Introduction to Convolutional Neural Networks and Computer Vision Applications

Goran Dubajić

11/04/2018

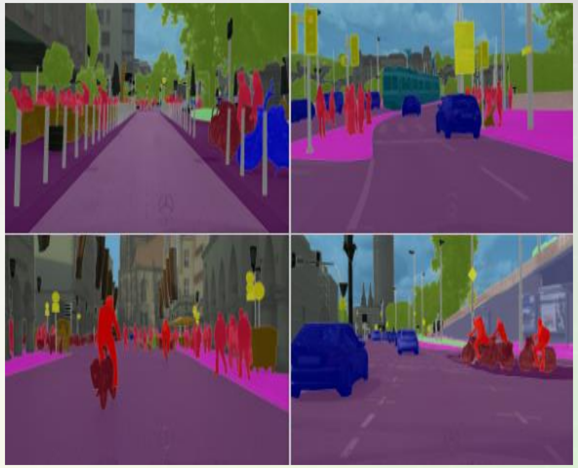
Microsoft
Development Center Serbia

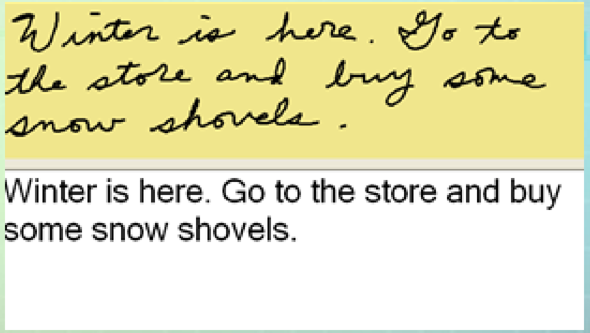
Outline

- Motivation
- Convolution and pooling layer
- Basic elements of ConvNet architecture
- History
- Notable architectures for image classification

Computer vision

- Spatial correlation
- Invariance to translation, rotation, lighting…
- Hierarchical structure





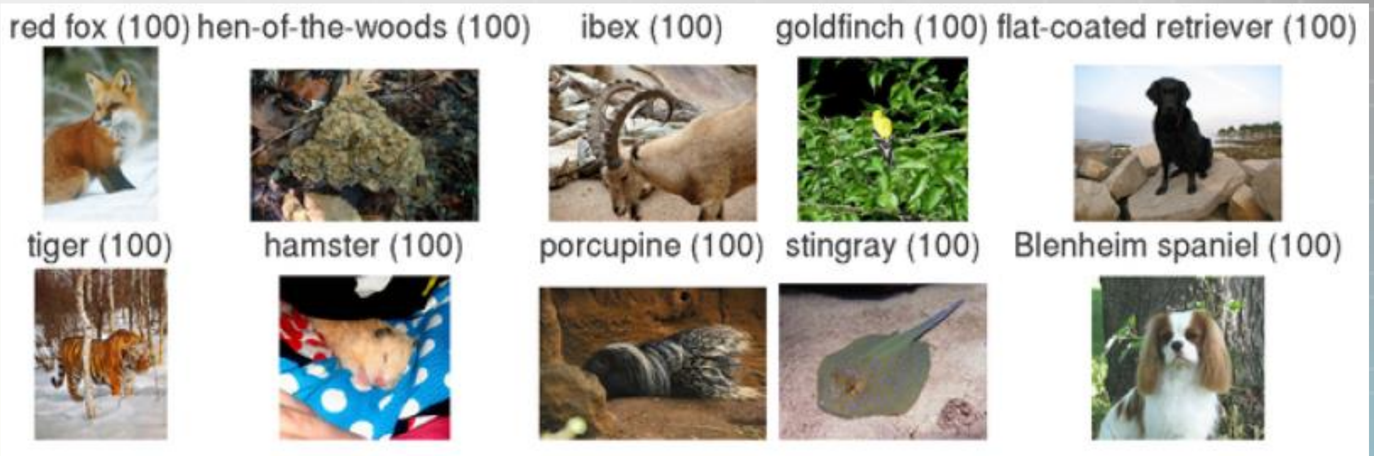
Microsoft Development Center Serbia

Convolutional neural networks (ConvNets)

- Extension of multilayer perceptron
- Feed-forward architecture
- Building blocks suitable for CV problems
- Biologically inspired
- Led to breakthroughs in many CV problems in recent years

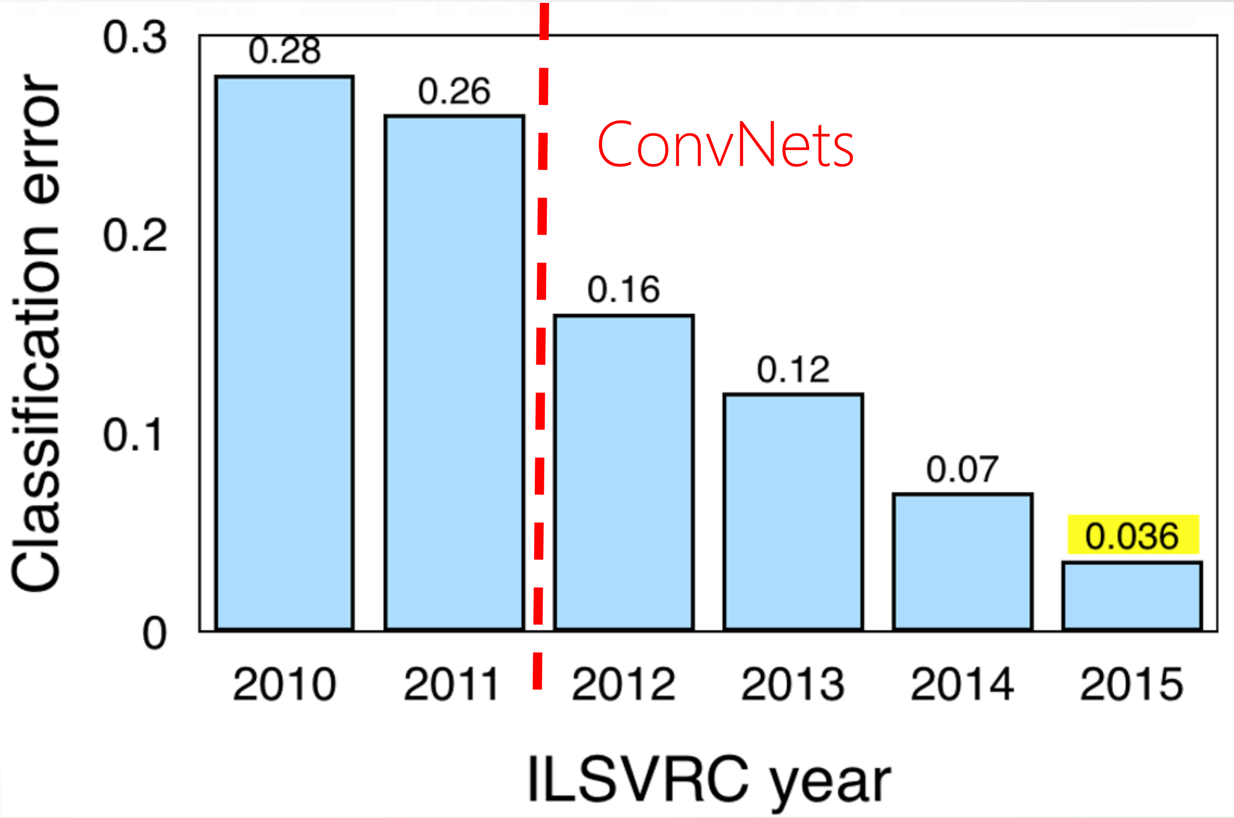
ImageNet 1K classification challenge (2010-2014)

- 1000 classes
- 1.28 million training images
- 50.000 test images



Russakovsky et al. ImageNet Large Scale Visual Recognition Challenge, IJCV, 2015
http://www.image-net.org/challenges/LSVRC/

ImageNet 1K classification challenge (2010-2014)



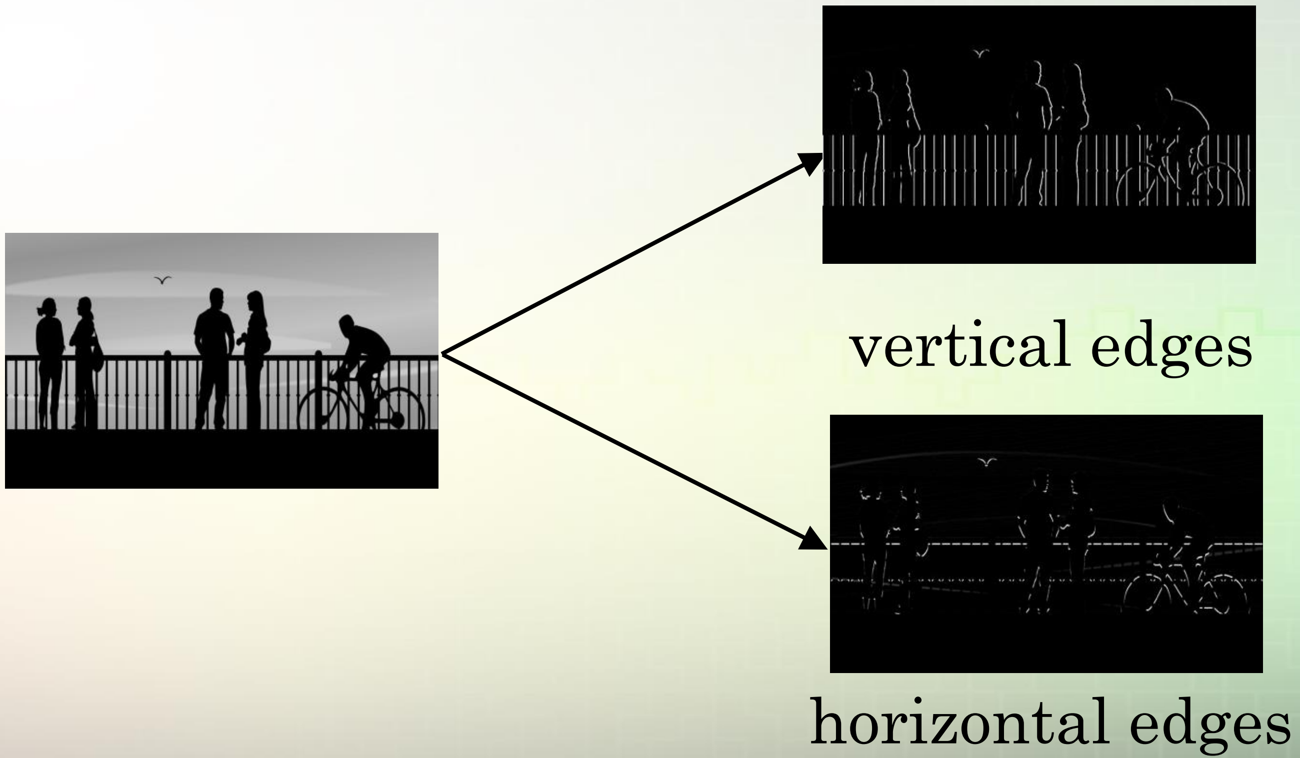
http://image-net.org/challenges/talks/ILSVRC+MSCOCO_12_17_15_introduction.pdf

Outline

- Motivation
- Convolution and pooling layer
- Basic elements of ConvNet architecture
- History
- Notable architectures for image classification

Microsoft Development Center Serbia

Computer Vision Problem



Vertical edge detection examples

10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0

*

1	0	-1
1	0	-1
1	0	-1

=

0	30	30	0
0	30	30	0
0	30	30	0
0	30	30	0

0	0	0	10	10	10
0	0	0	10	10	10
0	0	0	10	10	10
0	0	0	10	10	10
0	0	0	10	10	10
0	0	0	10	10	10

*

1	0	-1
1	0	-1
1	0	-1

=

0	-30	-30	0
0	-30	-30	0
0	-30	-30	0
0	-30	-30	0

Learning to detect edges

3	0	1	2	7	4
1	5	8	9	3	1
2	7	2	5	1	3
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

$*$

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

$=$

Padding

- Allowing filter to go outside input image
- Usually pad with zeros
- Used for adjusting output size
  - Example: stride = 1, padding = (kernel size – 1) / 2

Input

0	0	0	0	0	0
0	-1	-2	-2	2	0
0	-2	0	2	0	0
0	-2	0	2	0	0
0	0	0	0	0	0
0	0	0	0	0	0

Weights

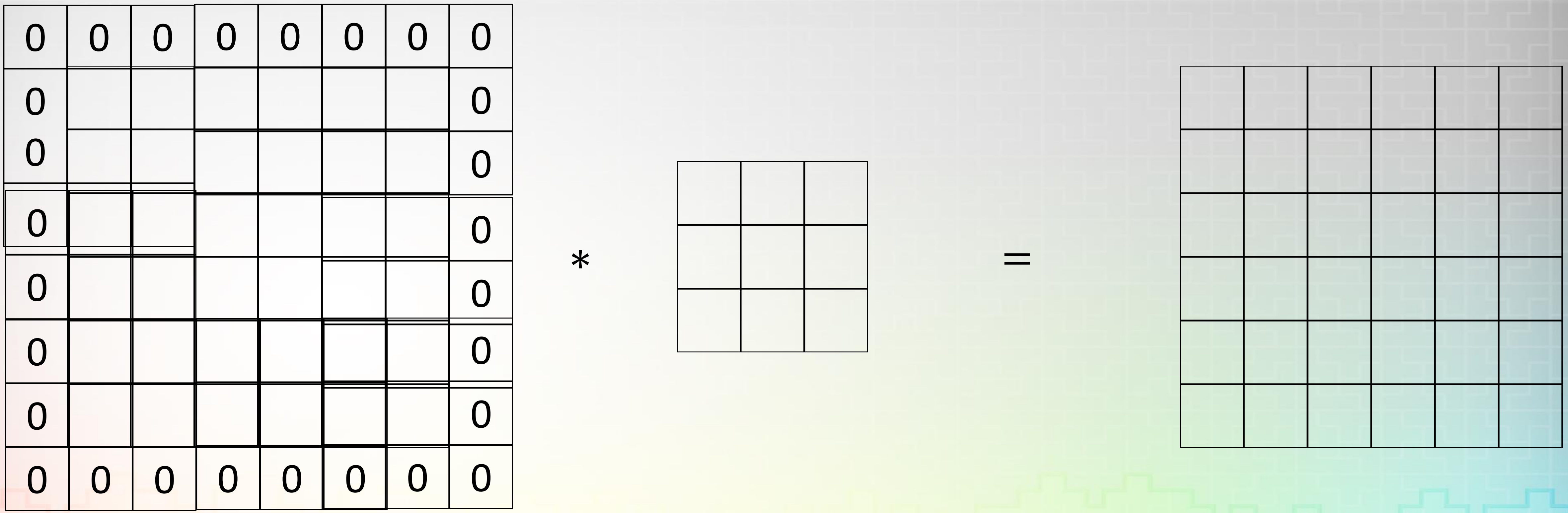
0	2	2
2	0	0
2	0	0

Output

0	-6	-4	0
-6	-16	0	12
-4	0	4	4
-4	4	4	0

Microsoft Development Center Serbia

Padding - Valid and Same convolutions



"Valid": Only convolve with valid pixels

"Same": Pad so that output size is the same as the input size.

