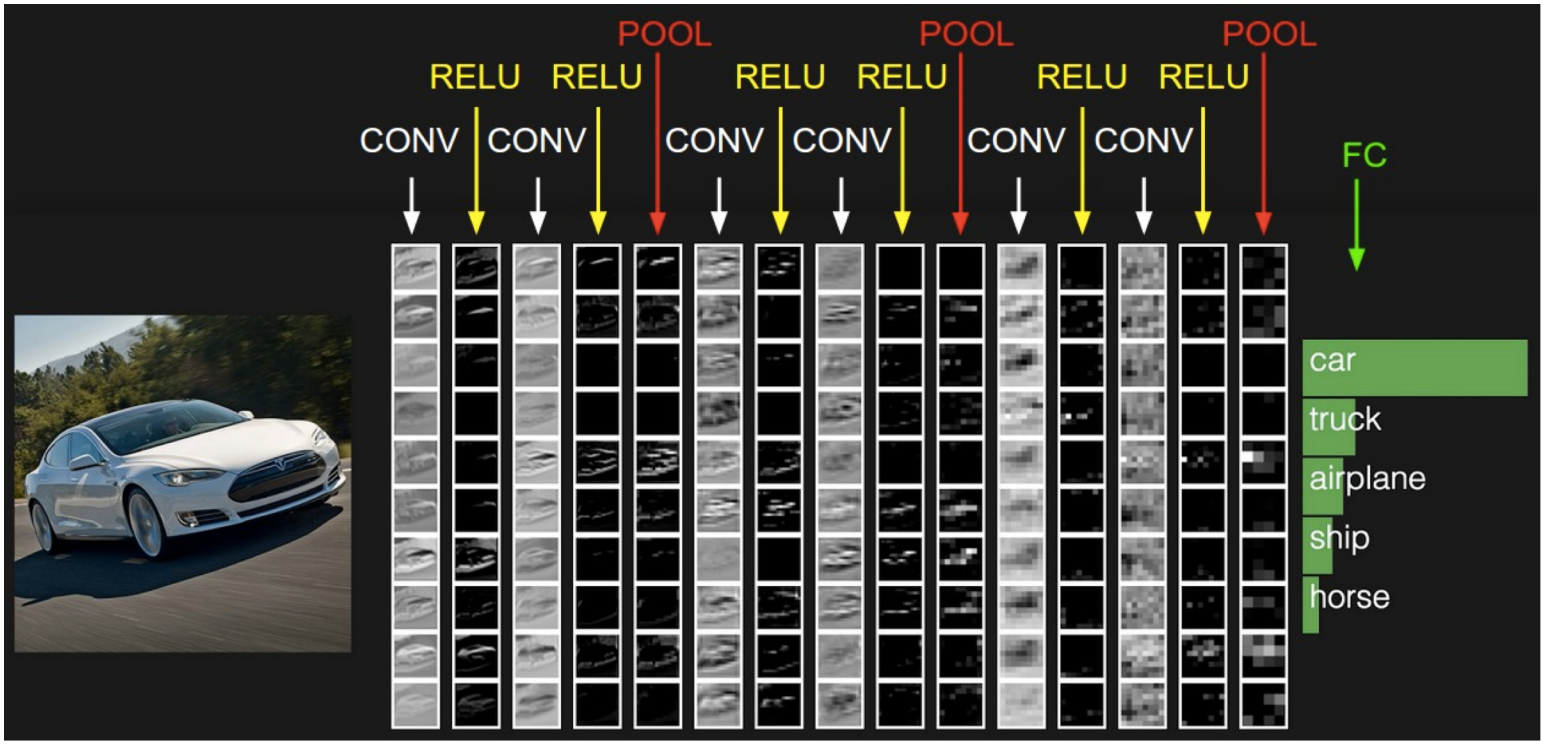
Convolutional neural networks

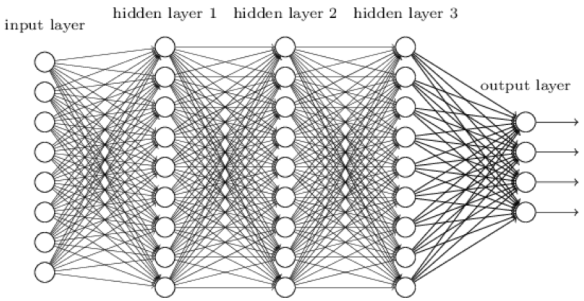


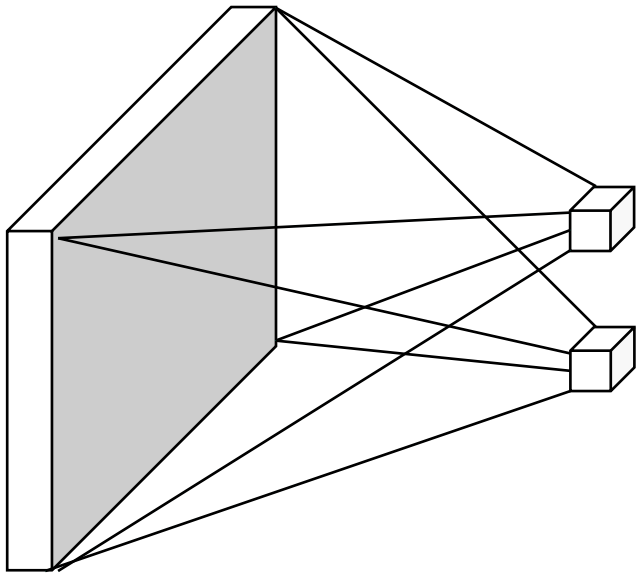
Many slides from Rob Fergus, Andrej Karpathy

Outline

- Building blocks
  - Convolutional layers and backprop rules
  - Pooling layers and nonlinearities
- Architectures:
  - 1st generation (2012-2013): AlexNet
  - 2nd generation (2014): VGGNet, GoogLeNet
  - 3rd generation (2015): ResNet
  - 4th generation (2016): ResNeXt, DenseNet

Convolutional layer anatomy

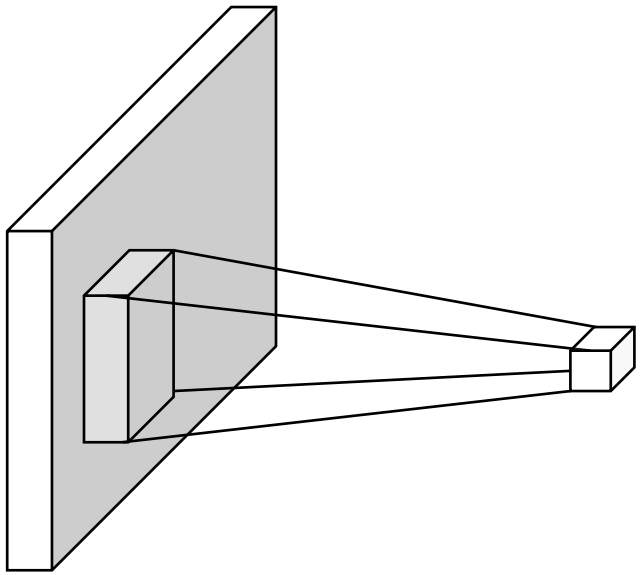




image

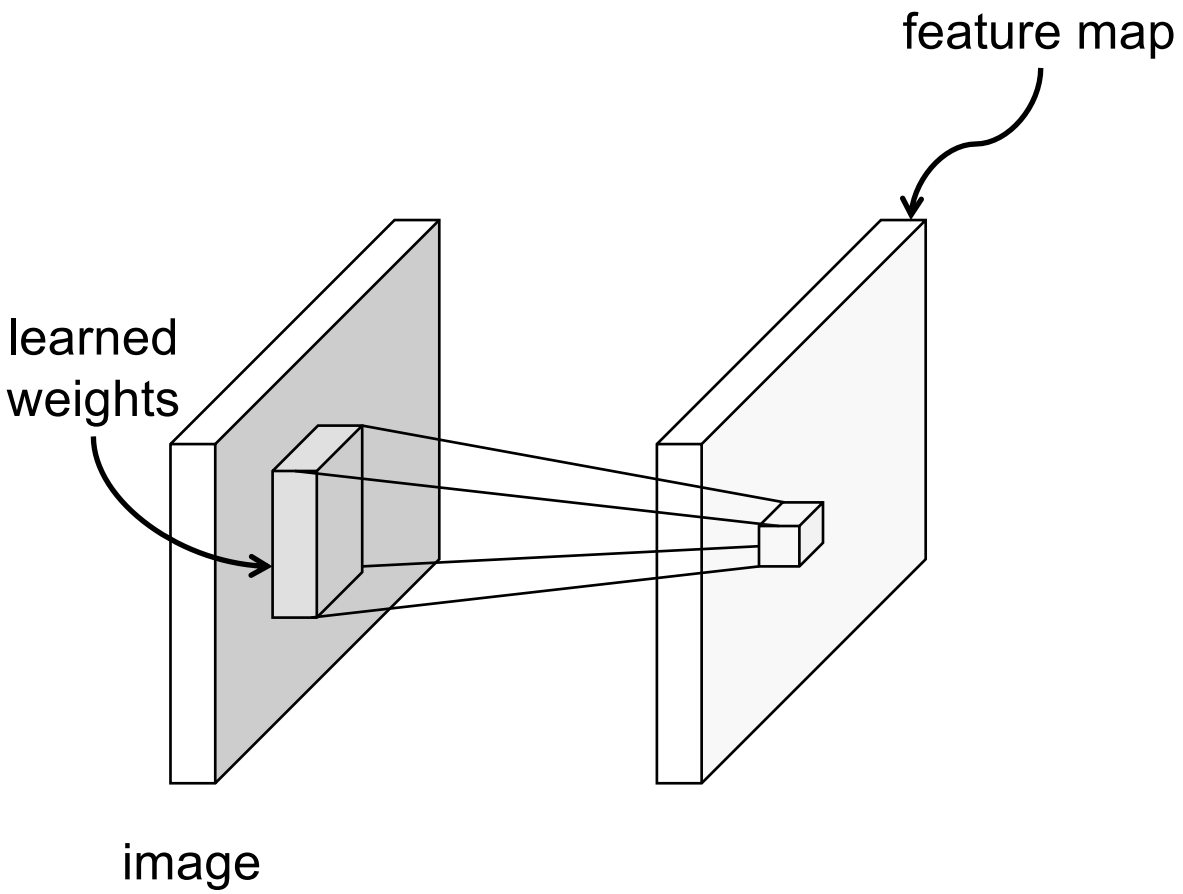
Fully connected layer

Convolutional layer anatomy

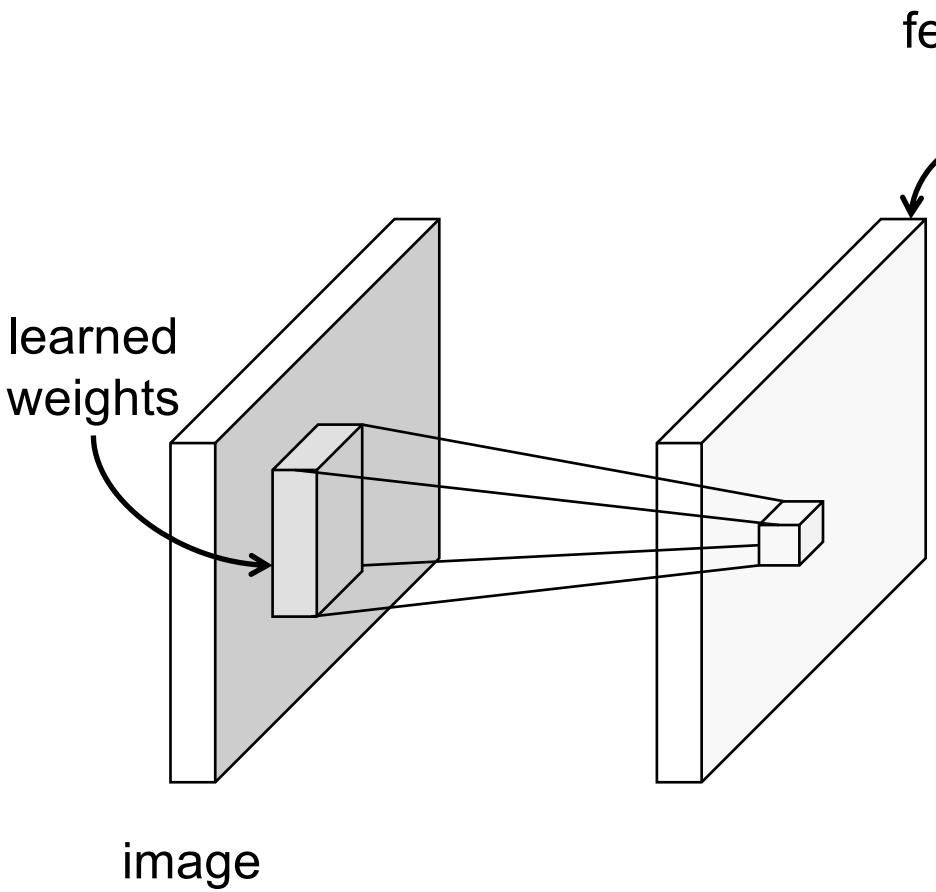


image

Convolutional layer anatomy

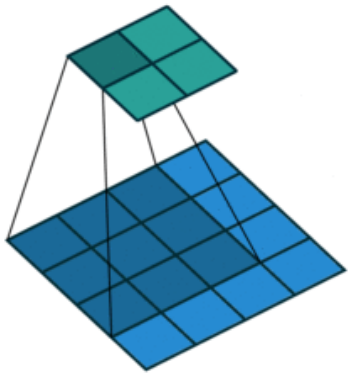


Convolutional layer anatomy



What is the feature map resolution?

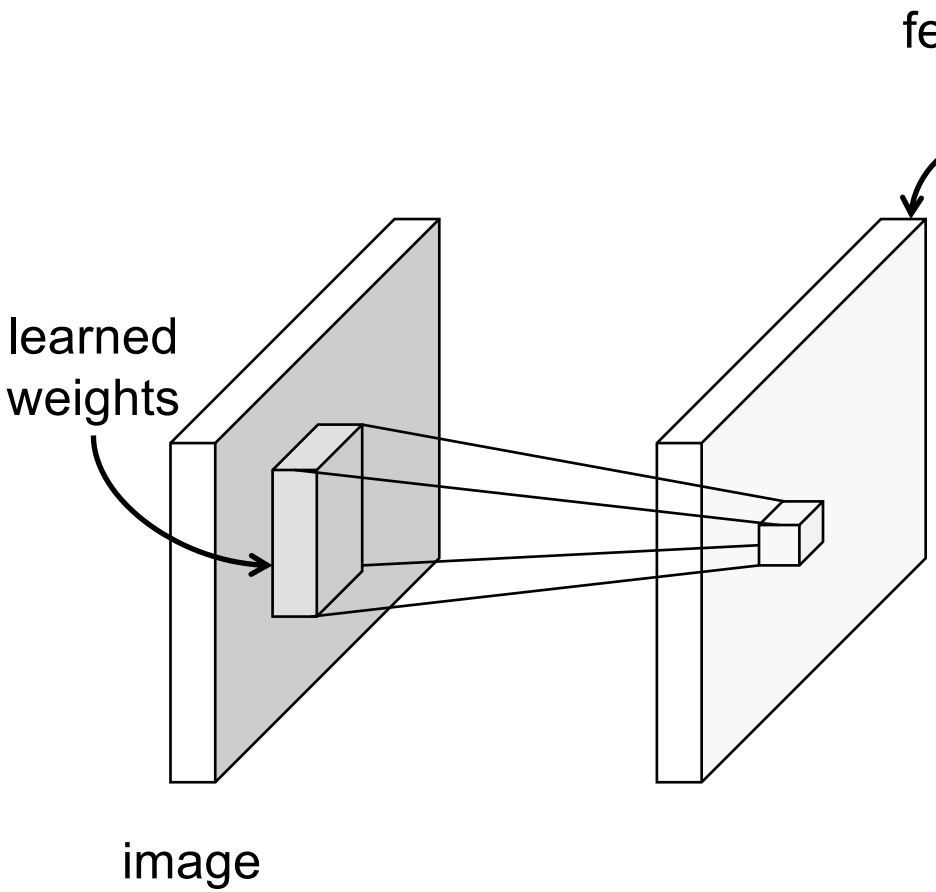
Depends on *padding* and *stride*:



No padding, stride 1

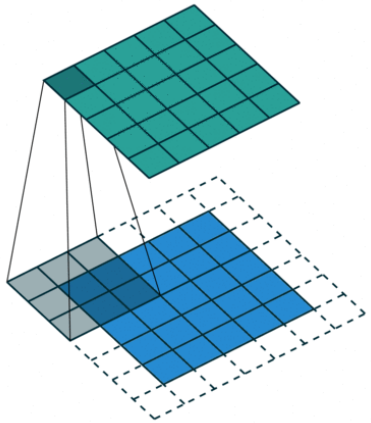
Animation source

Convolutional layer anatomy



What is the feature map resolution?

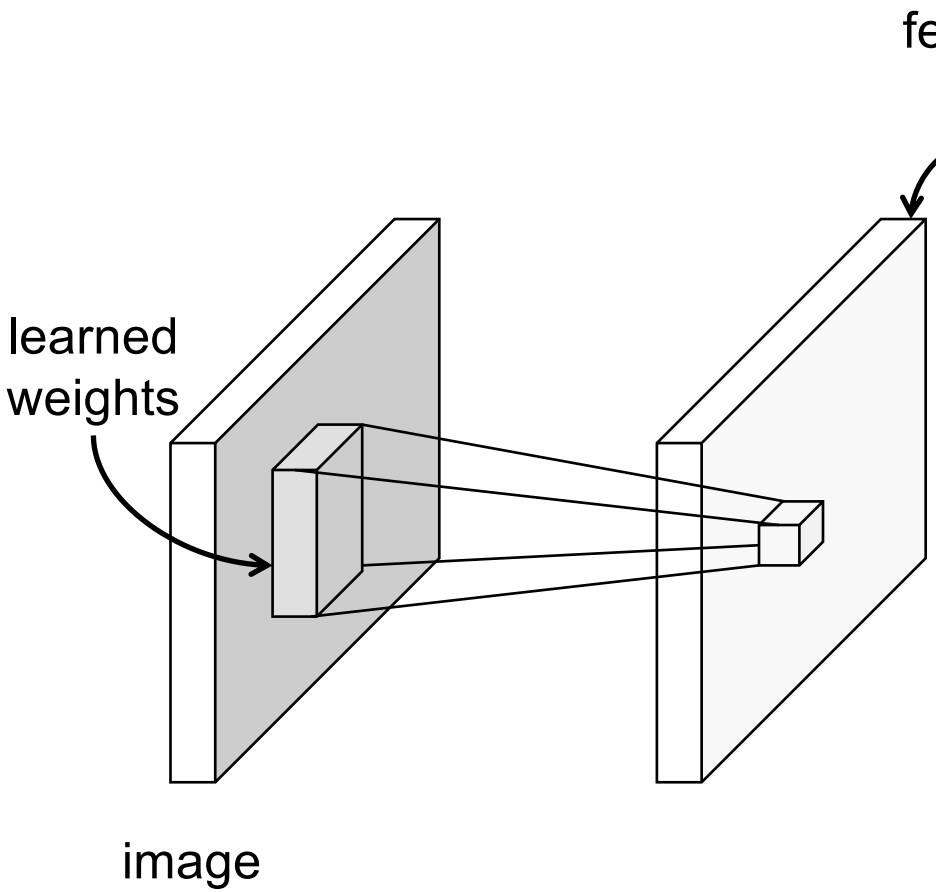
Depends on *padding* and *stride*:



