Convolutional Neural Networks (CNNs)

Pattern Recognition

Dennis Madsen

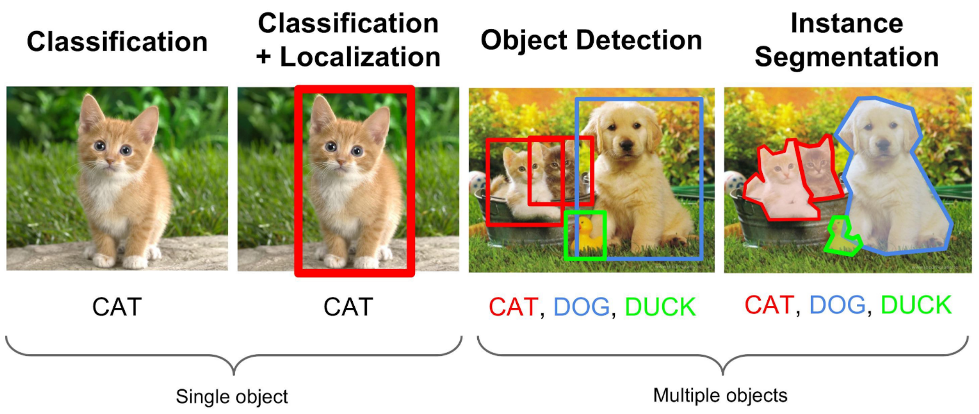
Topic overview

- *Neural Networks and Deep Learning*
- *Improving DNN: Hyperparameter tuning, regularization, optimization*
- **Convolutional Neural Networks (CNN)**
- **CNN popular architectures**
- Sequence Models/Recurrent neural networks (RNN)
- Beyond the basics (object detection and segmentation)

Topic overview

- **Convolutional Neural Networks (CNN)**
  - Convolutions and cross correlation
  - Image filtering examples
  - Dimensionality reduction - Pooling
  - Weight visualization
  - Convolutional neural networks
- **CNN popular architectures**
  - LeNet, AlexNet, ResNet, InceptionNet, U-Net

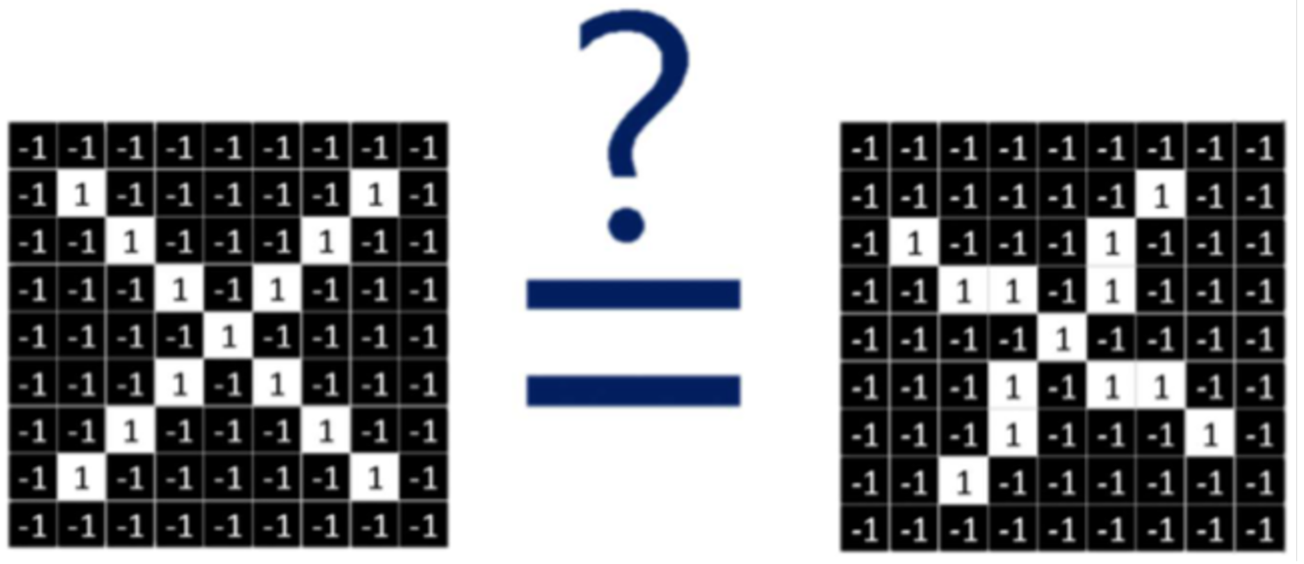
Computer Vision Problems



Neural style transfer

Motivation

- An image is represented as a matrix of pixel values.
- We want the classification to be invariant to: rotation, shift, deformation, scaling



-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1

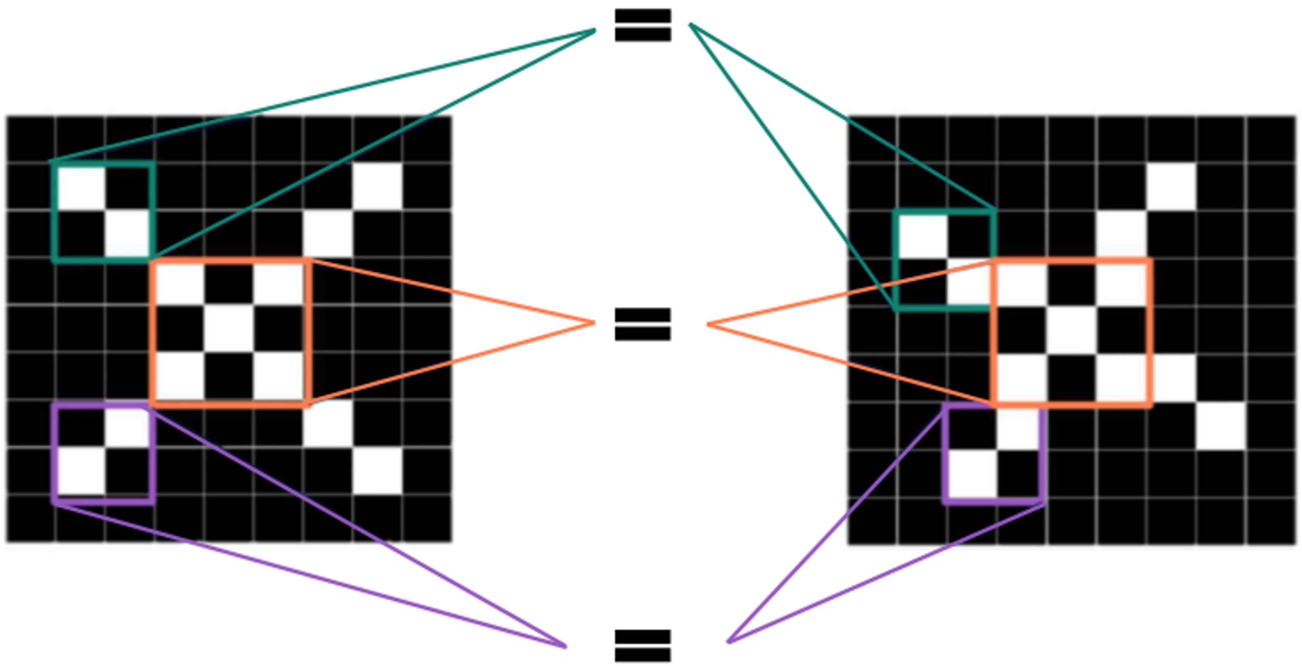
$\stackrel{?}{=}$

-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	1	-1	-1
-1	1	-1	-1	-1	1	-1	-1	-1
-1	-1	1	1	-1	1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1
-1	-1	-1	1	-1	1	1	-1	-1
-1	-1	-1	1	-1	-1	-1	1	-1
-1	-1	1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1

Source: http://introtodeeplearning.com/

Motivation

- Compare image features found in the images



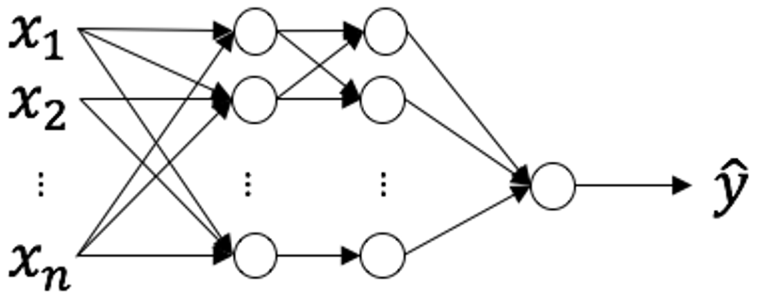
Source: http://introtodeeplearning.com/

Problem with a fully connected network

Image of dimensions 1000x1000x3 has 3 million feature inputs to a network.

$$x \in \mathbb{R}^{3M}$$

$$W^{(1)}\ (1000, 3M) \ = 3 \text{ billion parameters}$$

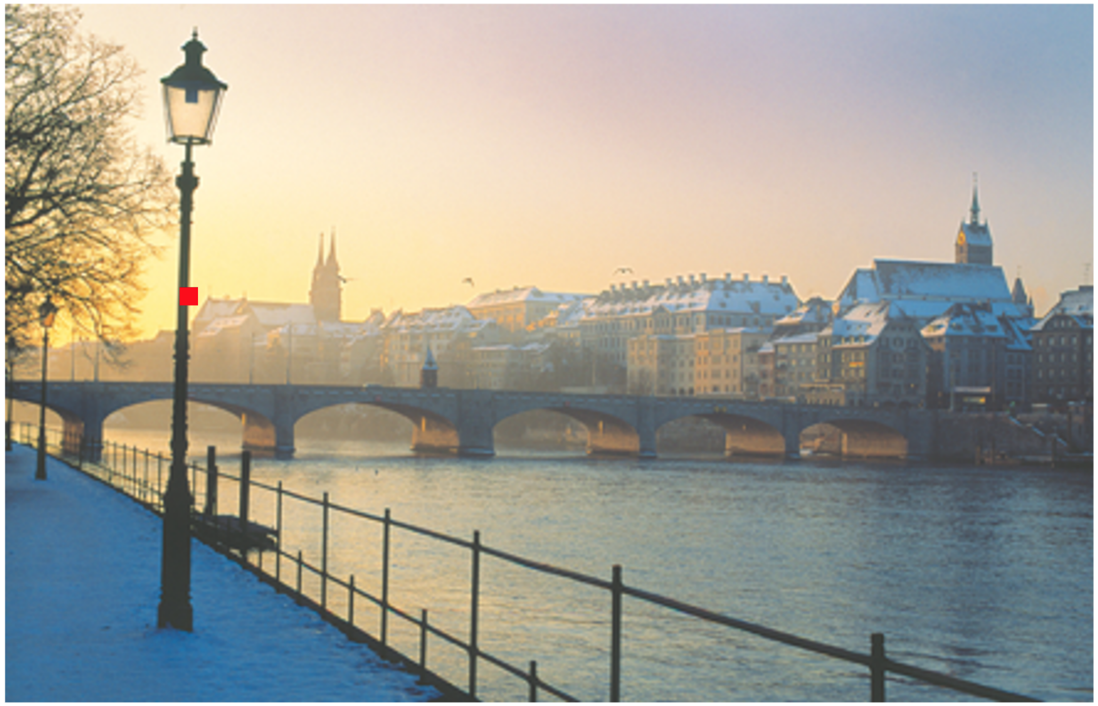


With 1000 neurons in the first hidden layer, we will already have 3 billion parameters to update.

Source: https://www.coursera.org/learn/convolutional-neural-networks/

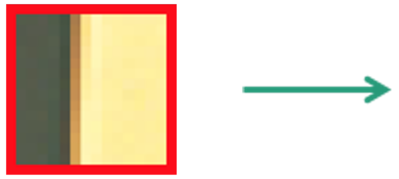
Fundamental Properties of Images

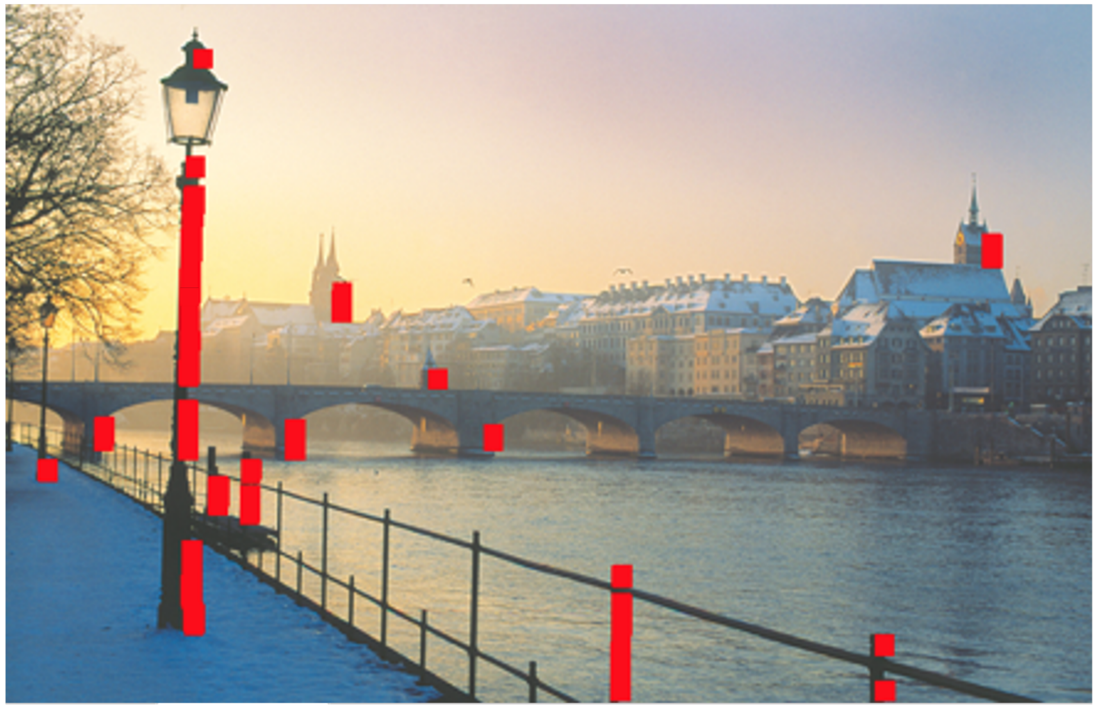
Property 1: Image statistics are locally correlated/structured.



Fundamental Properties of Images

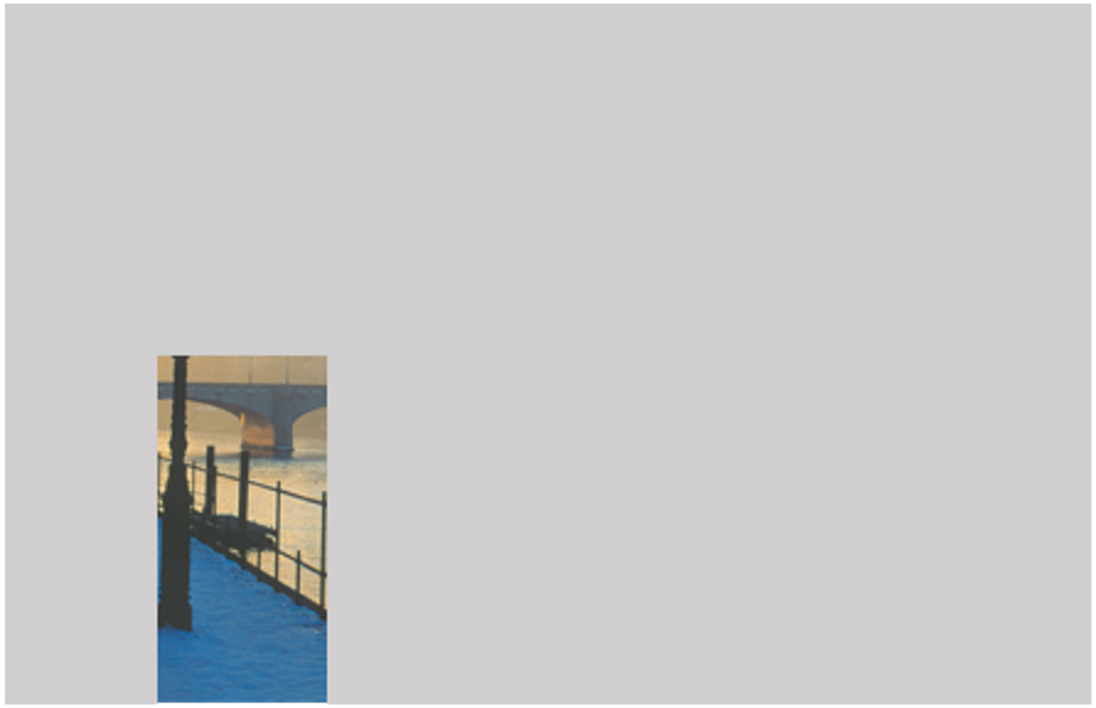
Property 2: Redundancy of structures.





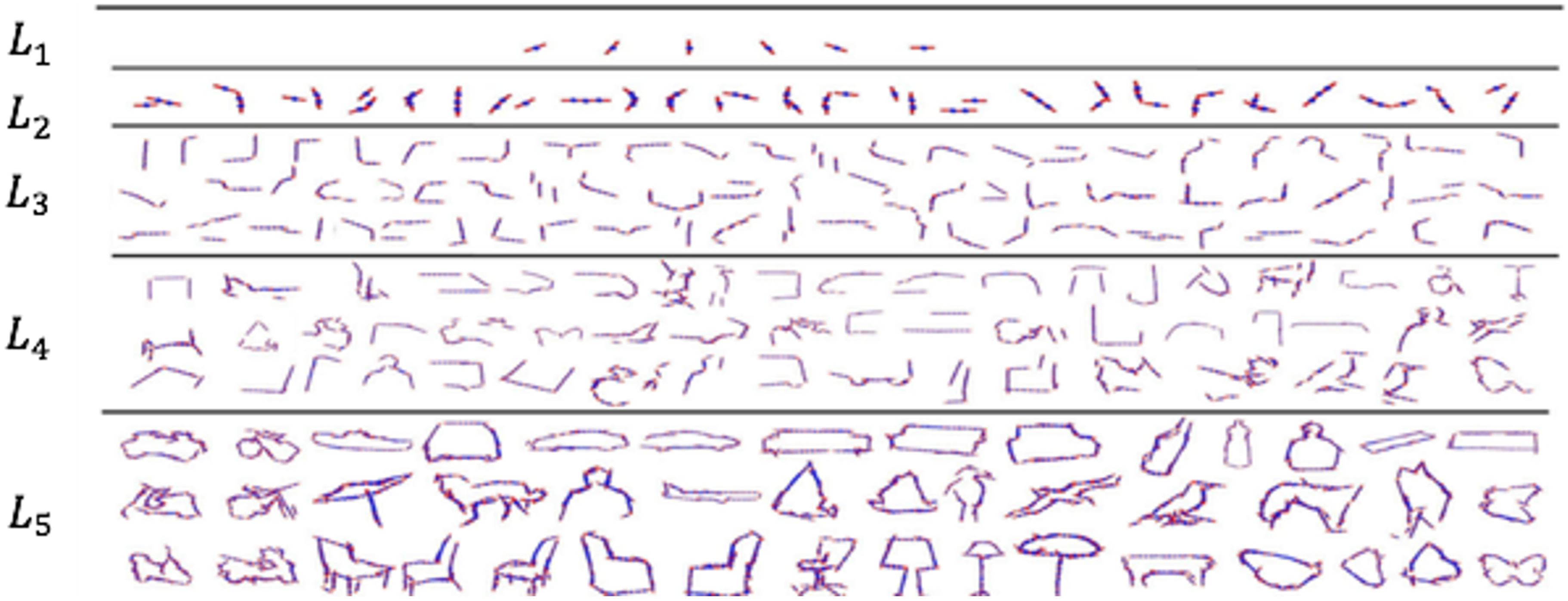
Fundamental Properties of Images

Property 3: Global Correlation.



Fundamental Properties of Images

Property 4: Compositionality of objects - a small set of building blocks (L1) is enough to build complex objects (L5) vis recursive composition.



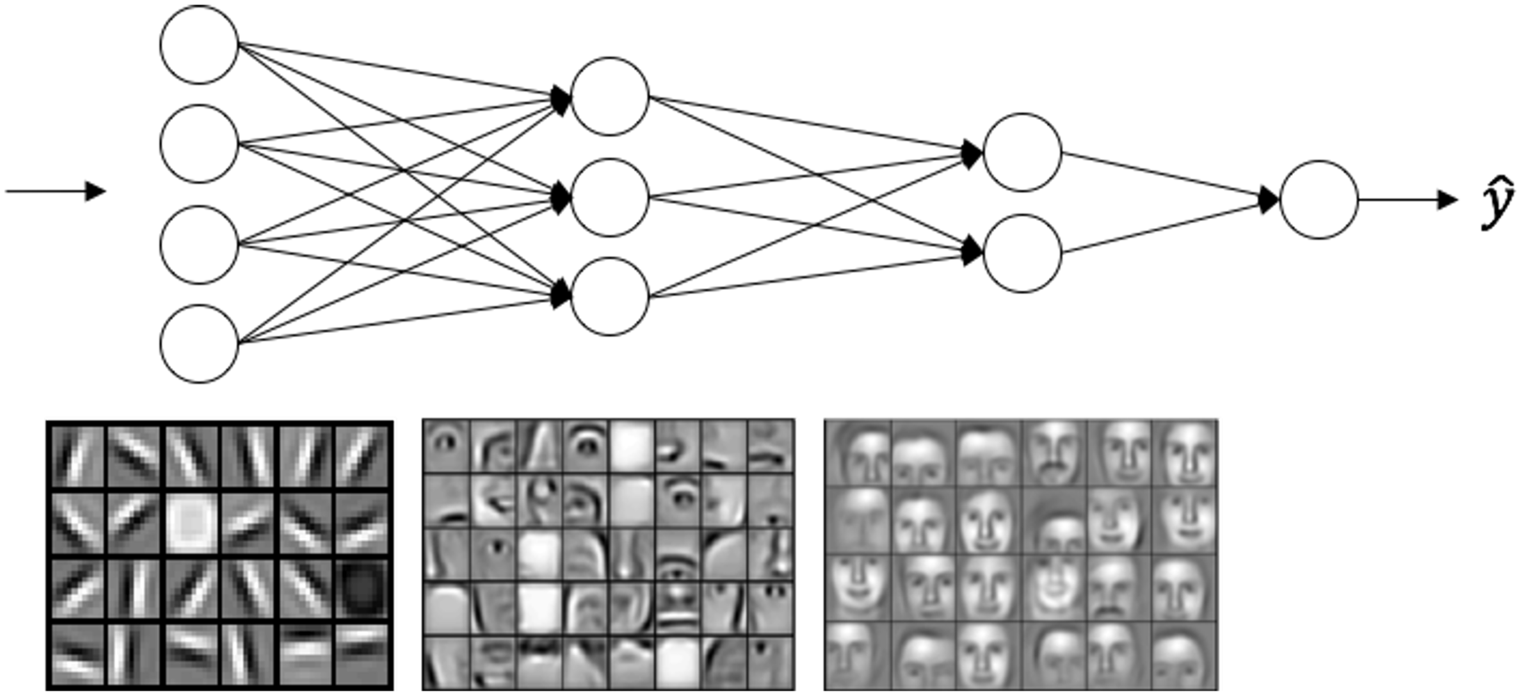
Convolutional Neural Networks

- Key Idea: *Constrain* the networks *architecture* to reduce the amount of network parameters.