Stride

- Distance between consecutive kernel applications
- Used for reducing spatial resolution

Input

-1	-2	-2	2	2
-2	0	2	0	0
-2	0	2	0	0
0	0	0	0	0
0	0	0	0	0

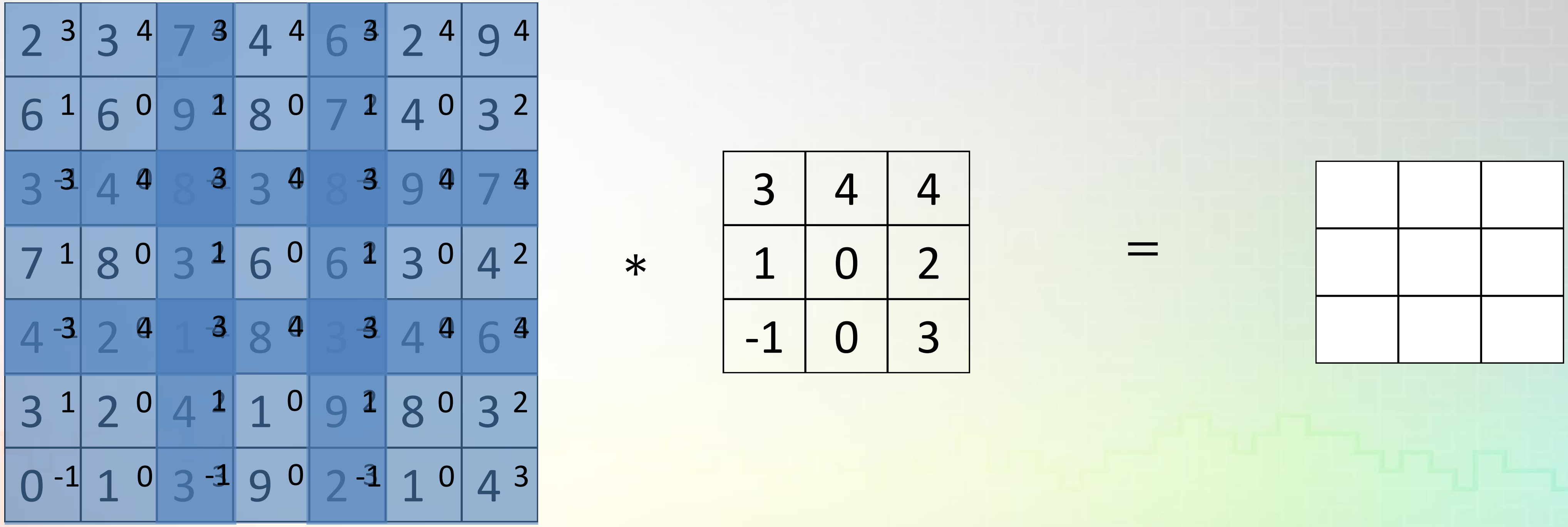
Weights

0	2	2
2	0	0
2	0	0

Output

-16	16
4	0

Strided convolution



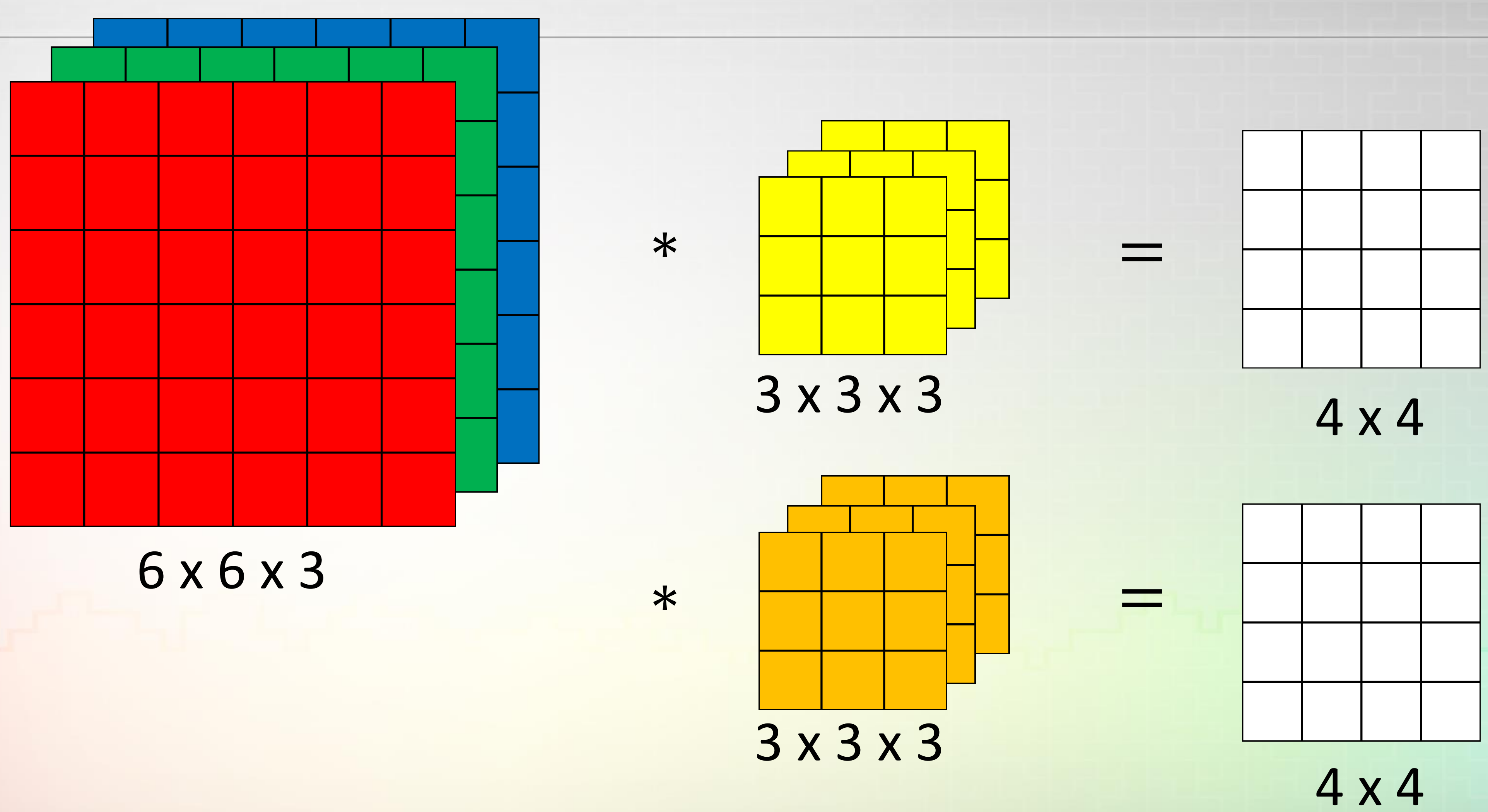
Summary of convolutions

$n \times n$ image $\qquad f \times f$ filter

padding p $\qquad$ stride s

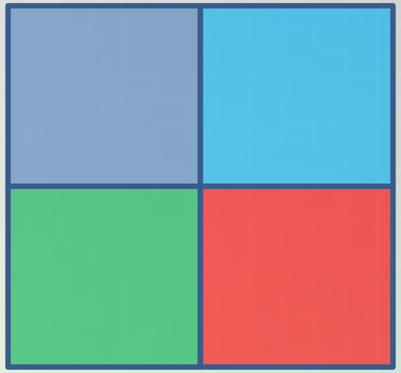
$$\left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor \times \left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor$$

Convolutions over volumes



Pooling layer: Max and Average pooling

1	3	2	1
2	9	1	1
1	3	2	3
5	6	1	2



Hyperparameters:

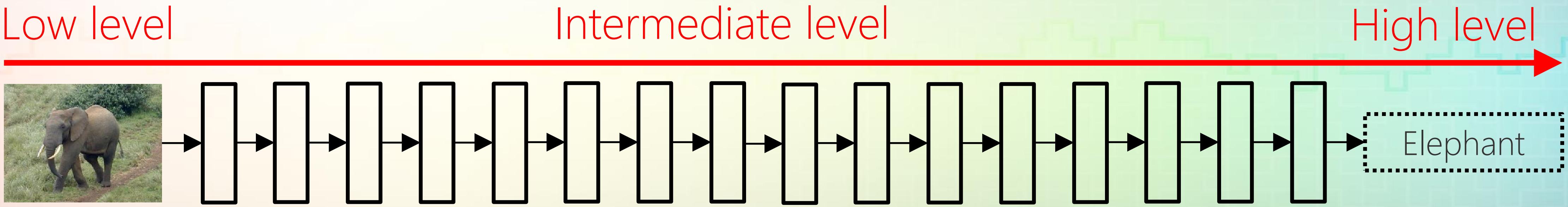
- f : filter size
- s : stride

Outline

- Motivation
- Convolution and pooling layer
- Basic elements of ConvNet architecture
- History
- Notable architectures for image classification

ConvNets are a form of deep learning

- Many simple nonlinear layers
- Features are learned from data, not handcrafted
- Features are hierarchical



Traditional approach

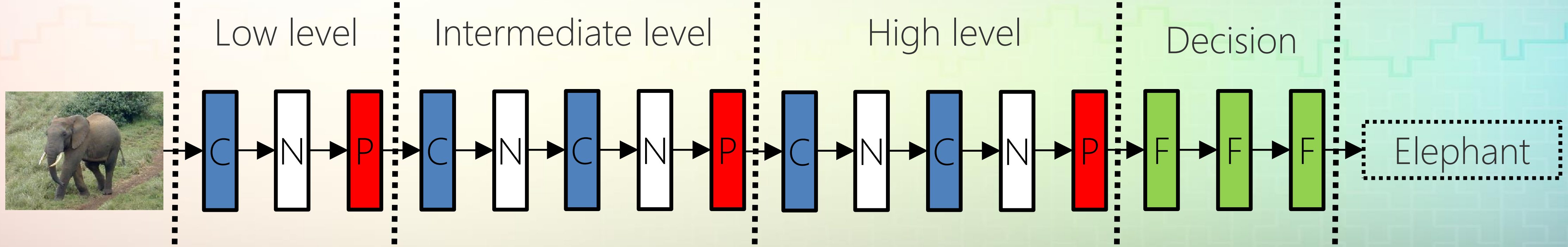
- Sequence of nonlinear transformations
- Handcrafted components
- Machine learned components trained independently

SIFT HoG … → Quantization (clustering) → Spatial pyramid → Histogram → SVM → Elephant

SIFT HoG …	Quantization (clustering)	Spatial pyramid	Histogram	SVM
Design	Learning	Design	Design	Learning

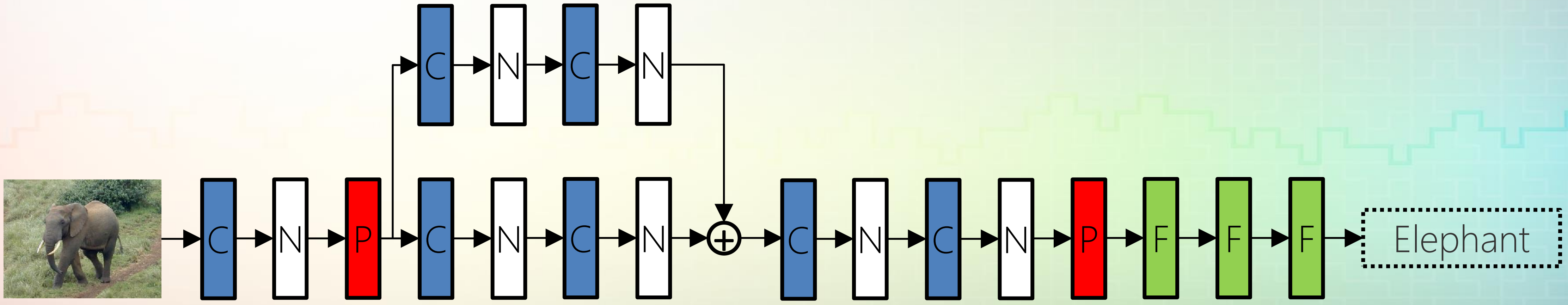
Basic ConvNet architecture

- Convolution layer
- Pooling layer
- Fully connected layer (like MLP)
- Various normalization layers
- Other...