With padding, stride 1

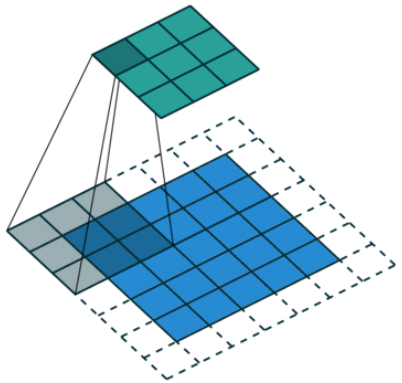
Animation source

Convolutional layer anatomy



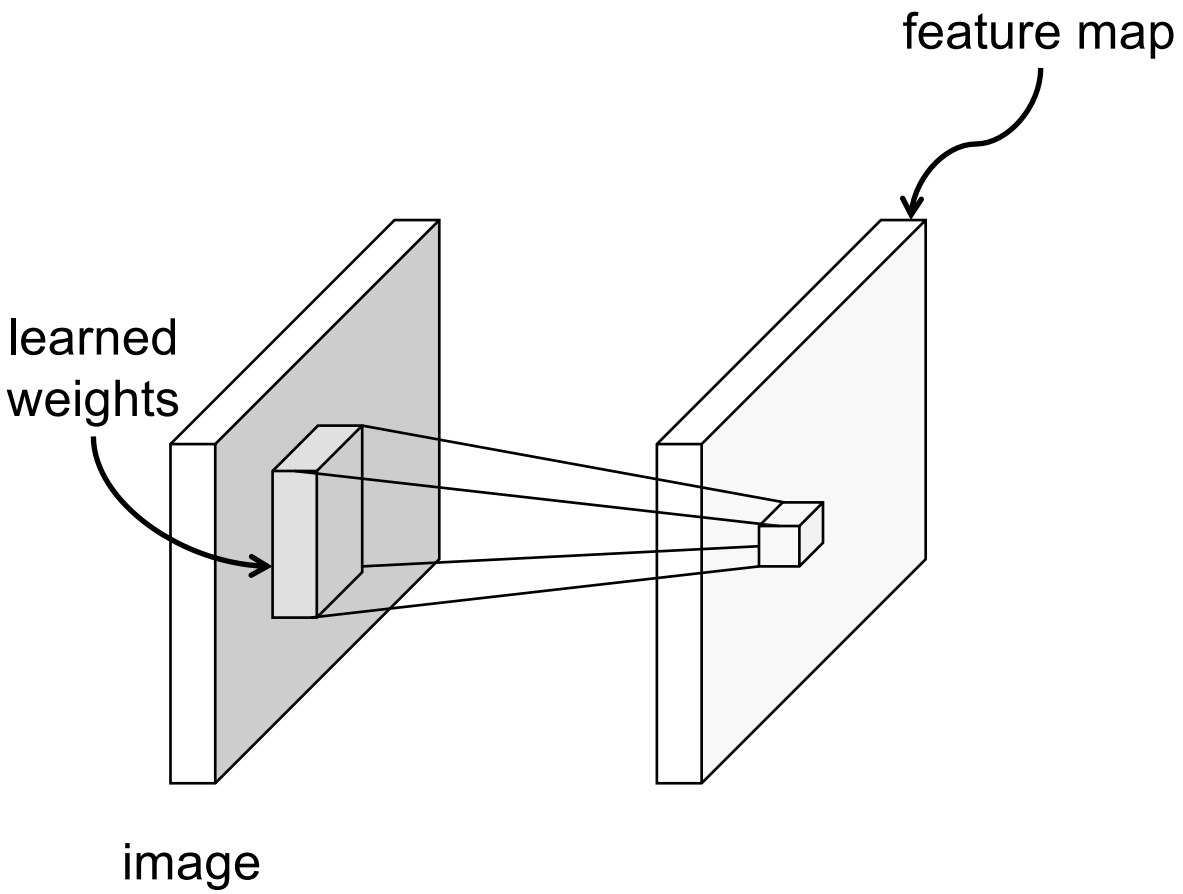
What is the feature map resolution?

Depends on *padding* and *stride*:

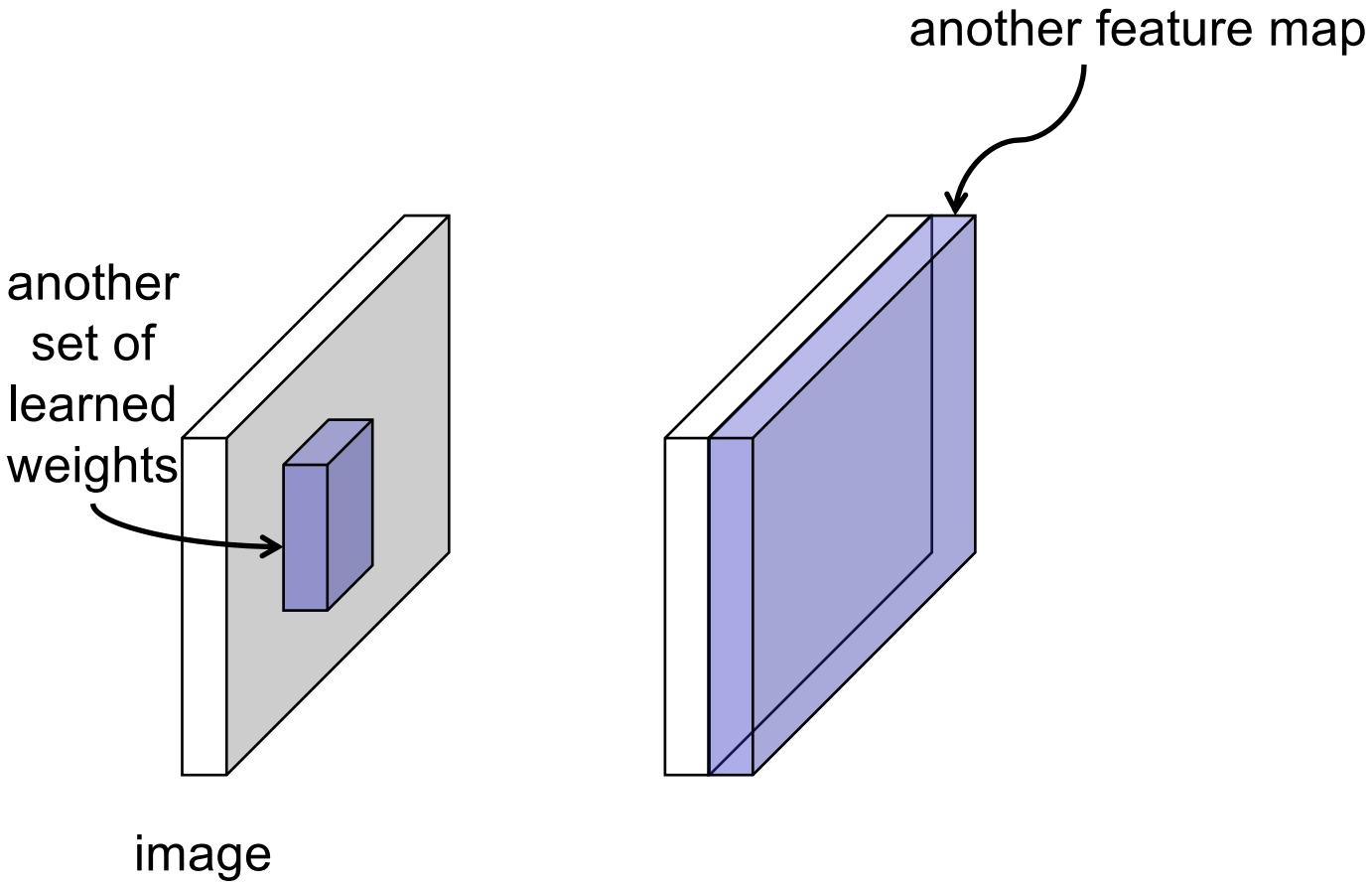


With padding, stride 2

Convolutional layer anatomy

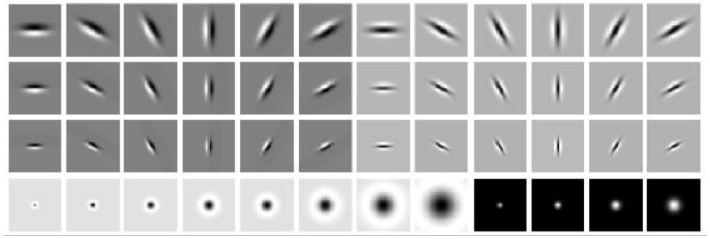


Convolutional layer anatomy

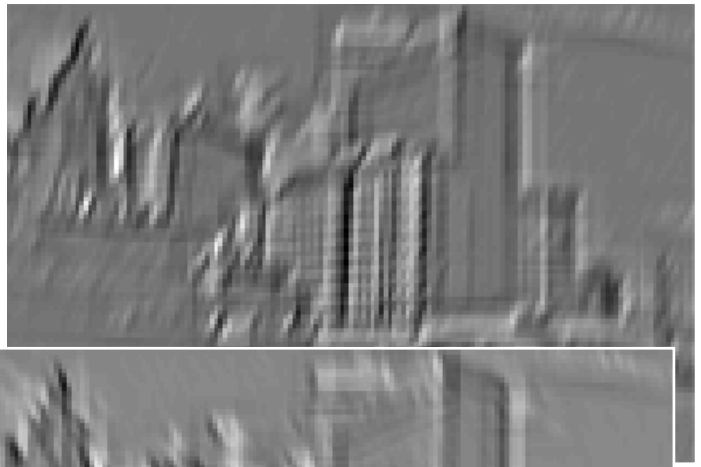


Convolution and traditional feature extraction

bank of K filters



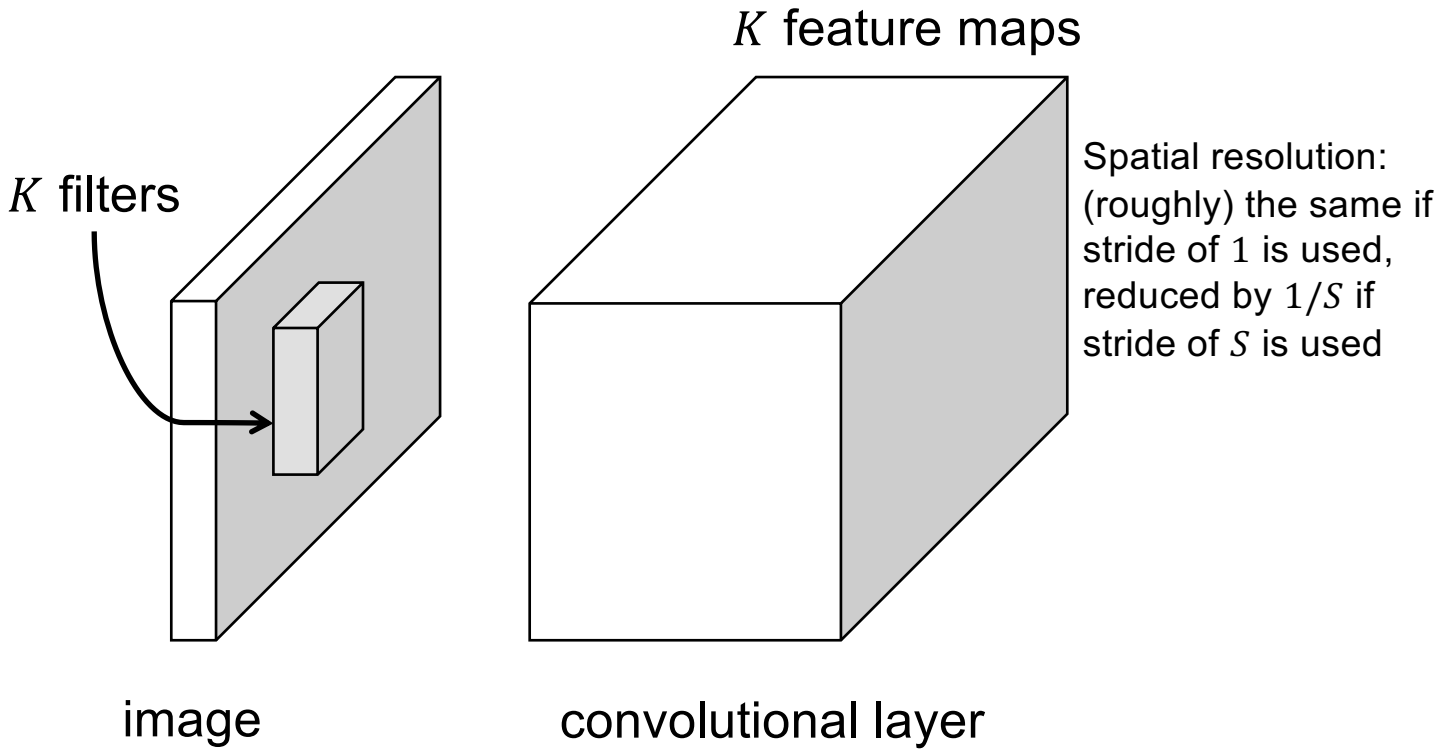
K feature maps



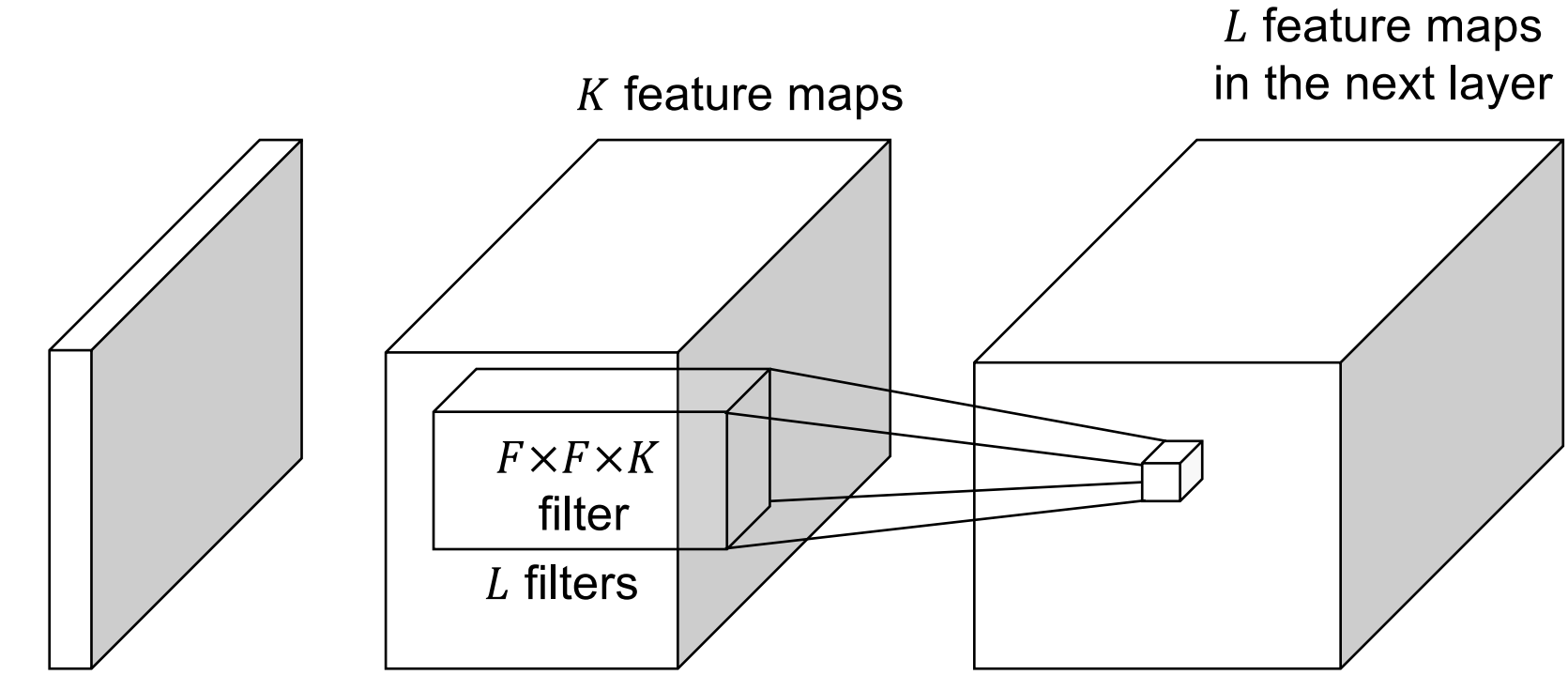
image

feature map

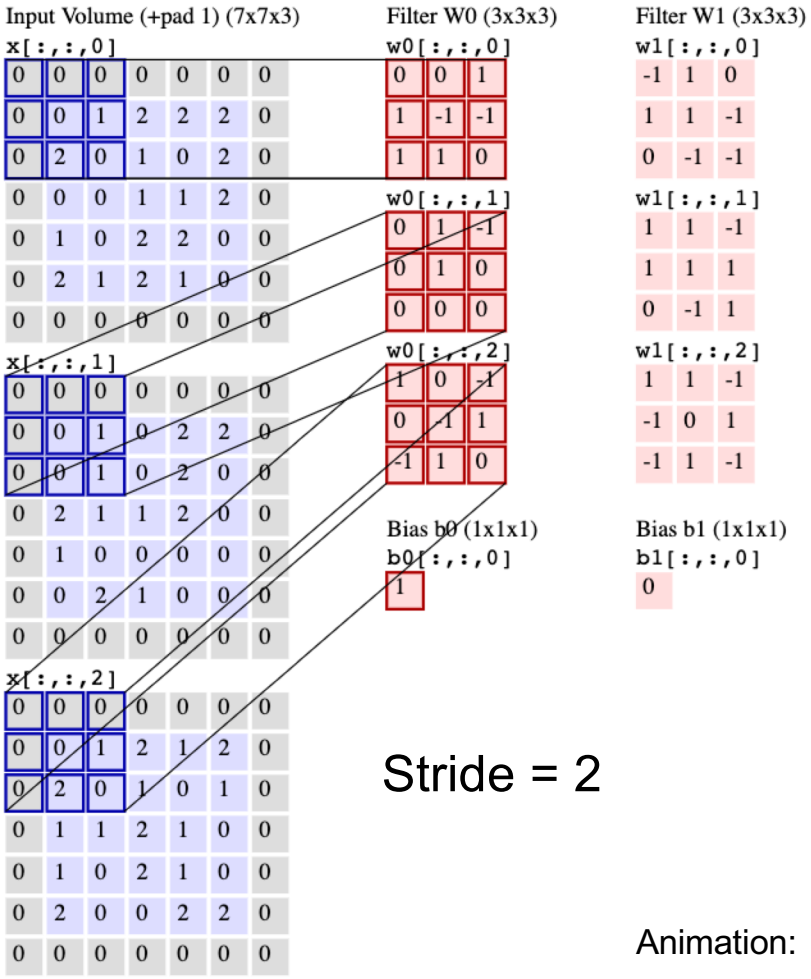
Convolutional layer anatomy



Convolutional layer anatomy



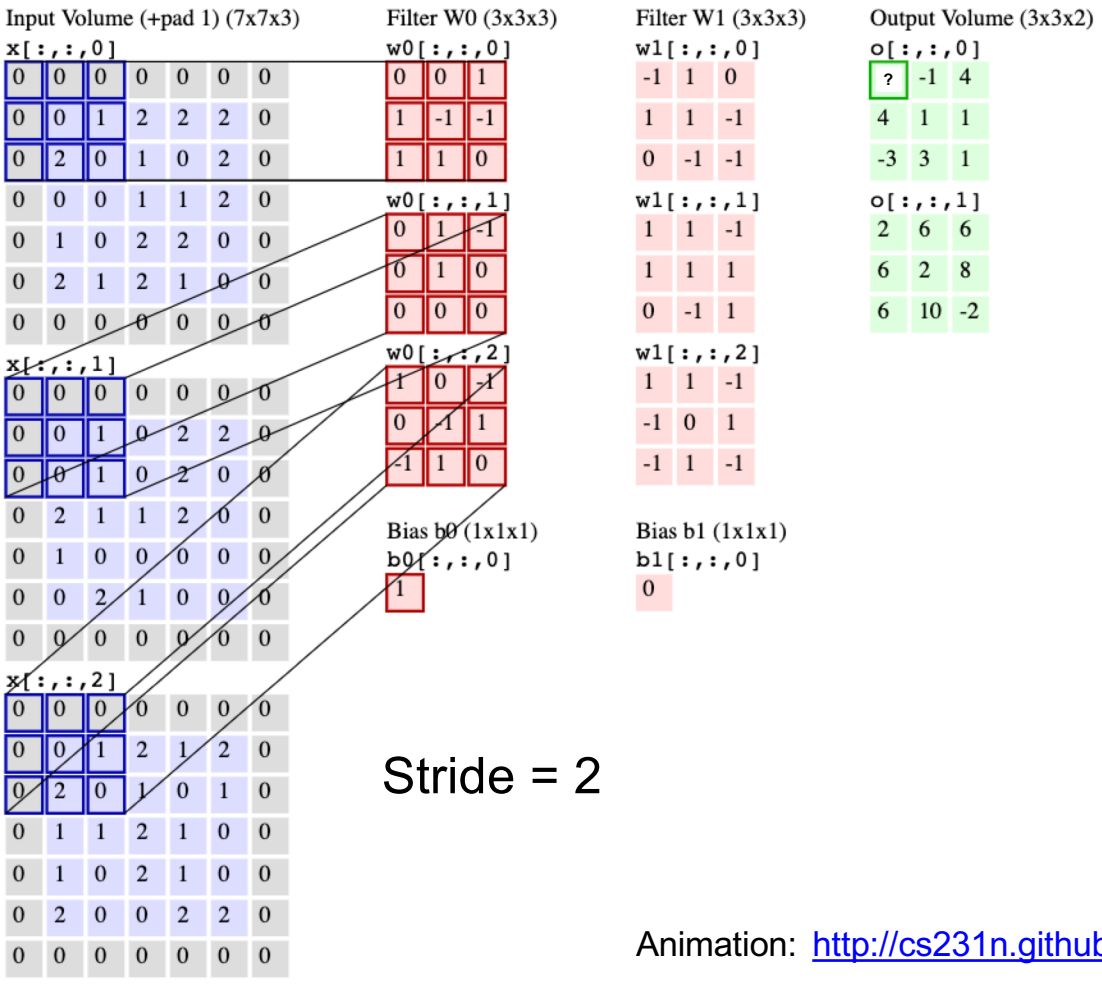
Convolutional layer example



Stride = 2

Animation: http://cs231n.github.io/convolutional-networks/#conv

Convolutional layer example



Animation: http://cs231n.github.io/convolutional-networks/#conv

Efficient implementation of convolutions

- Reshape all image neighborhoods into columns (im2col operation), do matrix-vector multiplication