- The network is constrained such that:
  - Hidden units are locally connected
  - Weights shared among hidden units
  - Hidden layers are subsampled

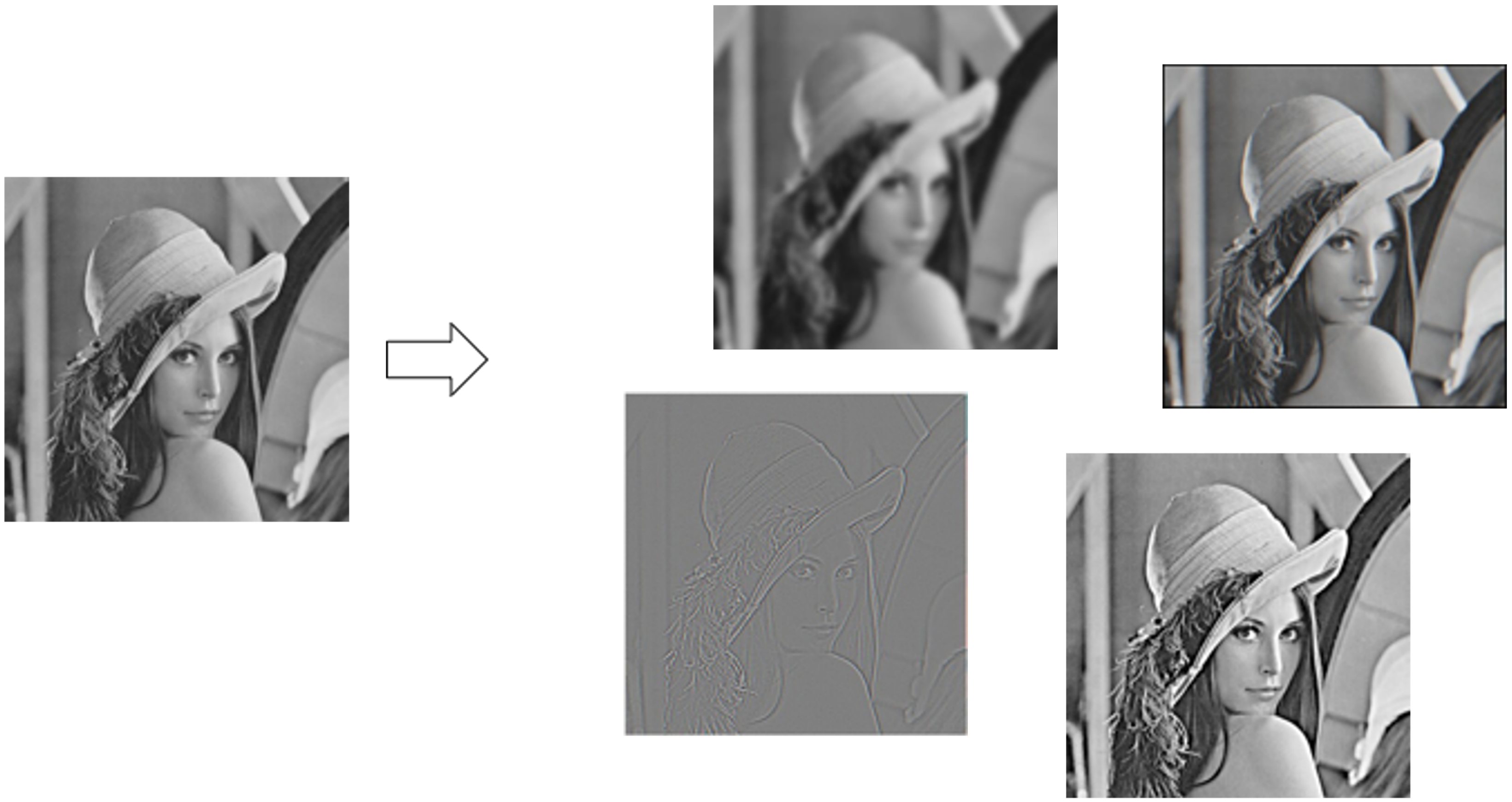
- These changes to the network architecture reflect properties which are specific to images.

Intuition about deep representation



Source: https://www.coursera.org/learn/neural-networks-deep-learning

Image Filtering (linear)

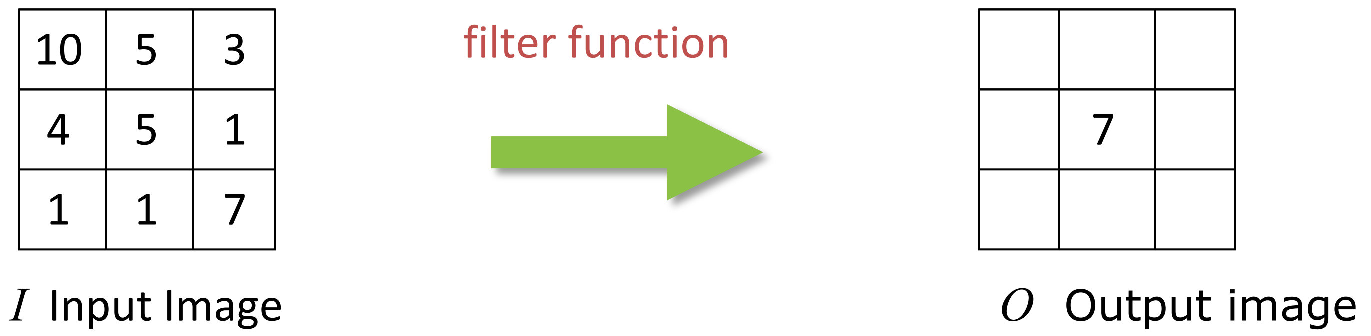


For more details see: Digital Image Processing *by R.C. Gonzales & R.E. Woods*

Image Filtering (linear)

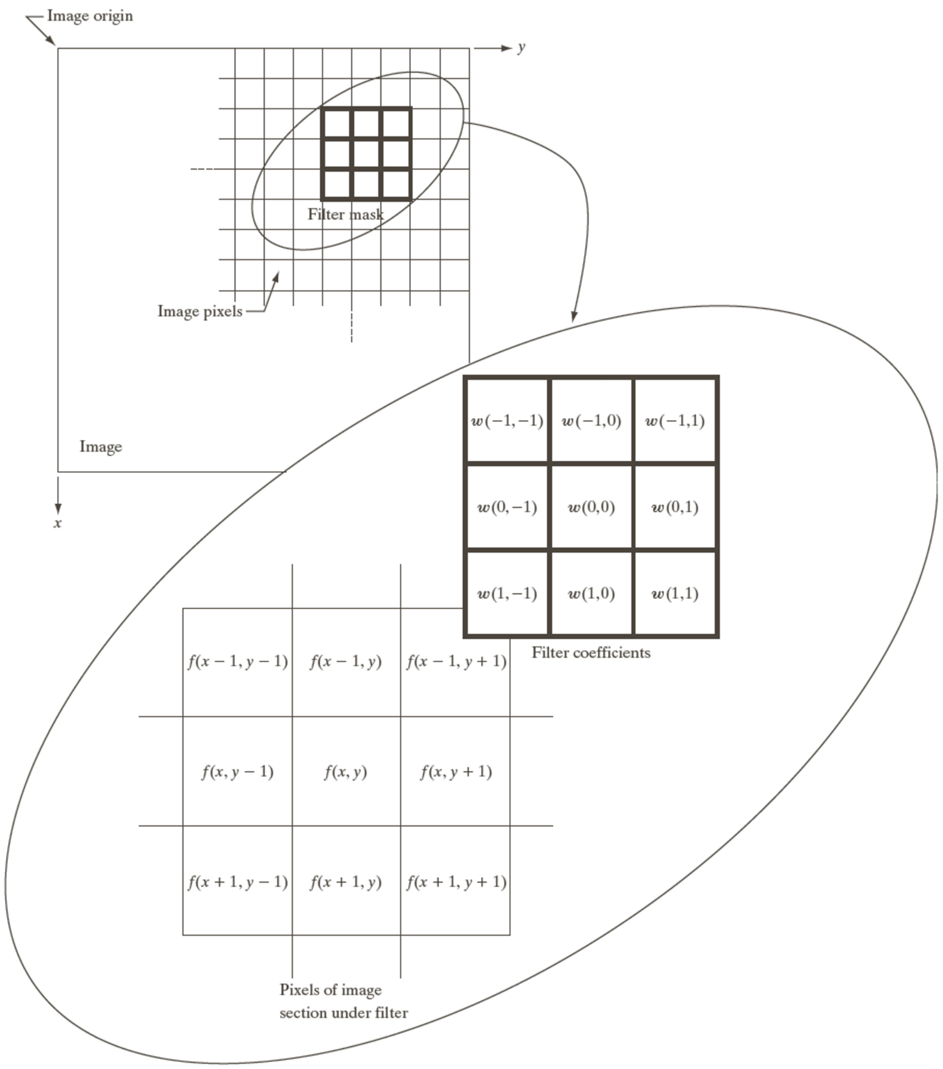
- Each novel output pixel value $O(x,y)$ is as linear function of the neighboring pixel values of $I(x,y)$.
  The linear weights are stored in the filter kernel $K(s,t)$ (also called filter or filter mask)

$$O[x,y] = \sum_{s=-a}^{a} \sum_{t=-b}^{b} k[s,t] I[x+s, y+t]$$



For more details see: Digital Image Processing *by R.C. Gonzales & R.E. Woods*

Spatial Filtering



For more details see: Digital Image Processing *by R.C. Gonzales & R.E. Woods*

Linear Filtering as correlation or convolution

- Cross-correlation: $$O[x,y]=\sum_{s=-a}^{a}\sum_{t=-b}^{b}k[s,t]I[x+s,y+t]$$

  Symbol: $O=k\otimes I$

- Convolution: $$O[x,y]=\sum_{s=-a}^{a}\sum_{t=-b}^{b}k[s,t]I[x-s,y-t]$$

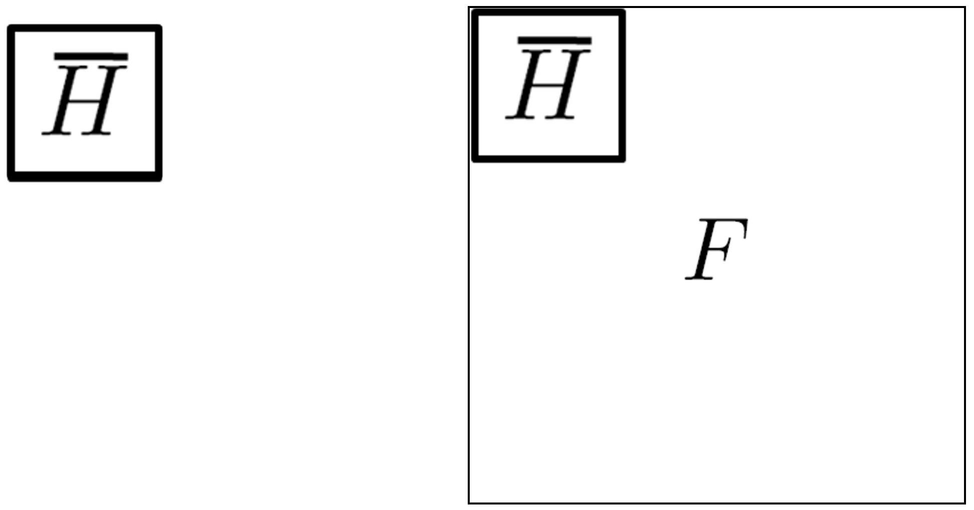
  Symbol: $O=k*I$

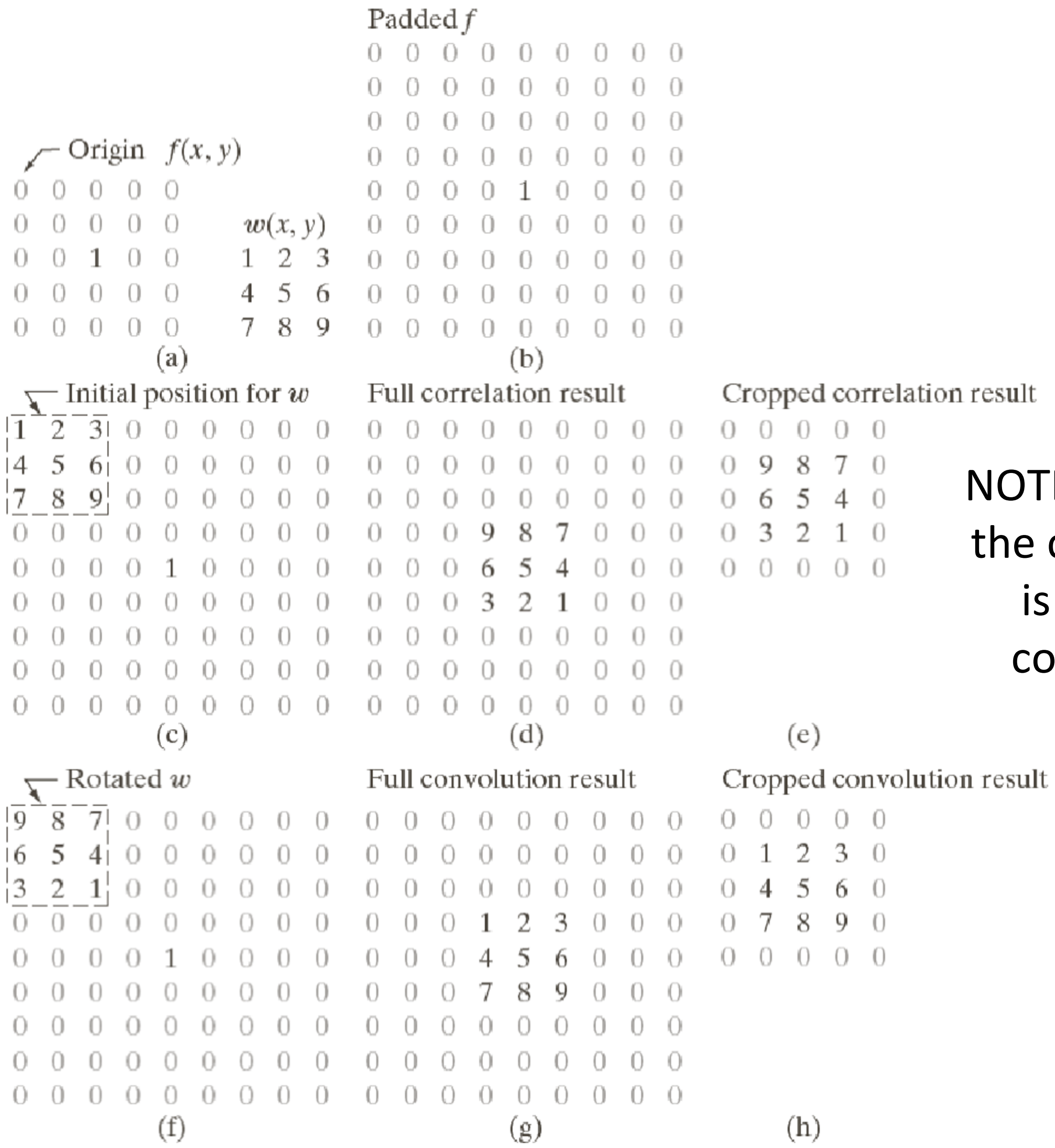
  Convolution is **commutative** and **associative**

For symmetric kernels there is no difference !!!

For more details see: Digital Image Processing *by R.C. Gonzales & R.E. Woods*

Convolution

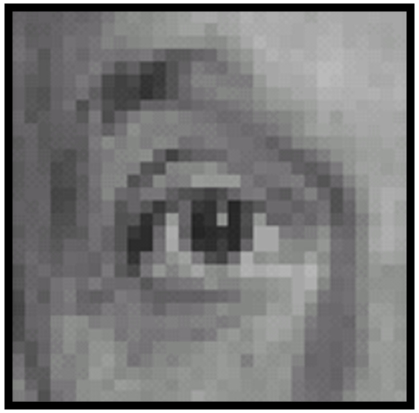




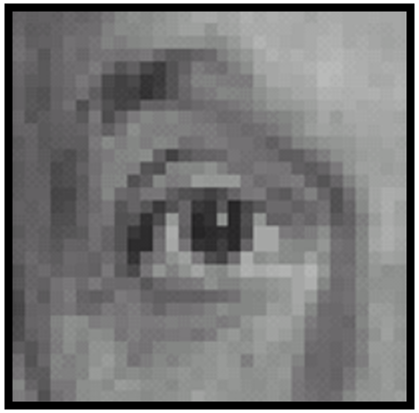
NOTE: In Neural Networks, the convolution operation is technically a cross-correlation operation.

For more details see: Digital Image Processing *by R.C. Gonzales & R.E. Woods*

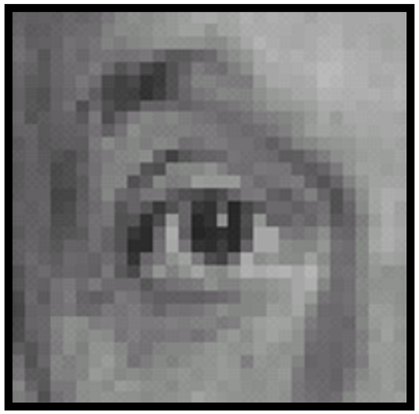
Linear filters: examples


Original

$$* \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} =$$

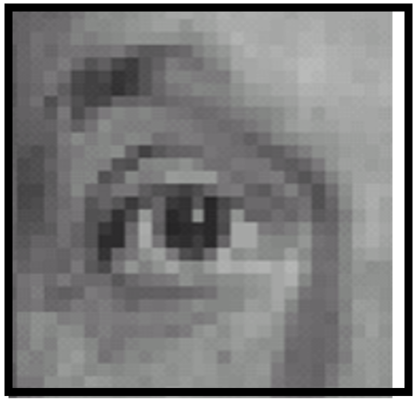

Identical image

Linear filters: examples



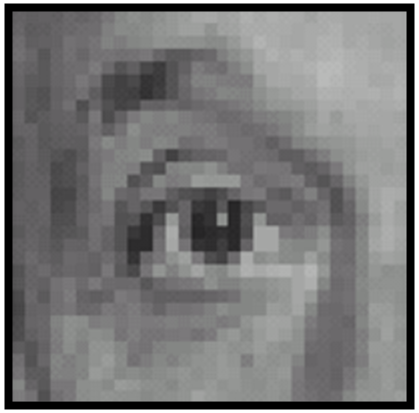
Original

$$* \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} =$$

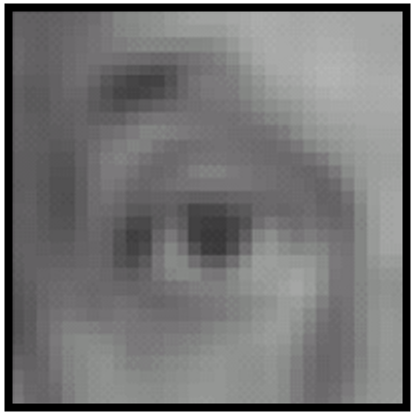


Shifted left
By 1 pixel

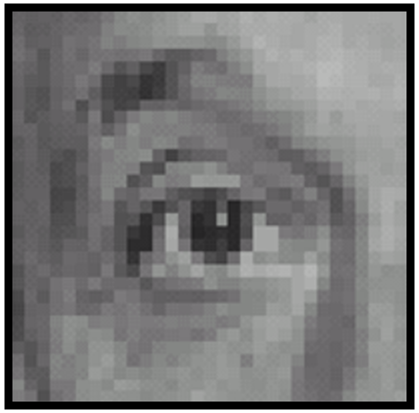
Linear filters: examples


Original

$$* \quad \frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix} =$$

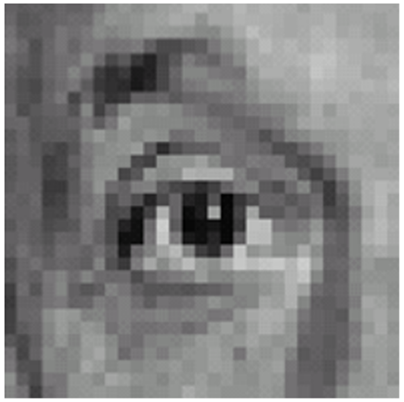

Blur (with a mean filter)

Linear filters: examples



Original

$$* \left(\begin{array}{|c|c|c|} \hline 0 & 0 & 0 \\ \hline 0 & 2 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array} - \frac{1}{9} \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array} \right) =$$



Sharpening filter
(accentuates edges)

Padding

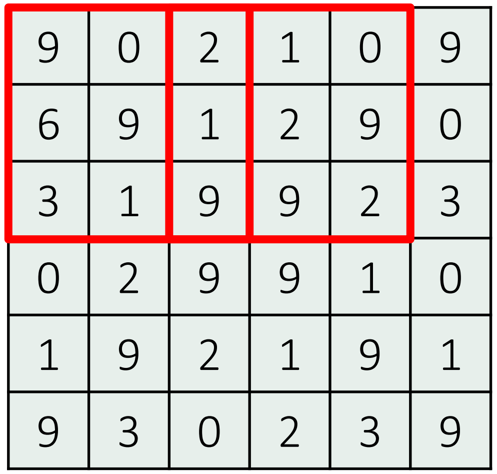
The image input and output from a convolution is not of the same dimensions. To account for this, we can add padding to the input.

