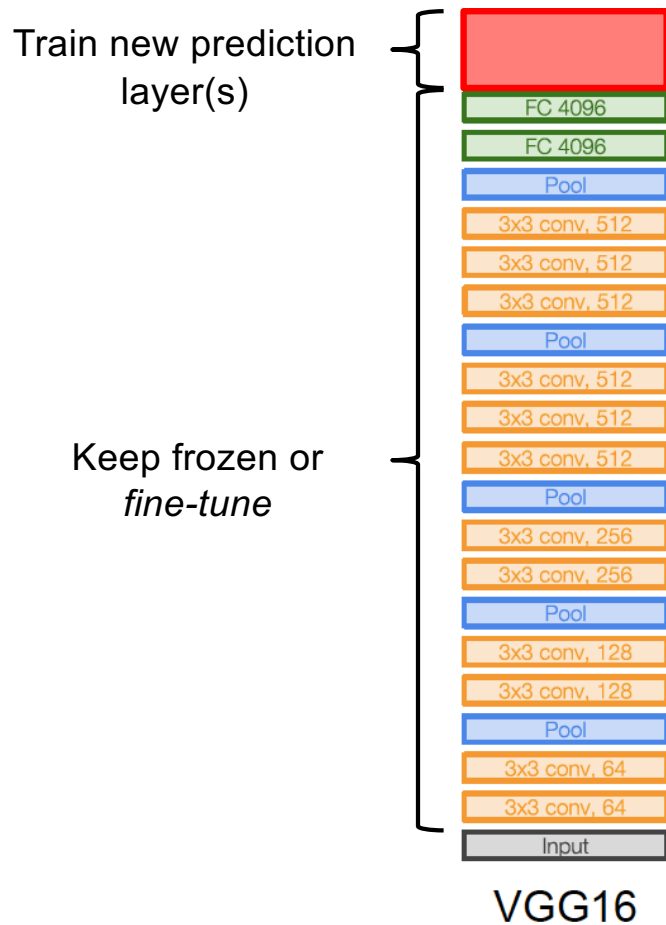
Example: CNNs for image captioning



**FC vectors from
pre-trained network**

O. Vinyals, A. Toshev, S. Bengio, and D. Erhan. [Show and tell: A neural image captioning generator](#). CVPR 2015

How to use a pre-trained network for a new task?



- Strategy 2: Transfer learning