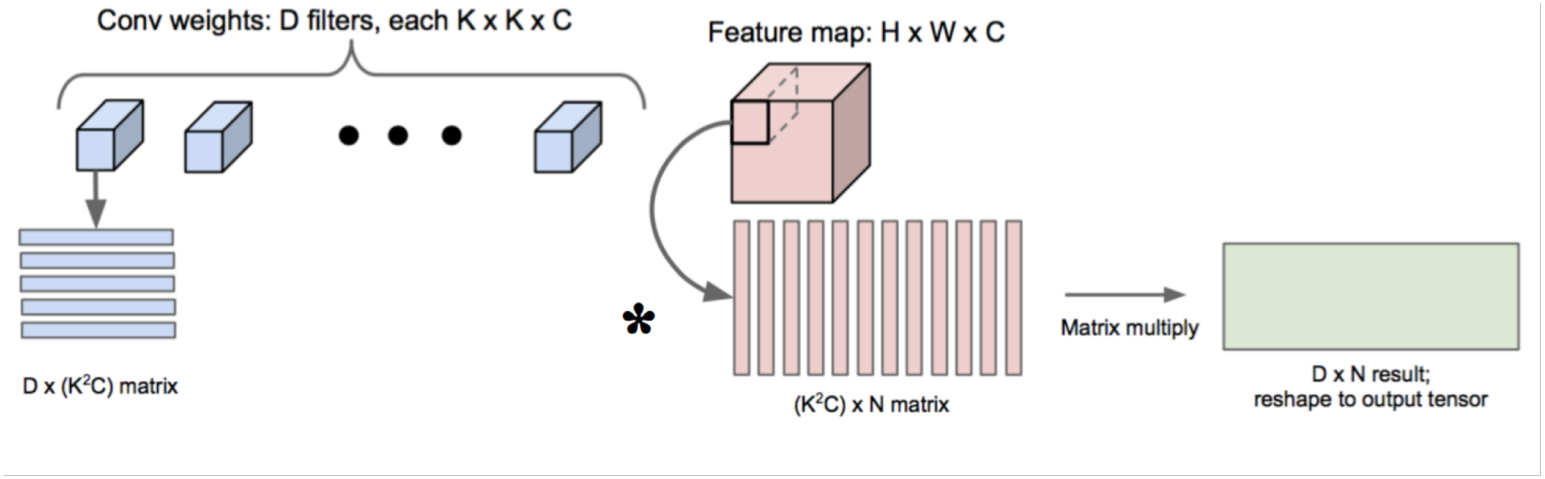
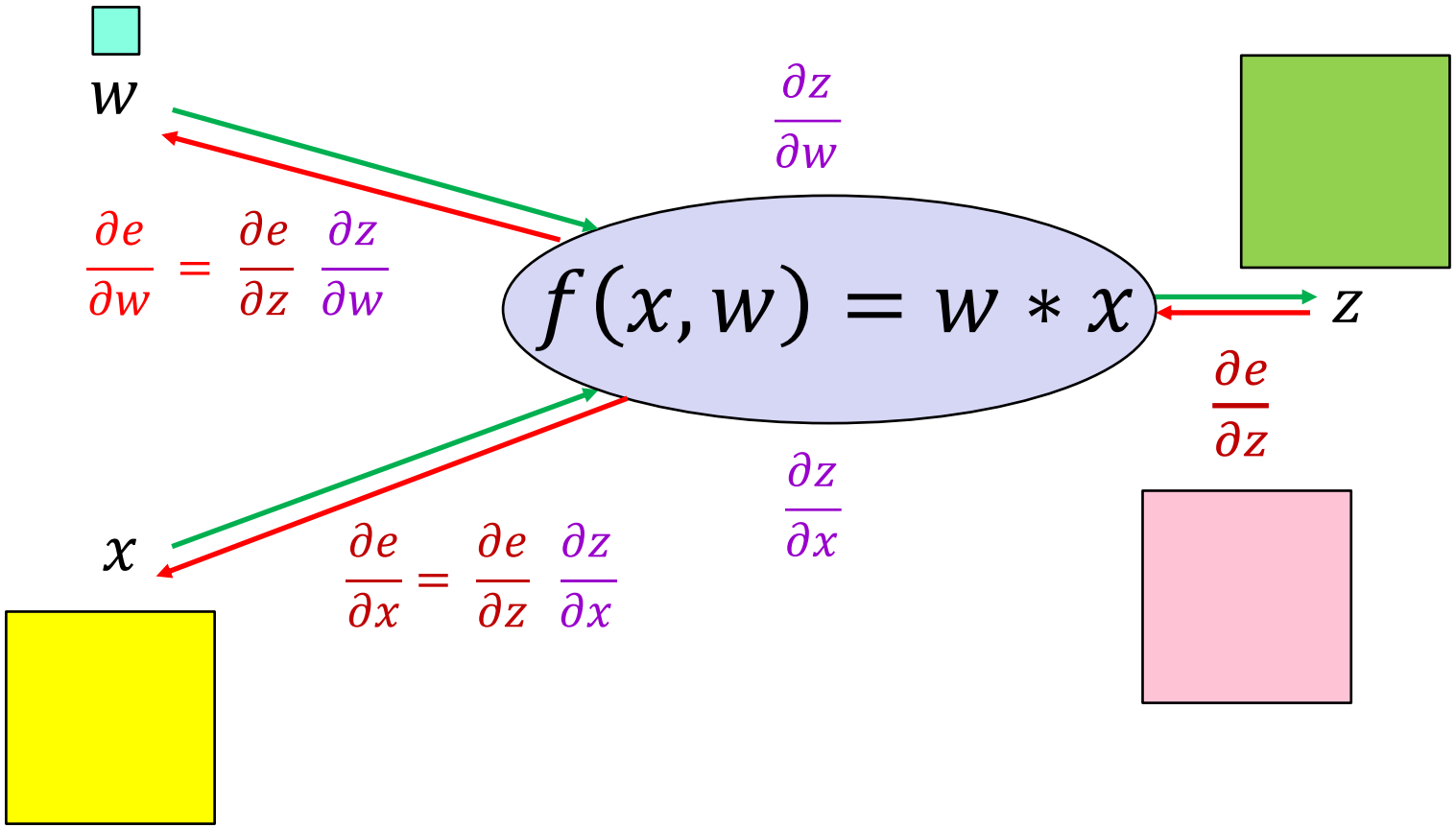


Image source

Backpropagation for convolutional layer



Backpropagation for convolutional layer

$$\frac{\partial e}{\partial w_{ij}}$$

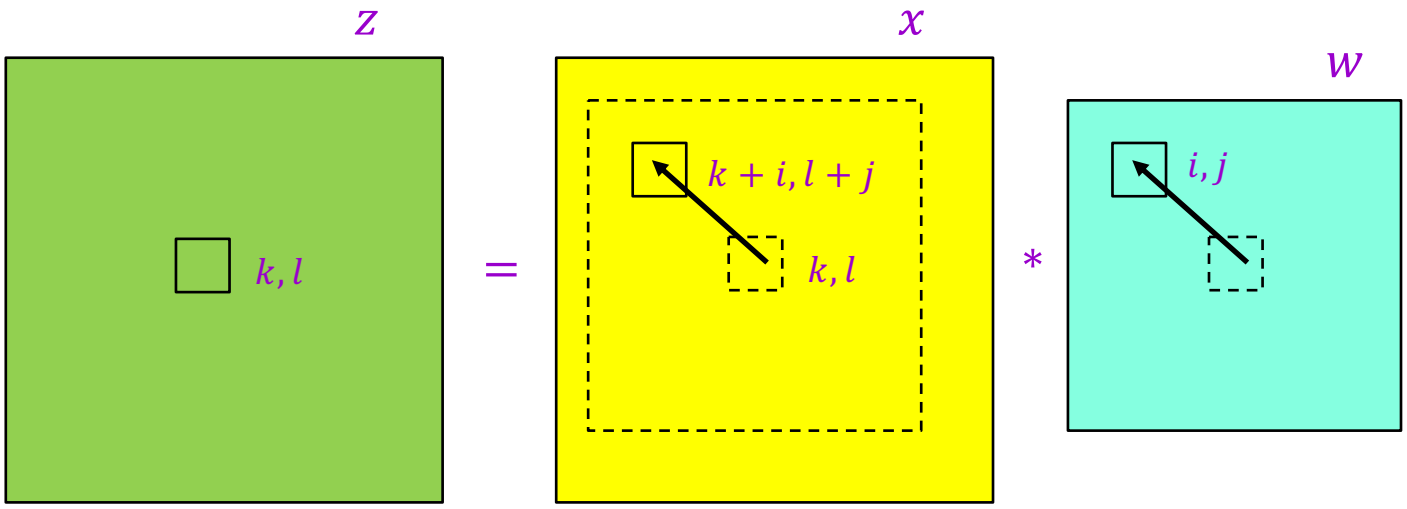
Backpropagation for convolutional layer

$$\frac{\partial e}{\partial w_{ij}} = \frac{\partial e}{\partial z}\frac{\partial z}{\partial w_{ij}} = \sum_{k,l}\frac{\partial e}{\partial z_{kl}}\frac{\partial z_{kl}}{\partial w_{ij}}$$

$$z_{kl} = \sum_{i,j=-f}^{f} w_{ij} x_{k+i,\,l+j}$$

For simplicity, assume filter indices go from $-f$ to f

$$\frac{\partial z_{kl}}{\partial w_{ij}} = x_{k+i,\,l+j}$$



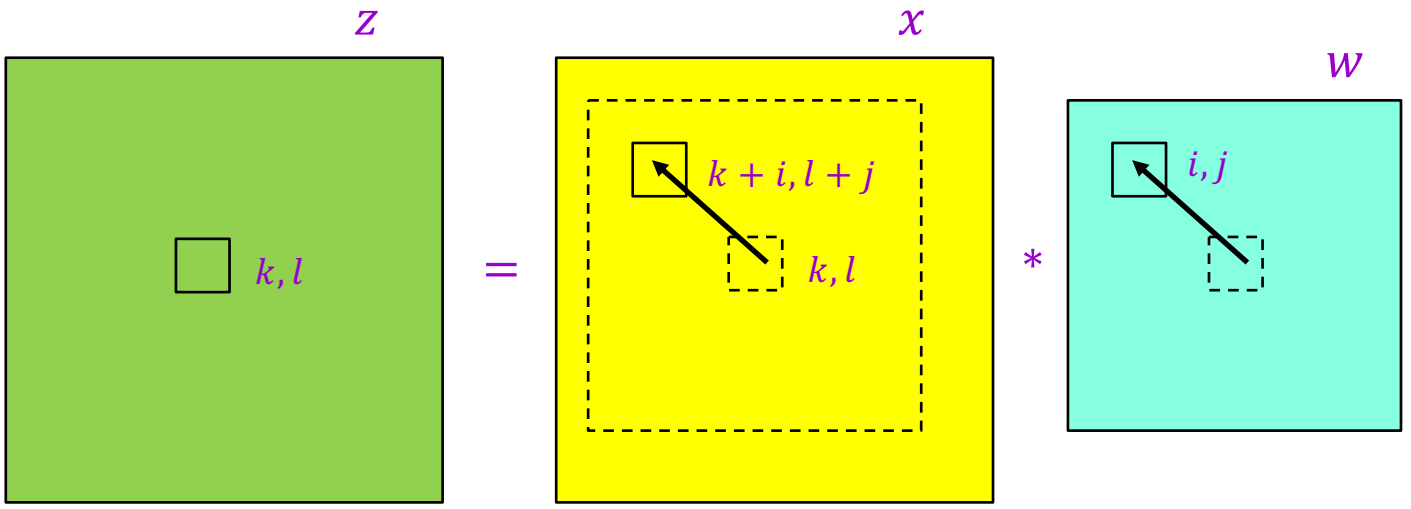
Backpropagation for convolutional layer

$$\frac{\partial e}{\partial w_{ij}} = \frac{\partial e}{\partial z}\frac{\partial z}{\partial w_{ij}} = \sum_{k,l}\frac{\partial e}{\partial z_{kl}}\frac{\partial z_{kl}}{\partial w_{ij}} = \boxed{\sum_{k,l}\frac{\partial e}{\partial z_{kl}}x_{k+i,\,l+j}}$$

$$z_{kl} = \sum_{i,j=-f}^{f} w_{ij}\,x_{k+i,\,l+j}$$

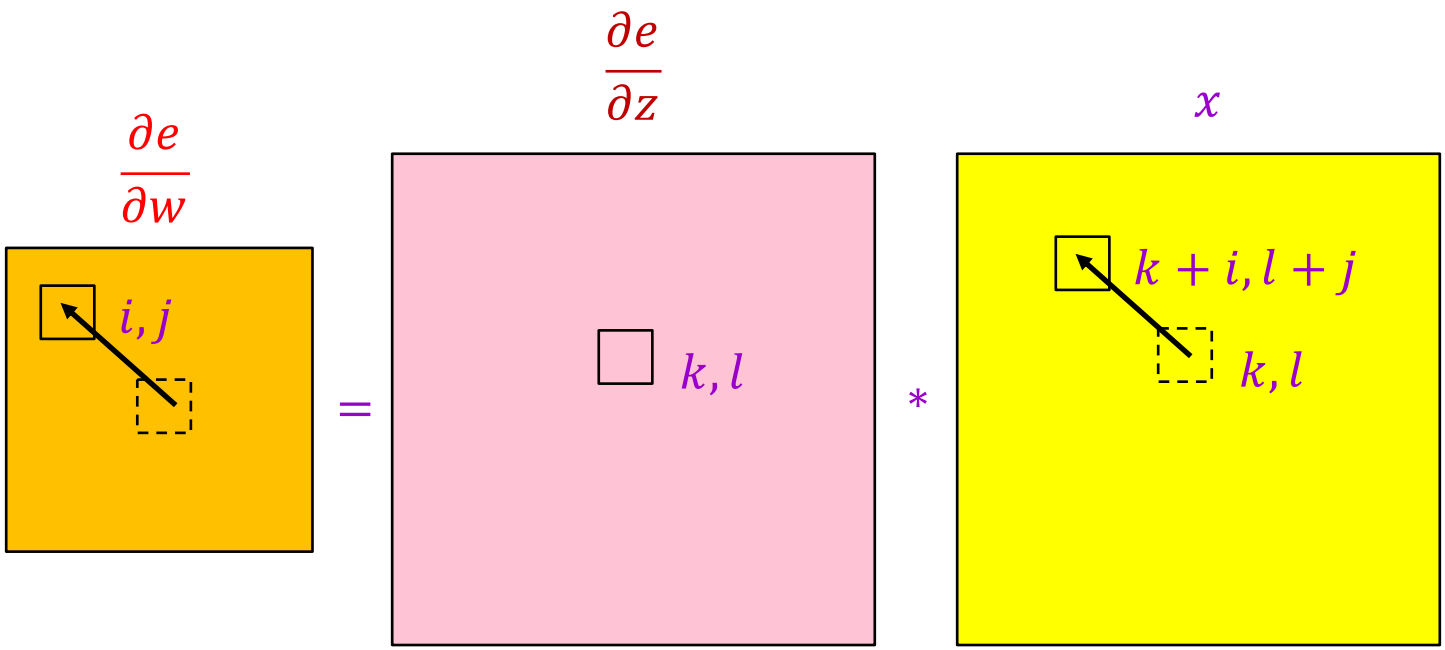
For simplicity, assume filter indices go from $-f$ to f

$$\frac{\partial z_{kl}}{\partial w_{ij}} = x_{k+i,\,l+j}$$



Backpropagation for convolutional layer

$$\frac{\partial e}{\partial w_{ij}} = \sum_{k,l} \frac{\partial e}{\partial z_{kl}} x_{k+i,\, l+j}$$