- "Valid" padding: No padding
  - Output image dimensions: $(n \times n) \ast (f \times f) \longrightarrow (n - f + 1) \times (n - f + 1)$
- "Same" padding: Pad so output size is the same as the input.
  - Padding amount to input image: $p = \frac{f-1}{2}$

Stride

- Stride decides the sliding amount:
  - Padding: *p*
  - Stride: *s*
  - Output with padding and stride:

$$(n \times n) \ * (f \times f) \ \longrightarrow (\tfrac{n+2p-f}{s} + 1) \times (\tfrac{n+2p-f}{s} + 1)$$

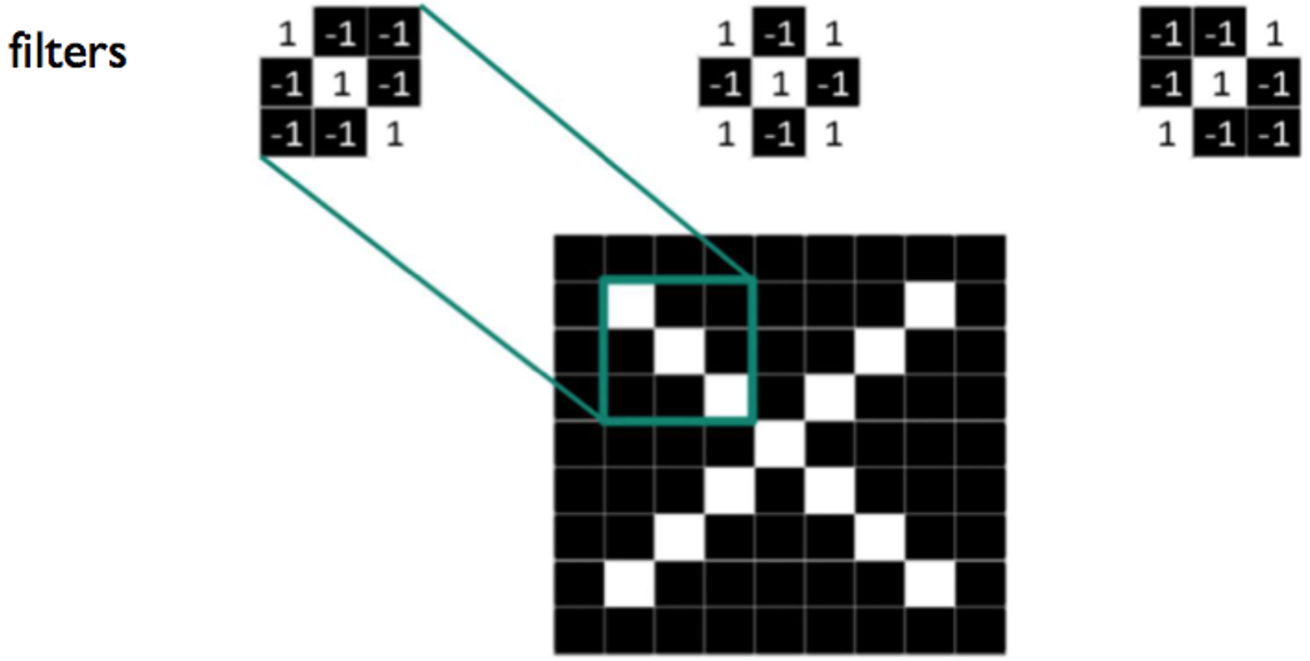


9	0	2	1	0	9
6	9	1	2	9	0
3	1	9	9	2	3
0	2	9	9	1	0
1	9	2	1	9	1
9	3	0	2	3	9

Stride = 2

Motivation

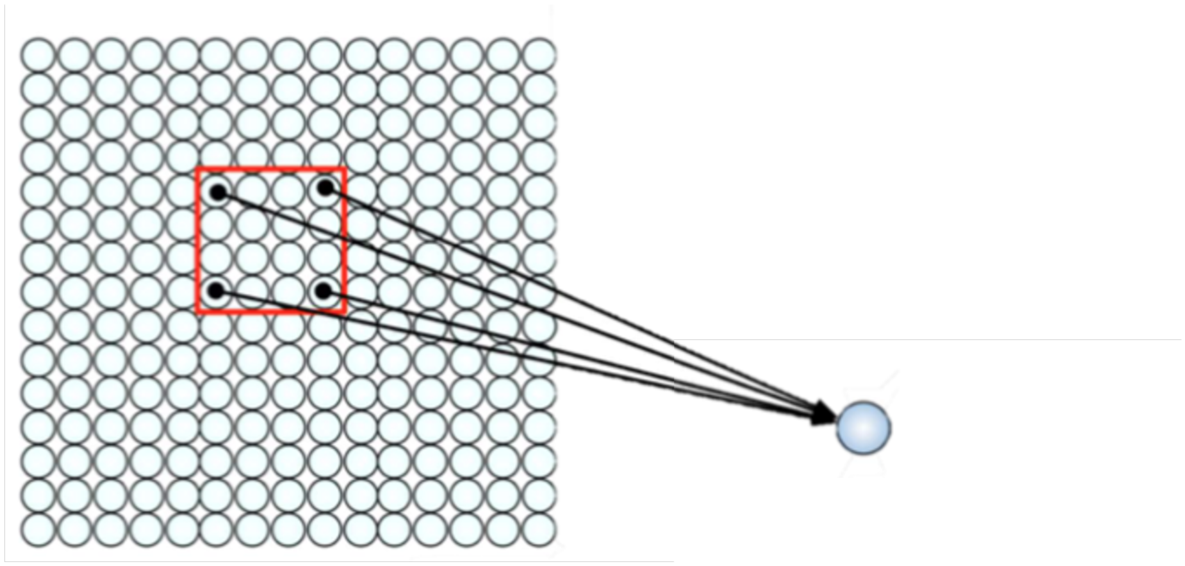
- Compare image features found in the images



Source: http://introtodeeplearning.com/

Use Spatial Structure

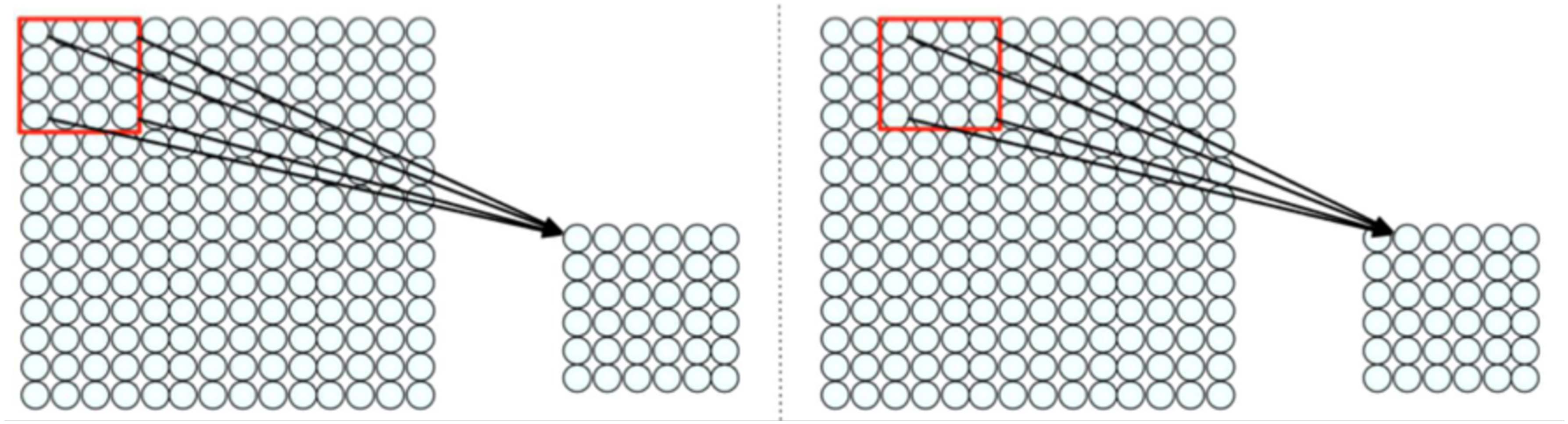
- Connect patches of input to neurons in hidden layer.
  - Neuron connected to a region of the input only "sees" this area.



Source: http://introtodeeplearning.com/

Use Spatial Structure

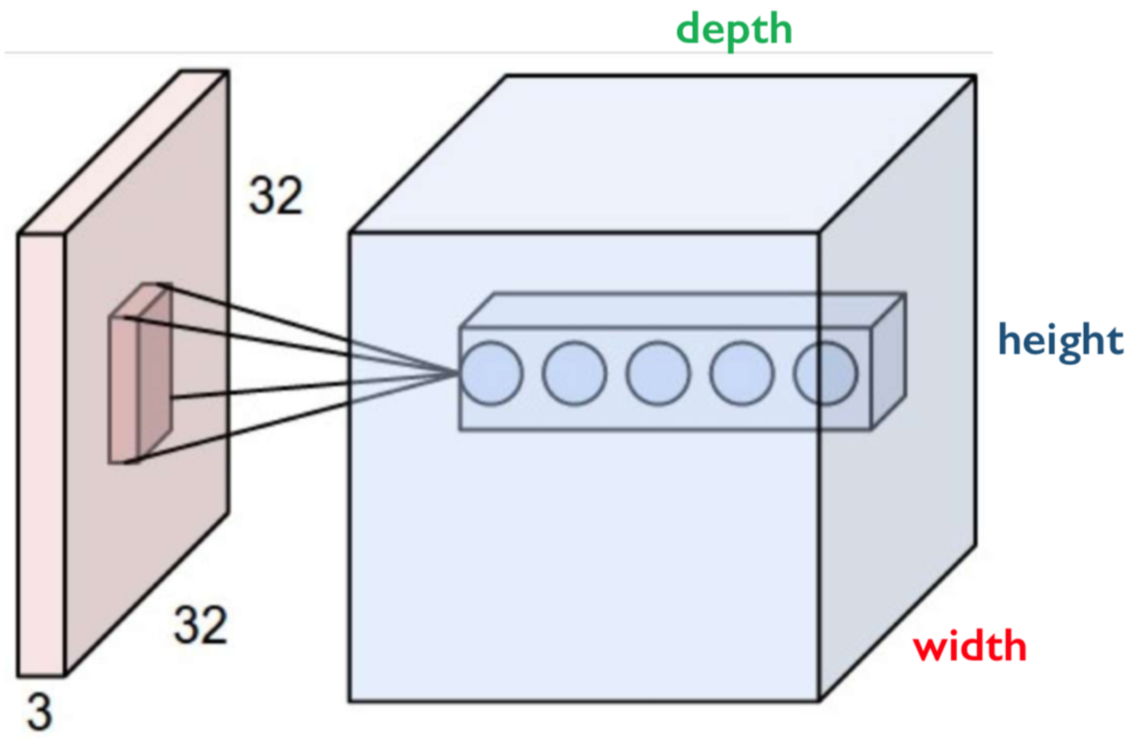
- Connect patches of input to neurons in hidden layer.
  - Neuron connected to a region of the input only "sees" this area.
  - Configure the network as in a *sliding window* approach.



- Use multiple filters to extract different features.
- Spatially share the parameters of each filter.

Source: http://introtodeeplearning.com/

Spatial Arrangement of Output Volume



Layer Dimensions:

$h \; x \; w \; x \; d$

where h and w are spatial dimensions
d (depth) = number of filters

Stride:

Filter step size

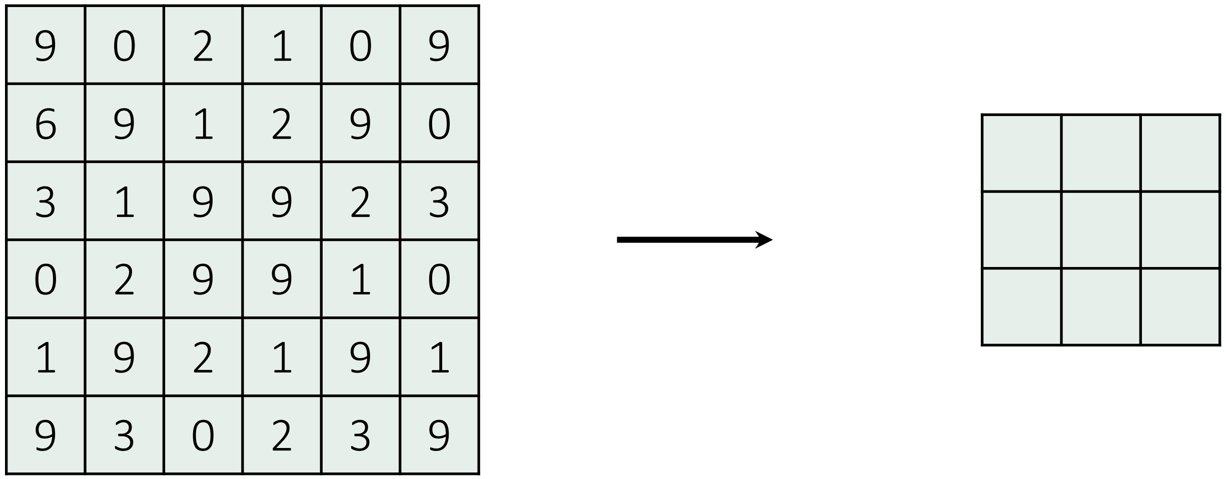
Receptive Field:

Locations in input image that
a node is path connected to

Source: http://introtodeeplearning.com/

Pooling

- Dimensionality reduction method/down-sampling process, that locally pools feature responses together



9	0	2	1	0	9
6	9	1	2	9	0
3	1	9	9	2	3
0	2	9	9	1	0
1	9	2	1	9	1
9	3	0	2	3	9

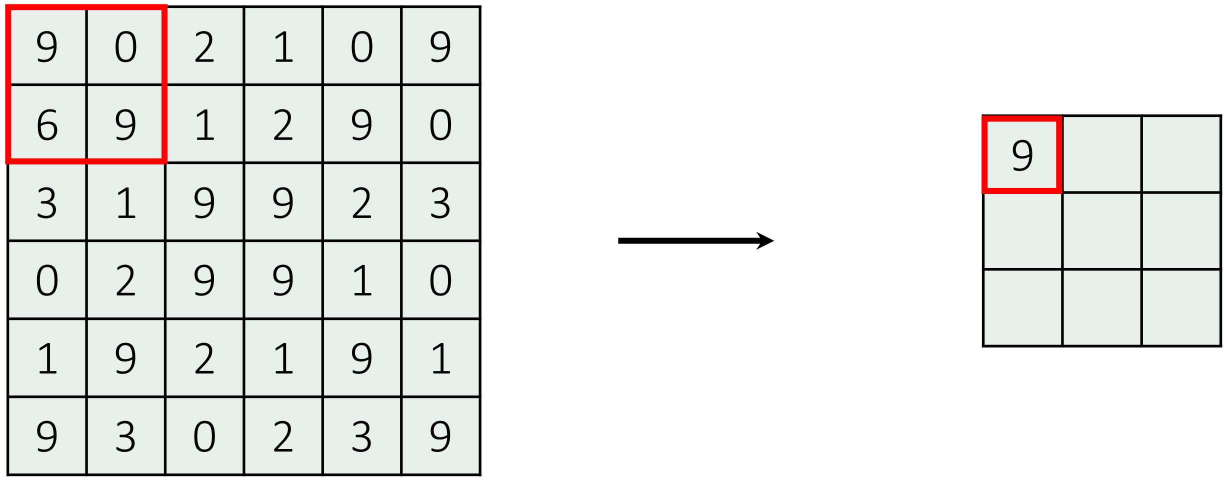
Max Pooling

- Max pooling example for down-sampling. Locally pools maximum feature responses together.

$$\begin{array}{|c|c|c|c|c|c|}\hline 9 & 0 & 2 & 1 & 0 & 9 \\ \hline 6 & 9 & 1 & 2 & 9 & 0 \\ \hline 3 & 1 & 9 & 9 & 2 & 3 \\ \hline 0 & 2 & 9 & 9 & 1 & 0 \\ \hline 1 & 9 & 2 & 1 & 9 & 1 \\ \hline 9 & 3 & 0 & 2 & 3 & 9 \\ \hline \end{array} \longrightarrow \begin{array}{|c|c|c|}\hline & & \\ \hline & & \\ \hline & & \\ \hline \end{array}$$

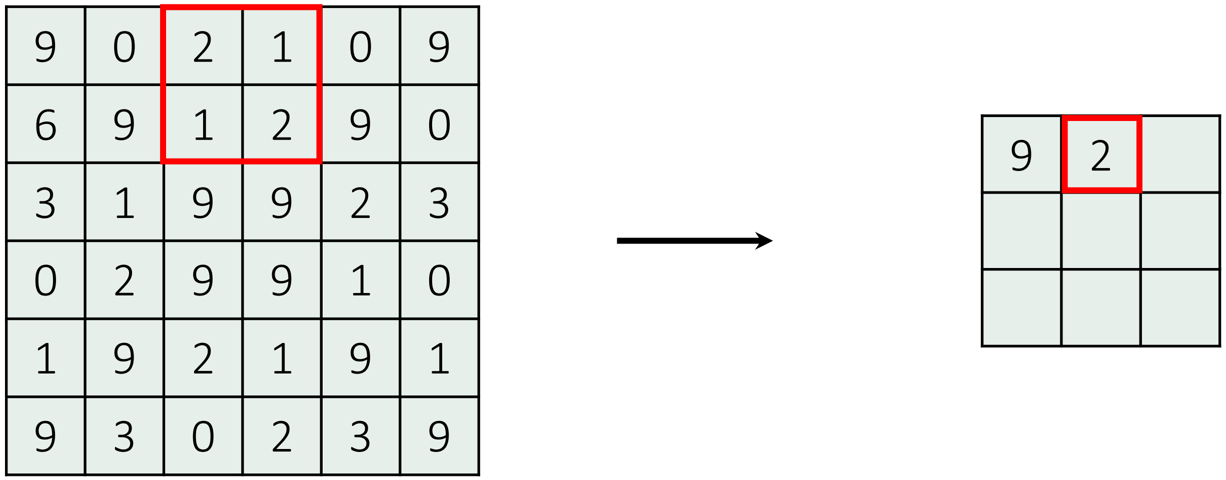
Max Pooling

- Max pooling example for down-sampling. Locally pools maximum feature responses together.



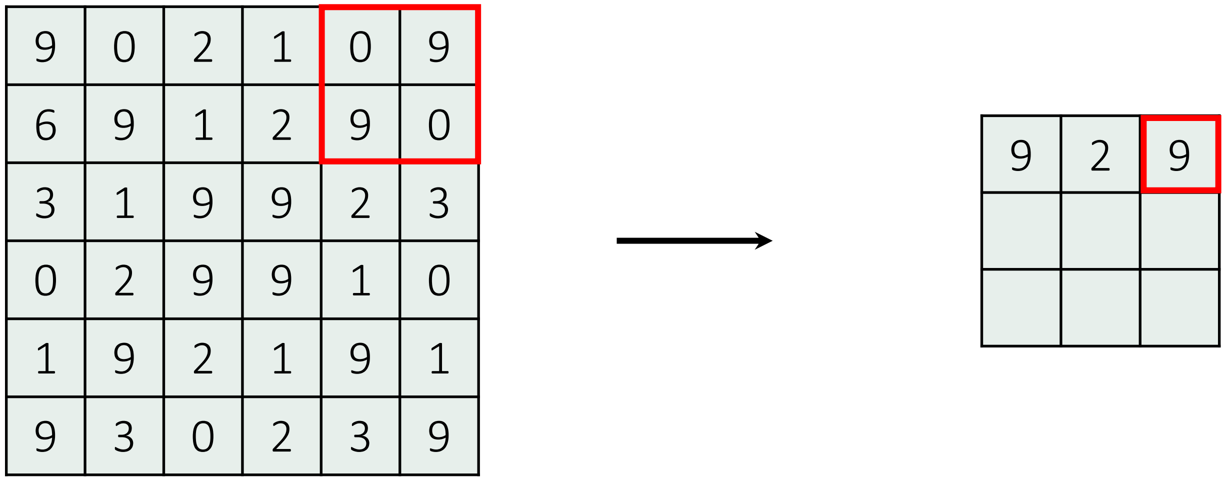
Max Pooling

- Max pooling example for down-sampling. Locally pools maximum feature responses together.



Max Pooling

- Max pooling example for down-sampling. Locally pools maximum feature responses together.

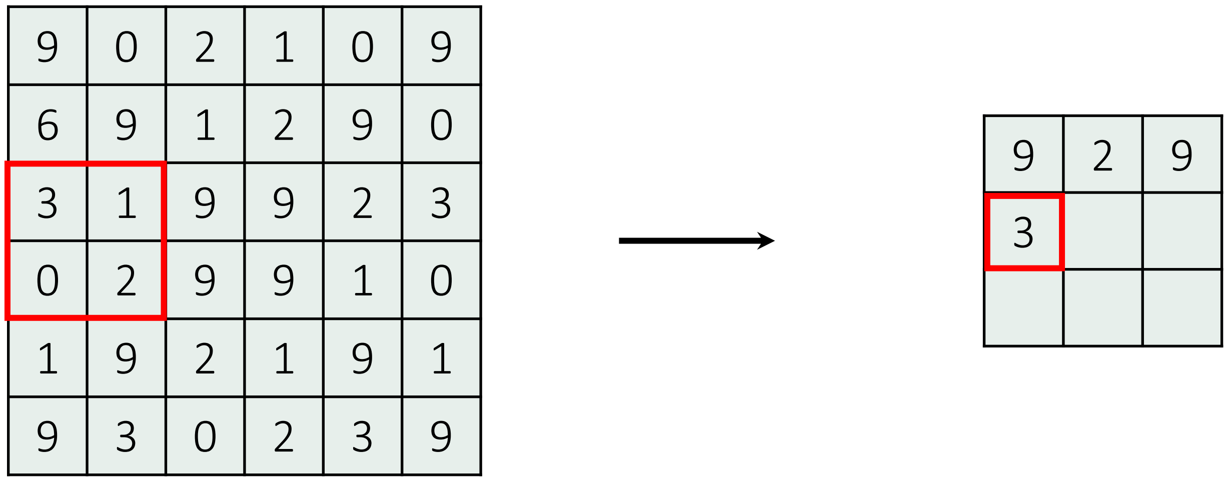


9	0	2	1	0	9
6	9	1	2	9	0
3	1	9	9	2	3
0	2	9	9	1	0
1	9	2	1	9	1
9	3	0	2	3	9

9	2	9

Max Pooling

- Max pooling example for down-sampling. Locally pools maximum feature responses together.