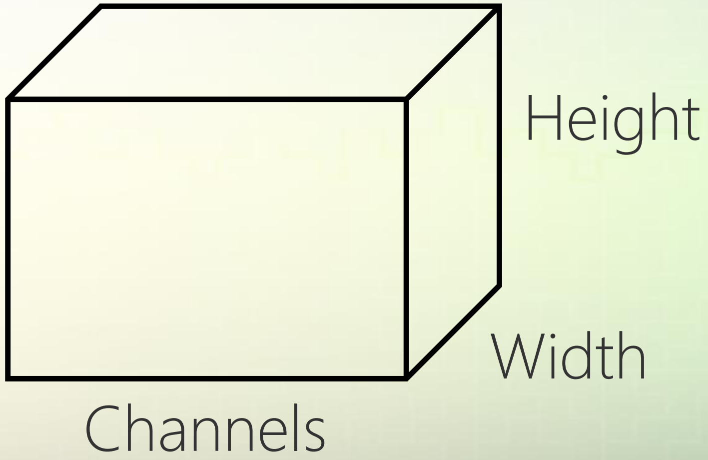
More recent ConvNet architectures

- Contain parallel branches
- Directed acyclic graph (DAG)



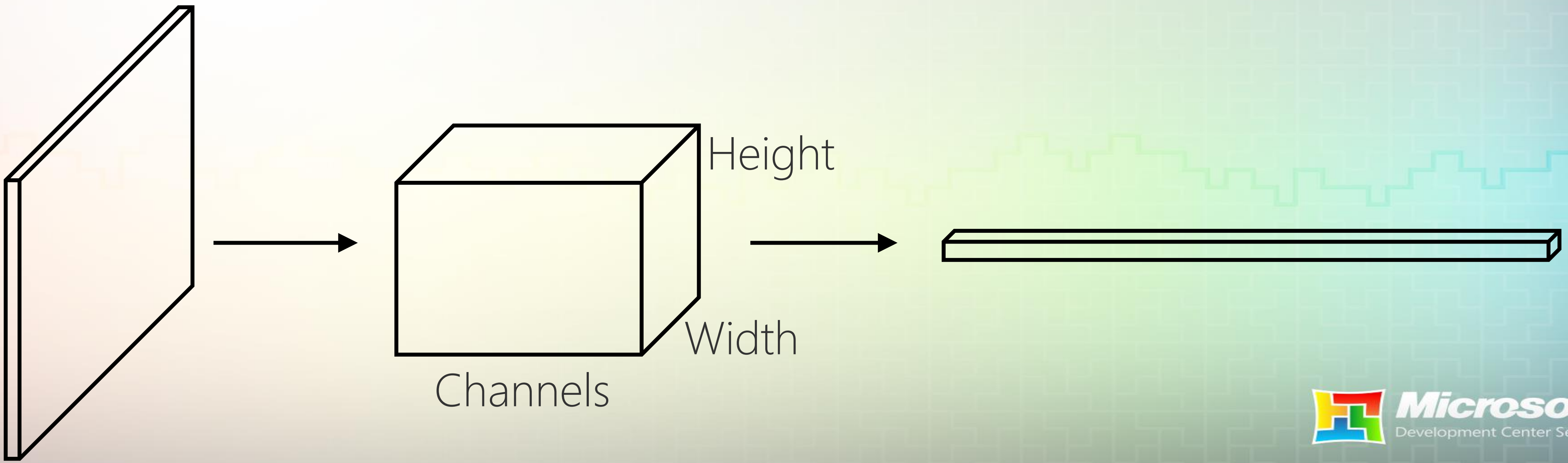
Activations

- Interpreted as multi-channel “images”
  - Network input: 1 channel (grayscale) or 3 channels (RGB)
  - Other activations can have more channels
- Channels are also called feature maps



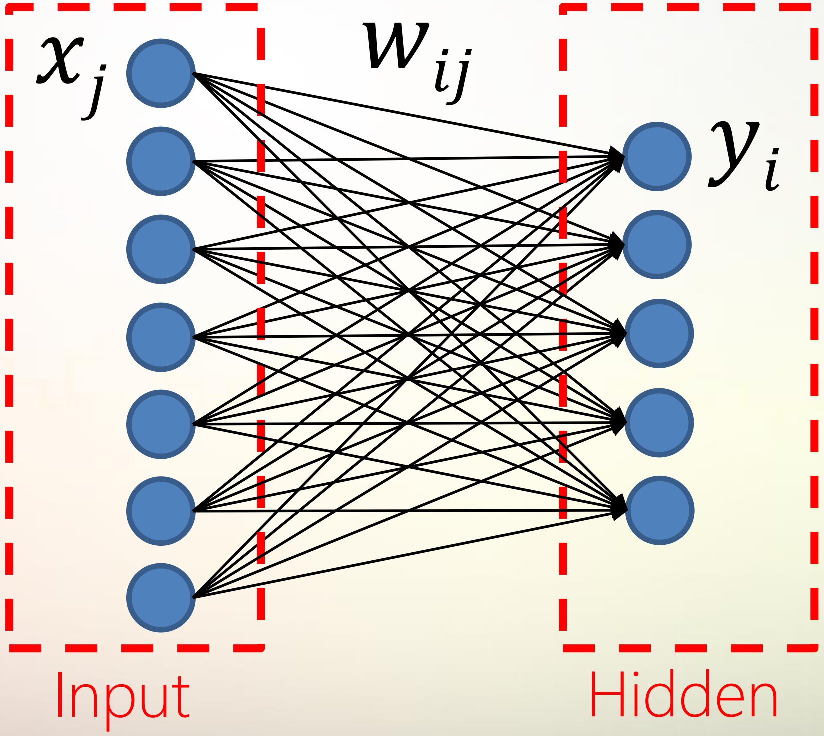
Activations

- Usually in practice
  - Spatial dimensions decrease with depth
  - Number of channels increases with depth



Fully connected (FC) layer

- Same as "hidden" layer in MLP



Activation function

Input

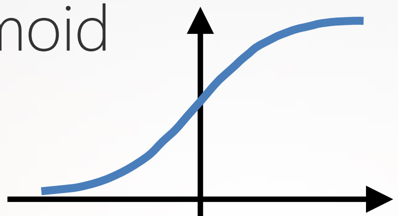
$$y_i = g\left(\sum_j w_{ij} x_j\right)$$

Output

Weight

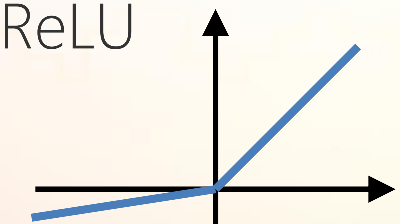
Types of activation functions

Sigmoid



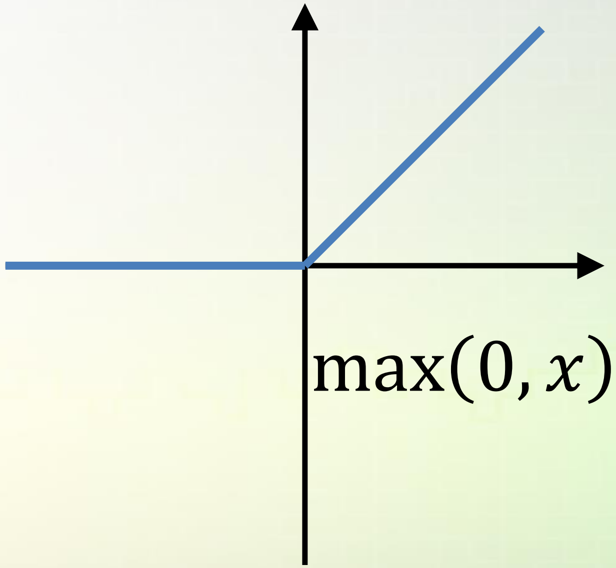
$$\frac{1}{1+e^{-x}}$$

"Leaky" ReLU



$$\begin{cases} x, x \geq 0 \\ ax, x < 0 \end{cases}$$

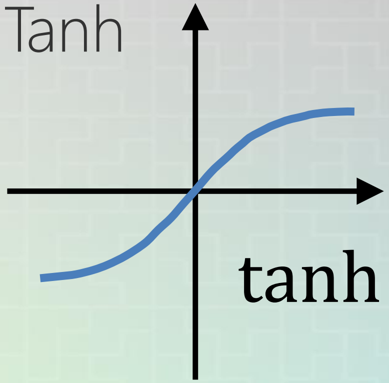
Rectified linear (ReLU)



$$\max(0, x)$$

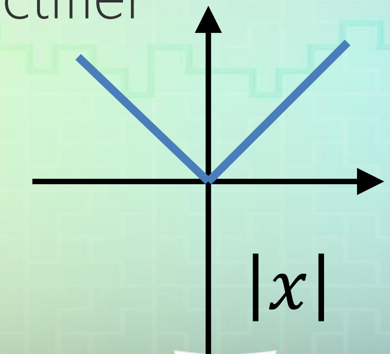
Most popular

Tanh



$$\tanh x$$

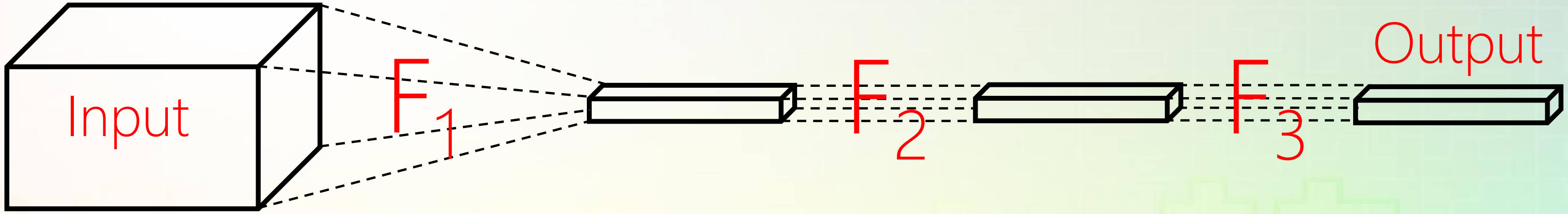
Rectifier

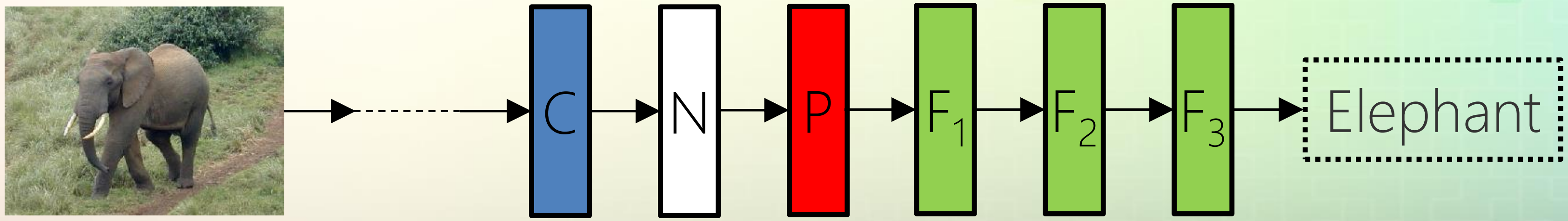


$$|x|$$

Fully connected layer

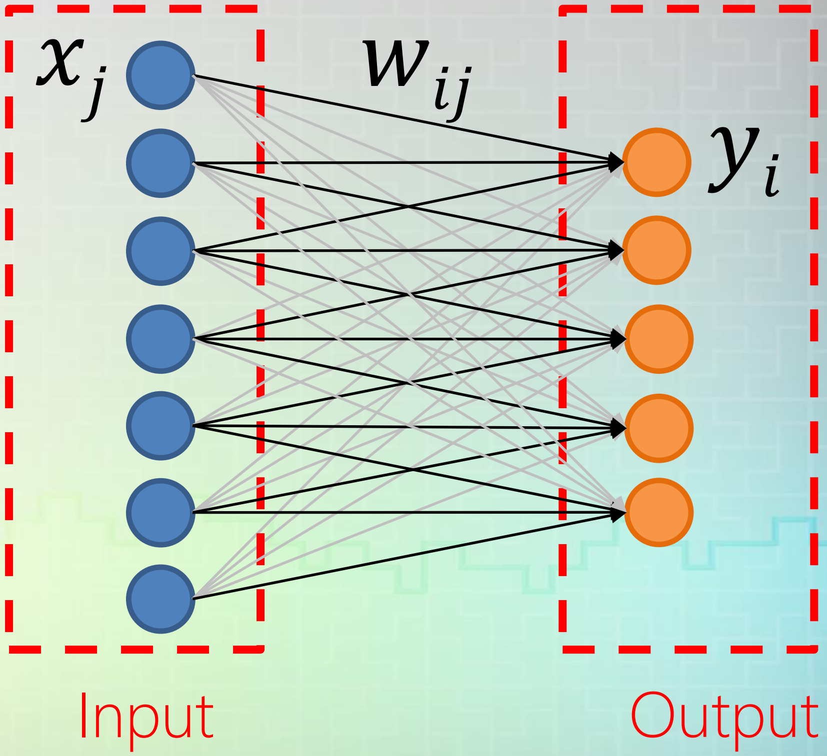
- Output has 1 x 1 spatial size
- Last FC layer is followed by softmax function
  - Converts activations to probabilities





Local connectivity

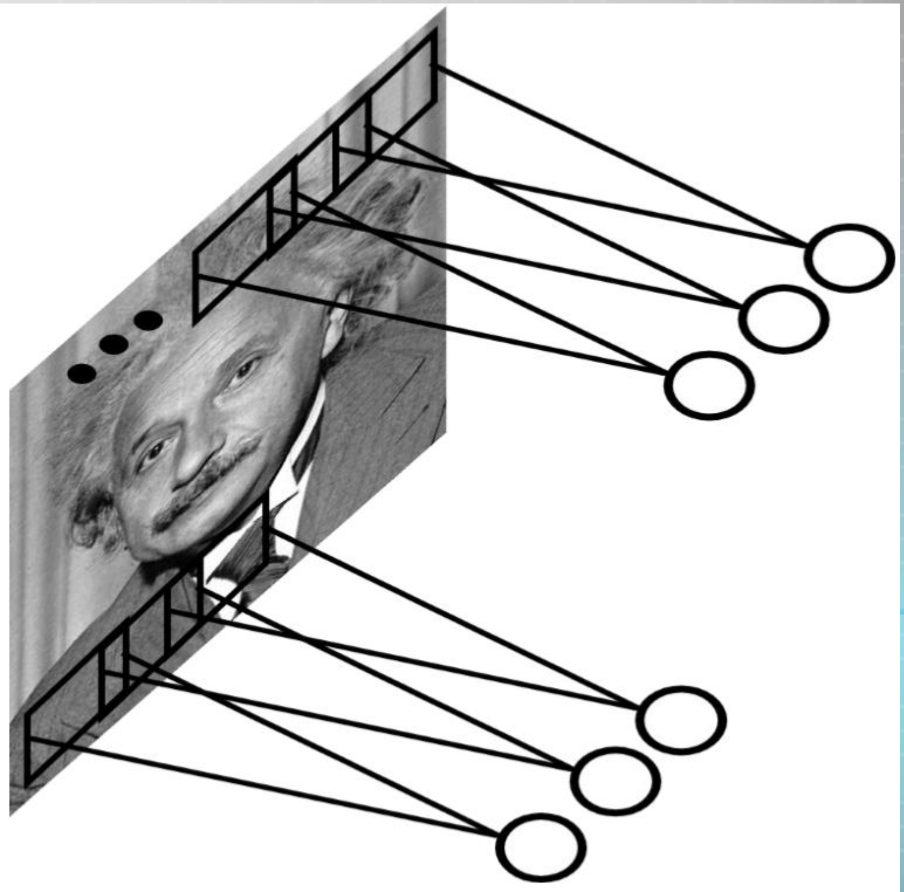
- Output neuron is connected only to "nearby" input neurons
  - Neighborhood in spatial coordinates
- Fewer parameters and computation
  - Many zero weights