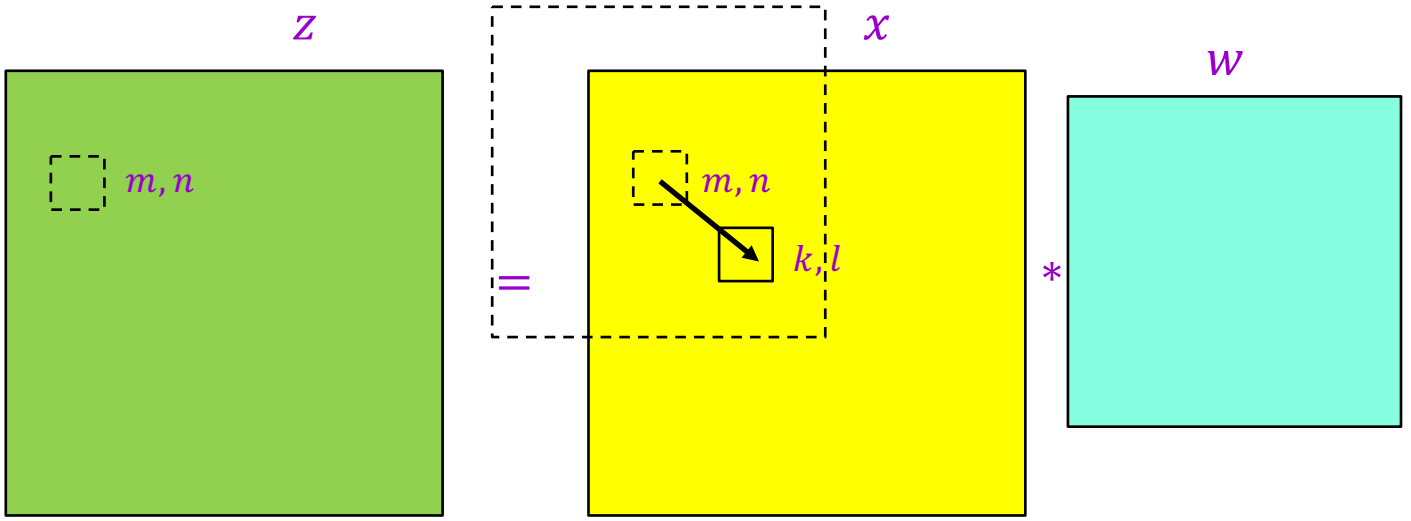


Backpropagation for convolutional layer

$$\frac{\partial e}{\partial x_{kl}}$$

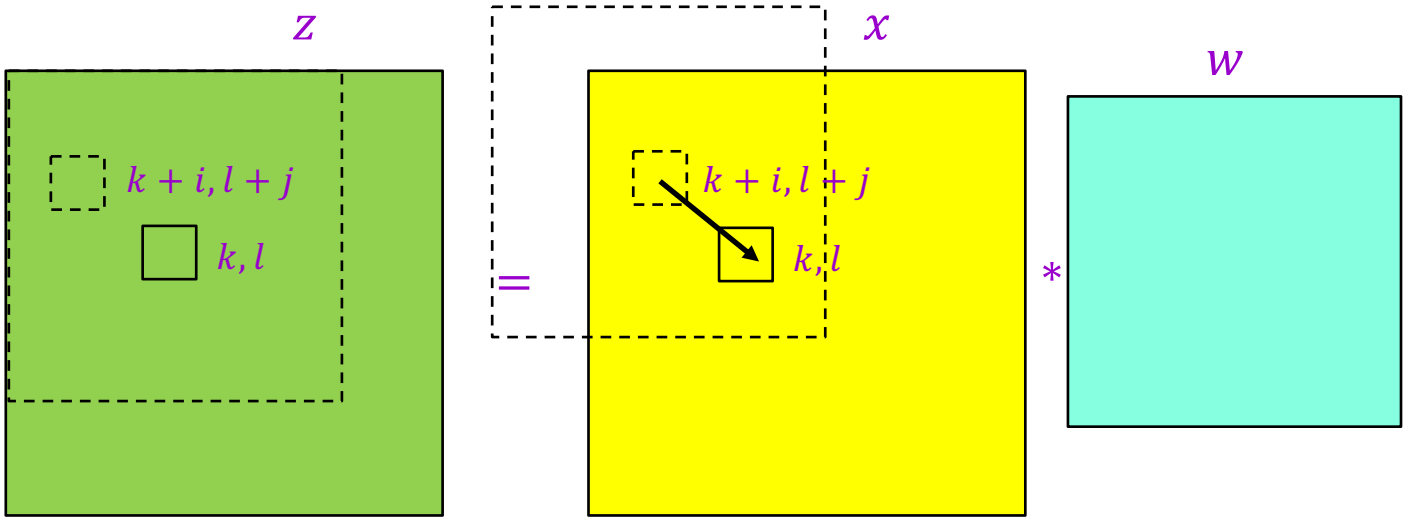
Backpropagation for convolutional layer

$$\frac{\partial e}{\partial x_{kl}} = \frac{\partial e}{\partial z}\,\frac{\partial z}{x_{kl}} = \sum_{m,n} \frac{\partial e}{\partial z_{mn}}\frac{\partial z_{mn}}{\partial x_{kl}}$$



Backpropagation for convolutional layer

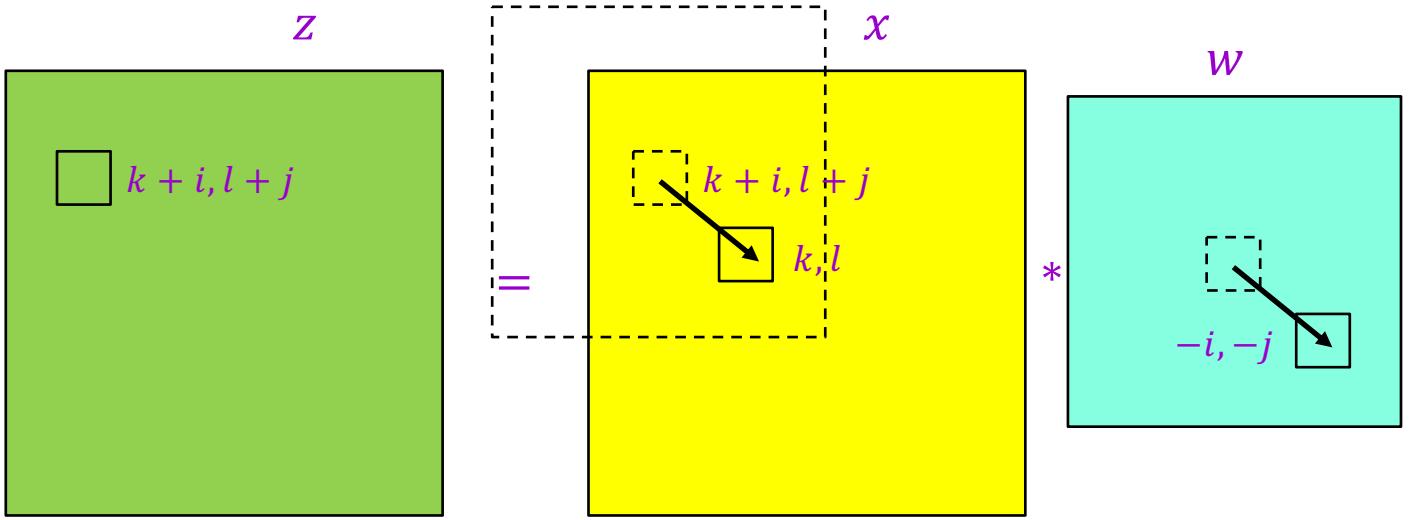
$$\frac{\partial e}{\partial x_{kl}} = \frac{\partial e}{\partial z}\,\frac{\partial z}{x_{kl}} = \sum_{m,n} \frac{\partial e}{\partial z_{mn}}\frac{\partial z_{mn}}{\partial x_{kl}} = \sum_{i,j} \frac{\partial e}{\partial z_{k+i,l+j}}\frac{\partial z_{k+i,l+j}}{\partial x_{kl}}$$



Backpropagation for convolutional layer

$$\frac{\partial e}{\partial x_{kl}} = \sum_{i,j} \frac{\partial e}{\partial z_{k+i,l+j}} \frac{\partial z_{k+i,l+j}}{\partial x_{kl}}$$

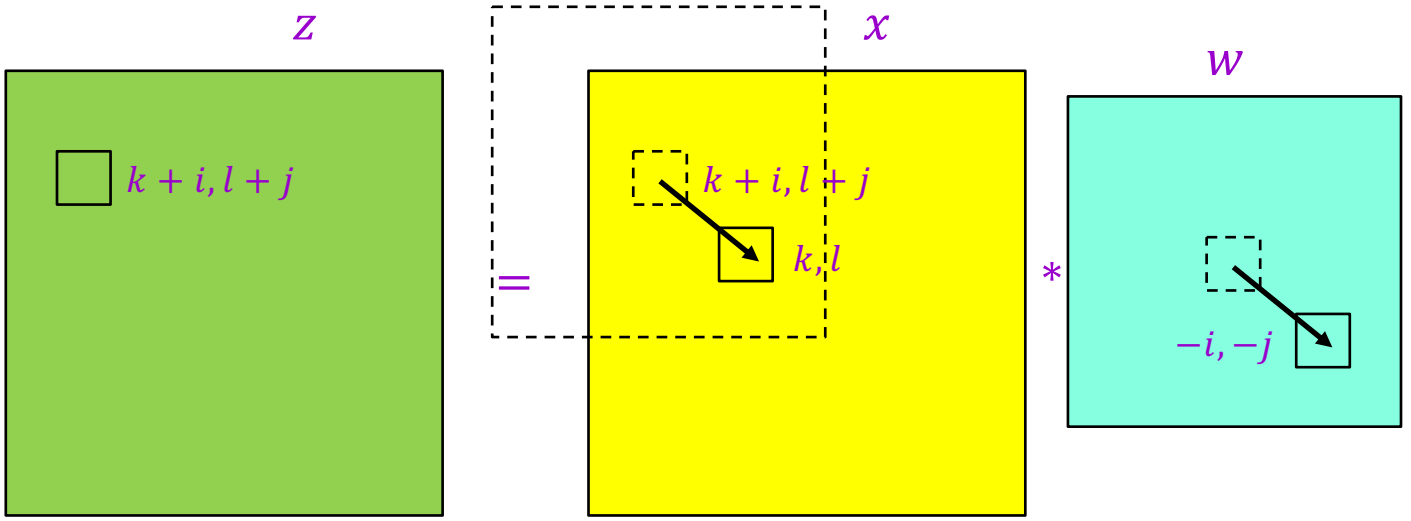
$$\frac{\partial z_{k+i,l+j}}{\partial x_{kl}} = w_{-i,-j}$$



Backpropagation for convolutional layer

$$\frac{\partial e}{\partial x_{kl}} = \sum_{i,j} \frac{\partial e}{\partial z_{k+i,l+j}} \frac{\partial z_{k+i,l+j}}{\partial x_{kl}} = \boxed{\sum_{i,j} \frac{\partial e}{\partial z_{k+i,l+j}} w_{-i,-j}}$$

$$\frac{\partial z_{k+i,l+j}}{\partial x_{kl}} = w_{-i,-j}$$

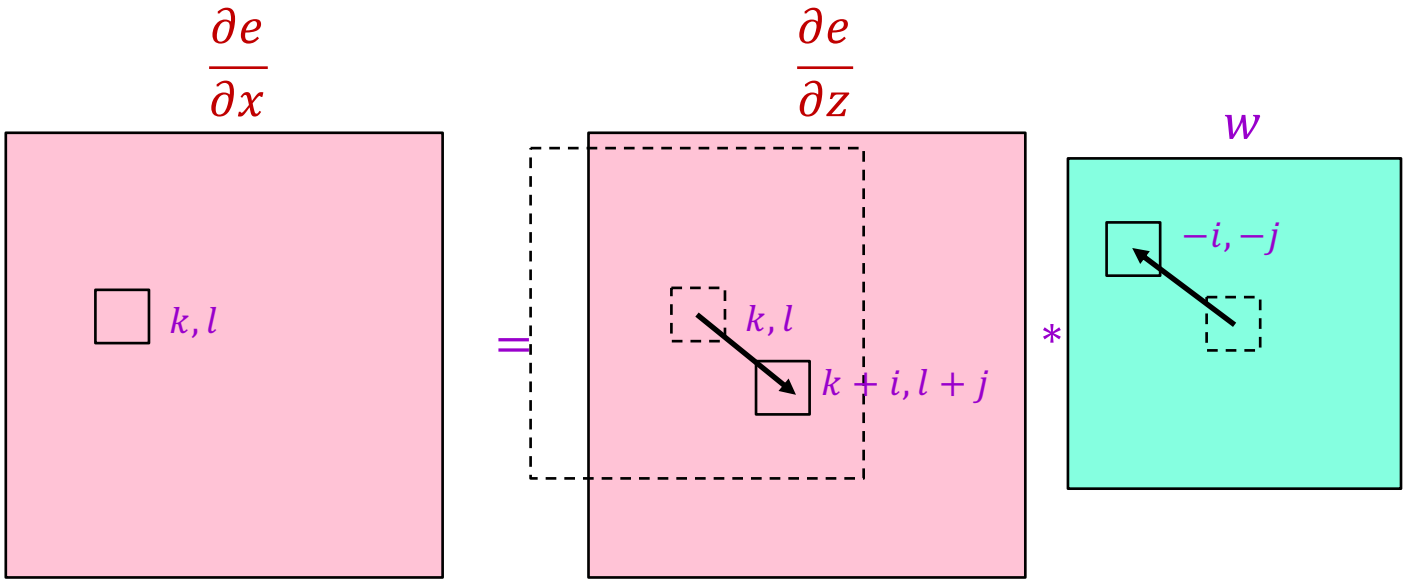


Backpropagation for convolutional layer

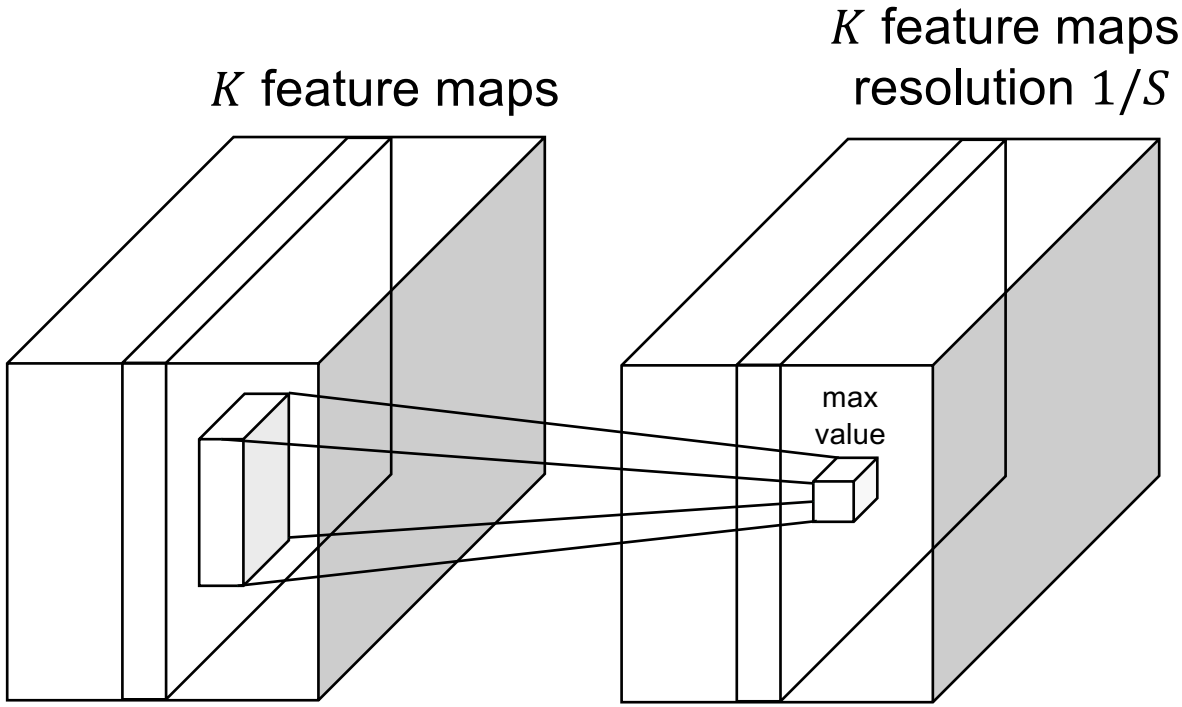
$$\frac{\partial e}{\partial x_{kl}} = \sum_{i,j} \frac{\partial e}{\partial z_{k+i,l+j}} w_{-i,-j}$$

Backpropagation for convolutional layer

$$\frac{\partial e}{\partial x_{kl}} = \sum_{i,j} \frac{\partial e}{\partial z_{k+i,l+j}} w_{-i,-j}$$



Max pooling layer

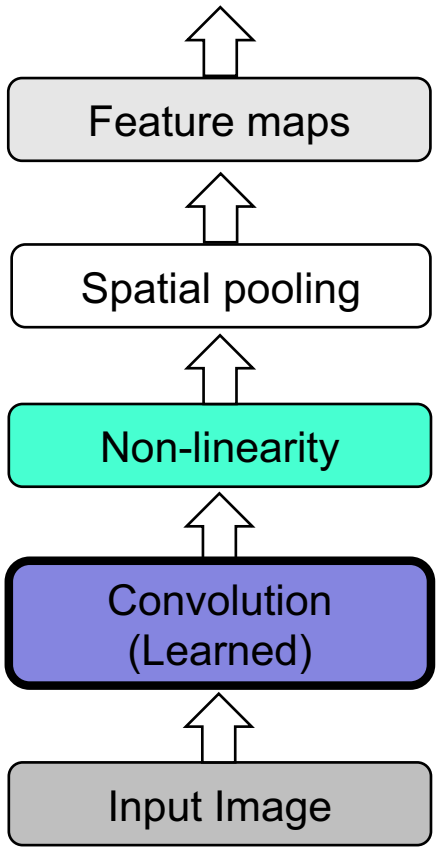


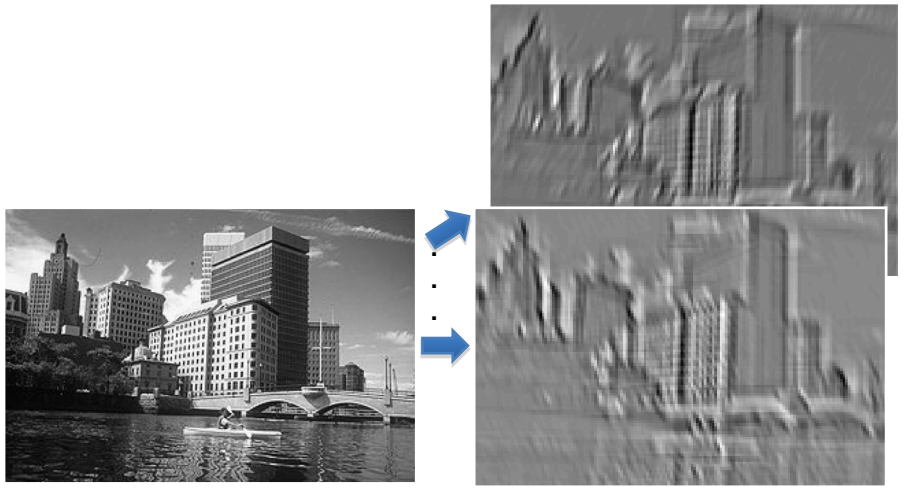
$F \times F$ pooling window, stride S

Usually: $F = 2$ or 3, $S = 2$

Backward pass: upstream gradient is passed back only to the unit with max value

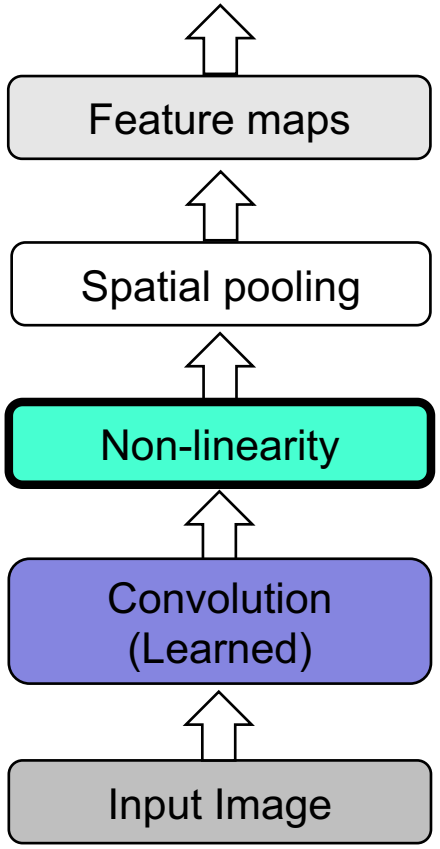
Summary: CNN pipeline





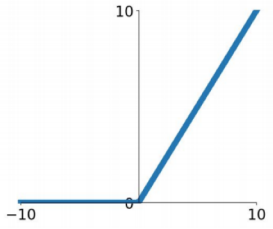
Source: R. Fergus, Y. LeCun

Summary: CNN pipeline



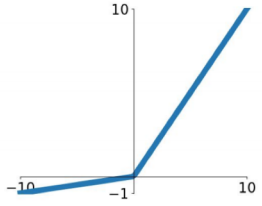
ReLU

$$\max(0, x)$$



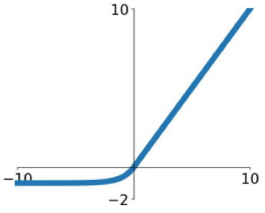
Leaky ReLU

$$\max(0.1x, x)$$



ELU

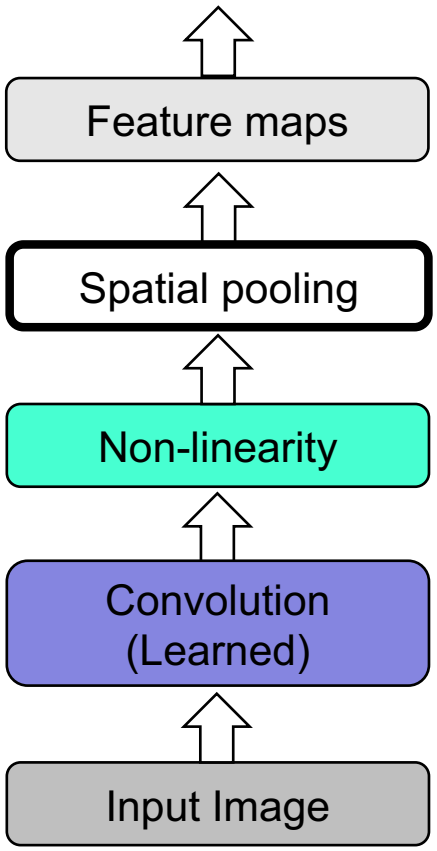
$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$