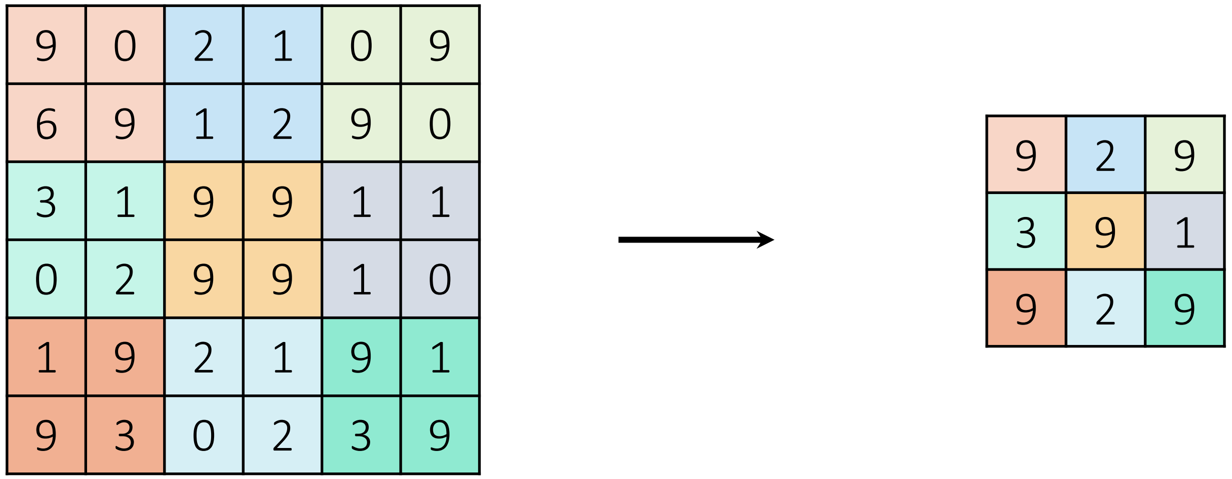


9	0	2	1	0	9
6	9	1	2	9	0
3	1	9	9	2	3
0	2	9	9	1	0
1	9	2	1	9	1
9	3	0	2	3	9

9	2	9
3		

Max Pooling

- Max pooling example for down-sampling. Locally pools maximum feature responses together.



9	0	2	1	0	9
6	9	1	2	9	0
3	1	9	9	1	1
0	2	9	9	1	0
1	9	2	1	9	1
9	3	0	2	3	9

$\longrightarrow$

9	2	9
3	9	1
9	2	9

Max Pooling

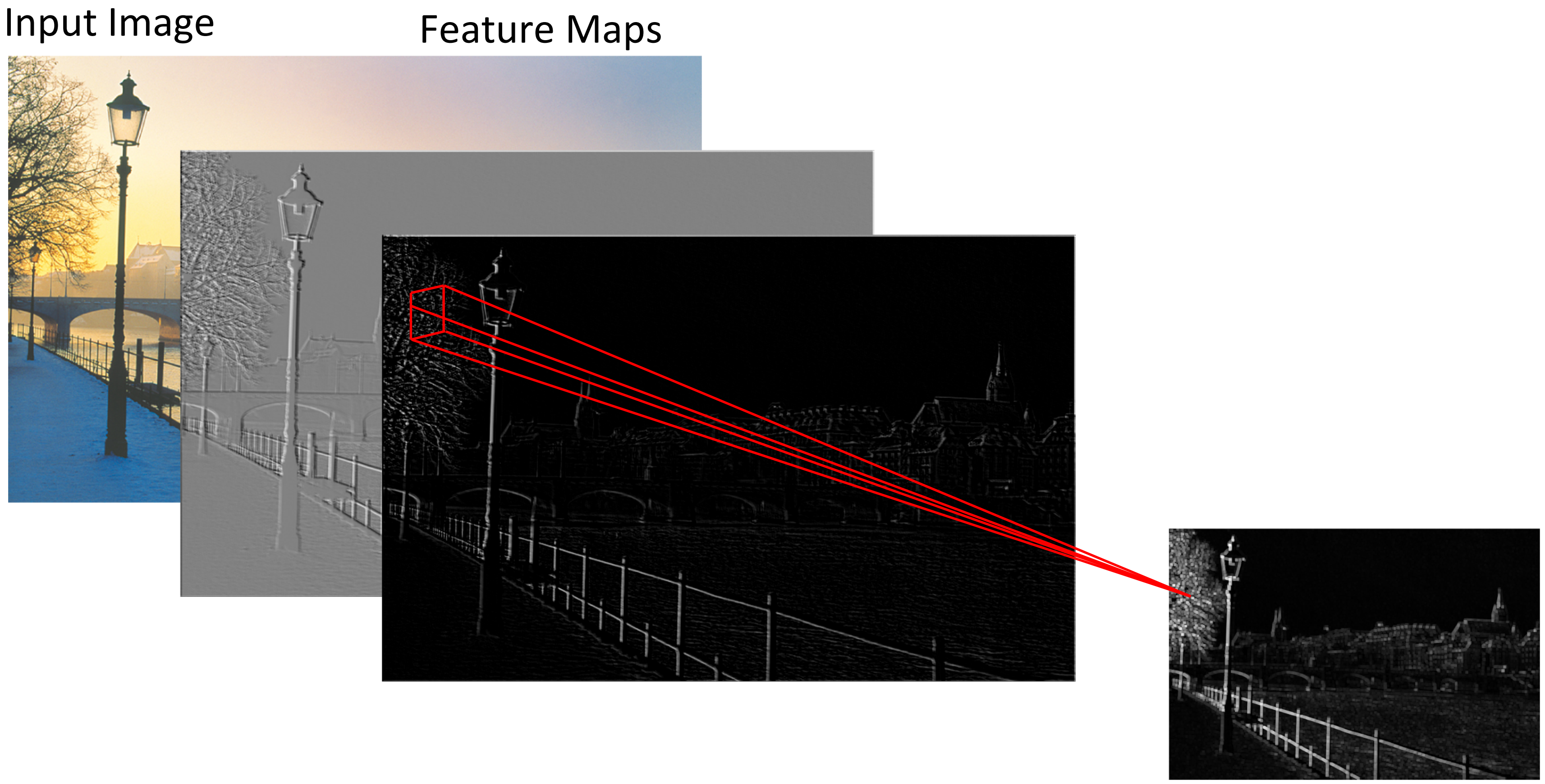
- Max pooling is a down-sampling process, that locally pools feature responses together. Its main benefits are:
    1. Dimensionality reduction
        - Reduces the number of parameters
        - Simplifies discovery of global patterns
    2. Invariance to small changes of the input signal

$$\begin{bmatrix} 9 & 0 & 2 & 1 & 0 & 9 \\ 6 & 9 & 1 & 2 & 9 & 0 \\ 3 & 1 & 9 & 9 & 1 & 1 \\ 0 & 2 & 9 & 9 & 1 & 0 \\ 1 & 9 & 2 & 1 & 9 & 1 \\ 9 & 3 & 0 & 2 & 3 & 9 \end{bmatrix} \longrightarrow \begin{bmatrix} 9 & 2 & 9 \\ 3 & 9 & 1 \\ 9 & 2 & 9 \end{bmatrix}$$

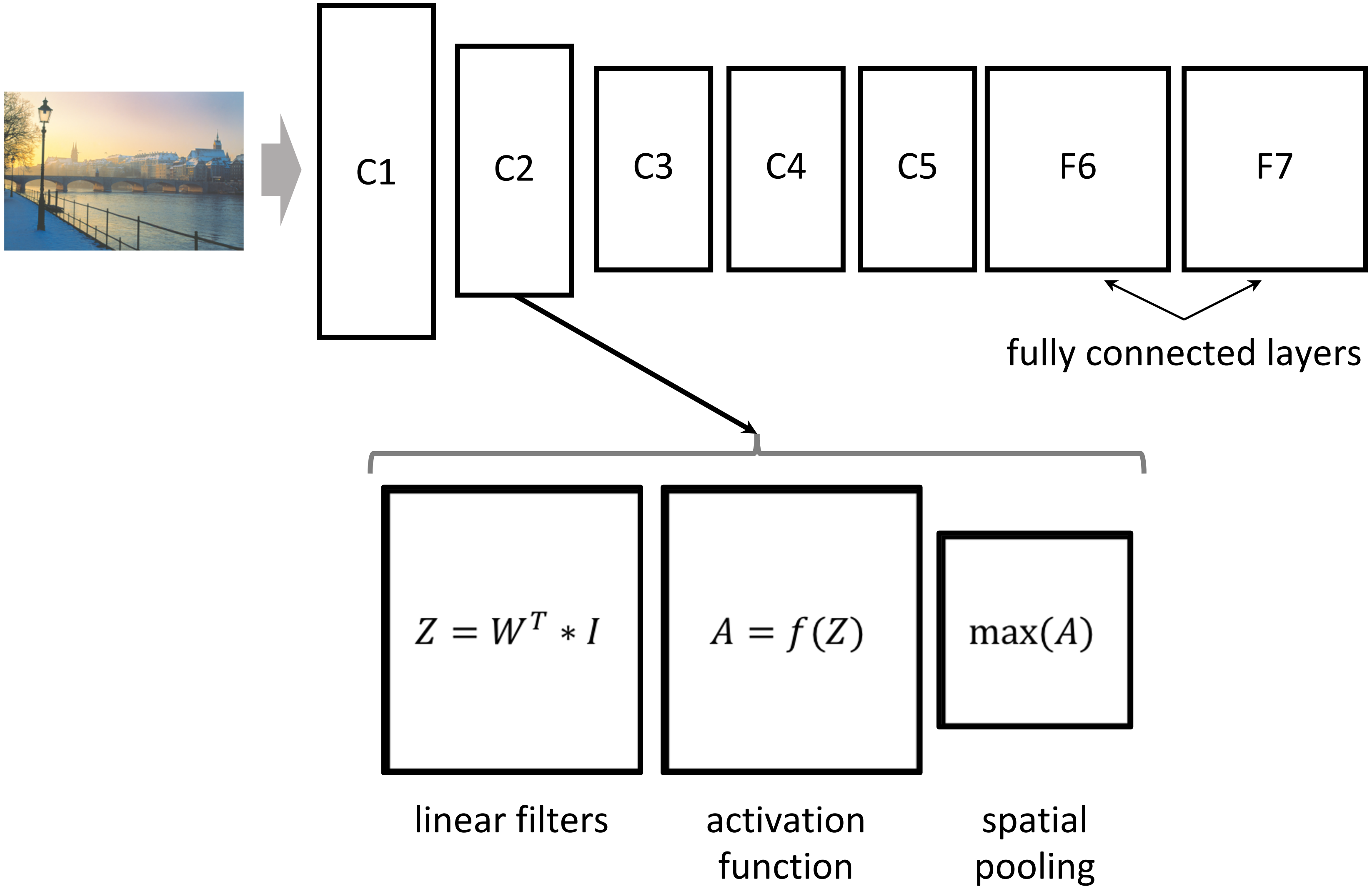
Other used pooling strategies:

- Average Pooling

Pooling Layer

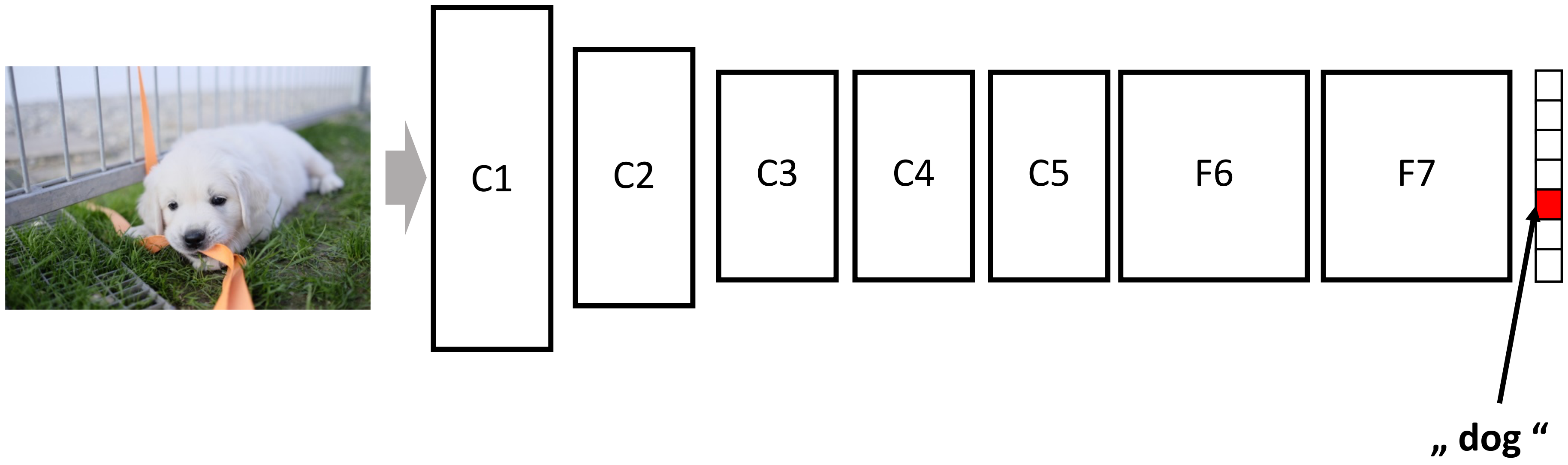


Layered CNN Architecture



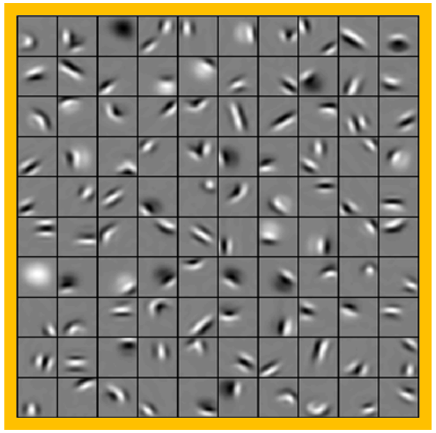
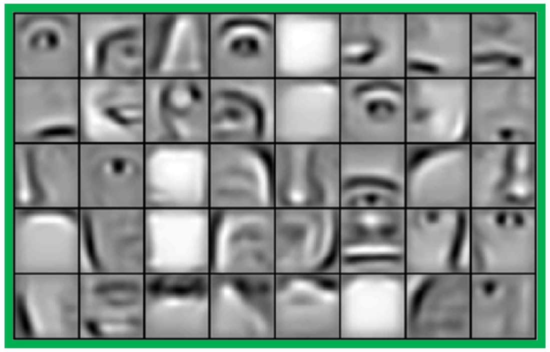
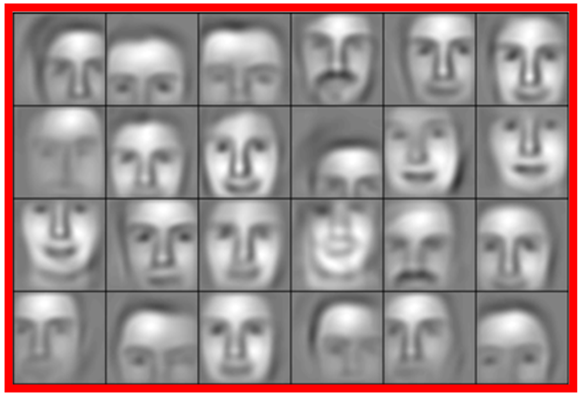
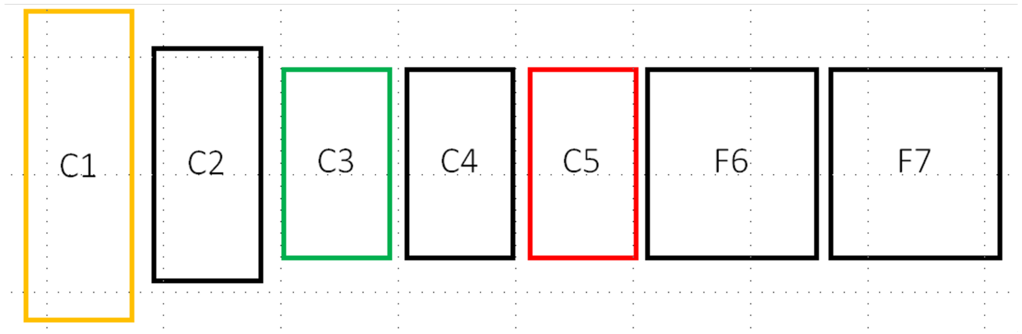
Classification

- Add an output layer and train the weights with backpropagation



Visualization of the learned weights

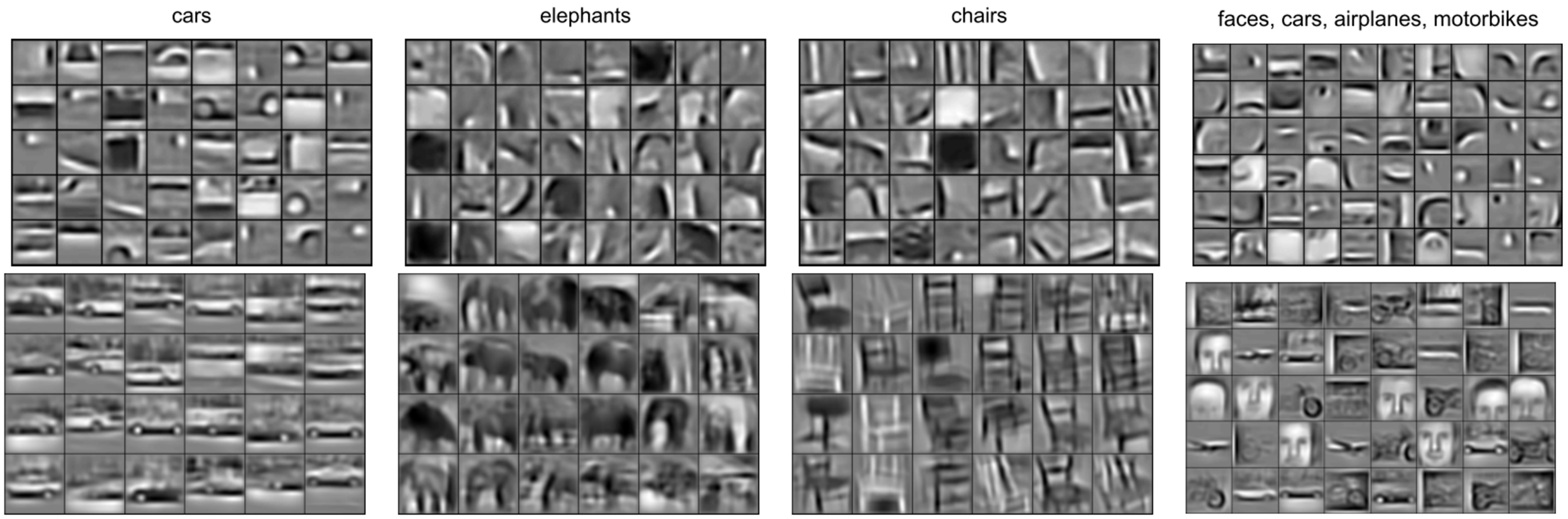
- When trained for face detection:

Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. Lee, Honglak, et al. 2009

Visualization of the learned weights

- When trained for different object classes:

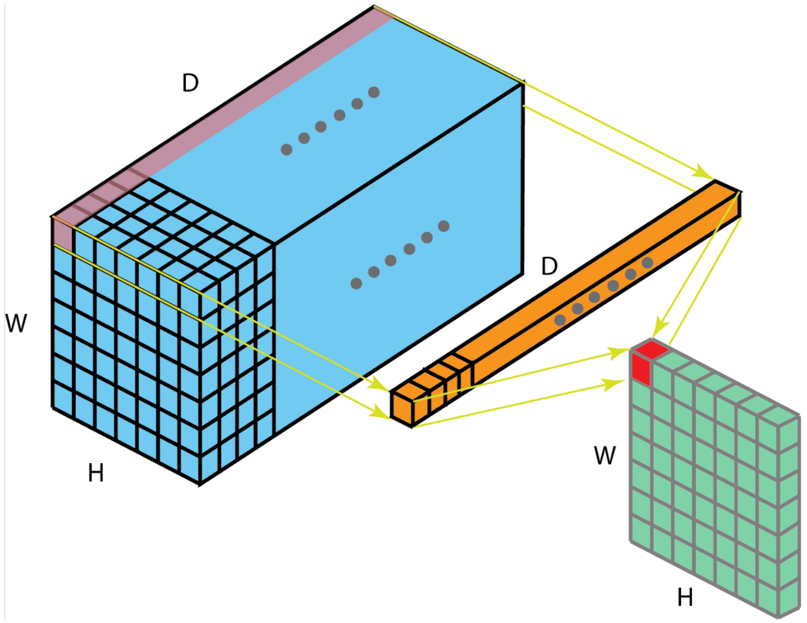


Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. Lee, Honglak, et al. 2009

1x1 Convolutions

Network in network

- Used for convolutions on the feature maps.
- Pooling shrinks the height and width of an image/feature map
- 1x1 Convolutions shrinks the number of filters
    - Ex. 28x28x192 to 28x28x32

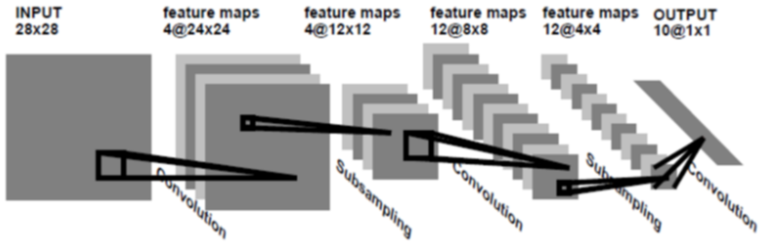


Source: Network in Network, Min Lin et al, University of Singapore
https://towardsdatascience.com/a-comprehensive-introduction-to-different-types-of-convolutions-in-deep-learning-669281e58215

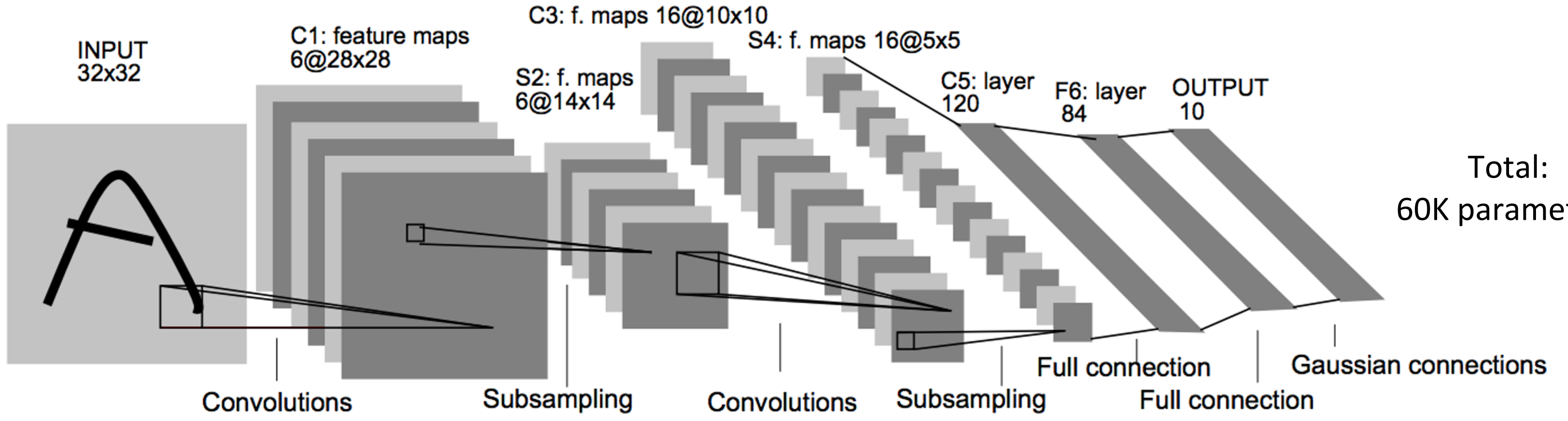
Popular/historical CNN Architectures

LeNet-1 1989 to LeNet-5 1998

- Used to automatically classify handwritten digits on bank cheques in USA.
- Convolutions for local receptive fields and weight sharing.
- Output: Gaussian (RBF) kernel before output layer.
- Average pooling for subsampling.