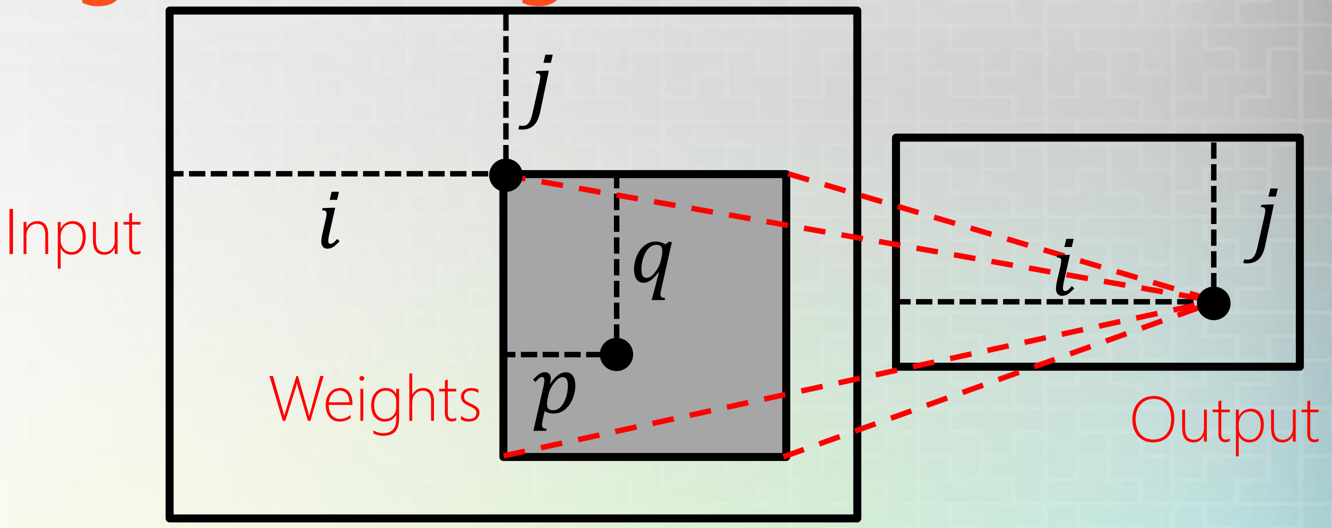
Weight sharing

Y. LeCun, M. A. Ranzato

- All output neurons have the same set of weights
- Stationarity: same features are of interest in all parts of image



Local connectivity + weight sharing = convolution



Window (kernel) size

Input

Activation function

$$y_{ij} = g\left(\boxed{\sum_{p,q=1}^{k} w_{pq} x_{i+p,j+q}}\right) = g\left((W * x)_{ij}\right)$$

Output

Weight

CONVOLUTION (actually correlation)

Convolution

- Like in image processing, but filter coefficients are learned

Activation function

Input

Output

$$y = g(W * x)$$

Kernel (filter)

- Variant with additive (*bias*) and multiplicative constants

Scaling (trained)

Bias

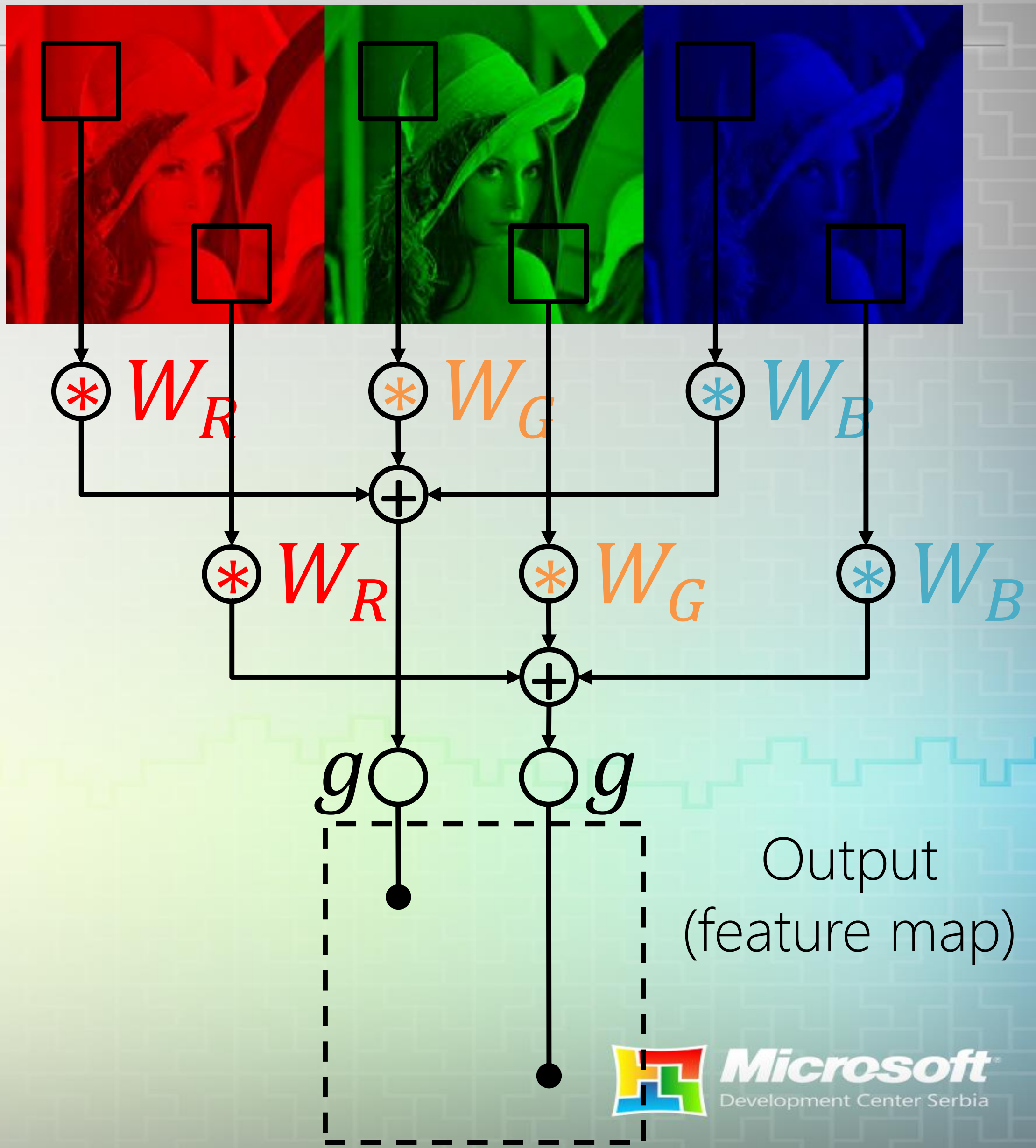
$$y = cg(W * x + b)$$

Multichannel input

- Each input has its own filter
- Results are added pixelwise
  - Before applying activation function

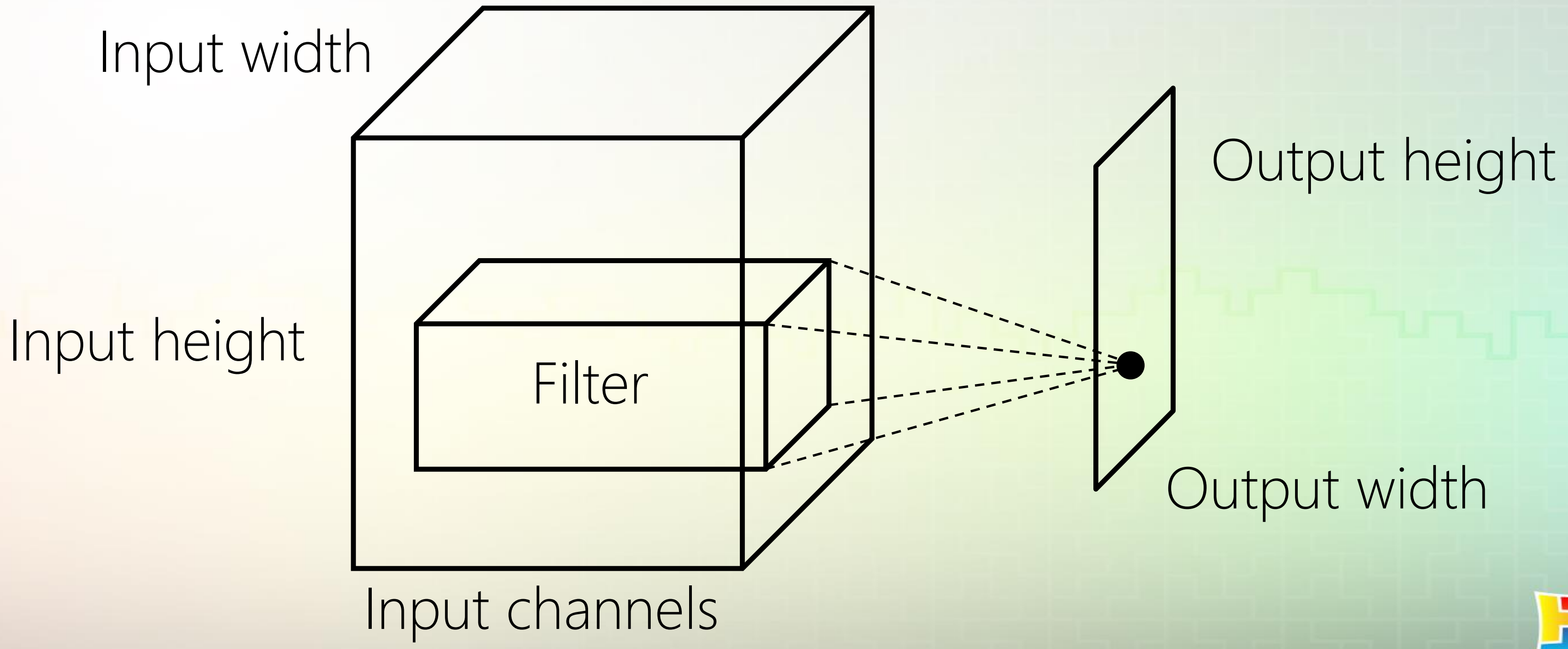
$$y = g\left(\sum_c W_c * x_c\right)$$

Index of input channel



Multichannel input: equivalent view

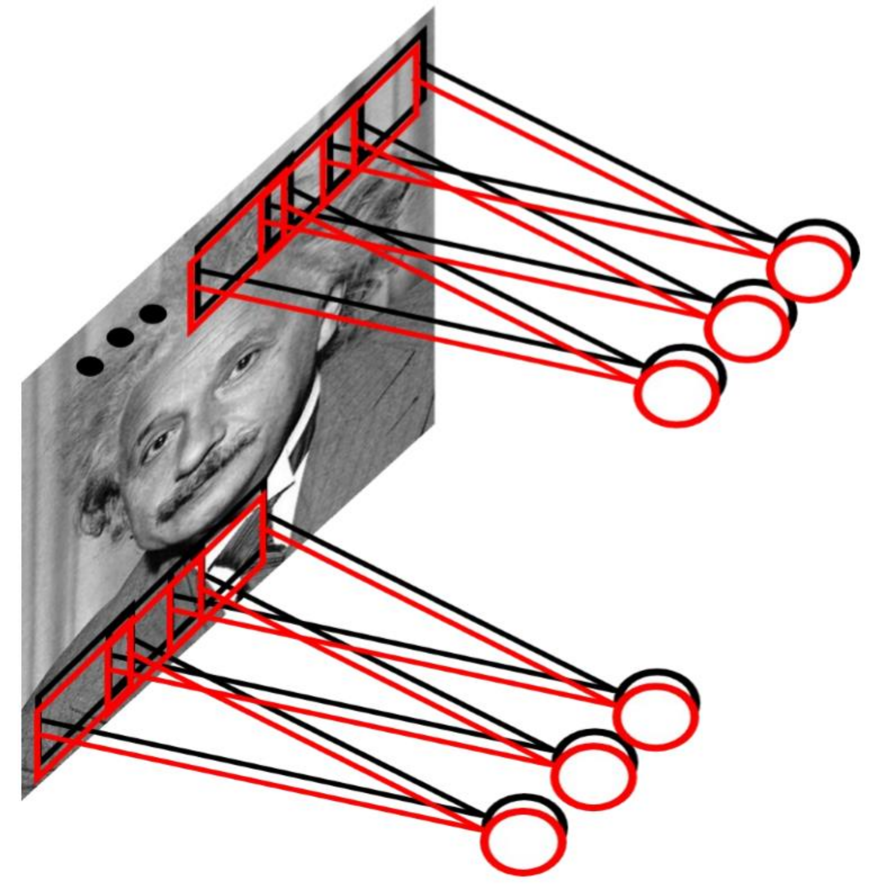
- A 3D filter “slides” across multichannel input image



Y. LeCun, M. A. Ranzato

Multichannel output

- Computing multiple feature maps of the same input
- All neurons “looking” at some region compute feature vector for that region
- Similar to hand-engineered features (e. g. Gabor) but trained



$$y_n = g\left(\sum_c W_{nc} * x_c\right)$$

Output channel (feature map)