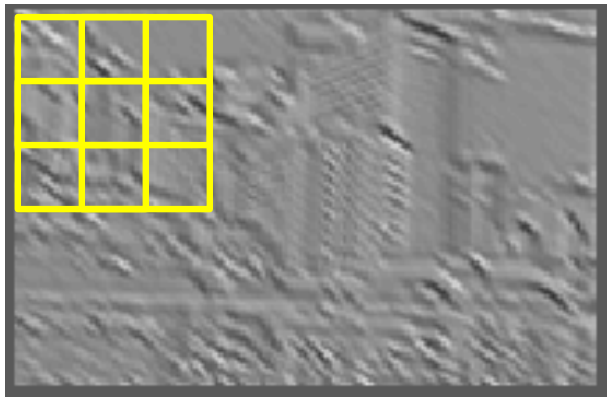


Source: R. Fergus, Y. LeCun

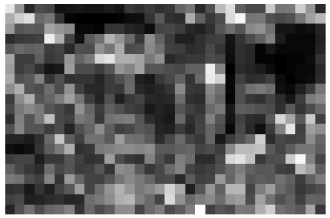
Source: Stanford 231n

Summary: CNN pipeline



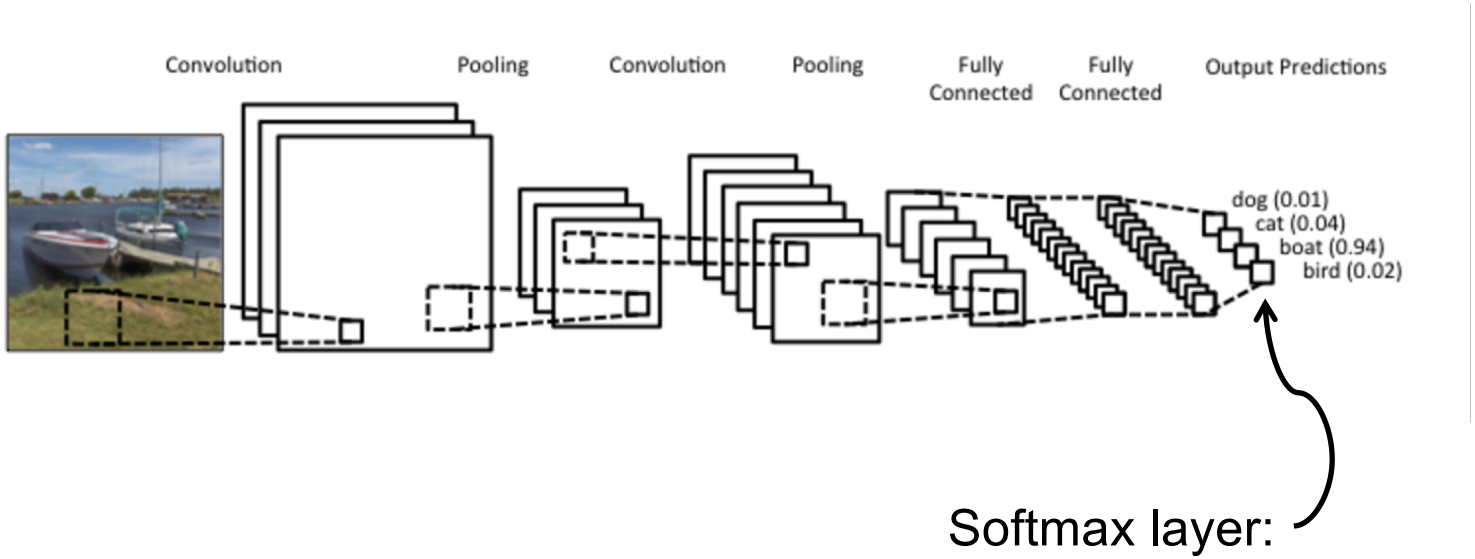


Max
(or Avg)



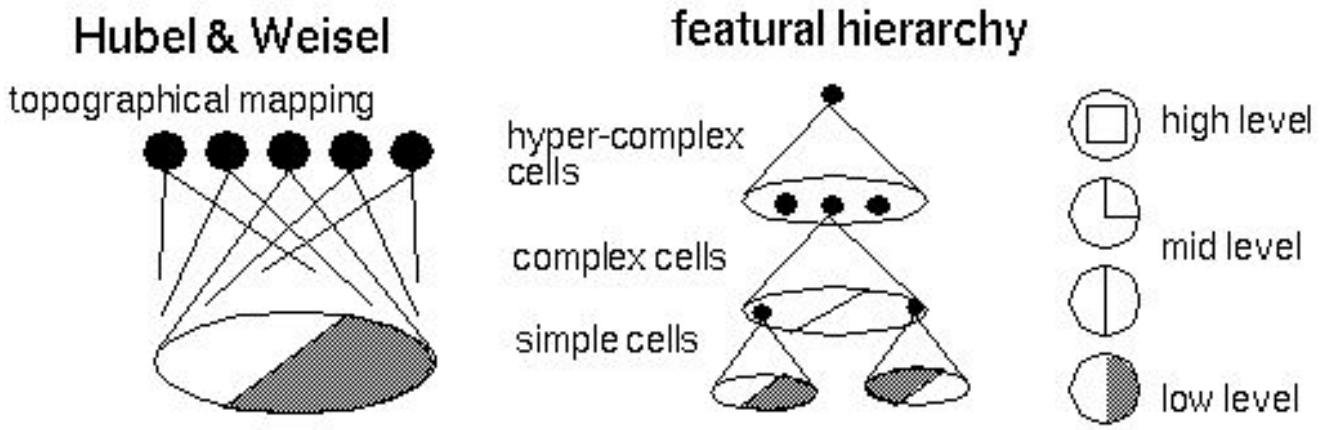
Source: R. Fergus, Y. LeCun

Summary: CNN pipeline



Backstory

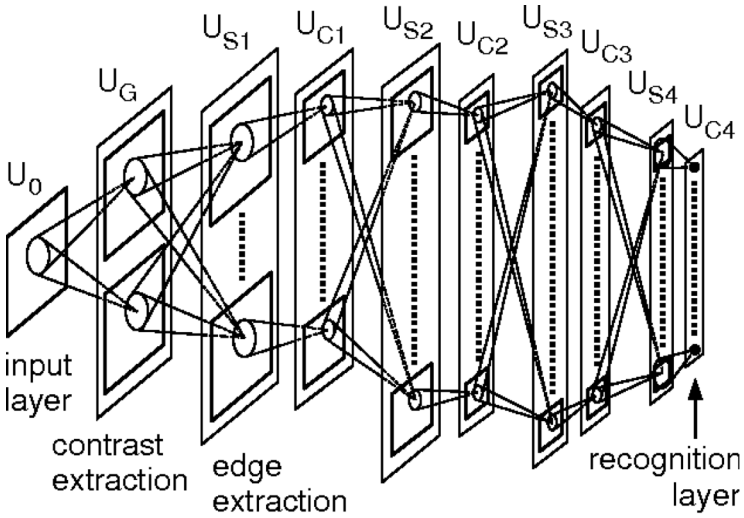
- Biological inspiration: D. Hubel and T. Wiesel (1959, 1962, Nobel Prize 1981)
  - Visual cortex consists of a hierarchy of *simple*, *complex*, and *hyper-complex* cells



Source

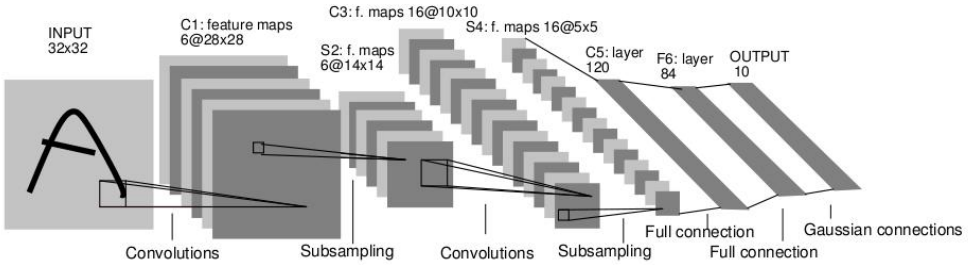
Backstory

Neocognitron



K. Fukishima. Neocognitron: A Self-organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position. Biological Cybernetics, 1980

LeNet-5



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

Outline

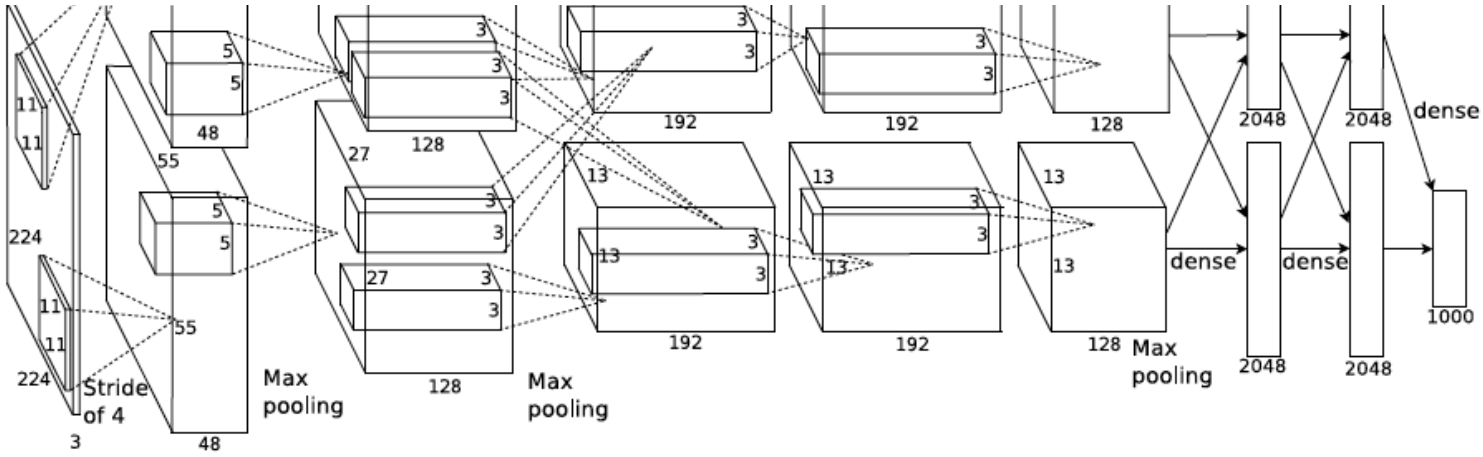
- Building blocks
  - Convolutional layers and backprop rules
  - Pooling layers and nonlinearities
- Architectures:
  - 1st generation (2012-2013): AlexNet
  - 2nd generation (2014): VGGNet, GoogLeNet
  - 3rd generation (2015): ResNet
  - 4th generation (2016): ResNeXt, DenseNet

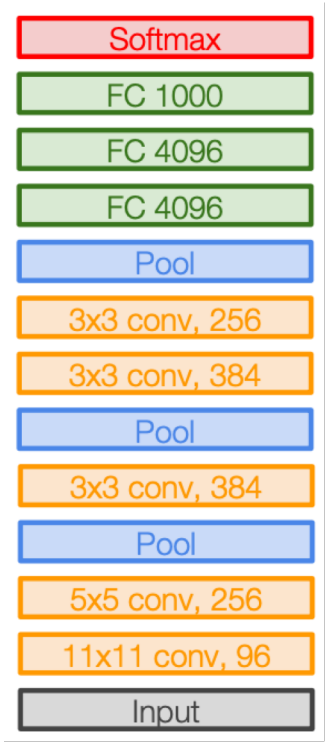
ImageNet Challenge

- ~14 million labeled images, 20k classes
- Images gathered from Internet
- Human labels via Amazon MTurk
- ImageNet Large-Scale Visual Recognition Challenge (ILSVRC):
  1.2 million training images, 1000 classes

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

AlexNet: ILSVRC 2012 winner





- Changes from LeNet-5:
  - Max pooling, ReLU nonlinearity
  - 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
  - Dropout regularization

A. Krizhevsky, I. Sutskever, and G. Hinton, ImageNet Classification with Deep Convolutional Neural Networks, NIPS 2012

Clarifai: ILSVRC 2013 winner

- Refinement of AlexNet

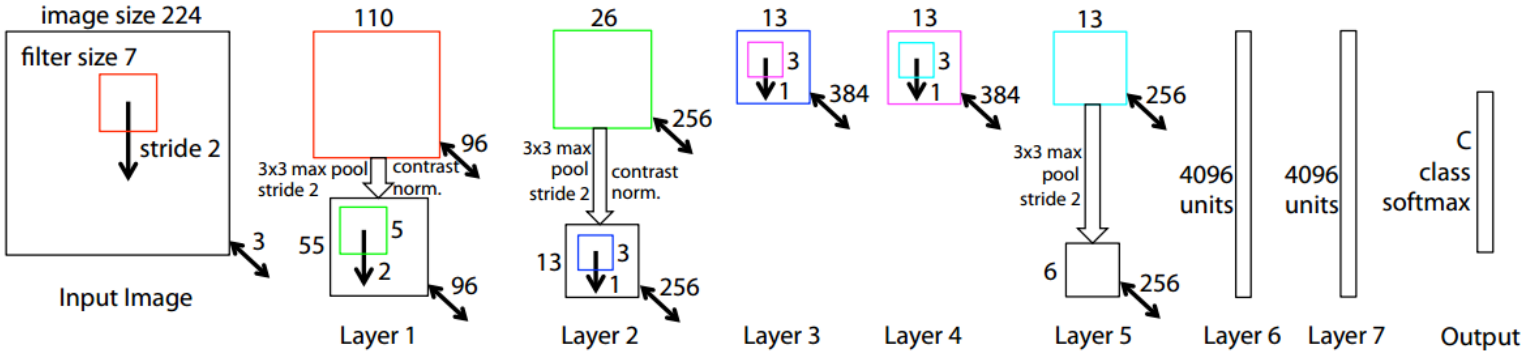


Figure 3. Architecture of our 8 layer convnet model. A 224 by 224 crop of an image (with 3 color planes) is presented as the input. This is convolved with 96 different 1st layer filters (red), each of size 7 by 7, using a stride of 2 in both x and y. The resulting feature maps are then: (i) passed through 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

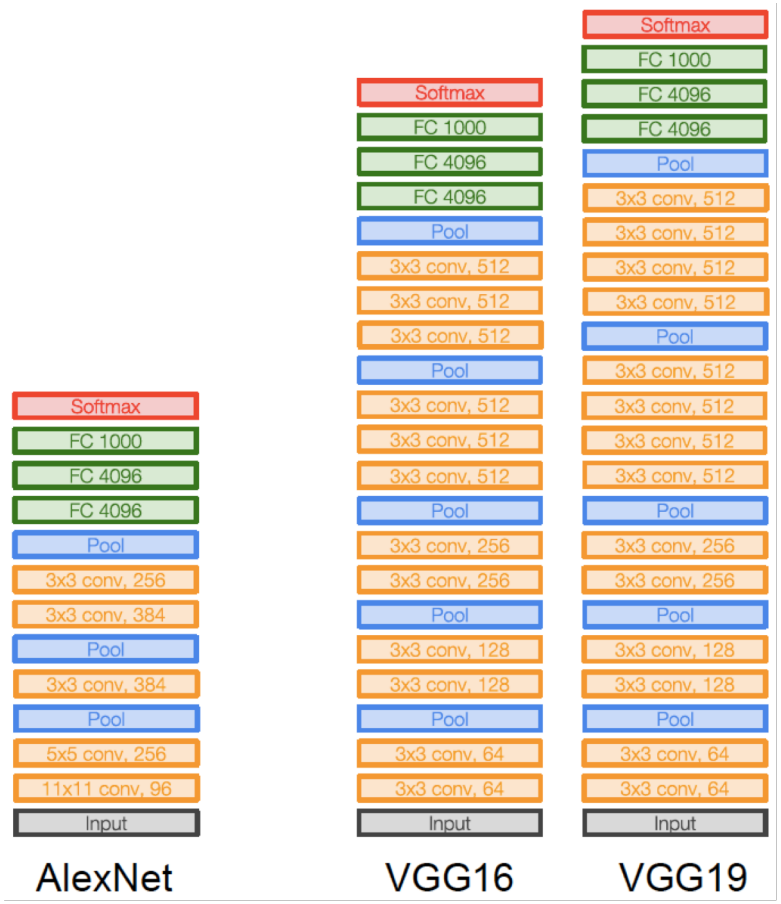


Image source

K. Simonyan and A. Zisserman, Very Deep Convolutional Networks for Large-Scale Image Recognition, ICLR 2015

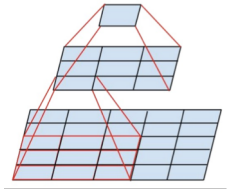
VGGNet: ILSVRC 2014 2nd place

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 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 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 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

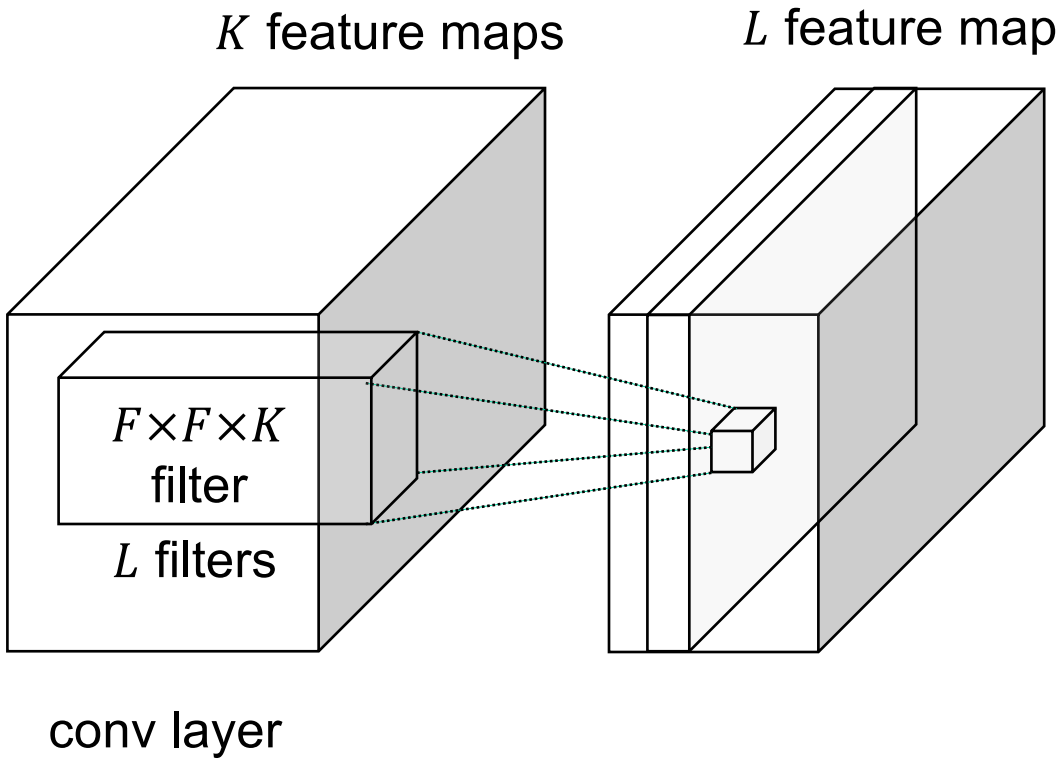
- Sequence of deeper networks trained progressively
- Large receptive fields replaced by successive layers of 3x3 convolutions (with ReLU in between)