Today a softmax output is used.

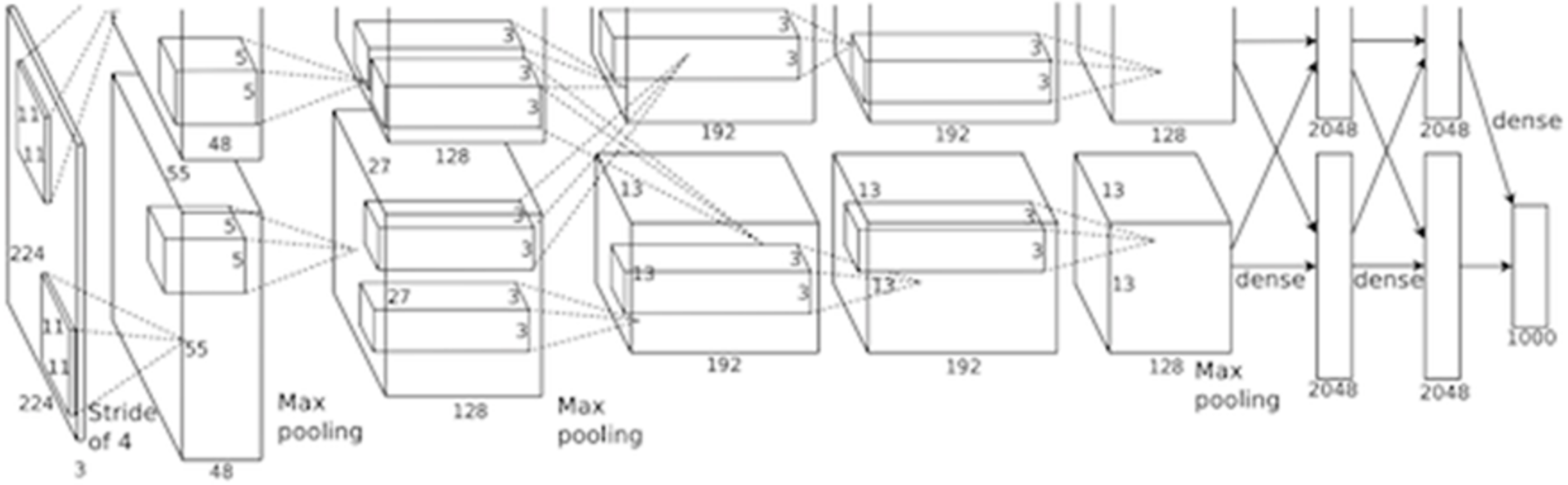


Total:
60K parameters

Source: Gradient-based learning applied to document recognition - LeNet-5, Y. LeCun, L. Bottou, Y. Bengio, P. Haffner

AlexNet - 2012

- Max-pooling instead of average pooling.
- ReLU instead of tanH activation function.
- Data augmentation to prevent overfitting.
  - Mirroring, random crops, intensity change.
- Dropout (longer convergence time, but avoids overfitting).



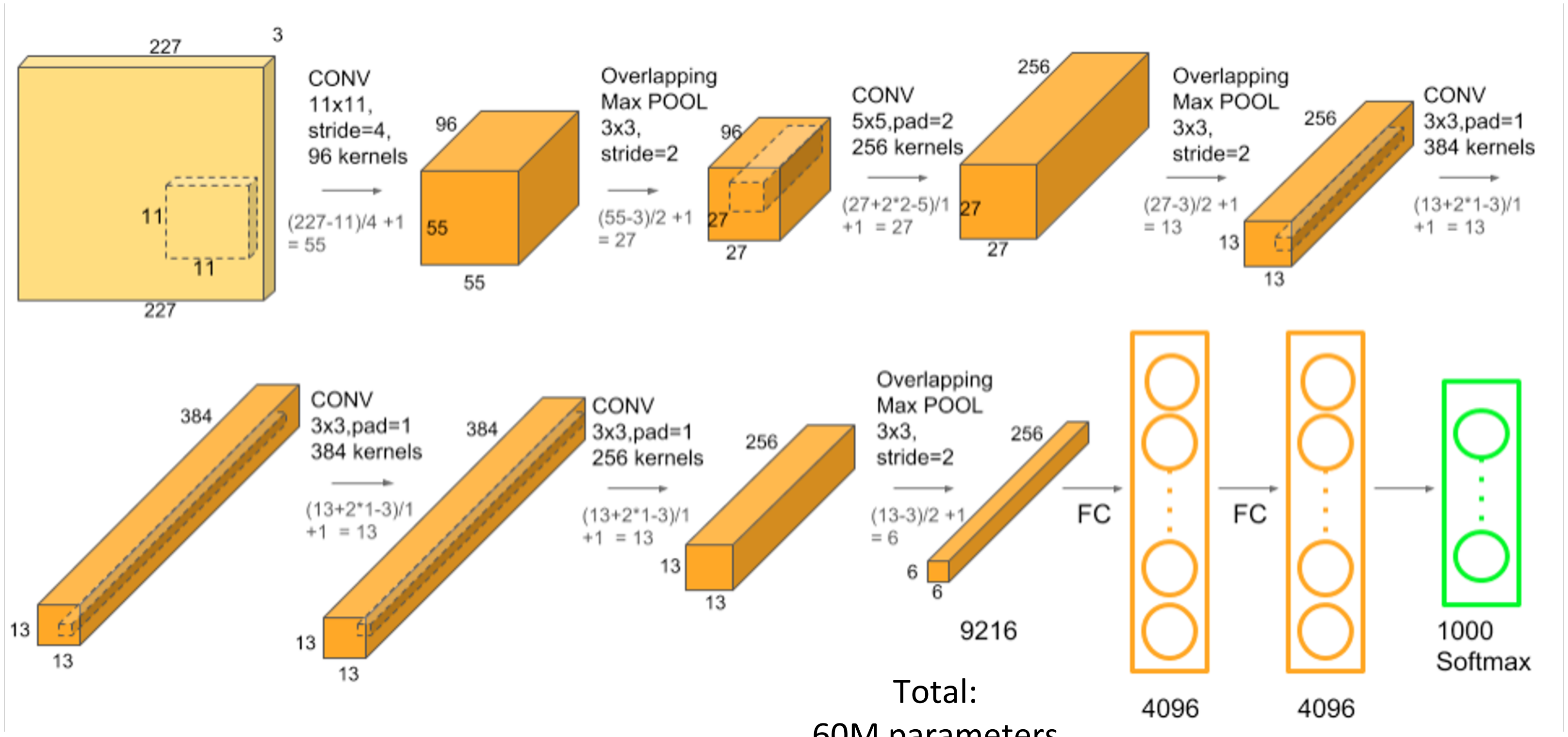
Trained on 2 GPUs

Total:
60M parameters

Source: ImageNet Classification with Deep Convolutional Neural Networks, Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton

AlexNet - 2012

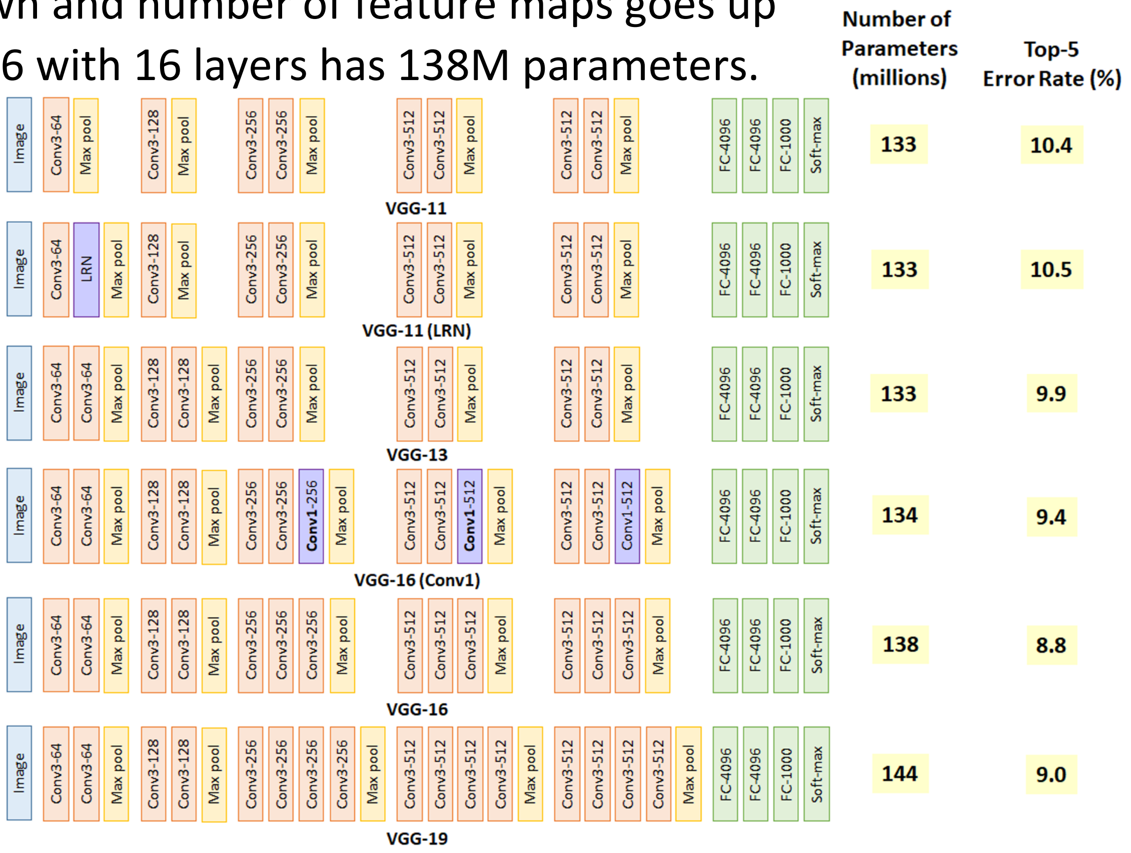
- Max-pooling instead of average pooling.
- ReLU instead of tanH activation function.
- Data augmentation to prevent overfitting.
  - Mirroring, random crops, intensity change.
- Dropout (longer convergence time, but avoids overfitting).



Source: https://www.learnopencv.com/understanding-alexnet/

VGG (Visual Geometry Group) - 2014

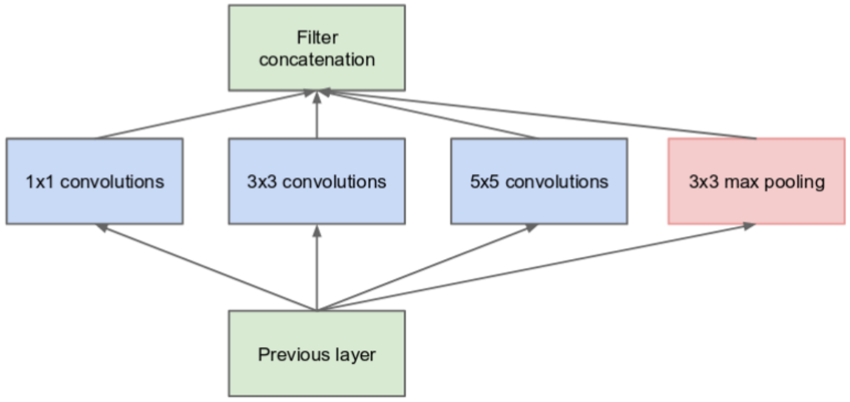
- Idea: simplify convolutional layers, repeat:
  - 3x3 filters, stride=1, same convolutions.
  - Max-pool 2x2, s=2.
  - Layer size goes down and number of feature maps goes up
- Deeper network, VGG-16 with 16 layers has 138M parameters.

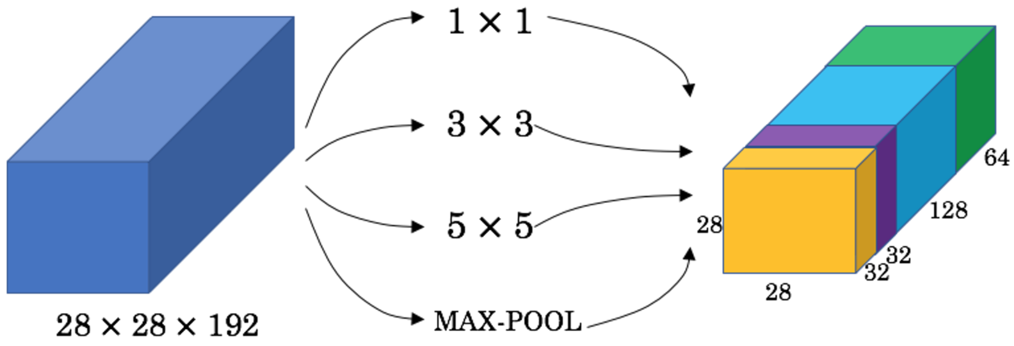

Number of Parameters (millions)
Top-5 Error Rate (%)
Image, Conv3-64, Max pool, Conv3-128, Max pool, Conv3-256, Conv3-256, Max pool, Conv3-512, Conv3-512, Max pool, Conv3-512, Conv3-512, Max pool, FC-4096, FC-4096, FC-1000, Soft-max
VGG-11
133
10.4
Image, Conv3-64, LRN, Max pool, Conv3-128, Max pool, Conv3-256, Conv3-256, Max pool, Conv3-512, Conv3-512, Max pool, Conv3-512, Conv3-512, Max pool, FC-4096, FC-4096, FC-1000, Soft-max
VGG-11 (LRN)
133
10.5
Image, Conv3-64, Conv3-64, Max pool, Conv3-128, Conv3-128, Max pool, Conv3-256, Conv3-256, Max pool, Conv3-512, Conv3-512, Max pool, Conv3-512, Conv3-512, Max pool, FC-4096, FC-4096, FC-1000, Soft-max
VGG-13
133
9.9
Image, Conv3-64, Conv3-64, Max pool, Conv3-128, Conv3-128, Max pool, Conv3-256, Conv3-256, Conv1-256, Max pool, Conv3-512, Conv3-512, Conv1-512, Max pool, Conv3-512, Conv3-512, Conv1-512, Max pool, FC-4096, FC-4096, FC-1000, Soft-max
VGG-16 (Conv1)
134
9.4
Image, Conv3-64, Conv3-64, Max pool, Conv3-128, Conv3-128, Max pool, Conv3-256, Conv3-256, Conv3-256, Max pool, Conv3-512, Conv3-512, Conv3-512, Max pool, Conv3-512, Conv3-512, Conv3-512, Max pool, FC-4096, FC-4096, FC-1000, Soft-max
VGG-16
138
8.8
Image, Conv3-64, Conv3-64, Max pool, Conv3-128, Conv3-128, Max pool, Conv3-256, Conv3-256, Conv3-256, Conv3-256, Max pool, Conv3-512, Conv3-512, Conv3-512, Conv3-512, Max pool, Conv3-512, Conv3-512, Conv3-512, Conv3-512, Max pool, FC-4096, FC-4096, FC-1000, Soft-max
VGG-19
144
9.0

Source: Very Deep Convolutional Networks for Large-Scale Image Recognition, Karen Simonyan & Andrew Zisserman, University of Oxford

InceptionNet (GoogLeNet) - 2014

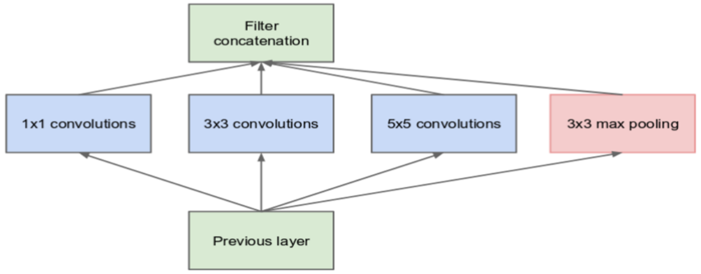
- Do not pick the operation to do - do them all.
    - 1x1 convolution
    - 3x3 convolution (same convolutions)
    - 5x3 convolution (same convolutions)
    - Pooling (with padding)
- Repeat the simple structure multiple times



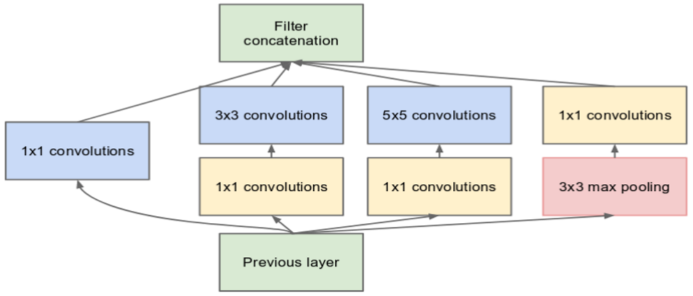


Source: Going deeper with convolutions, Christian Szegedy et al. Google (2014)

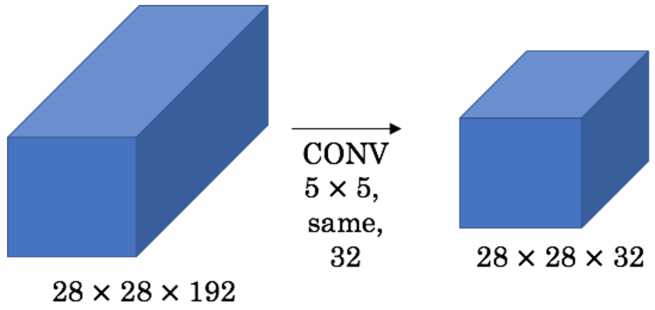
InceptionNet (GoogLeNet) - 2014

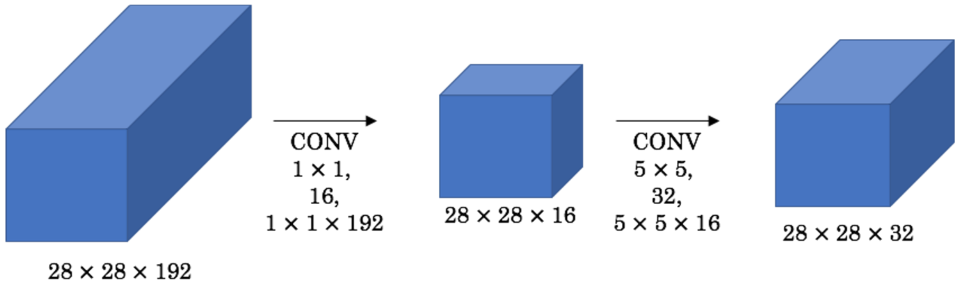


(a) Inception module, naïve version



(b) Inception module with dimension reductions





$$(28 \times 28 \times 32) \times (5 \times 5 \times 192) = 120M$$

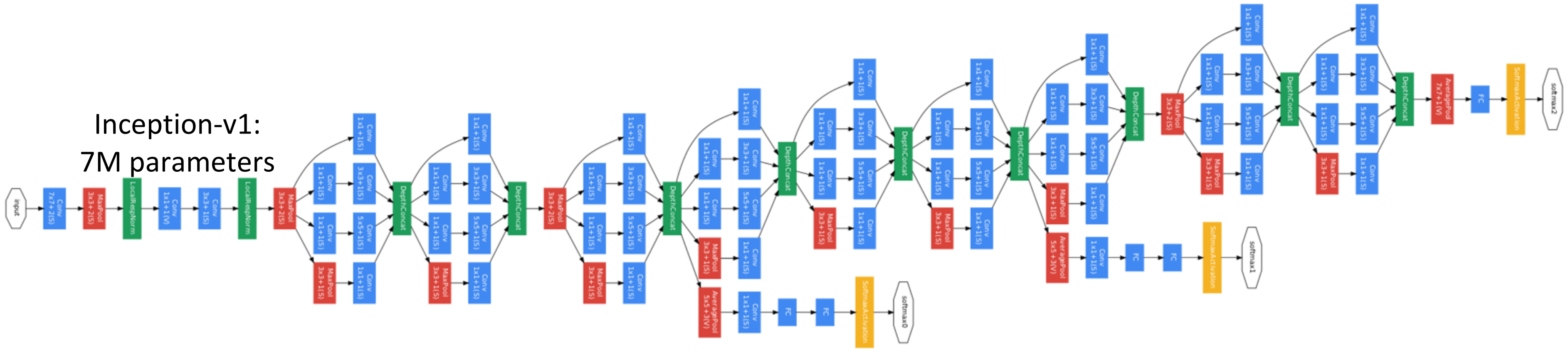
$$(28 \times 28 \times 16) \times 192 = 2.4M$$
$$(28 \times 28 \times 32) \times (5 \times 5 \times 16) = 10M$$
$$2.4 + 10M = 12.4M$$

Source: Going deeper with convolutions, Christian Szegedy et al. Google (2014)
https://www.coursera.org/learn/convolutional-neural-networks

InceptionNet (GoogLeNet) - 2014

- Side branches - use intermediate layers to do the prediction.
- Use final output and side branches to compute the total cost.