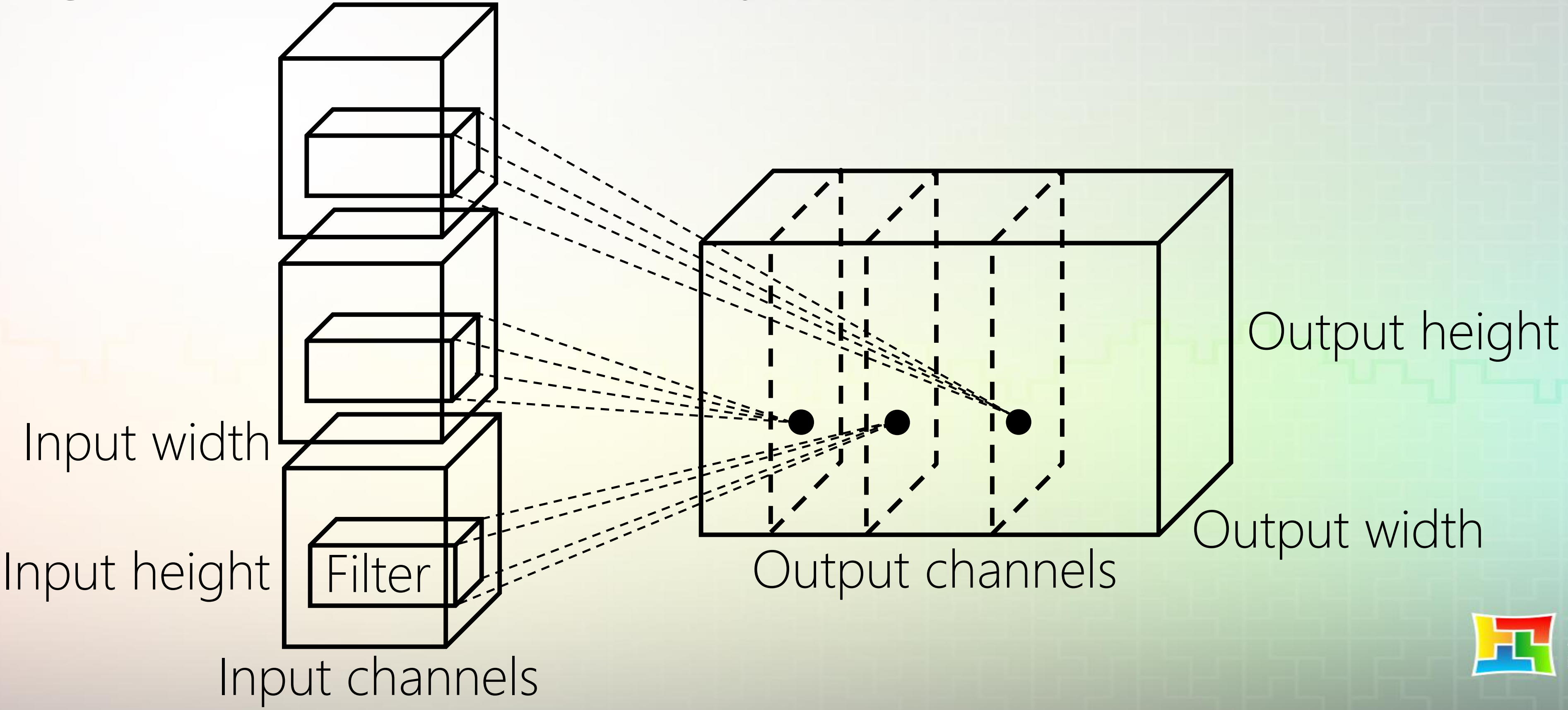
Kernel (filter)

Input channel

Microsoft Development Center Serbia

Multichannel output: equivalent view

- Weights of convolutional layer form a 4D tensor



Convolutional layer

- Set of convolutional filters with activation function
- Output (spatial) size ≈
  (input size + pad – kernel size + 1) / stride
- Parameter count =
  Input channels x output channels x kernel size^2
- Operation (multiply + add) count =
  input height x input width x input channels x
  output height x output width x output channels

1 x 1 convolutional layer

- All kernels have spatial size 1 x 1
- Used for adjusting channel count
- Equivalent to applying the same FC layer to each pixel's feature vector
- Input and output have the same spatial size

Pooling

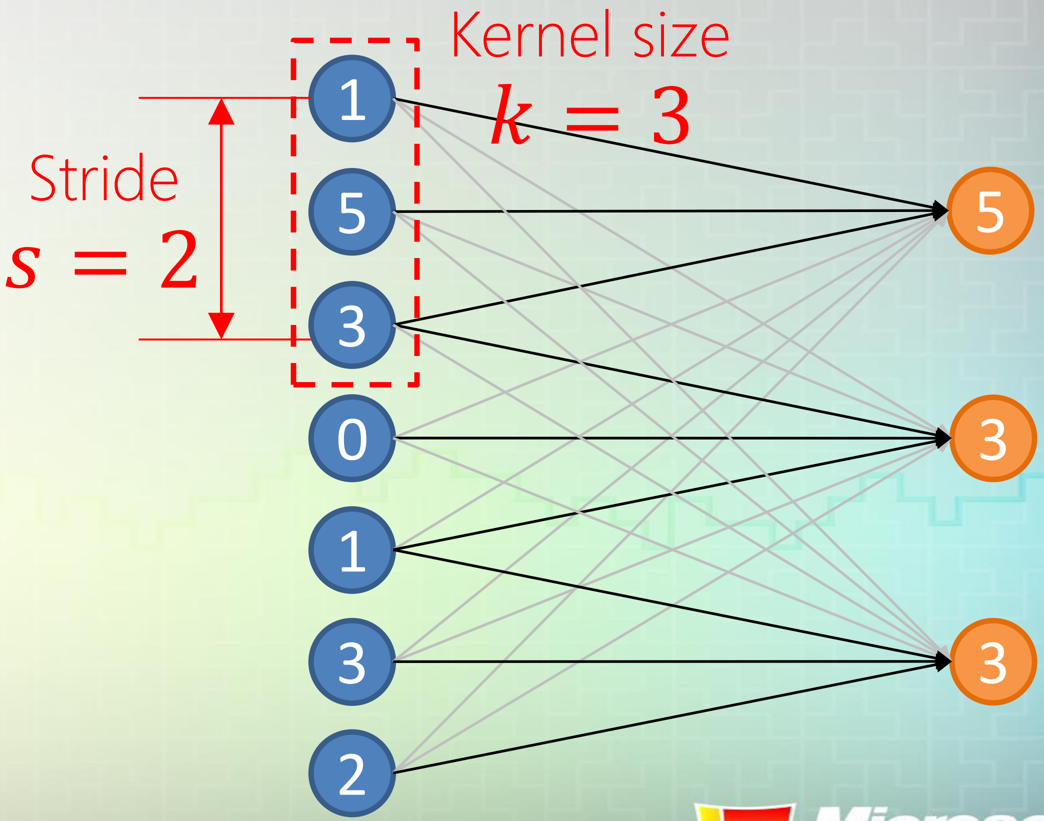
- Combining outputs of nearby input neurons
  - Max pooling

$$y_{ij} = \max_{p,q} x_{i+p,j+q}$$

  - Average pooling

$$y_{ij} = \frac{1}{k^2} \sum_{p,q=1}^{k} x_{i+p,j+q}$$

  - L1, L2 norm…



Example

- Max pooling with kernel size 3 x 3 and stride 2

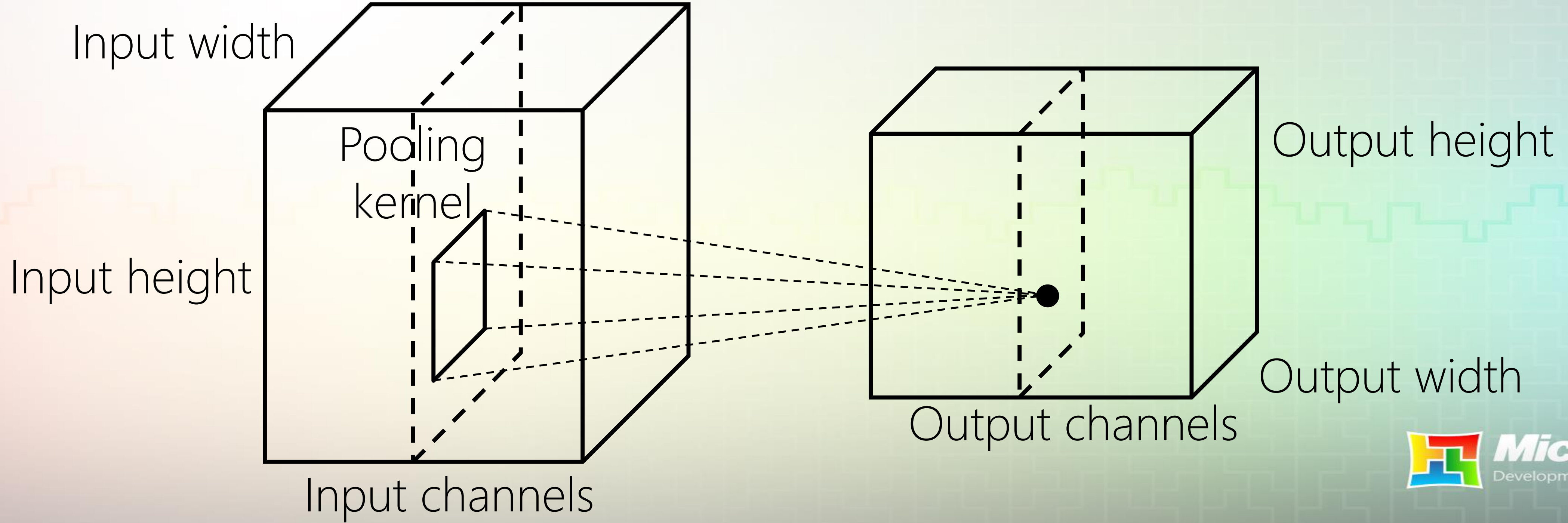
0	0	7	0	0
0	5	-3	6	2
0	0	1	0	0
0	2	0	0	-1
2	0	0	5	0

Max pooling →

7	7
2	5

Multichannel input and output

- Applied independently to each input channel
- Input channel count = output channel count



Pooling layer

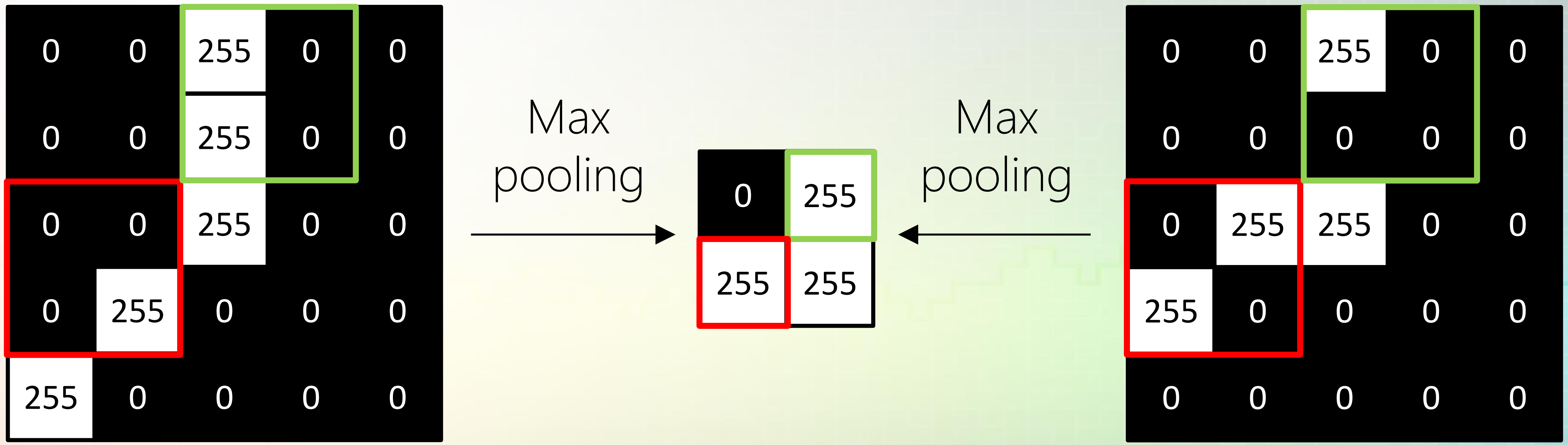
- Only pooling operation, without activation function
- Pooling operation can be nonlinear
  - E.g. max pooling
- Pooling operation is differentiable
  - Allows backpropagation
- Stride and padding

Pooling layer

- Output (spatial) size ≈
  (input size + padding – kernel size + 1) / stride
- Parameter count = 0
- Operation (multiply + add) count =
  output height x output width x channel count x kernel size^2

Invariance to local translation

- Locality is determined by kernel size

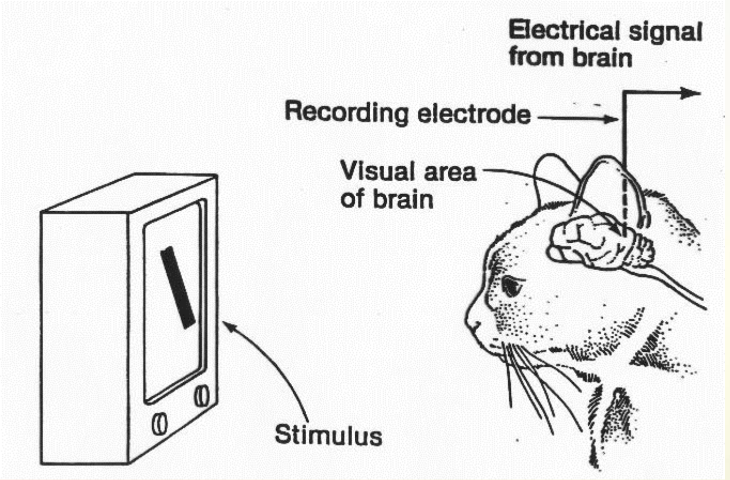


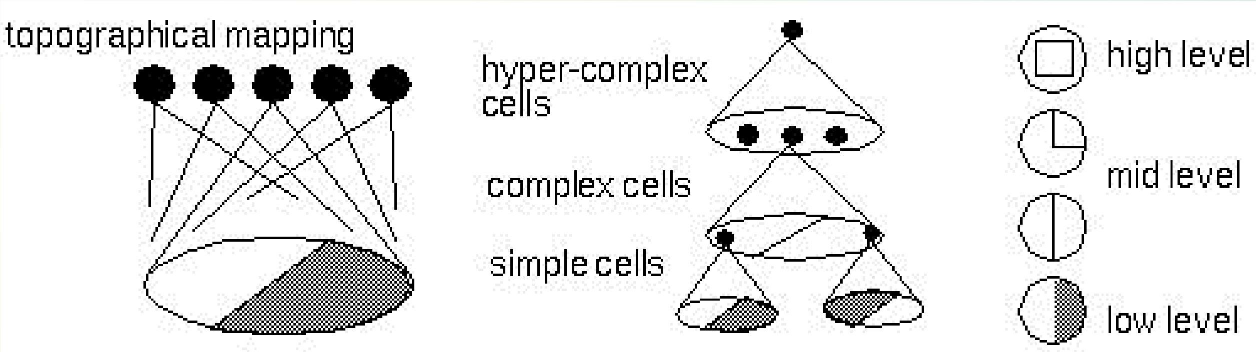
Outline

- Motivation
- Convolution and pooling layer
- Basic elements of ConvNet architecture
- History
- Notable architectures for image classification