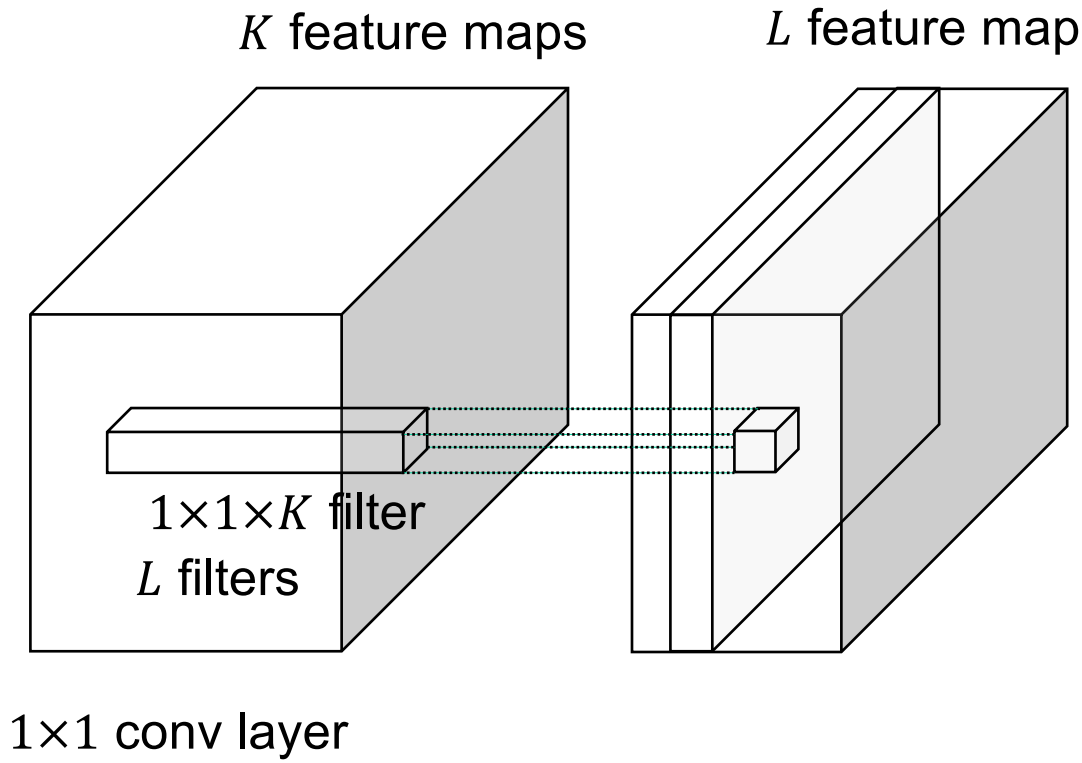
- One 7x7 conv layer with K feature maps needs $49K^2$ weights, three 3x3 conv layers need only $27K^2$ weights
- Experimented with 1x1 convolutions

K. Simonyan and A. Zisserman, Very Deep Convolutional Networks for Large-Scale Image Recognition, ICLR 2015

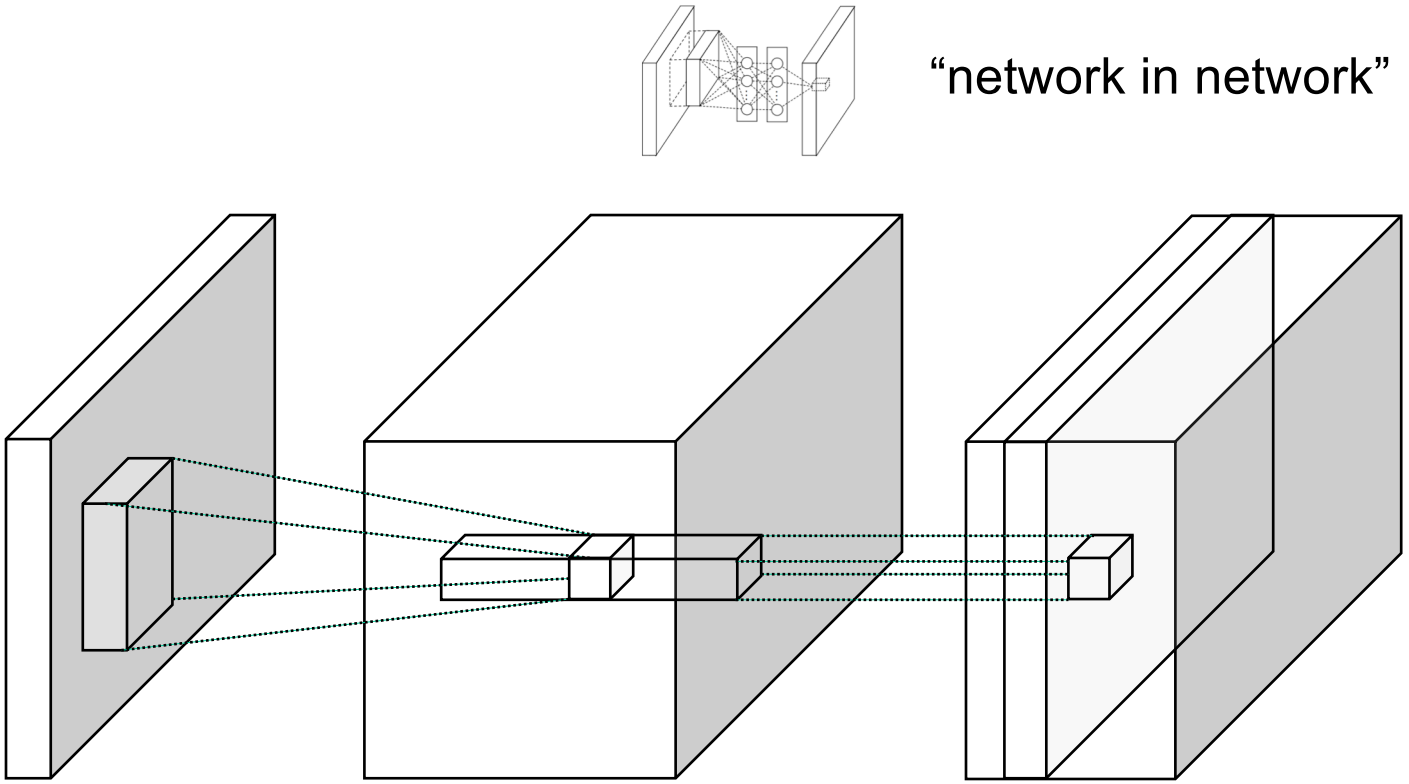
1x1 convolutions



1x1 convolutions



1x1 convolutions



M. Lin, Q. Chen, and S. Yan, Network in network, ICLR 2014

GoogLeNet: ILSVRC 2014 winner

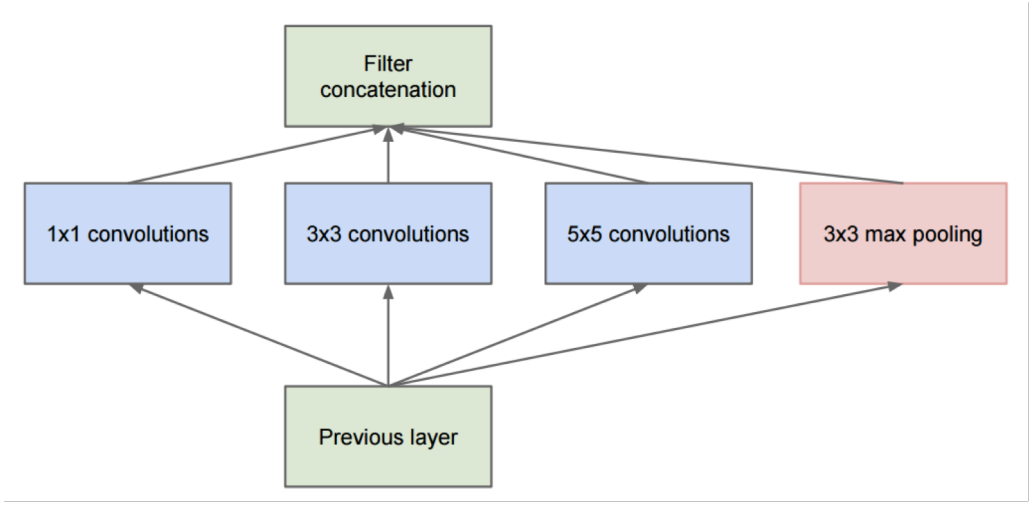
- The Inception Module

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

C. Szegedy et al., Going deeper with convolutions, CVPR 2015

GoogLeNet

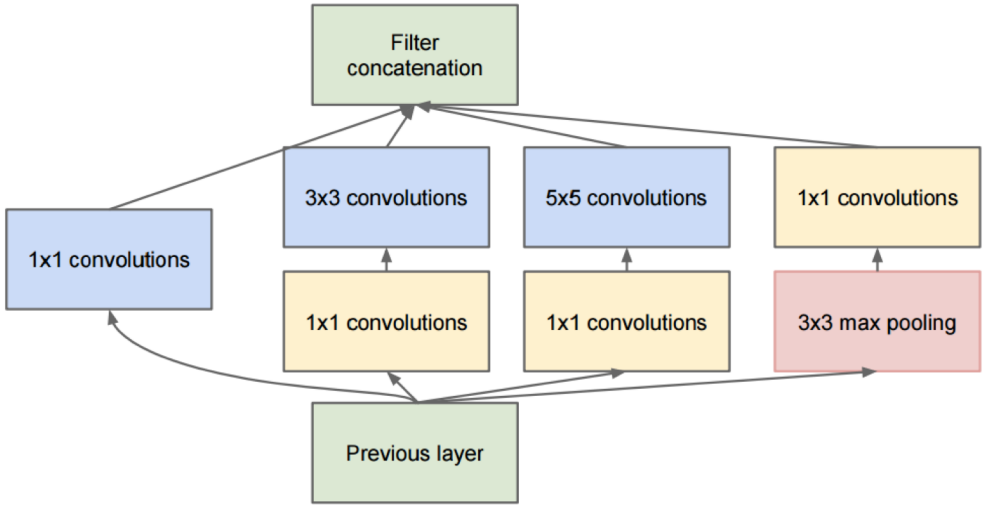
- The Inception Module
  - Parallel paths with different receptive field sizes and operations are meant to capture sparse patterns of correlations in the stack of feature maps



C. Szegedy et al., Going deeper with convolutions, CVPR 2015

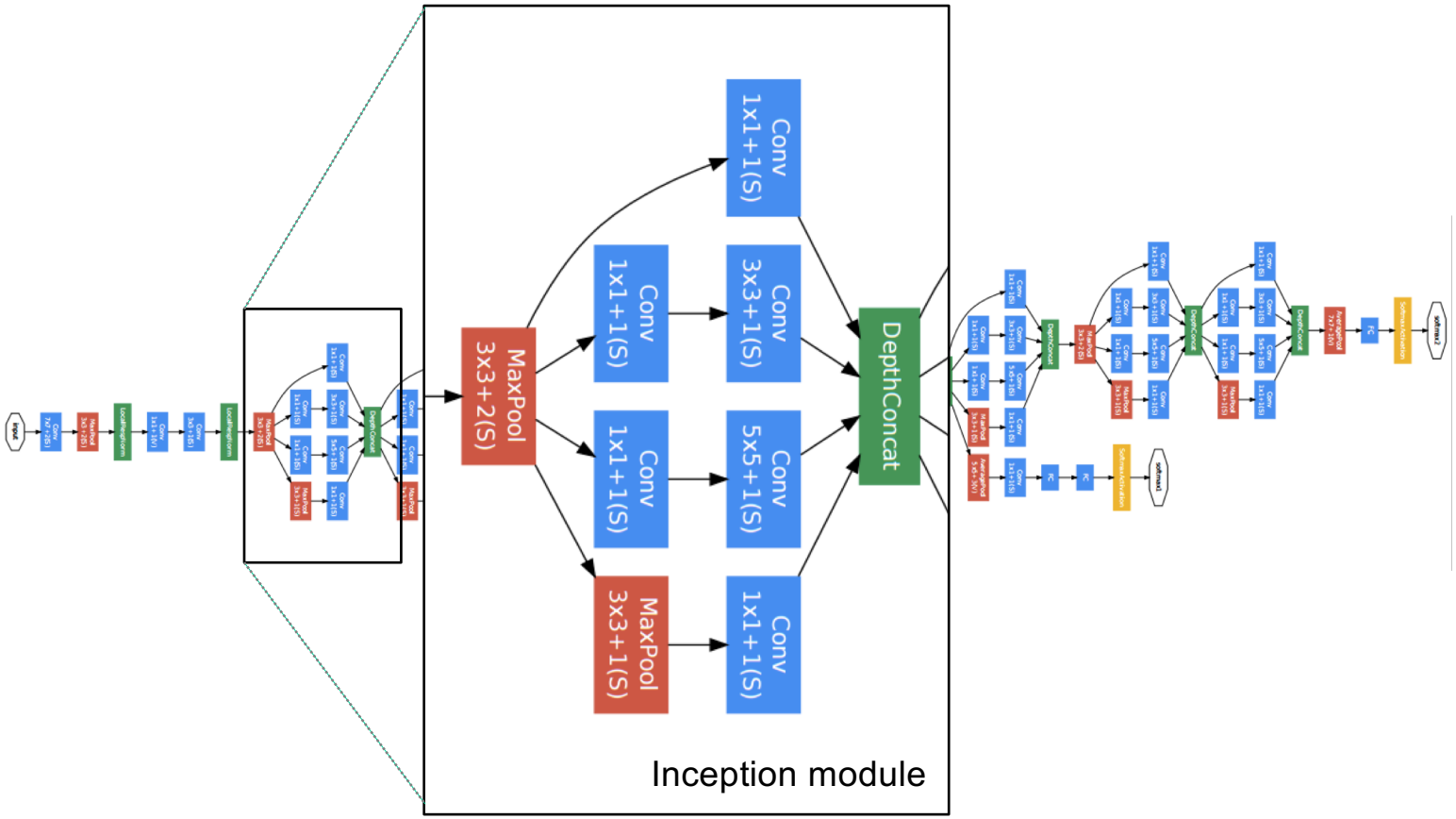
GoogLeNet

- The 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 1x1 convolutions for dimensionality reduction before expensive convolutions



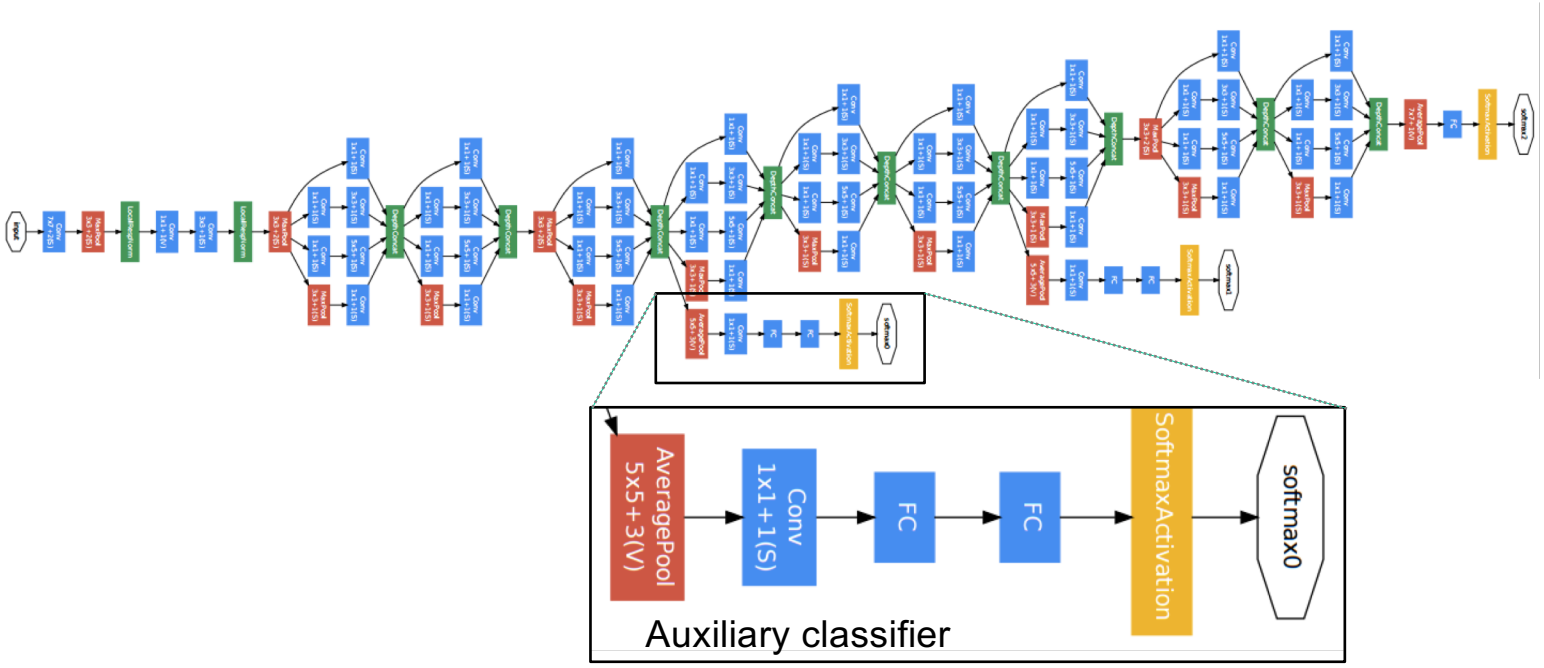
C. Szegedy et al., Going deeper with convolutions, CVPR 2015