Inception-v1:
7M parameters



Meme citation in original paper:

Source: Going deeper with convolutions, Christian Szegedy et al. Google

ResNet - 2015

- Idea: Skip connections - forward outputs from activations to future layer inputs.
- Allows to train really deep networks:
  - It avoid vanishing gradients.
- Skip connection doesn't hurt as it is easy to learn the identity function:

Skip is added before the second activation.

$$a^{(l+2)} = g(z^{(l+1)} + a^{(l)})$$

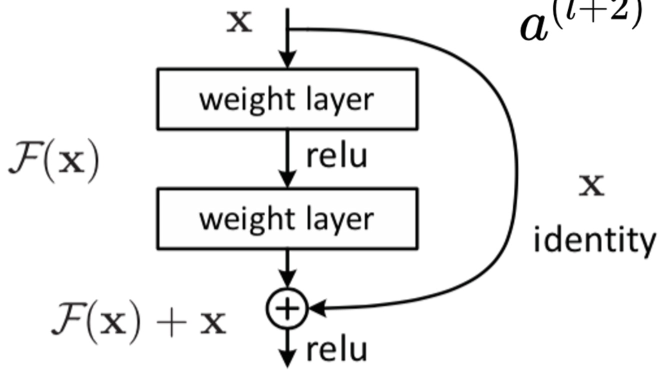
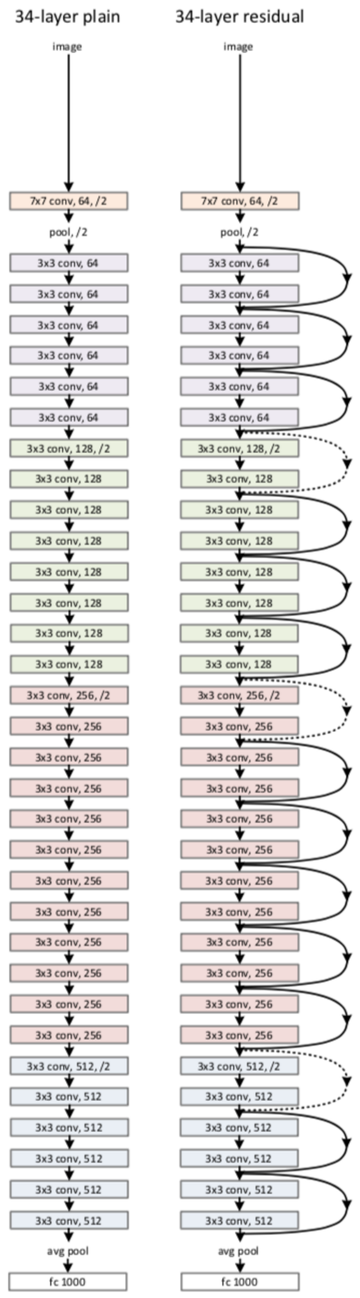

x
weight layer
relu
weight layer
$\mathcal{F}(\mathbf{x})$
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

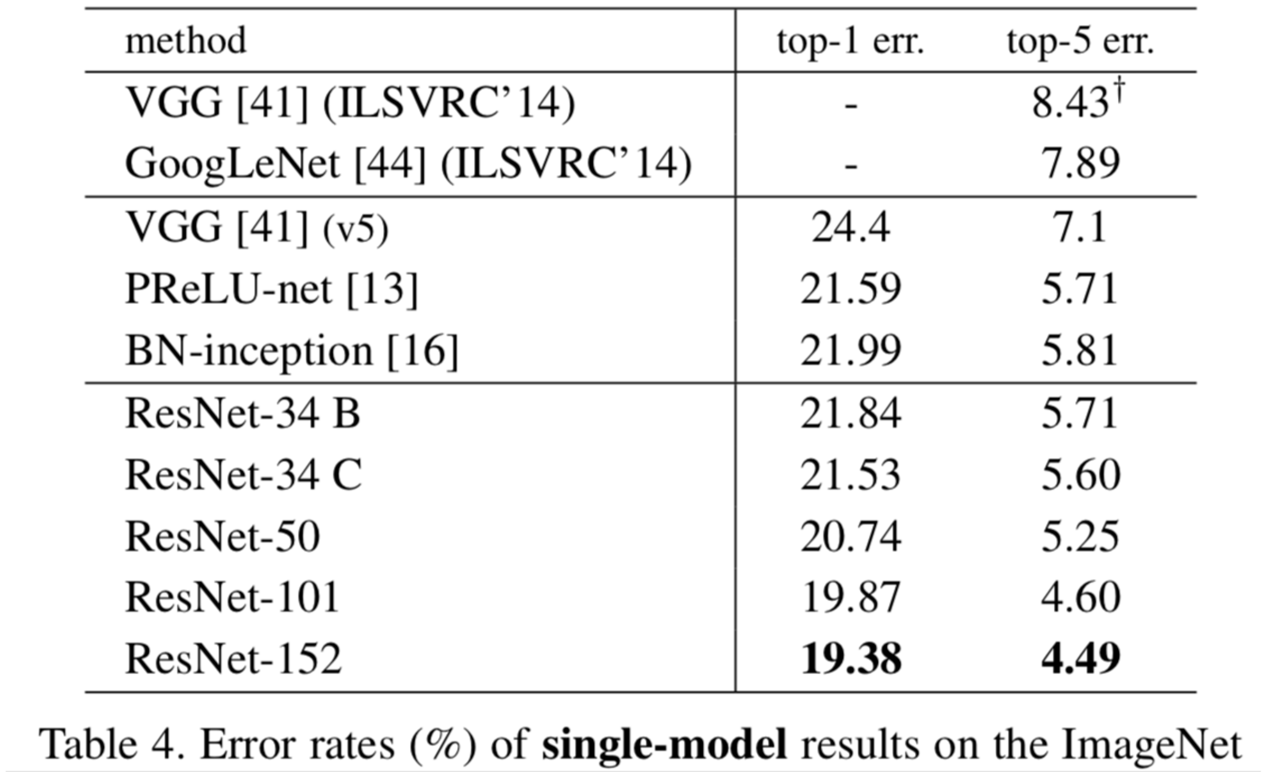
Figure 2. Residual learning: a building block.



Total (ResNet-152): 60M parameters

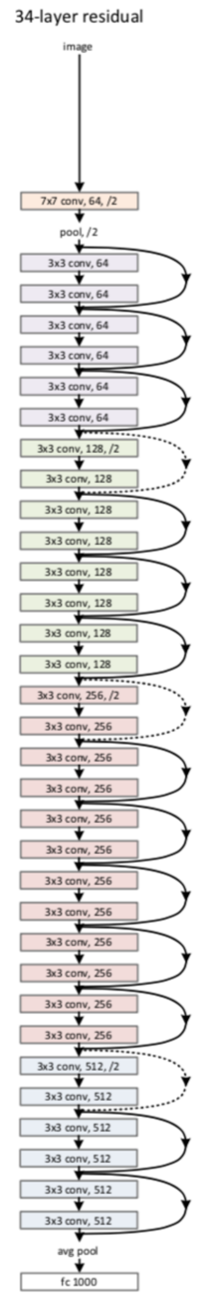
Source: Deep Residual Learning for Image Recognition, Kaiming He et al. Microsoft

ResNet - 2015



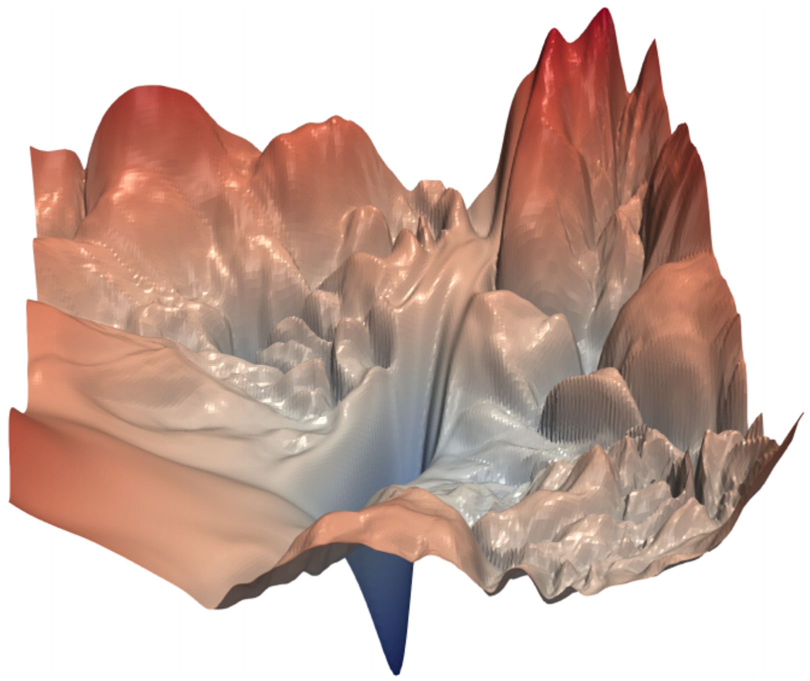
method	top-1 err.	top-5 err.
VGG [41] (ILSVRC'14)	-	8.43†
GoogLeNet [44] (ILSVRC'14)	-	7.89
VGG [41] (v5)	24.4	7.1
PReLU-net [13]	21.59	5.71
BN-inception [16]	21.99	5.81
ResNet-34 B	21.84	5.71
ResNet-34 C	21.53	5.60
ResNet-50	20.74	5.25
ResNet-101	19.87	4.60
ResNet-152	**19.38**	**4.49**

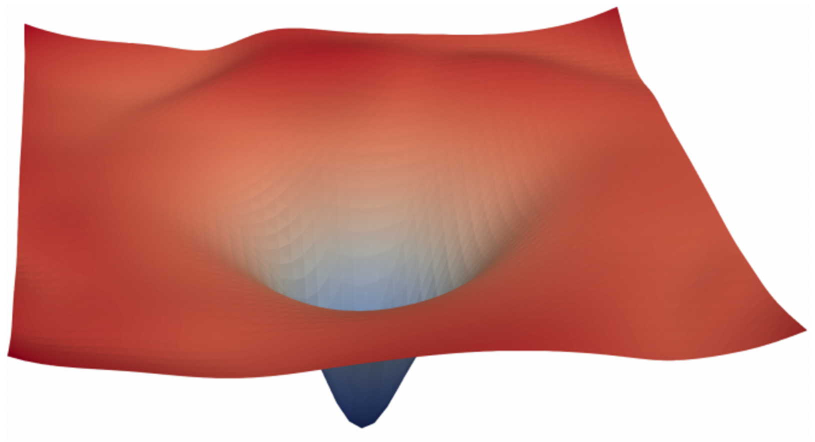
Table 4. Error rates (%) of **single-model** results on the ImageNet



Source: Deep Residual Learning for Image Recognition, Kaiming He et al. Microsoft

Skip connections


(a) without skip connections

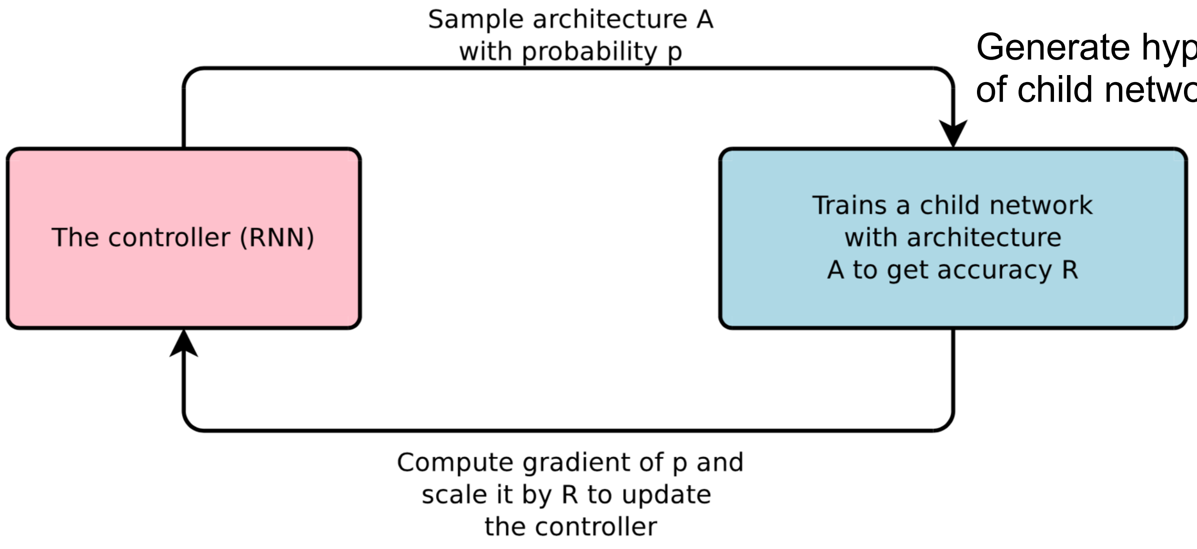

(b) with skip connections

Loss surface of ResNet-56 with/without skip connections.

Source: Visualizing the Loss Landscape of Neural Nets, NeurIPS 2018

Neural Architecture Search - 2017

NAS for finding good architectures with gradient-based search



Generate hyperparameters of child network

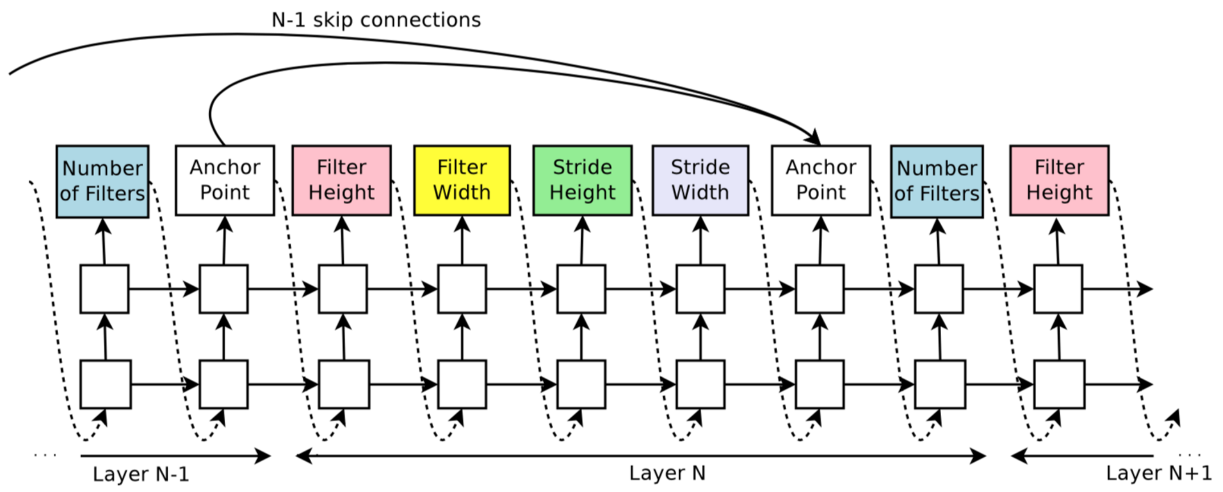
Learning for CIFAR-10 (image recognition) - Final test on ImageNet

Source: Neural Architecture Search with Reinforcement Learning, Barret Zoph, Quoc V. Le, Google Brain

Neural Architecture Search - 2017

Sampling of simple convolutional network. Predicts:

- Filter height + width
- Stride width
- Number of filters/layer + repeats
- (Skip)



Splitting computation across multiple machines with a central parameter server.

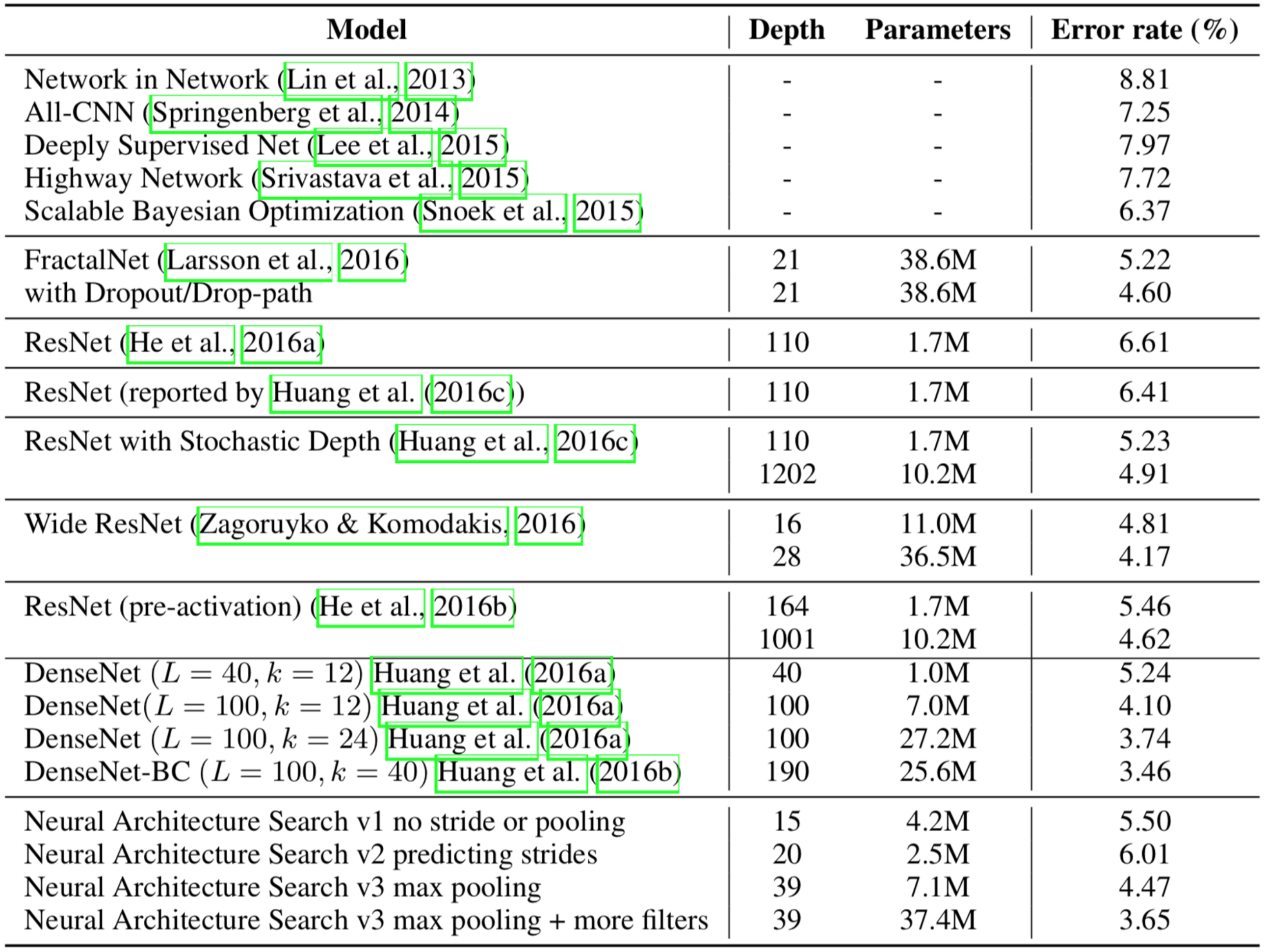
Controller trains 12.800 architectures -> then trains child till convergence

800 networks being trained on 800 GPU's - concurrently at any time!

Running time … 28 days!

Source: Neural Architecture Search with Reinforcement Learning, Barret Zoph, Quoc V. Le, Google Brain

Neural Architecture Search - 2017



Model	Depth	Parameters	Error rate (%)
Network in Network (Lin et al., 2013)	-	-	8.81
All-CNN (Springenberg et al., 2014)	-	-	7.25
Deeply Supervised Net (Lee et al., 2015)	-	-	7.97
Highway Network (Srivastava et al., 2015)	-	-	7.72
Scalable Bayesian Optimization (Snoek et al., 2015)	-	-	6.37
FractalNet (Larsson et al., 2016)	21	38.6M	5.22
with Dropout/Drop-path	21	38.6M	4.60
ResNet (He et al., 2016a)	110	1.7M	6.61
ResNet (reported by Huang et al. (2016c))	110	1.7M	6.41
ResNet with Stochastic Depth (Huang et al., 2016c)	110	1.7M	5.23
	1202	10.2M	4.91
Wide ResNet (Zagoruyko & Komodakis, 2016)	16	11.0M	4.81
	28	36.5M	4.17
ResNet (pre-activation) (He et al., 2016b)	164	1.7M	5.46
	1001	10.2M	4.62
DenseNet ($L = 40, k = 12$) Huang et al. (2016a)	40	1.0M	5.24
DenseNet($L = 100, k = 12$) Huang et al. (2016a)	100	7.0M	4.10
DenseNet ($L = 100, k = 24$) Huang et al. (2016a)	100	27.2M	3.74
DenseNet-BC ($L = 100, k = 40$) Huang et al. (2016b)	190	25.6M	3.46
Neural Architecture Search v1 no stride or pooling	15	4.2M	5.50
Neural Architecture Search v2 predicting strides	20	2.5M	6.01
Neural Architecture Search v3 max pooling	39	7.1M	4.47
Neural Architecture Search v3 max pooling + more filters	39	37.4M	3.65

Table 1: Performance of Neural Architecture Search and other state-of-the-art models on CIFAR-10.