Cat visual cortex

- Simple, complex, and hyper-complex cells

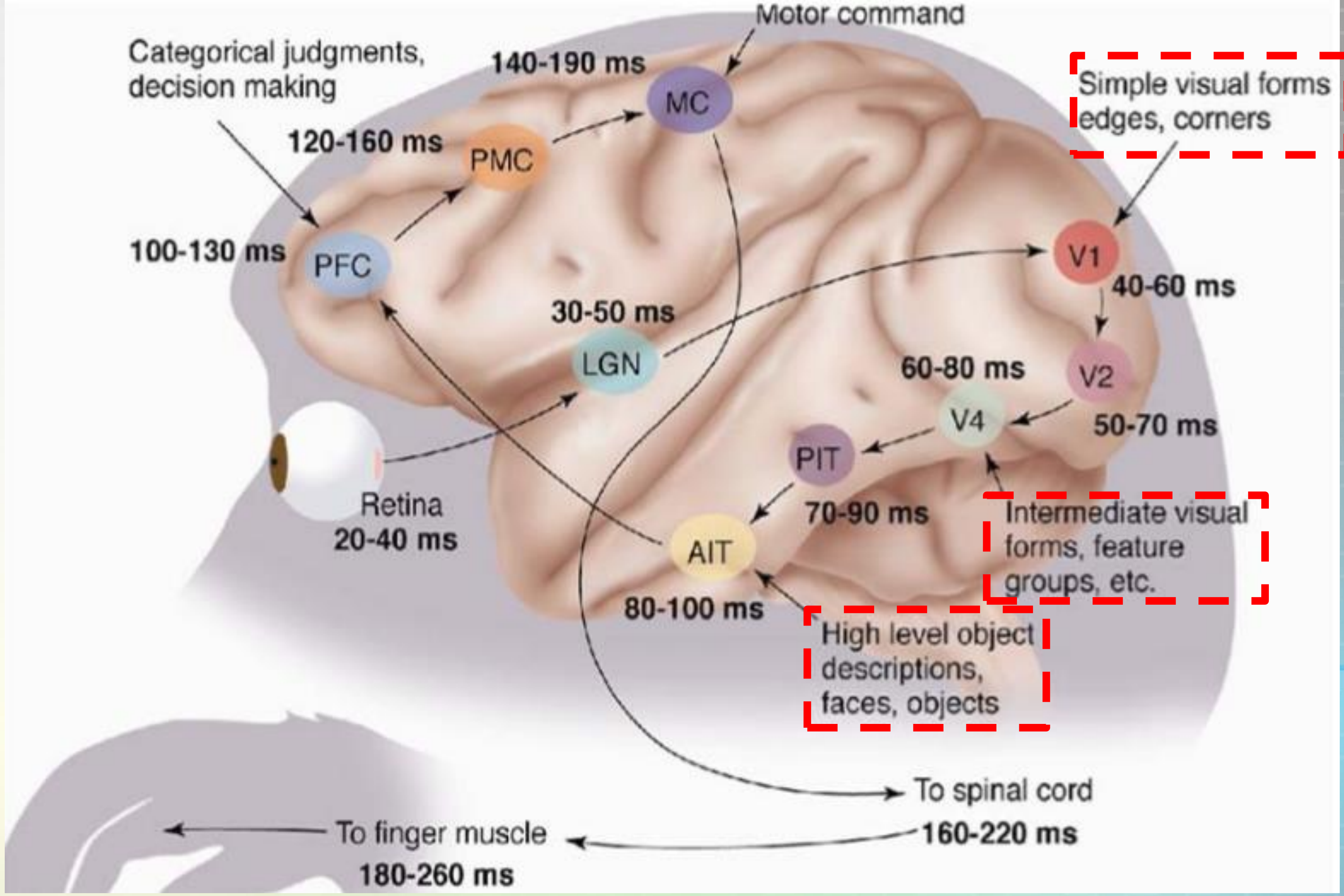




Hubel and Wiesel, Receptive fields of single neurones in the cat's striate cortex, 1959

Human visual cortex

- Hierarchy of features from low to high level

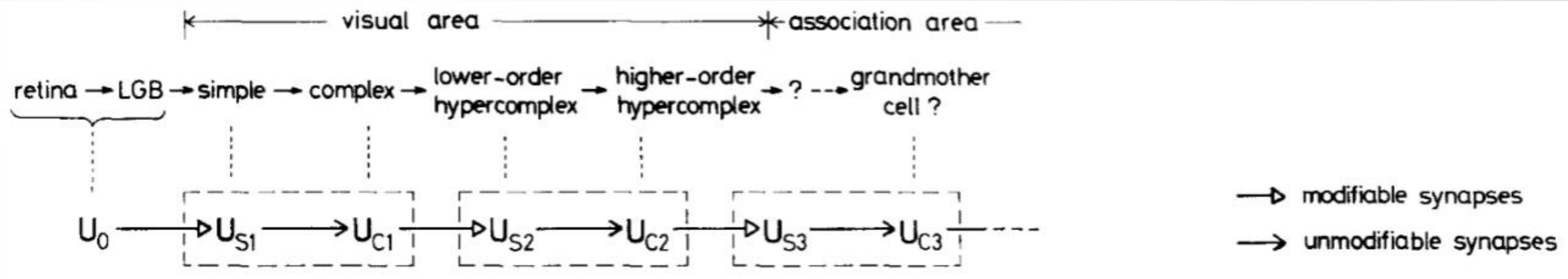


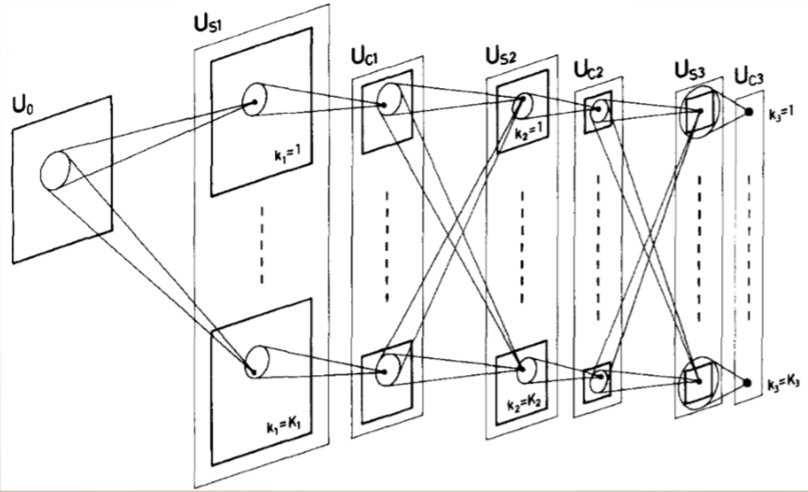
S. J. Thorpe, M. Fabre-Thorpe,
Seeking Categories in the Brain, Science, 2001.

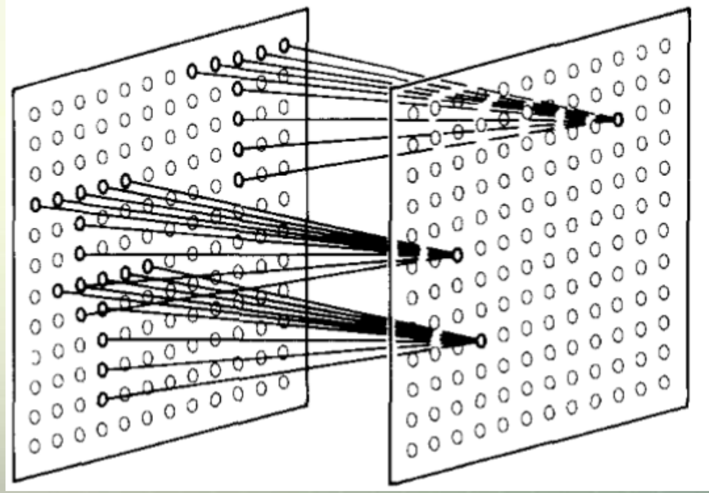
Microsoft Development Center Serbia

Neocognitron [Fukushima 1980]

- No supervised learning algorithm

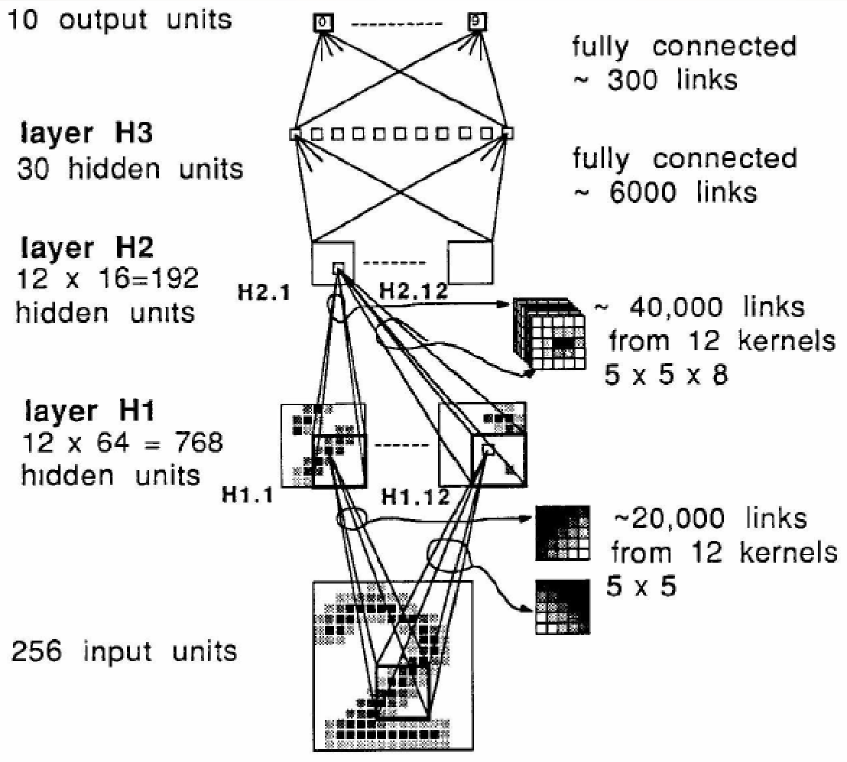


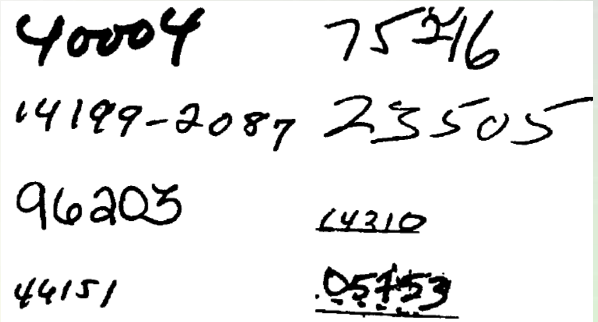


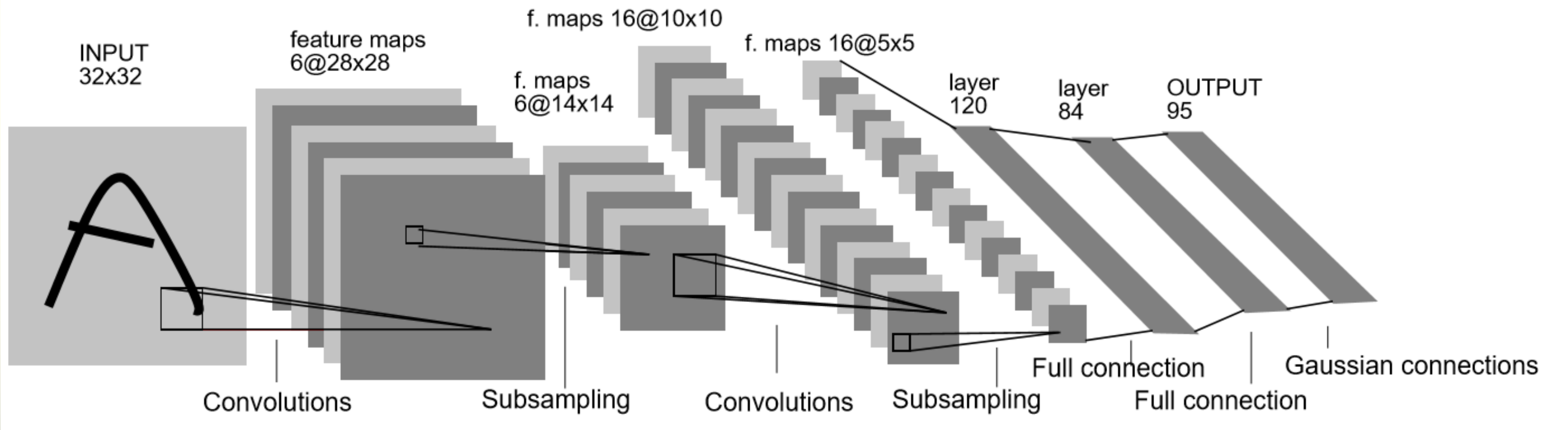


Convolutional network for handwriting recognition

[Le Cun et al. 1989-1998]







Fall and rise of convolutional networks

- Rise of Support Vector Machines (SVM) in mid-1990s
  - Pros: theory, convex optimization
  - Cons: handcrafted features

- Return of convolutional networks from ~2012
  - Availability of data and compute resources
  - Trained features outperform handcrafted features
  - Enables attacking harder problems