GoogLeNet



C. Szegedy et al., Going deeper with convolutions, CVPR 2015

GoogLeNet



C. Szegedy et al., Going deeper with convolutions, CVPR 2015

GoogLeNet

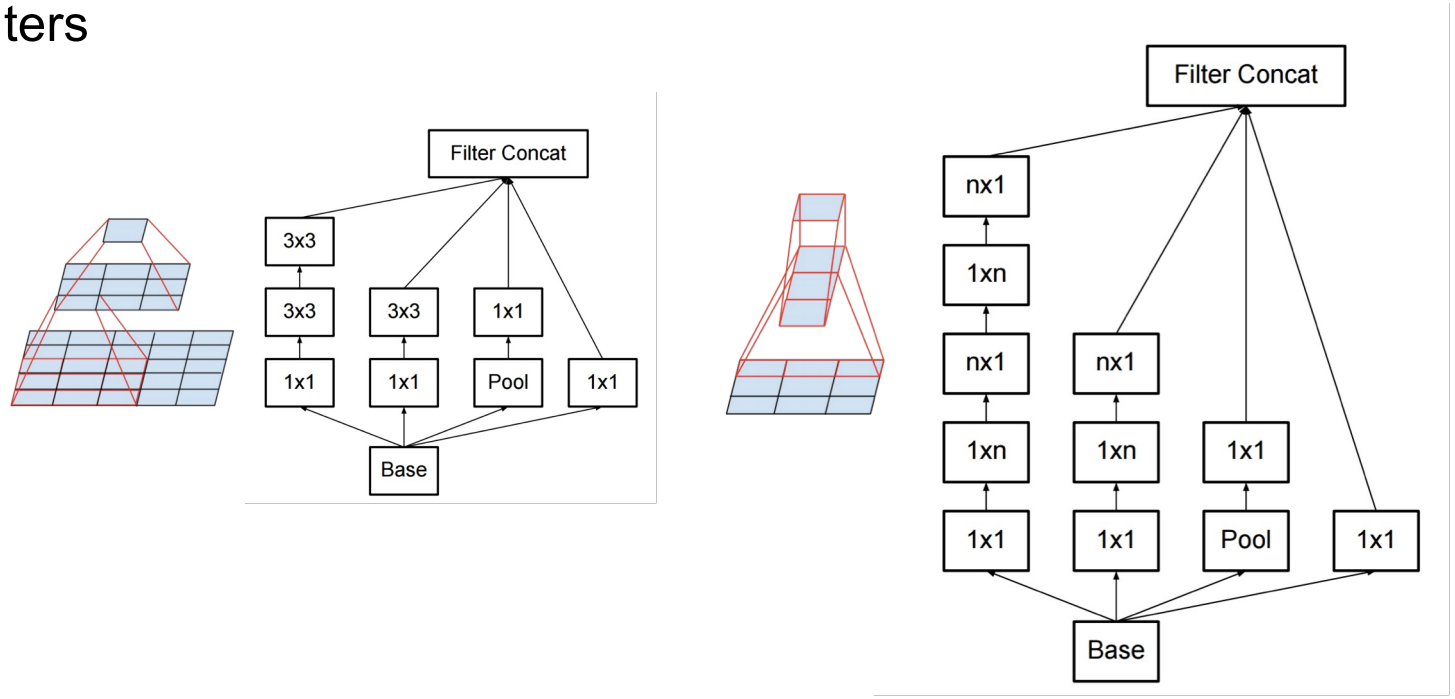
An alternative view:

type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M
inception (3b)		28×28×480	2	128	128	192	32	96	64	380K	304M
max pool	3×3/2	14×14×480	0								
inception (4a)		14×14×512	2	192	96	208	16	48	64	364K	73M
inception (4b)		14×14×512	2	160	112	224	24	64	64	437K	88M
inception (4c)		14×14×512	2	128	128	256	24	64	64	463K	100M
inception (4d)		14×14×528	2	112	144	288	32	64	64	580K	119M
inception (4e)		14×14×832	2	256	160	320	32	128	128	840K	170M
max pool	3×3/2	7×7×832	0								
inception (5a)		7×7×832	2	256	160	320	32	128	128	1072K	54M
inception (5b)		7×7×1024	2	384	192	384	48	128	128	1388K	71M
avg pool	7×7/1	1×1×1024	0								
dropout (40%)		1×1×1024	0								
linear		1×1×1000	1							1000K	1M
softmax		1×1×1000	0								

C. Szegedy et al., Going deeper with convolutions, CVPR 2015

Inception v2, v3

- Improve training with batch normalization, reducing importance of auxiliary classifiers
- More variants of inception modules with aggressive factorization of filters



C. Szegedy et al., Rethinking the inception architecture for computer vision, CVPR 2016

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/

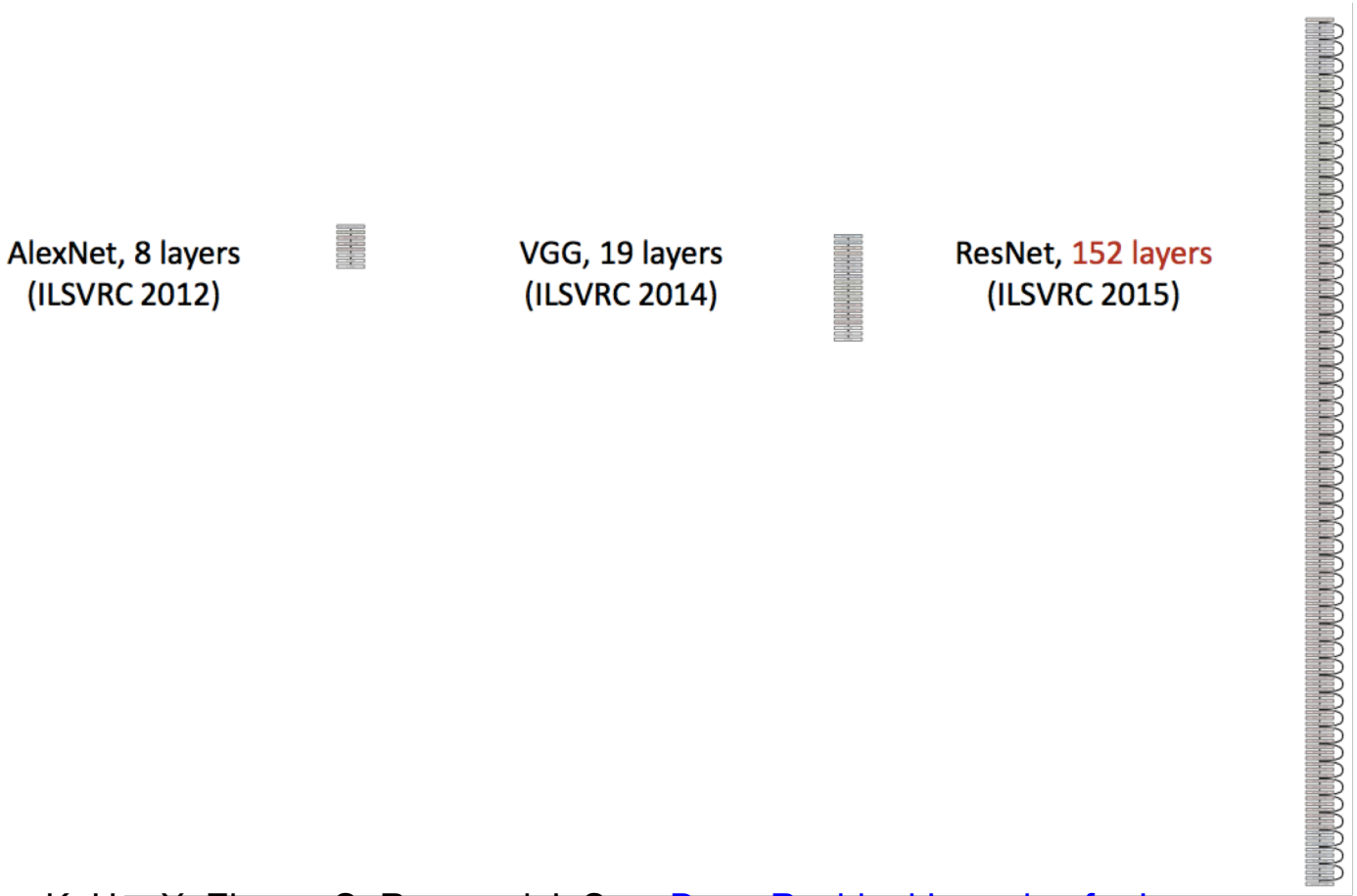
ResNet: ILSVRC 2015 winner

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



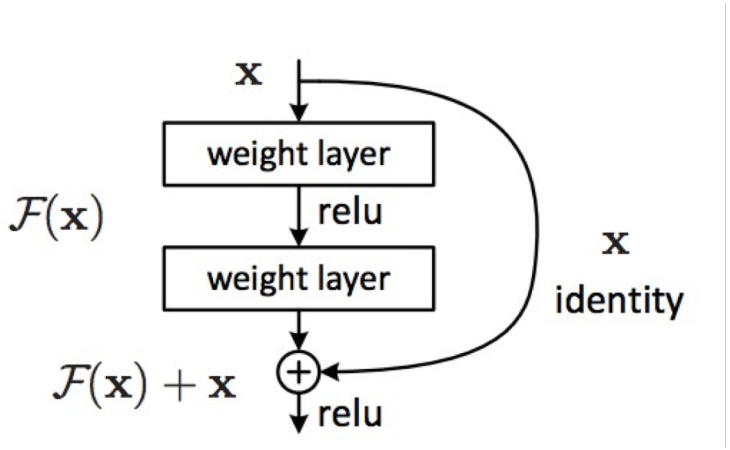
K. He, X. Zhang, S. Ren, and J. Sun, Deep Residual Learning for Image Recognition, CVPR 2016 (Best Paper)


I WAS WINNING IMAGENET
UNTIL A DEEPER MODEL CAME ALONG
imgflip.com

Source (?)

ResNet

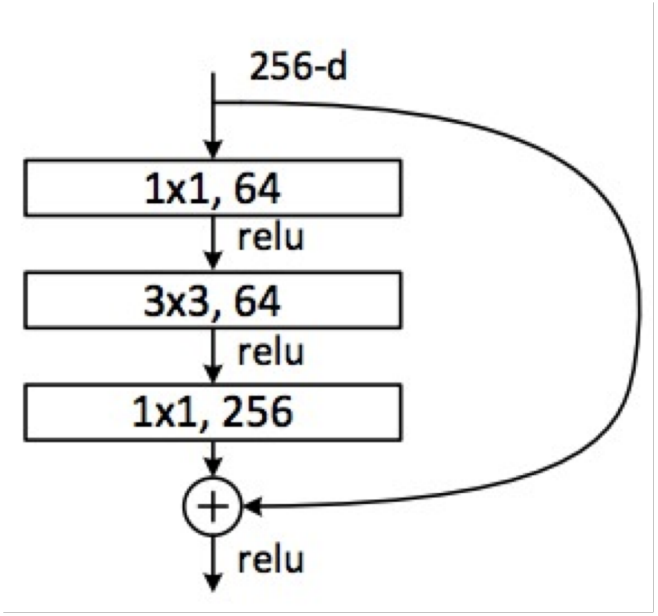
- The residual module
  - Introduce *skip* or *shortcut* connections (existing before in various forms in literature)
  - Make it easy for network layers to represent the identity mapping



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 3x3 convolutions with 256 feature maps at input and output:
  256 x 256 x 3 x 3 ~ 600K operations
- Using 1x1 convolutions to reduce 256 to 64 feature maps, followed by 3x3 convolutions, followed by 1x1 convolutions to expand back to 256 maps:
  256 x 64 x 1 x 1 ~ 16K
  64 x 64 x 3 x 3 ~ 36K
  64 x 256 x 1 x 1 ~ 16K
  Total: ~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				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix} \times 4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times 8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix} \times 6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times 23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times 36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times 3$
	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)

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%	

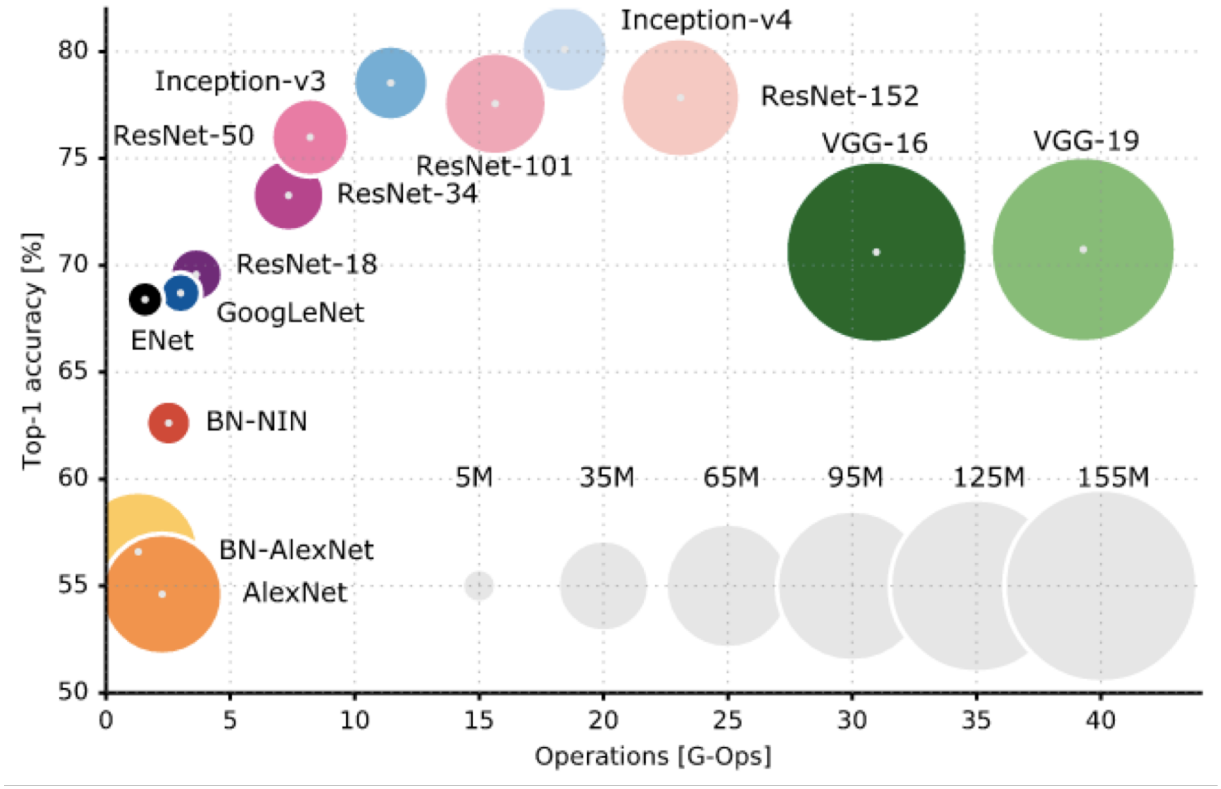
*Officially, there was no longer a classification competition and ResNet-based systems won in localization and detection

Techniques necessary for state-of-the-art performance

- Training tricks and details: initialization, regularization, normalization
- Training data augmentation
- Averaging classifier outputs over multiple crops/flips
- Ensembles of networks

(to be covered next time)

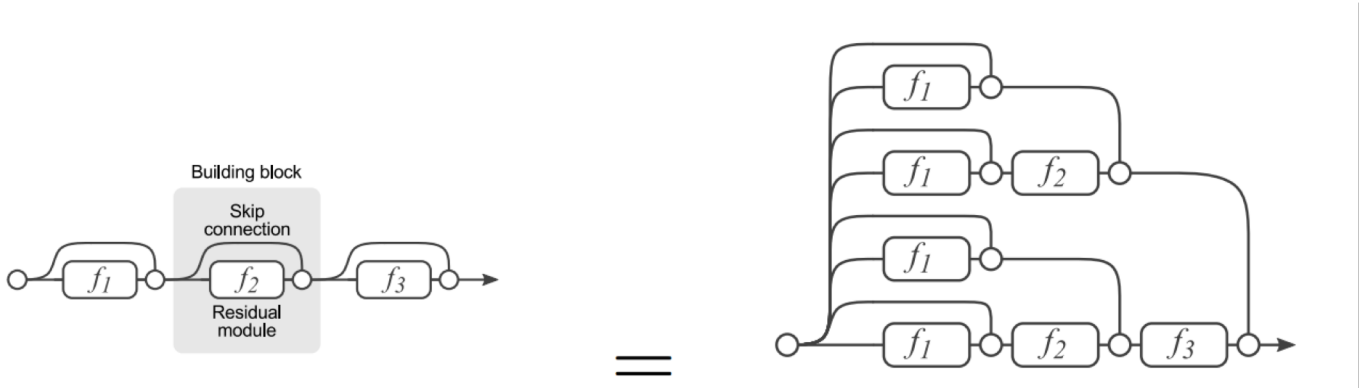
Comparing architectures



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

Beyond ResNets: Why do they work?

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



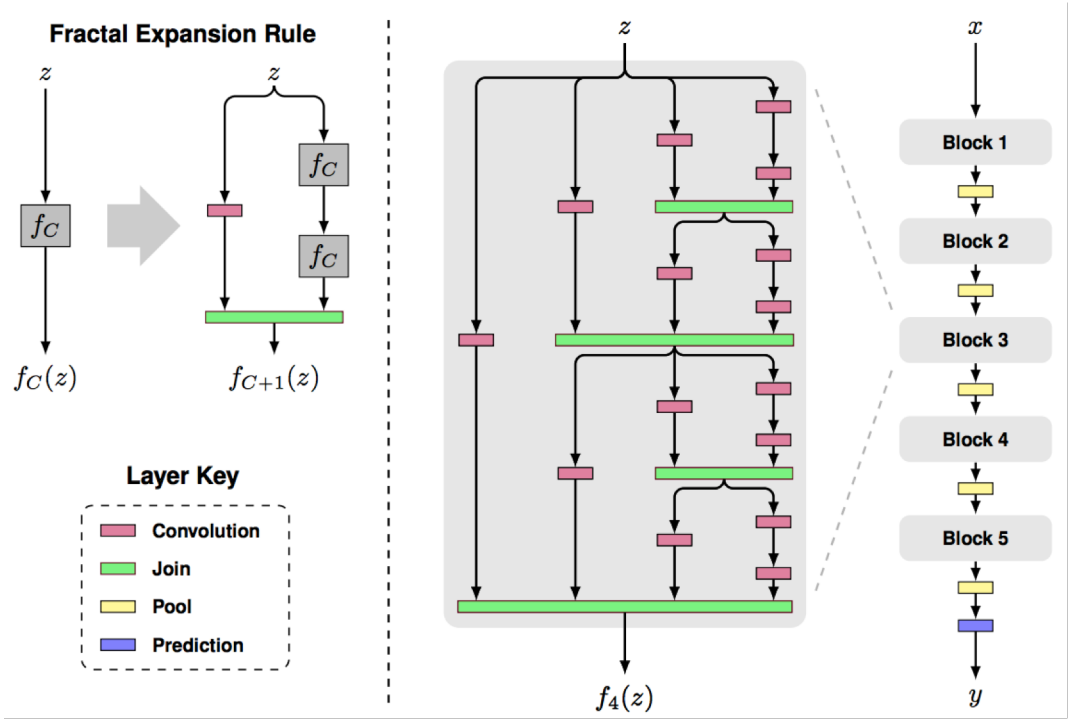
(a) Conventional 3-block residual network (b) Unraveled view of (a)

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

Beyond ResNet: FractalNet

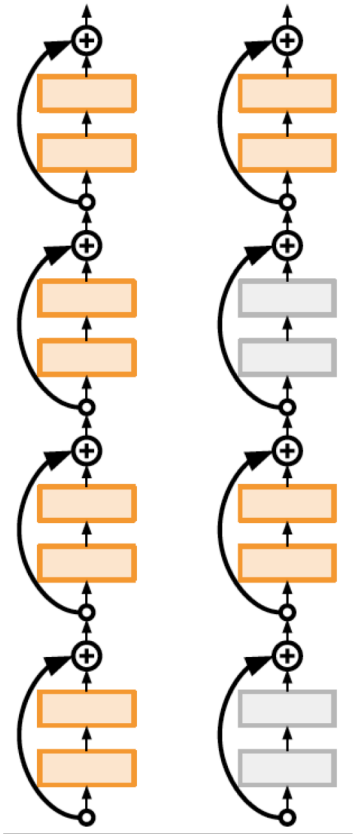
- Claim: key to good performance is not having skip connections (residuals), but having both shallow and deep paths



S. Larsson, M. Maire and G. Shakhnarovich, FractalNet: Ultra-Deep Neural Networks without Residuals, ICLR 2017

Beyond ResNet: Stochastic depth

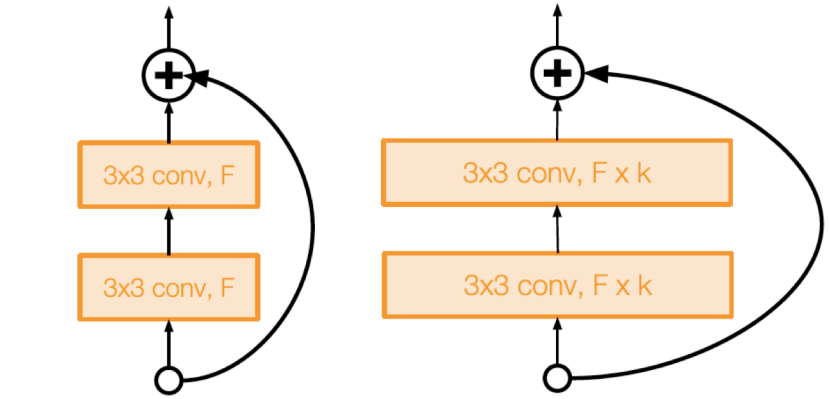
- At training time, in each mini-batch, randomly drop a subset of layers (bypass with identity function). At test time, use full network
  - Forces the network to learn better, speeds up convergence, improves accuracy