Source: Neural Architecture Search with Reinforcement Learning, Barret Zoph, Quoc V. Le, Google Brain

U-Net - 2015

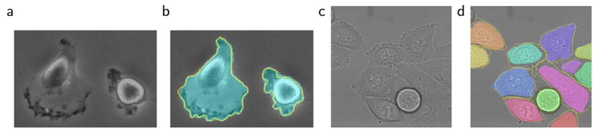
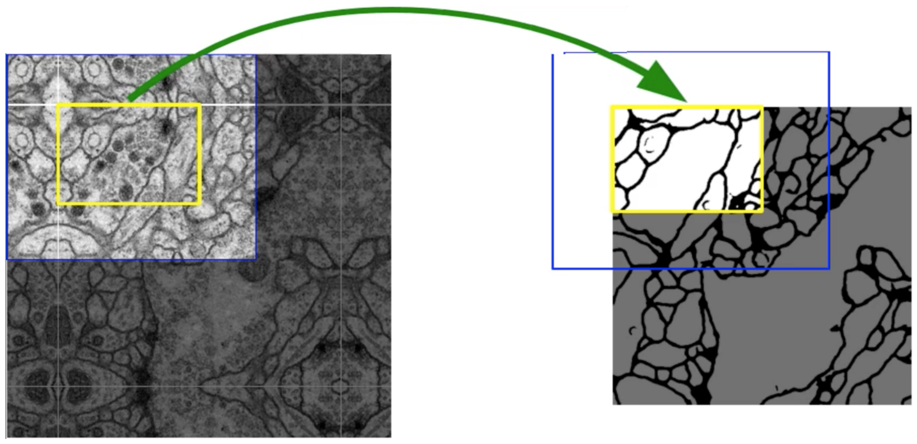
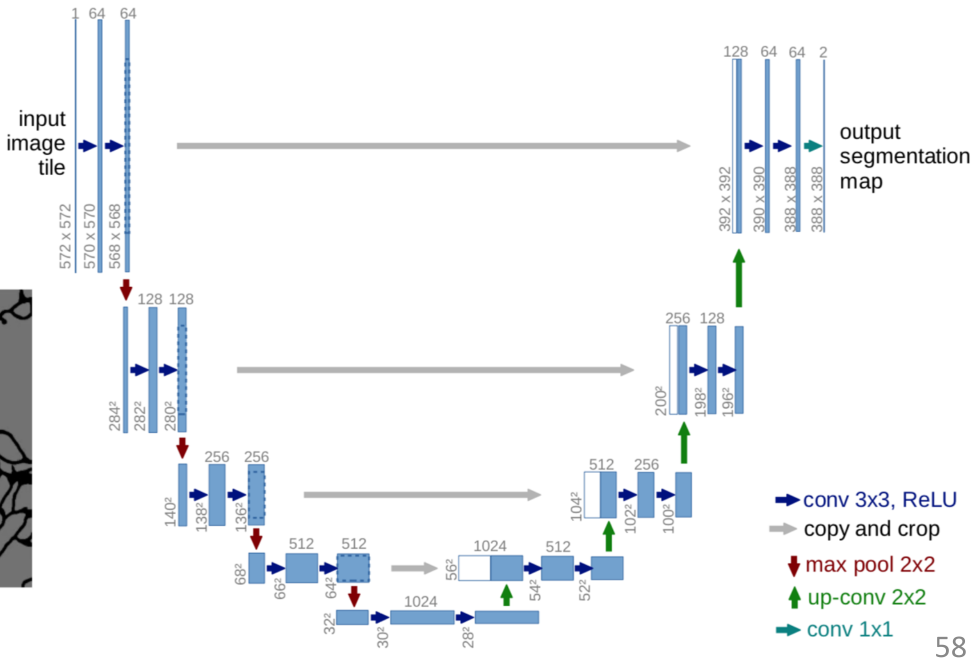


Fig. 4. Result on the ISBI cell tracking challenge. (**a**) part of an input image of the "PhC-U373" data set. (**b**) Segmentation result (cyan mask) with manual ground truth (yellow border) (**c**) input image of the "DIC-HeLa" data set. (**d**) Segmentation result (random colored masks) with manual ground truth (yellow border).

- Architecture for image segmentation - typical challenge within medical images.
- Works with very little available data and is trainable end-to-end.
- Encoder + decoder network.
- Only contains convolutional layers - no fully connected dense layers.
- Binary cross-entropy can be used to "classify" each pixel.

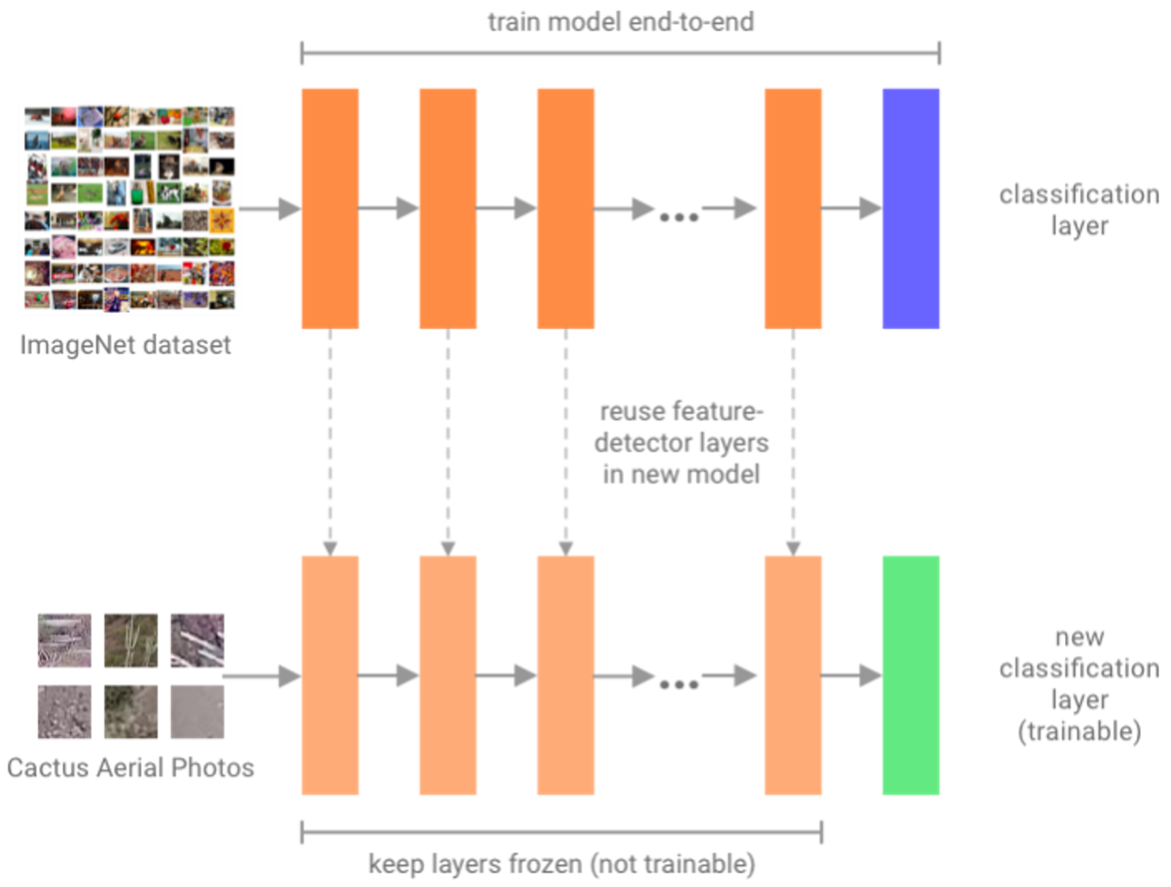




Source: U-Net Convolutional Network for Biomedical Image Segmentation, Olaf Ronneberger et al, University of Freiburg

Transfer learning

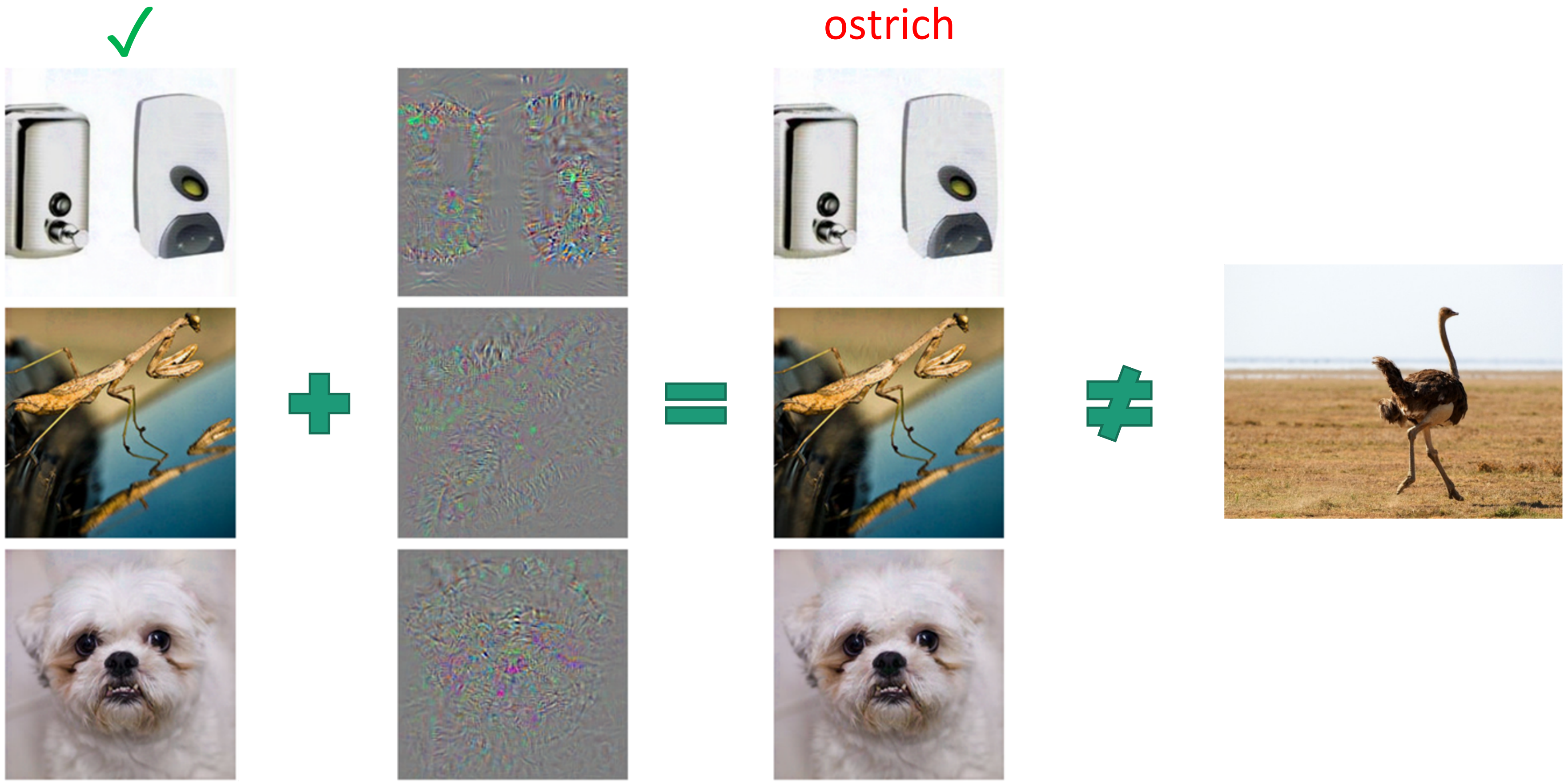
- Use a pretrained network on another tasks


train model end-to-end
ImageNet dataset
classification layer
reuse feature-detector layers in new model
Cactus Aerial Photos
new classification layer (trainable)
keep layers frozen (not trainable)

Source: https://towardsdatascience.com/easy-image-classification-with-tensorflow-2-0-f734fee52d13

Adversarial attacks on images

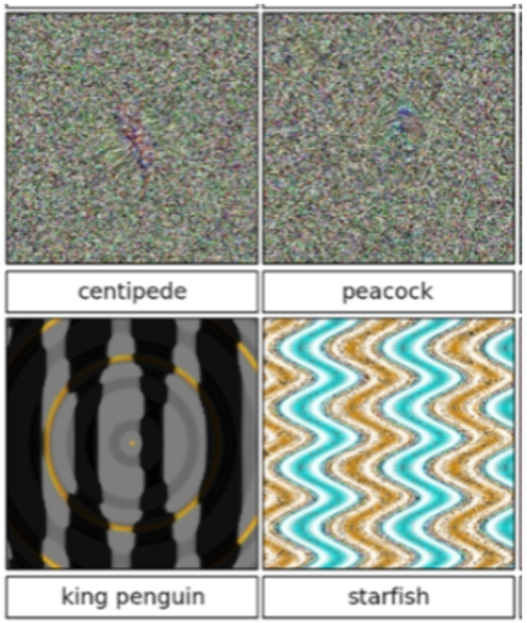
- Adding the “right” noise induces miss-classification

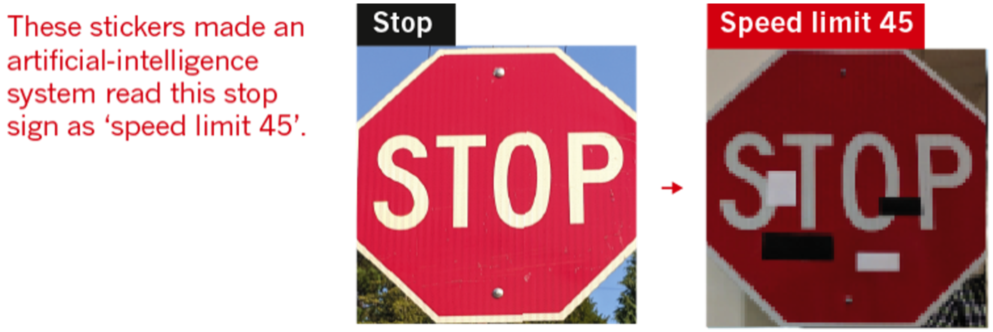


Source: Szegedy, Christian, et al. "Intriguing properties of neural networks." 2013

Adversarial attacks on images

- Generating "adversarial" examples – classification confidence > 99%




These stickers made an artificial-intelligence system read this stop sign as 'speed limit 45'.
Stop
Speed limit 45
STOP
STOP

- Security risk for DNN systems: self-driving cars detecting a stop sign or a pedestrian.
- One does not even need to know the DNN and its weights. By sending enough requests to the model, the internal mechanism can be inferred.
- Deep learning systems still does not understand the world!

Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. Nguyen, Anh, Jason Yosinski, and Jeff Clune. *2015*
https://www.nature.com/articles/d41586-019-03013-5

Beyond image classification

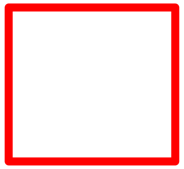
Object localization vs detection

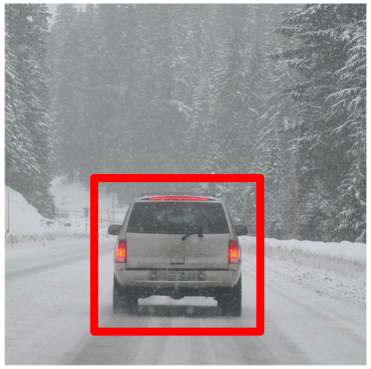
Image classification

Classification with localization

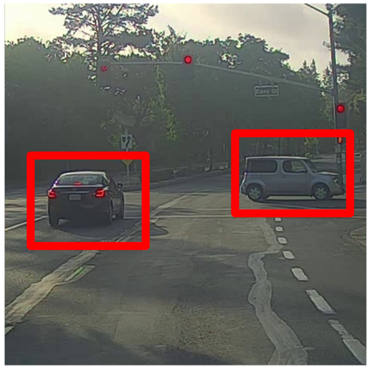
Detection

Naive method: Sliding window





$$y = \begin{bmatrix} 1 \\ b_x \\ b_y \\ b_h \\ b_w \\ 0 \\ 1 \\ 0 \end{bmatrix}$$



- Single object for image classification and localization.
- Multiple objects for object detection.

- Localization: Predict class + bounding box details (height, width, center): b_x, b_y, b_h, b_w

Example:

1. Pedestrian
2. Car
3. Motorcycle
4. Background

$$y = \begin{bmatrix} P_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

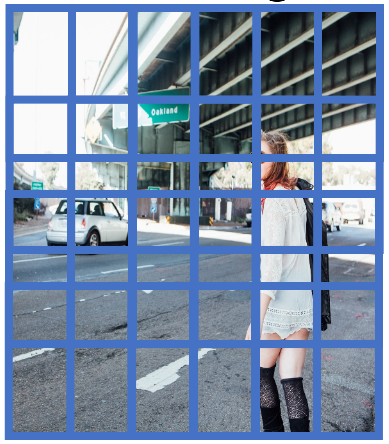
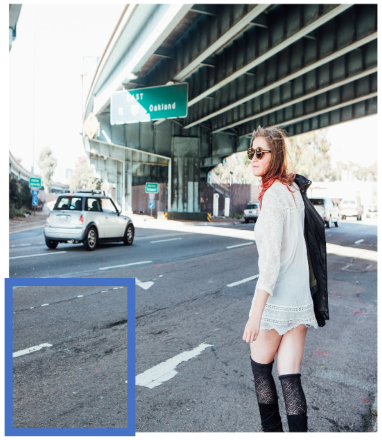
Probability of class existence?

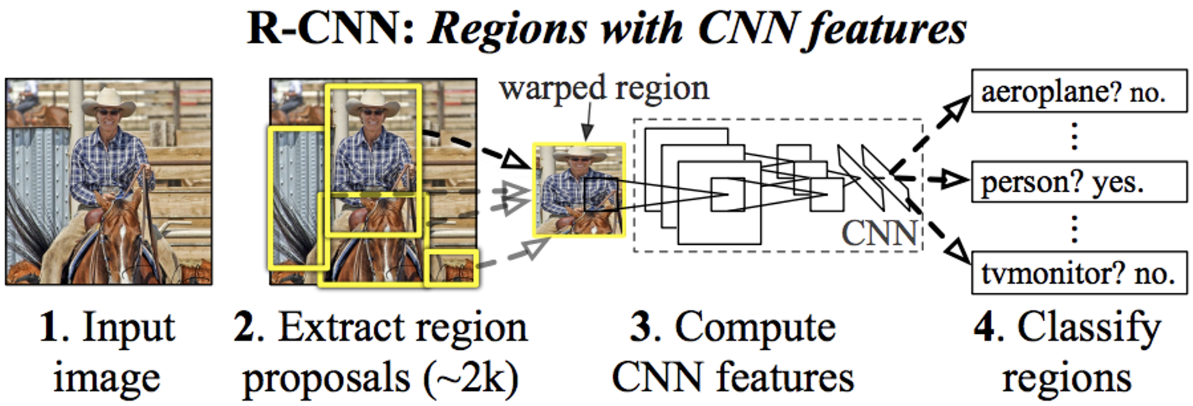
$$y = \begin{bmatrix} 0 \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \end{bmatrix}$$

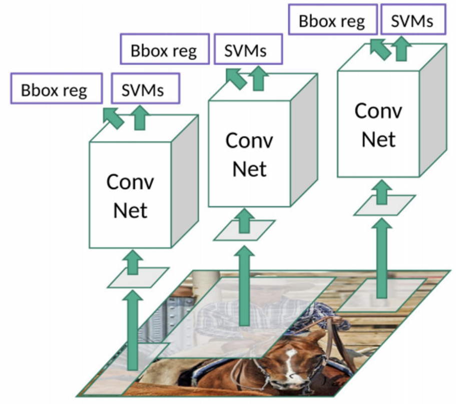
Source: https://www.coursera.org/learn/convolutional-neural-networks

Region-CNN (R-CNN)

- Bypass the problem of selecting a large number of regions.
- Segment the image into regions which we use a CNN to classify.



Nothing interesting!