Today: convolutional networks are everywhere

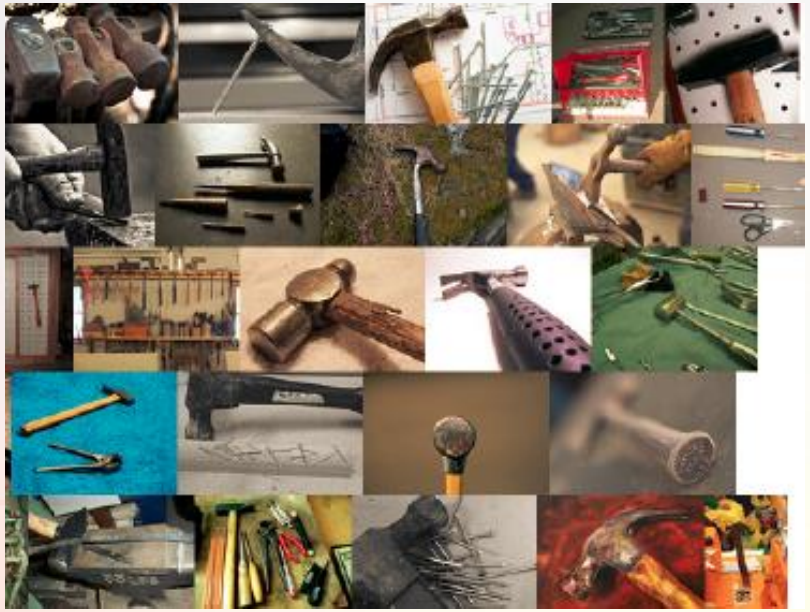
- Handwriting
- Objects in image
- Scene understanding
- OCR “in the wild”
- Traffic signs
- Pedestrians
- Image segmentation
- Activity in video
- Image captioning
- Depth estimation
- Textures
- Body pose
- …

Outline

- Motivation
- Convolution and pooling layer
- Basic elements of ConvNet architecture
- History
- Notable architectures for image classification

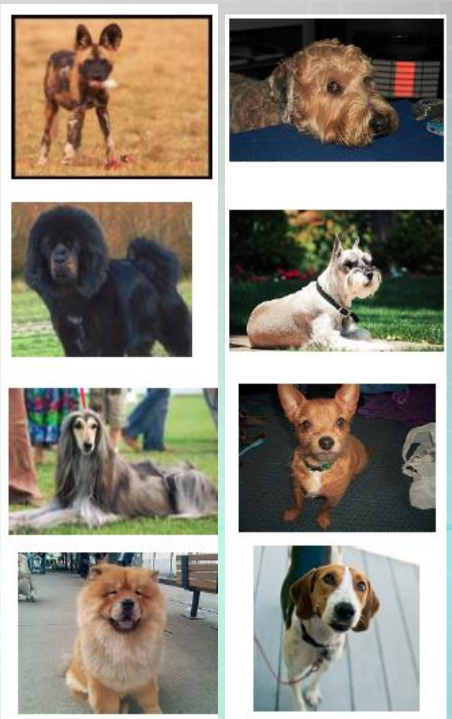
Image classification

hammer



chime

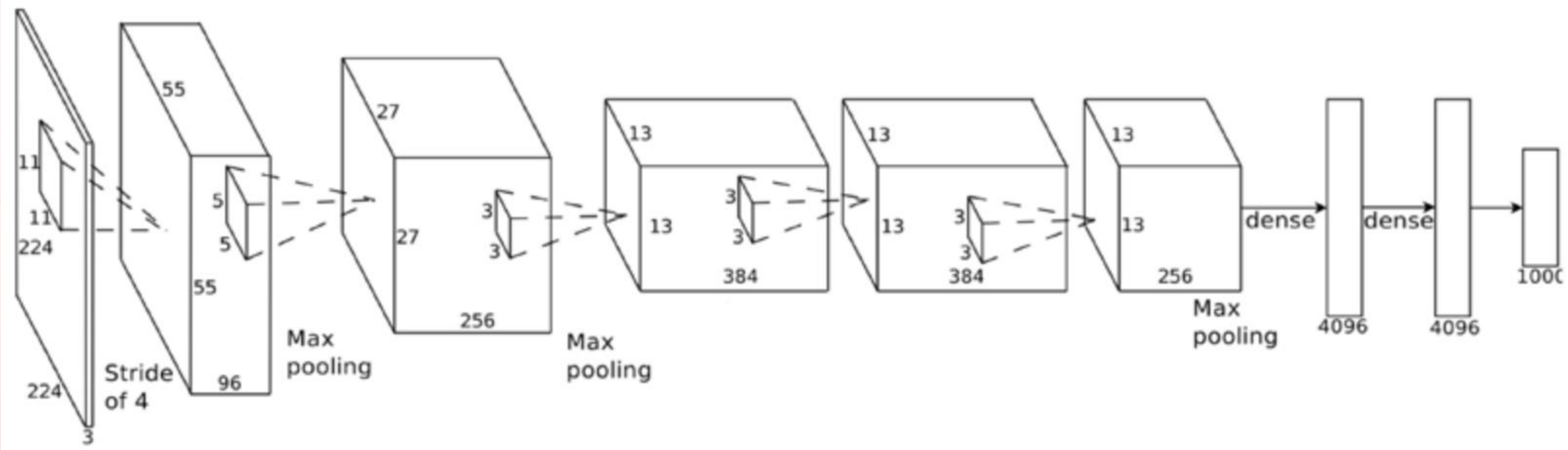
dog



ImageNet Large-scale Visual Recognition Challenge 2012
http://image-net.org/challenges/LSVRC/2012/ilsvrc2012.pdf

AlexNet [Krizhevsky et al. 2012]

- Restarted interest in convolutional networks in computer vision



Fully connected, 1000 outputs + softmax

Fully connected 4096 outputs + ReLU

Fully connected 4096 outputs + ReLU

Max pooling

Conv 3x3, 256 outputs + ReLU

Conv 3x3, 384 outputs + ReLU

Conv 3x3, 384 outputs + ReLU

Max pooling 2x2 subsampling

Local normalization

Conv 11x11, 256 outputs + ReLU

Max pooling 2x2 subsampling

Local normalization

Conv 11x11, 96 outputs + ReLU

Microsoft Development Center Serbia

AlexNet

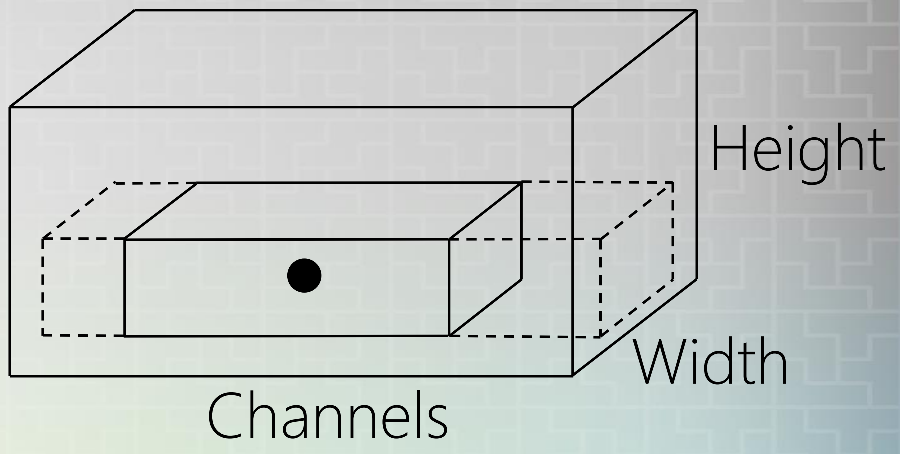
- 60 million parameters
- 832 million operations (multiply-adds)
- Top-5 classification error 16% on ImageNet 1K test
  - Winner of ILSVRC 2012 (classification and detection)
  - Previous record 26%

AlexNet training

- Supervised learning, gradient descent w/ backpropagation
  - 90 epochs of ImageNet 1K training set (1.3 million images)
  - 5-6 days on 2 x NVIDIA GTX 580 (3GB)
- Techniques
  - ReLU activation function
  - Local normalization
  - Dropout
  - Data augmentation

Local normalization

- Normalize activations by local statistics
  - E.g. mean and standard deviation
  - Statistics from a (3D) neighborhood
- Encourage "competition" for high activations
  - Prevent coadaptation of neurons
  - If all activations are high, they all get reduced by a lot
  - Bio-inspired: lateral inhibition

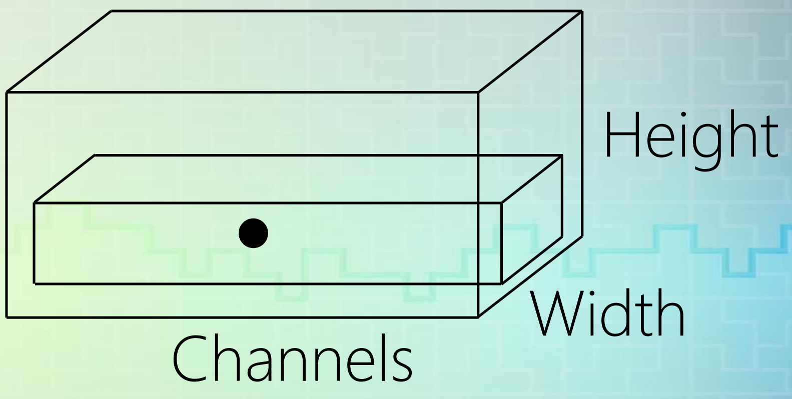


Local normalization

- AlexNet

$$b^i_{x,y} = a^i_{x,y} \Big/ \left(k + \alpha \sum_{j=\max(0,i-n/2)}^{\min(N-1,i+n/2)} (a^j_{x,y})^2 \right)^\beta$$

- Contrast normalization [Jarret et al. 2009]
  - Stats are computed from all channels
  - Weighted mean, weights decay with spatial distance as 2D Gaussian



Mean

$$y_{cij} = \frac{x_{cij} - m_{cij}}{\max(k, \sigma_{cij})}$$

Standard deviation

Dropout

- Regularization technique
- In each forward pass remove a random subset of neurons in a given layer
  - Those neurons do not participate in backpropagation either
  - Usually remove each neuron independently with fixed probability (usually 0.5)
- Prevents coadaptation of neurons, makes network more robust

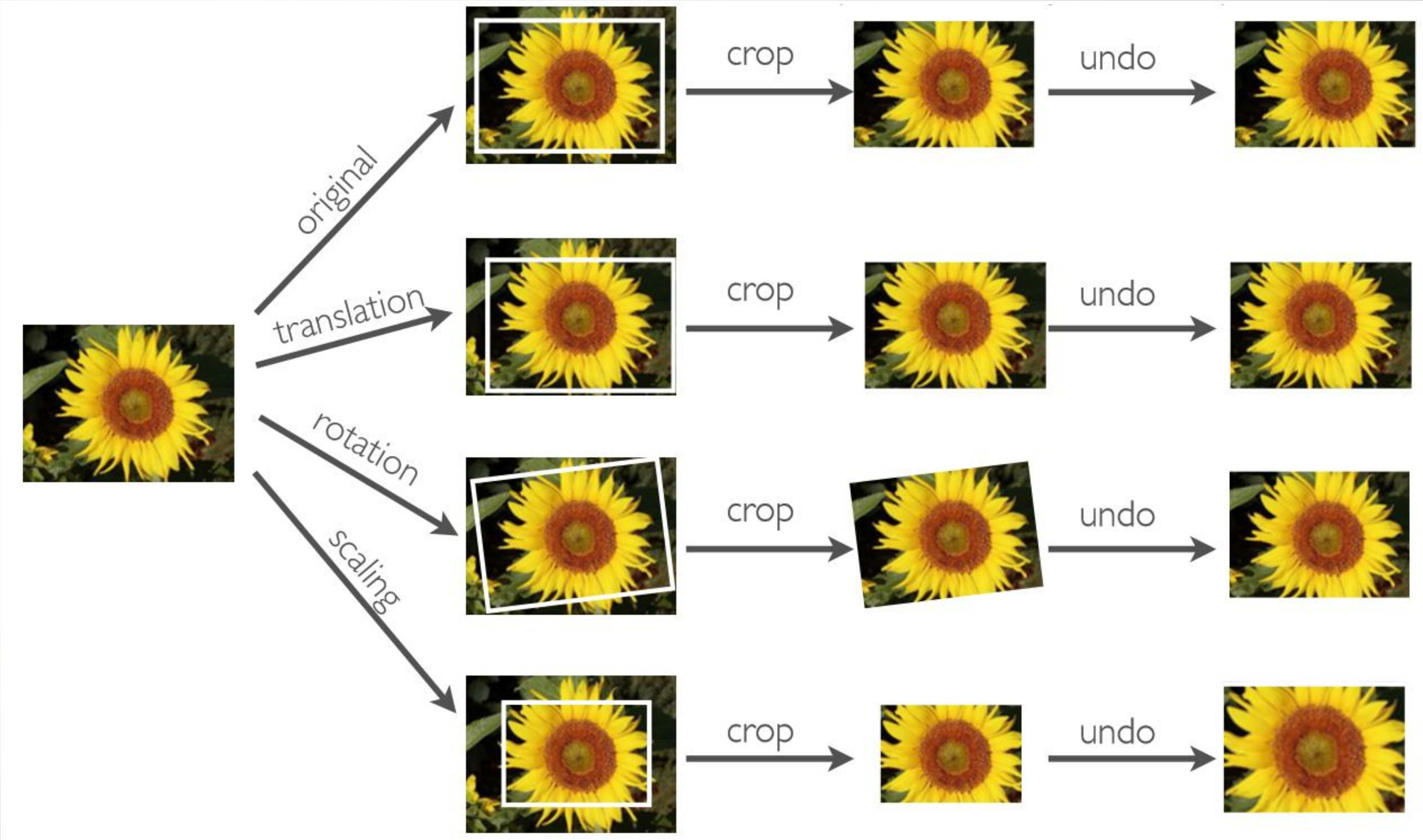
Dropout

- At runtime multiply activations of neurons in layers subject to dropout
  - Factor 1/(1-p), where p is the dropout probability
  - Exponential family of networks with shared weights
  - Expected activation of a randomly chosen network from the family
- Slows down convergence
- In AlexNet applied to first two FC layers

Data augmentation

- Problem: not enough training data (slow labeling)
- Data augmentation: synthesizing a large amount of “realistic” training examples from a small amount of real examples

Example: image classification



Types of variations

- Invariances built into the architecture
  - Local translation (due to pooling)
  - Local change in lighting (due to pooling, local normalization...)
- Most useful are those that are not built in
  - Rotation, scaling, noise...