G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Weinberger, Deep Networks with Stochastic Depth, ECCV 2016

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

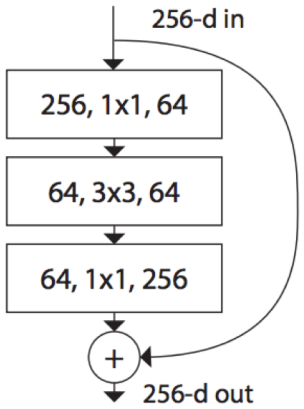
Image source

S. Zagoryuko and N. Komodakis, Wide Residual Networks, BMVC 2016

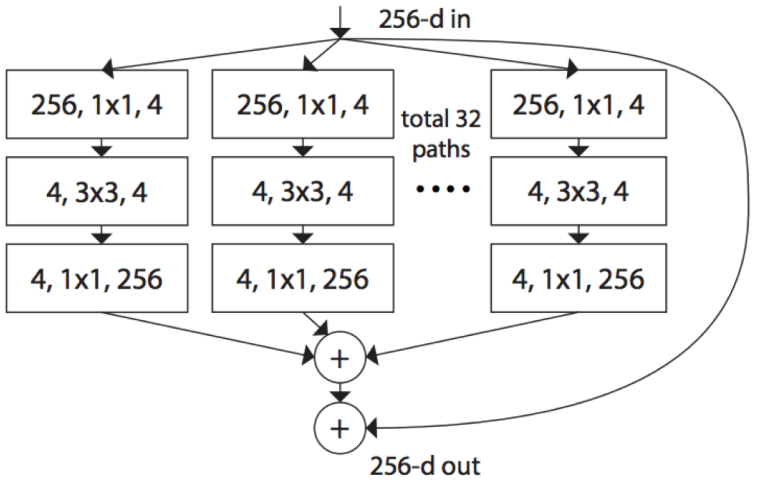
Beyond ResNet: ResNeXt

- 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

ResNet



ResNeXt, roughly same complexity



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

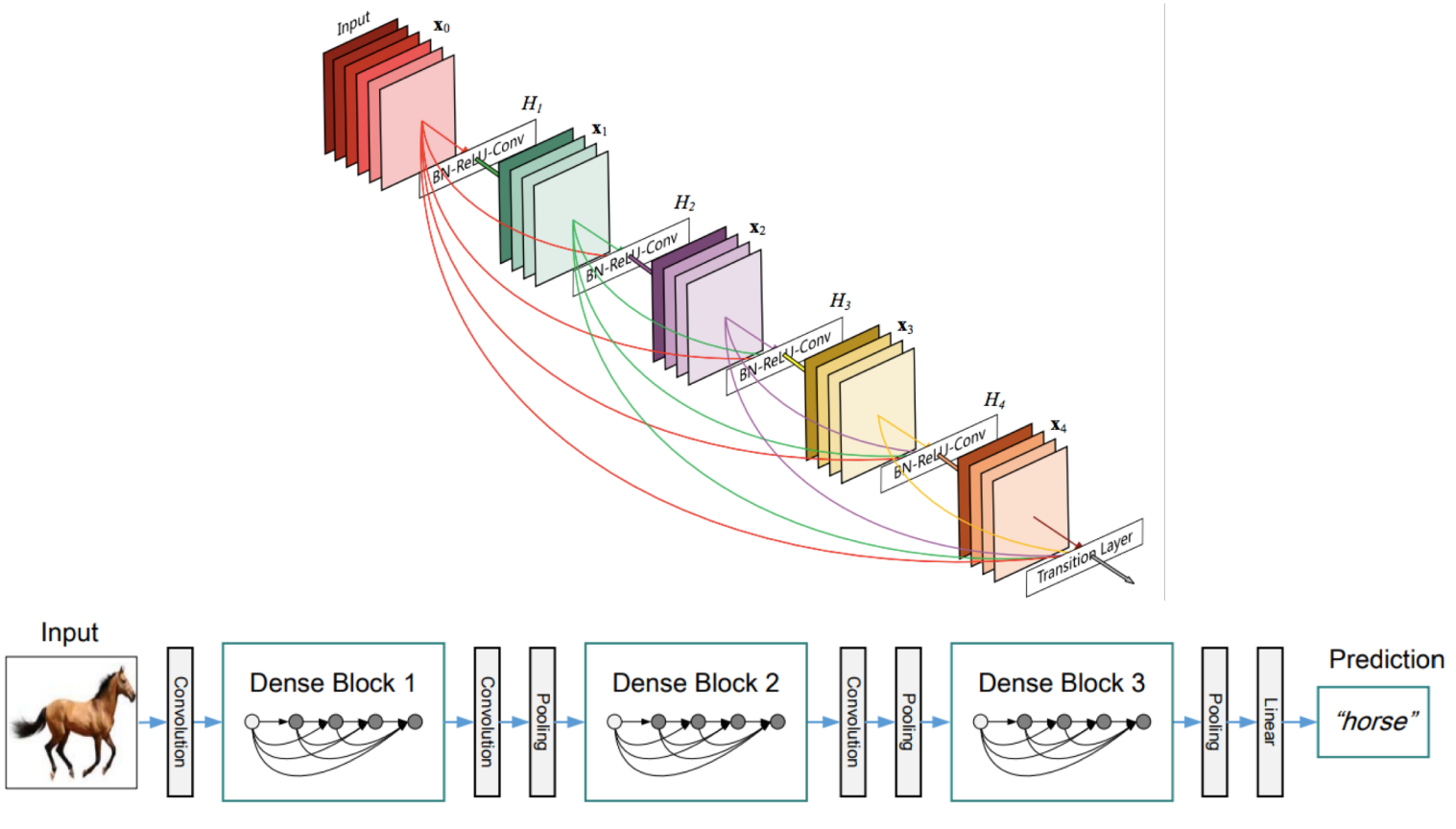
Beyond ResNet: ResNeXt

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



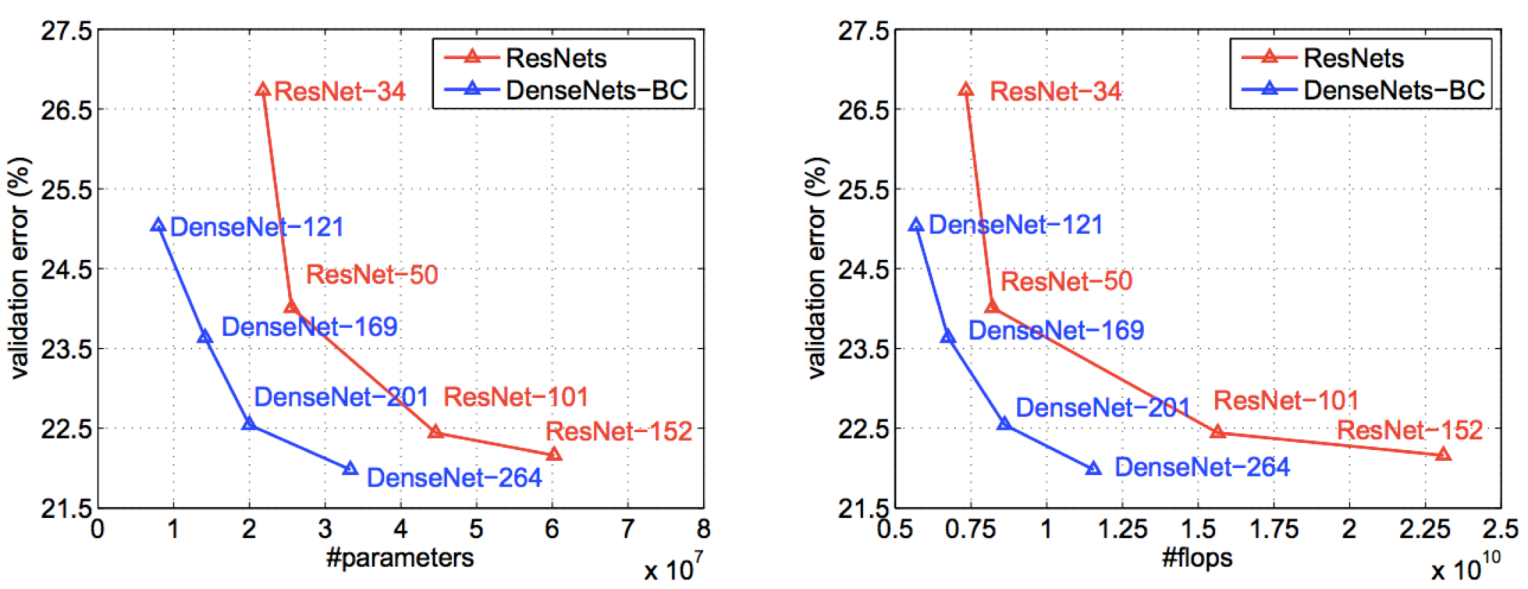
G. Huang, Z. Liu, and L. van der Maaten, Densely Connected Convolutional Networks, CVPR 2017 (Best Paper Award)

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	[1 × 1 conv; 3 × 3 conv] × 6	[1 × 1 conv; 3 × 3 conv] × 6	[1 × 1 conv; 3 × 3 conv] × 6	[1 × 1 conv; 3 × 3 conv] × 6
Transition Layer (1)	56 × 56	1 × 1 conv			
	28 × 28	2 × 2 average pool, stride 2			
Dense Block (2)	28 × 28	[1 × 1 conv; 3 × 3 conv] × 12	[1 × 1 conv; 3 × 3 conv] × 12	[1 × 1 conv; 3 × 3 conv] × 12	[1 × 1 conv; 3 × 3 conv] × 12
Transition Layer (2)	28 × 28	1 × 1 conv			
	14 × 14	2 × 2 average pool, stride 2			
Dense Block (3)	14 × 14	[1 × 1 conv; 3 × 3 conv] × 24	[1 × 1 conv; 3 × 3 conv] × 32	[1 × 1 conv; 3 × 3 conv] × 48	[1 × 1 conv; 3 × 3 conv] × 64
Transition Layer (3)	14 × 14	1 × 1 conv			
	7 × 7	2 × 2 average pool, stride 2			
Dense Block (4)	7 × 7	[1 × 1 conv; 3 × 3 conv] × 16	[1 × 1 conv; 3 × 3 conv] × 32	[1 × 1 conv; 3 × 3 conv] × 32	[1 × 1 conv; 3 × 3 conv] × 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



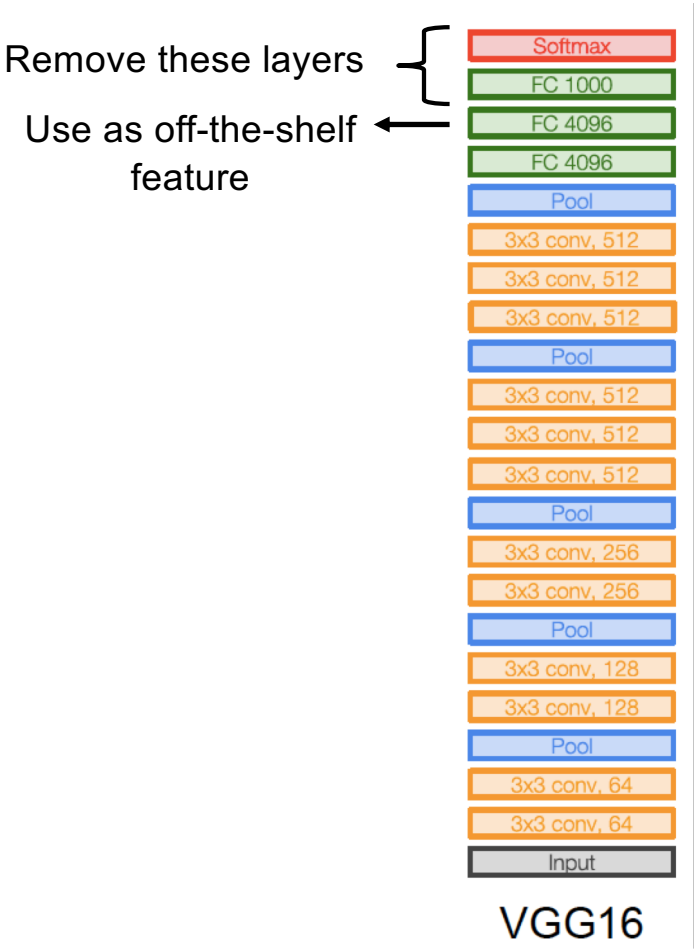
ImageNet validation error vs. number of parameters and test-time operations

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

Design principles

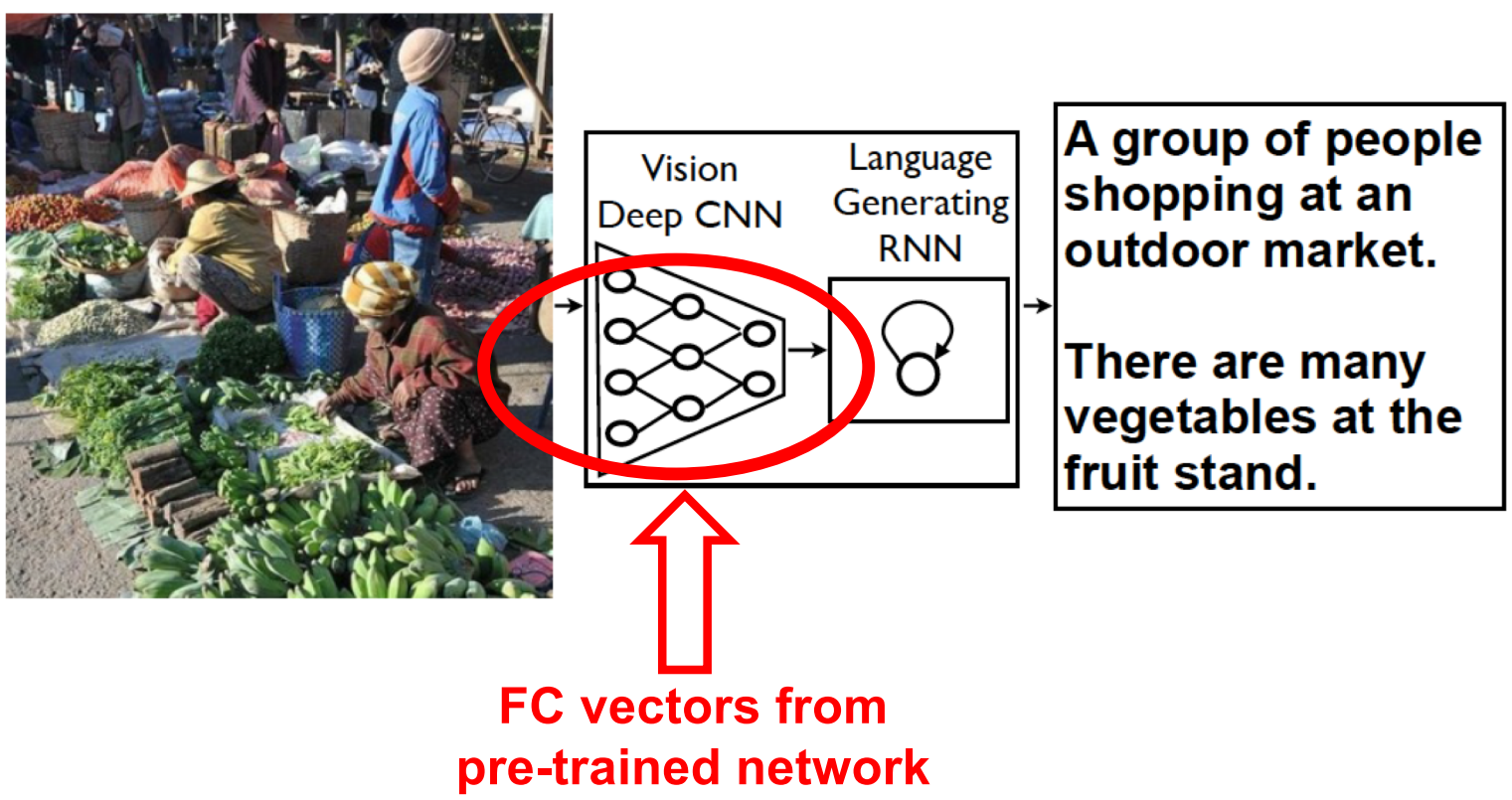
- Make networks parameter-efficient
  - Reduce filter sizes, factorize filters
  - Use 1x1 convolutions to reduce number of feature maps before more expensive operations
  - Minimize reliance on FC layers
- Reduce spatial resolution gradually, within each level of resolution replicate a given “block” multiple times
- Use skip connections and/or create multiple redundant paths through the network
- Play around with depth vs. width vs. “cardinality”

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



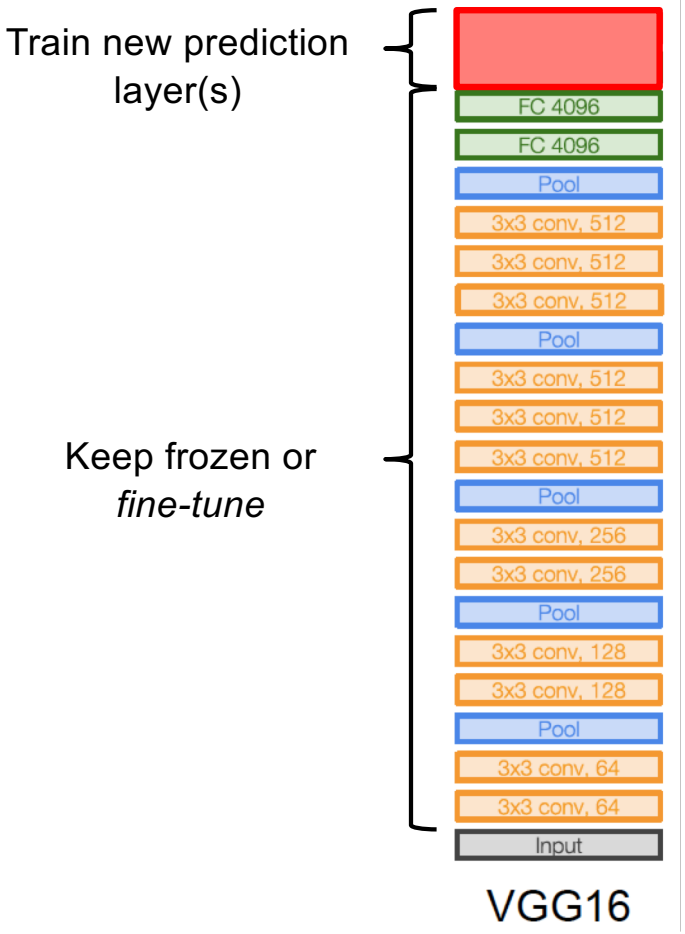
- Strategy 1: Use as feature extractor

Example: CNNs for image captioning



O. Vinyals, A. Toshev, S. Bengio, and D. Erhan. Show and tell: A neural image captio n generator. CVPR 2015

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



- Strategy 2: Transfer learning

References (incomplete)

- https://culurciello.github.io/tech/2016/06/04/nets.html
- Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE 86(11): 2278–2324, 1998.
- A. Krizhevsky, I. Sutskever, and G. Hinton, ImageNet Classification with Deep Convolutional Neural Networks, NIPS 2012
- M. Zeiler and R. Fergus, Visualizing and Understanding Convolutional Networks, ECCV 2014
- K. Simonyan and A. Zisserman, Very Deep Convolutional Networks for Large-Scale Image Recognition, ICLR 2015
- M. Lin, Q. Chen, and S. Yan, Network in network, ICLR 2014
- C. Szegedy et al., Going deeper with convolutions, CVPR 2015
- C. Szegedy et al., Rethinking the inception architecture for computer vision, CVPR 2016
- K. He, X. Zhang, S. Ren, and J. Sun, Deep Residual Learning for Image Recognition, CVPR 2016
- G. Huang, Z. Liu, and L. van der Maaten, Densely Connected Convolutional Networks, CVPR 2017