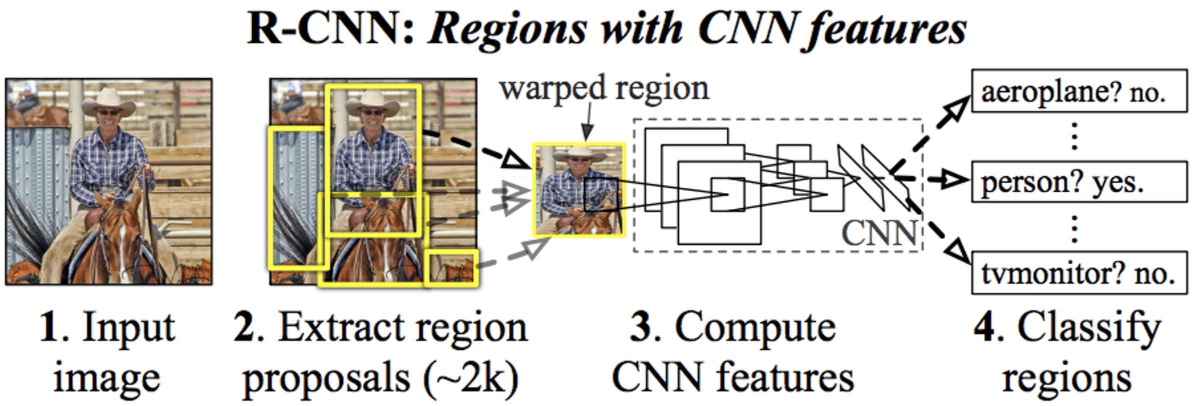


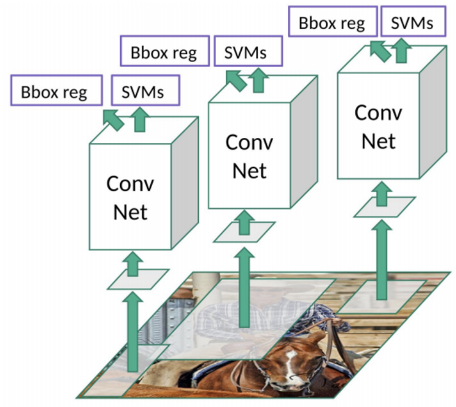


Source: Rich feature hierarchies for accurate object detection and semantic segmentation, Girshik et. al, 2013

Region-CNN (R-CNN) problems

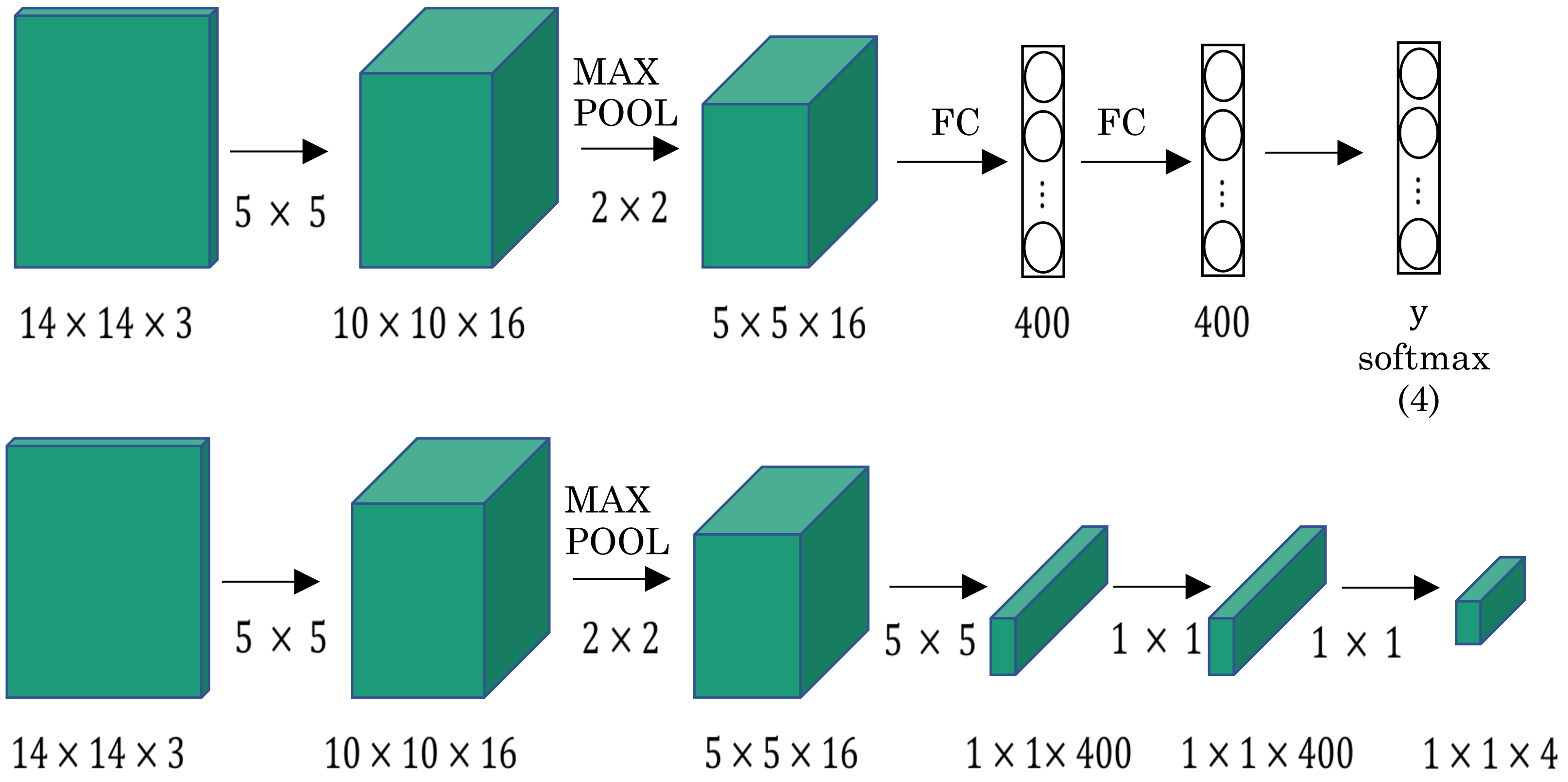
- It takes a long time to train the network where 2000 region proposals have to be classified.
- Not usable for real-time applications.
- No learning is happening for the region detections.





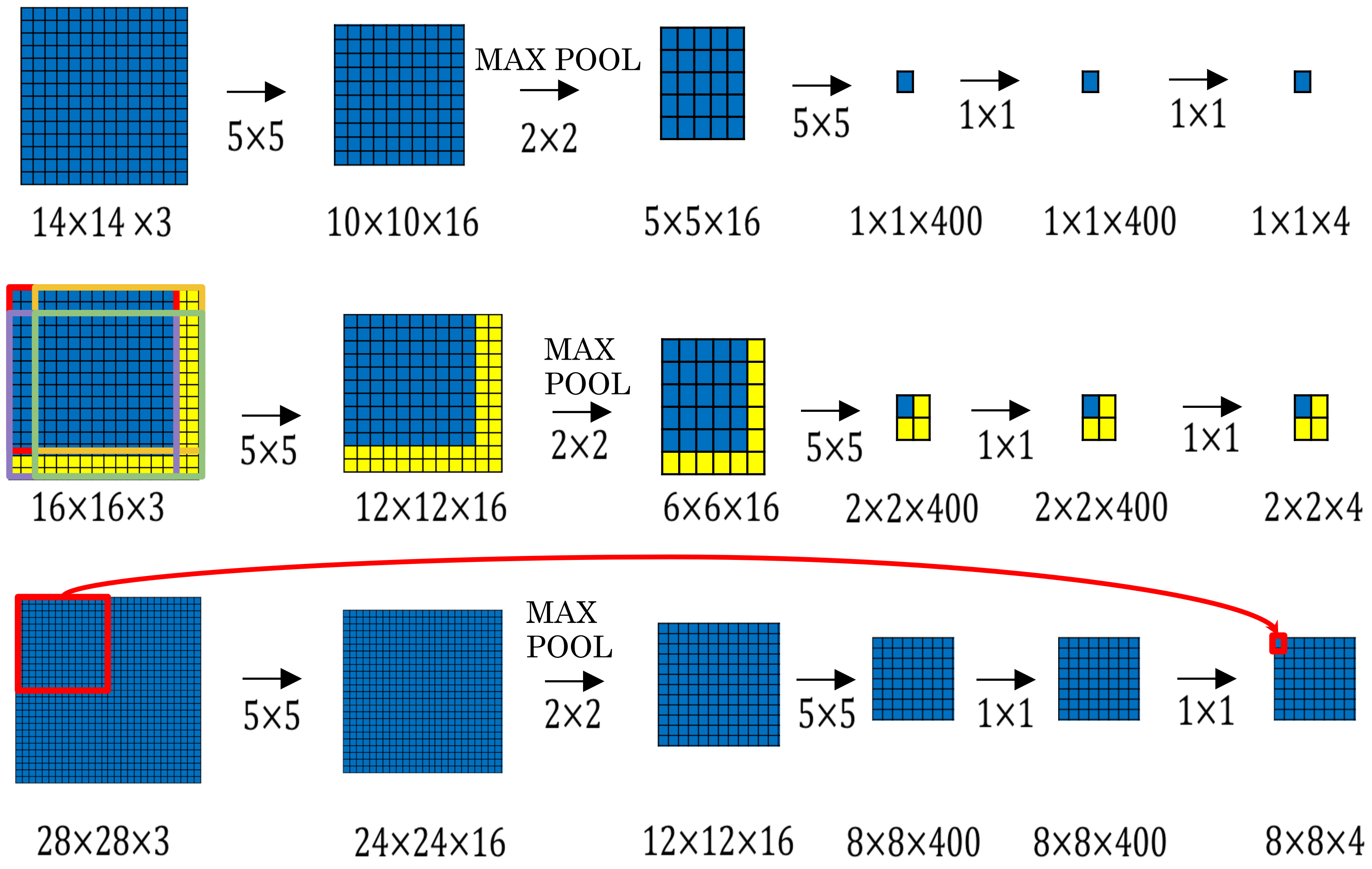
Source: Rich feature hierarchies for accurate object detection and semantic segmentation, Girshik et. al, 2013

Fully connected to convolutional



Source: https://www.coursera.org/learn/convolutional-neural-networks

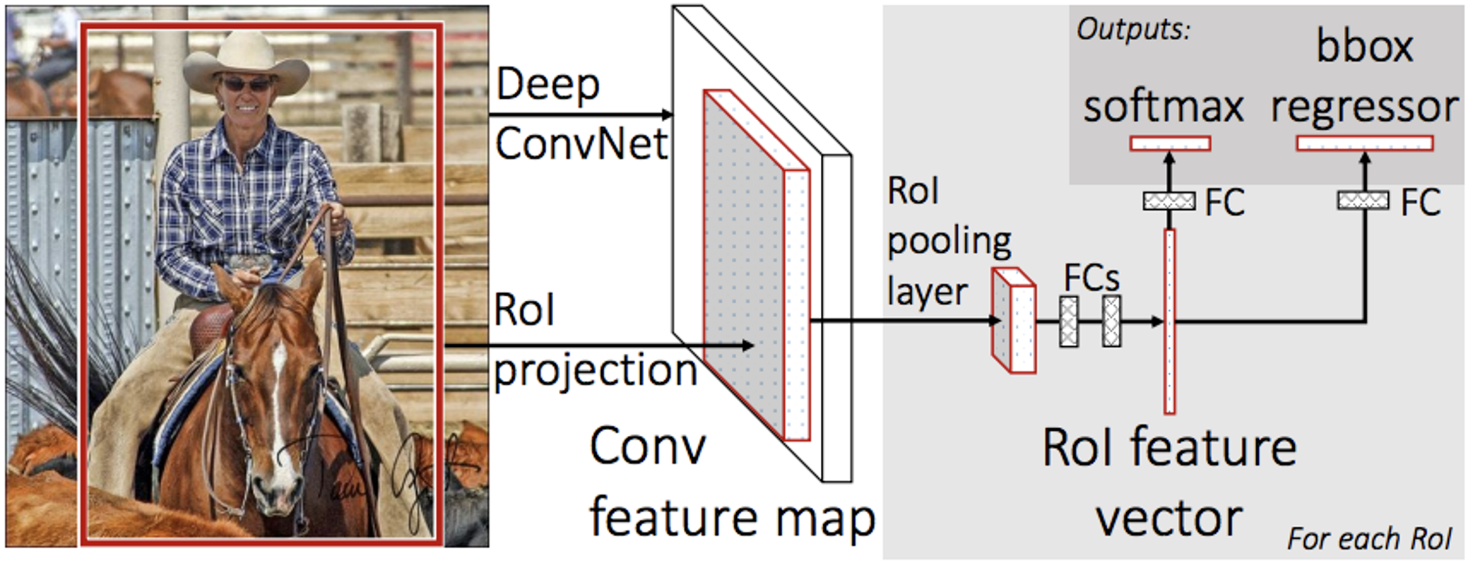
Convolutional sliding window



Source: https://www.coursera.org/learn/convolutional-neural-networks

Fast R-CNN

- "Fast" as the convolutional operation is implementing the sliding window to classify all the proposed regions.

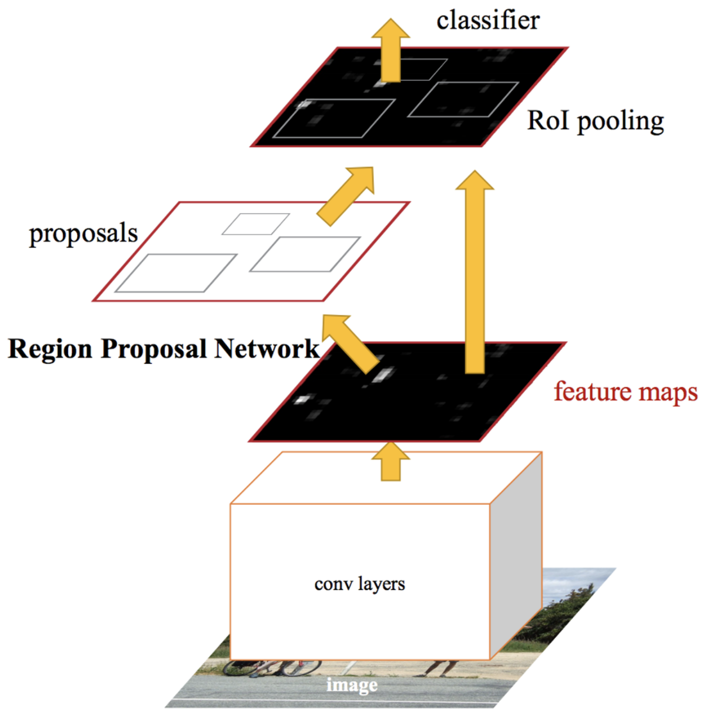


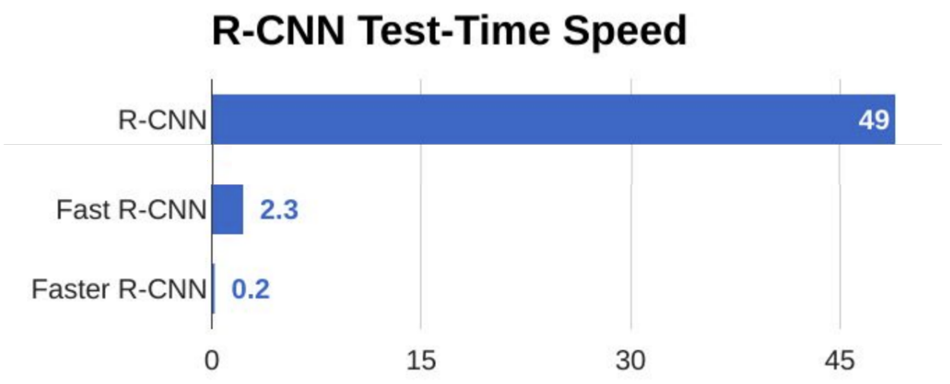
- Slow to propose regions

Source: Fast R-CNN, Girshik et. al, 2015

Faster-CNN

- Identify proposal regions from a convolutional neural network.

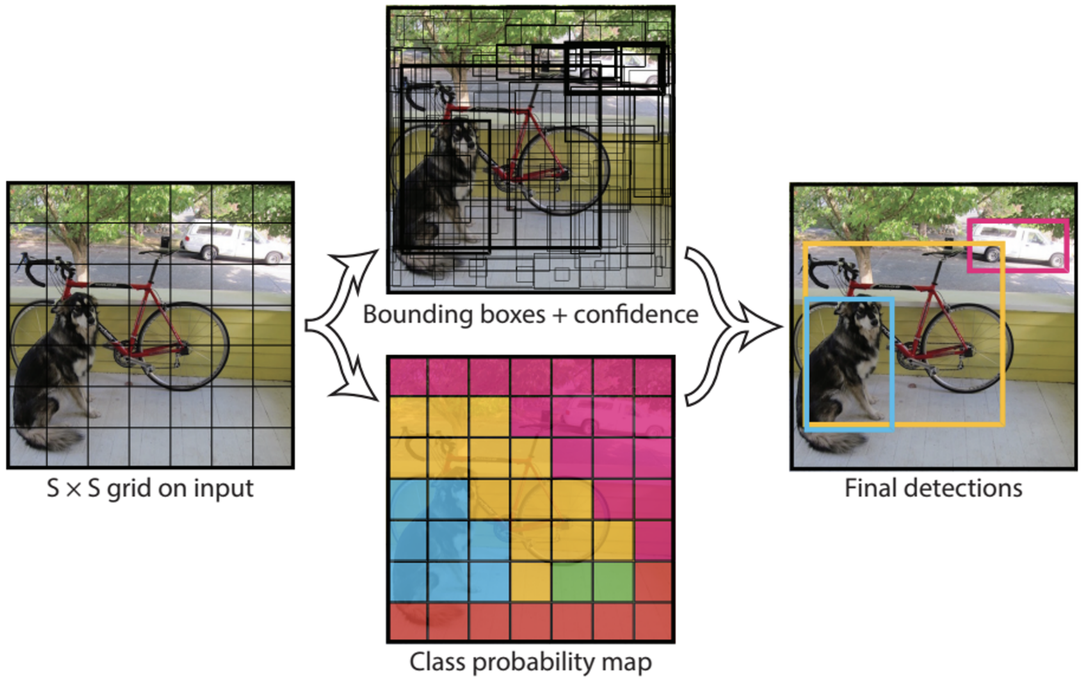




Source: Faster R-CNN: Towards real-time object detection with region proposal networks. Ren et. al, 2016

YOLO - You Only Look Once

- R-CNN variant all use regions to localize the object within the image.
- YOLO instead look at the complete image and divides it into a grid.
- Much faster than faster R-CNN.
- Problem with small objects within the image.



Source: You Only Look Once: Unified real-time object detection, Redmon et al. 2015

YOLO v4: You Only Look Once

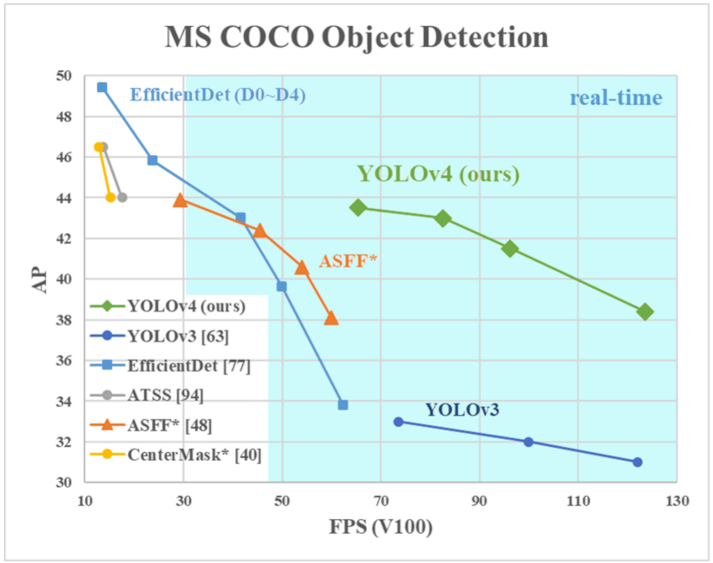
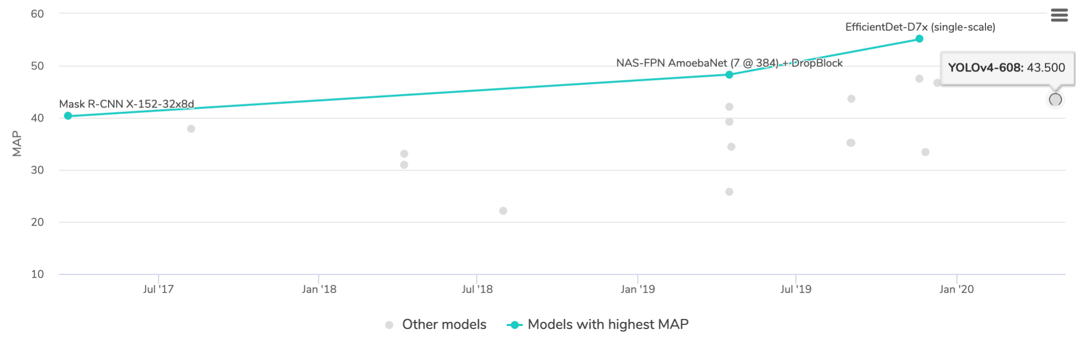
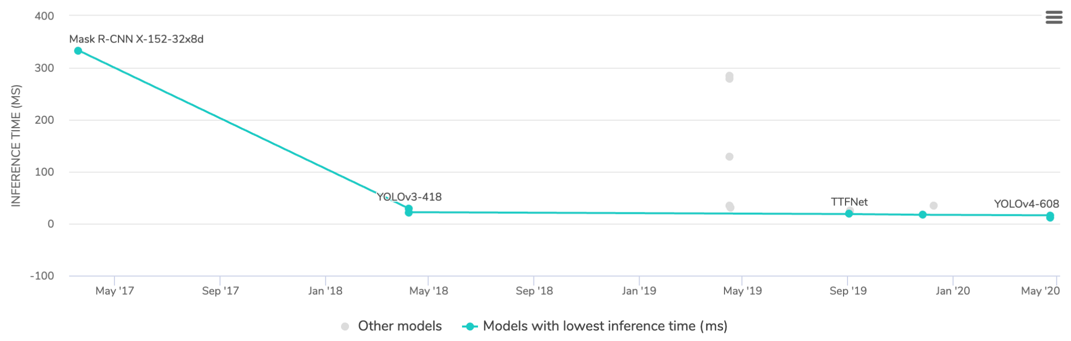


Figure 1: Comparison of the proposed YOLOv4 and other state-of-the-art object detectors. YOLOv4 runs twice faster than EfficientDet with comparable performance. Improves YOLOv3's AP and FPS by 10% and 12%, respectively.

Real-Time Object Detection on COCO



Real-Time Object Detection on COCO



Source: YOLOv4: Optimal Speed and Accuracy of Object Detection, Bochkovskiy et al. 2020
Source: https://paperswithcode.com/sota/real-time-object-detection-on-coco

Summary

- Fully connected networks are not feasible to use on image data.
- Use convolutional layers to detect features in images.
  - Objects are built from simple features.
  - CNNs are robust against object transformations.
- Convolutions in Neural networks is what is known from literature as cross-correlation.
- Additional hyperparameters:
  - Kernel/filter size, stride, padding.
- Use existing architectures with modifications for similar tasks.
- Use pre-trained networks and transfer learning - especially when data is sparse.

Credits

Books:

- https://www.deeplearningbook.org/
- http://neuralnetworksanddeeplearning.com/

Online Course from MIT:

- http://introtodeeplearning.com/

Online course from Stanford University:

- https://www.coursera.org/specializations/deep-learning?

Other

- cs231n.github.io
- appliedgo.net
- brohrer.github.io
- learnopencv.com