Data augmentation in AlexNet

- Random crop 224 x 224 pixels
- Horizontal flip: with probability 0.5 replace image with its mirror image (with respect to vertical axis)
- Lighting augmentation
  - For each image choose a random RGB displacement, add it to each pixel
  - "Realistic" RGB displacement is obtained from training set statistics
    - PCA (Principal Component Analysis) of all RGB pixel values

VGGNet [Simonyan and Zisserman 2014]

- Simplified design, increased depth
  - Convolution: kernel 3 x 3, stride 1, padding 1
  - Max pooling: kernel 2 x 2, stride 2
- Idea: replace 5 x 5 layer with two 3 x 3 layers
  - Less computation, more nonlinearity

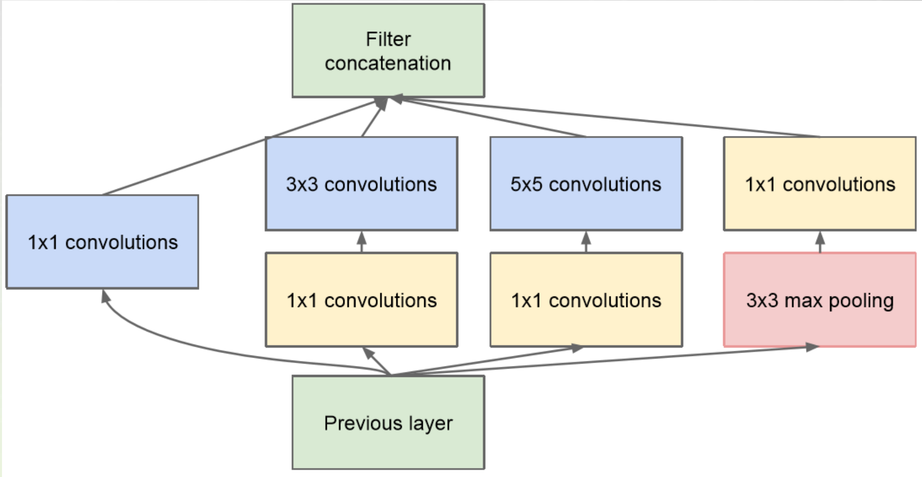
C	D	E
16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)		
conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool		
conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool		
conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool		
conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool		
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		

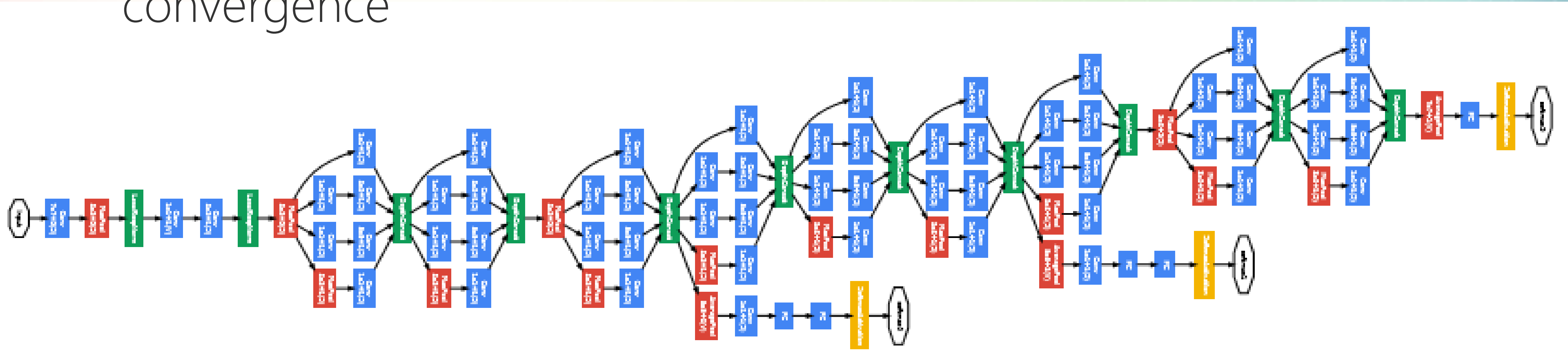
VGGNet [Simonyan and Zisserman 2014]

- Top-5 classification error 7.3% on ImageNet 1K test
  - Second place in ILSVRC 2014
- 138 million parameters (more than AlexNet)
- 15.3 billion operations (much slower than AlexNet)

GoogLeNet [Szegedy et al. 2014]

- Inception module
  - Branching
  - 1 x 1 convolutions for dimensionality reduction
  - 2 auxiliary loss functions improve convergence



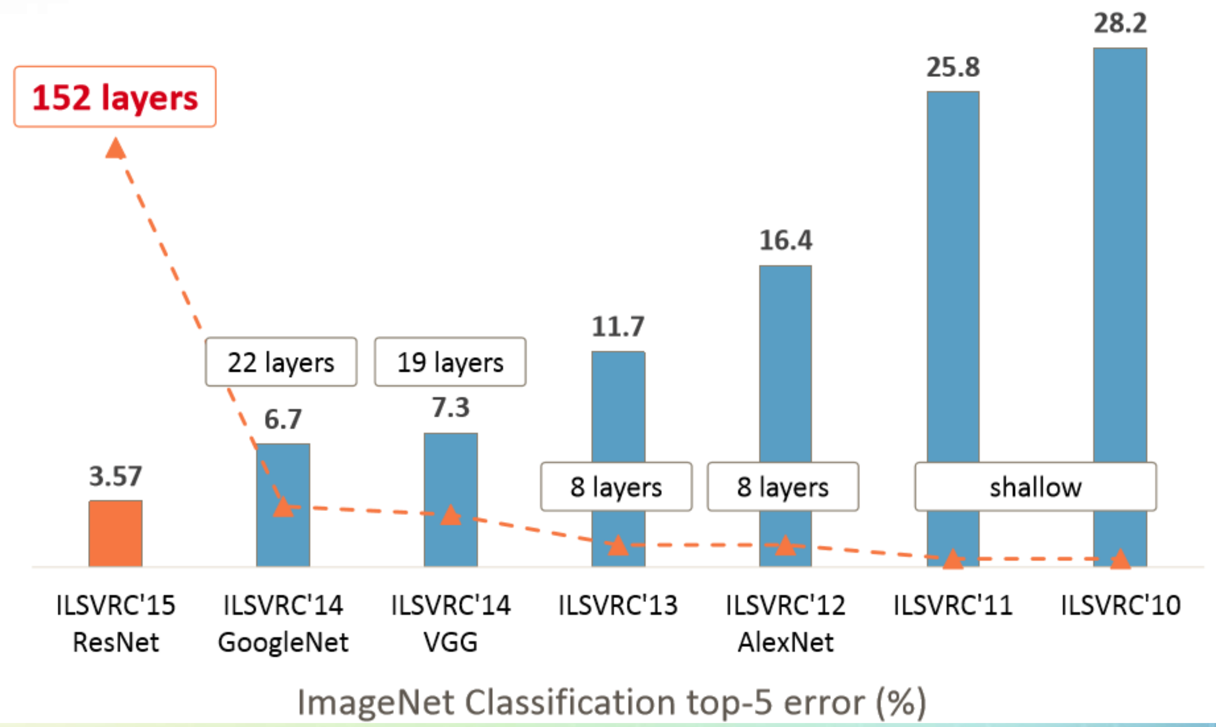


GoogLeNet [Szegedy et al. 2014]

- 22 layers with weights
- Only 5 million parameters (12x fewer than AlexNet)
  - No FC layers
- 1.5 billion operations (2x more than AlexNet)
- Top-5 classification error 6.7% on ImageNet 1K test
  - Winner of ILSVRC 2014

Residual networks (ResNets) [He et al. 2015]

- Extremely deep (152 layers)
- Top-5 classification error 3.6% on ImageNet 1K test
- Winner of all 5 disciplines in ILSVRC & COCO 2015

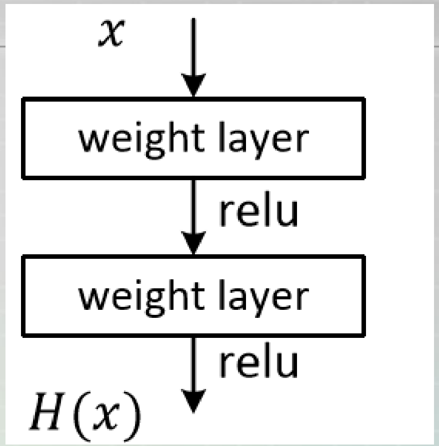


Kaiming He

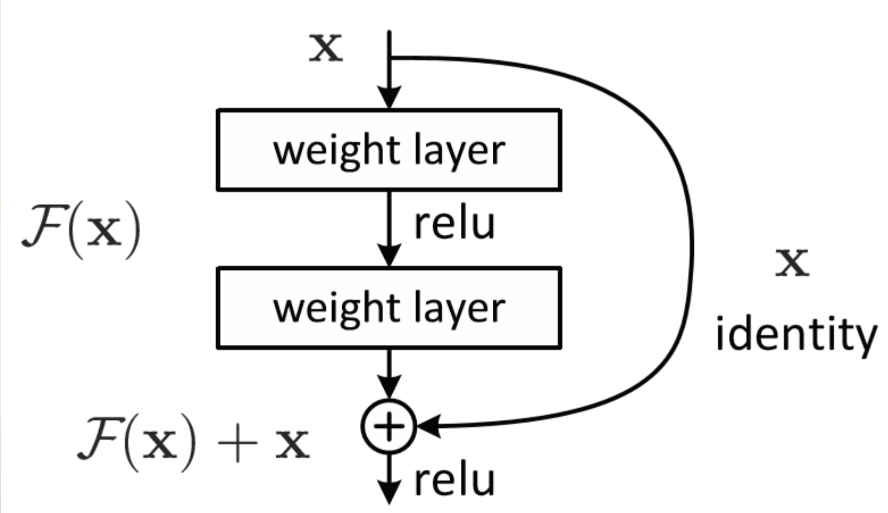
Residual unit

- Small number of convolution layers with ReLU activation
  - Plus normalization layers (not shown)
- Learns difference between its input and target output
- Improves convergence
  - Without residual approach, increasing depth hurts accuracy

Standardna mreža



Rezidualna mreža



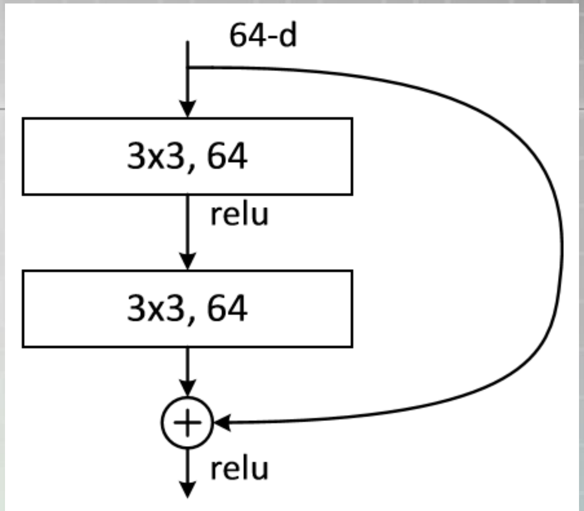
Kaiming He

Microsoft Development Center Serbia

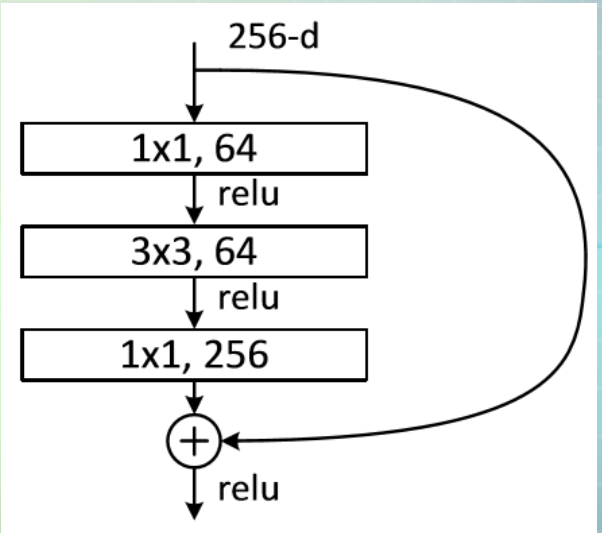
Residual networks with “bottleneck”

- Reduces the number of parameters and operations
- Internal dimension reduction
  - Also used in GoogLeNet
  - Bottleneck units have more channels, but equal complexity as non-bottleneck units

Standardna



Sa „uskim grlom“



Kaiming He

ResNet architectures

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

ResNet properties

- Almost no max-pooling
  - Reducing spatial dimensions is done in convolution layers
- No FC layers
- No dropout
- No local normalization
- Uses batch normalization
  - Further improves convergence

ResNet properties

- Training
  - 120 epochs of ImageNet 1K training (1.3 million images)
  - 2-3 weeks on 8 GPUs (a few days for ResNet-18)
- Even ResNet-152 is slightly faster than VGG-16

Batch normalization [Ioffe and Szegedy 2015]

- Problem: statistics of inputs to a given layer change over time
  - The change depends on weight updates in previous layers
  - Changes are more severe in deeper layers
  - This limits depth of networks that can be trained

$$y_c = \alpha_c \frac{x_c - \mathrm{E}[x_c]}{\sqrt{\mathrm{Var}[x_c]}} + \beta_c$$

Trained additive/multiplicative constants
(one value per channel)

All activations in channel c
(minibatch size x width x height)

Batch normalization

- Reduces dependence on initial weights
- Allows higher learning rate values
- Has regularization effect
  - Samples within the same minibatch influence each other
  - Adds "noise" coming from other samples
  - Reduces need for dropout and other normalizations

References

- CS231n Winter 2016
- Convolutional Neural Networks (2017)
- Coursera: Convolutional Neural Networks
- Coursera: Improving Deep Neural Networks: Hyperparameter tuning, Regularization and Optimization