

Convolutional Neural Networks (CNNs)

Deep learning for Computer Vision

Pattern Recognition

Fall 2019

Dennis Madsen

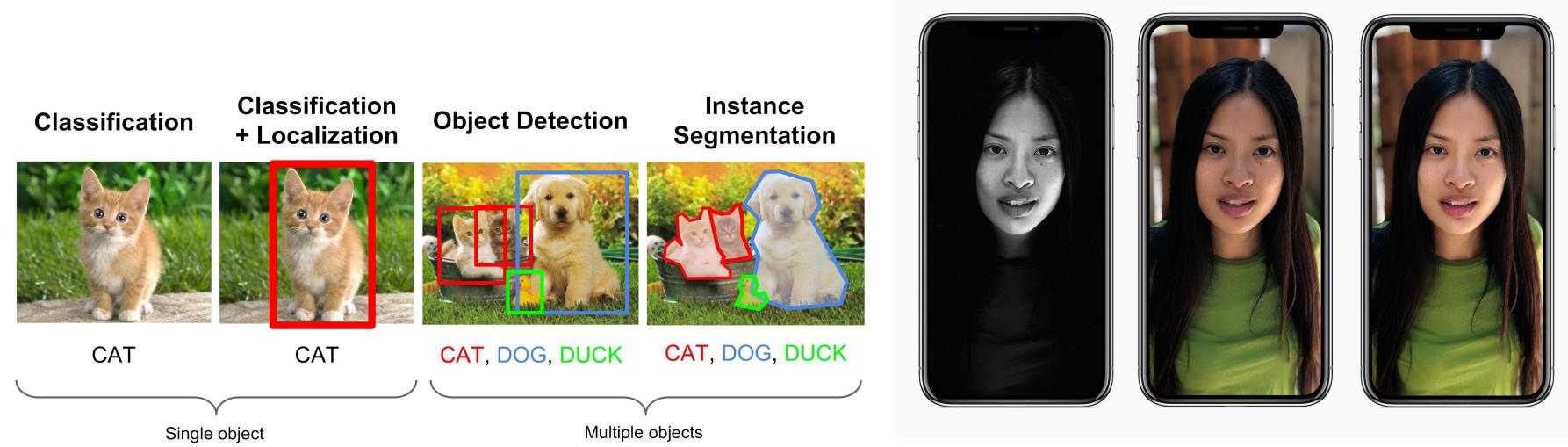
Topic overview

- Neural Networks (recap) 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

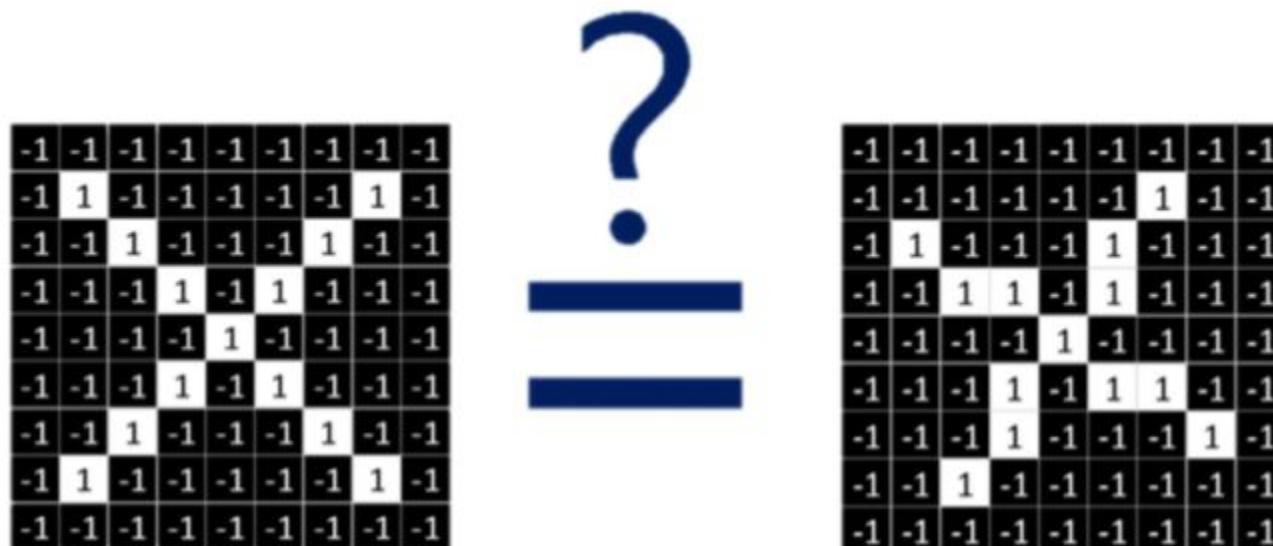


Neural style transfer



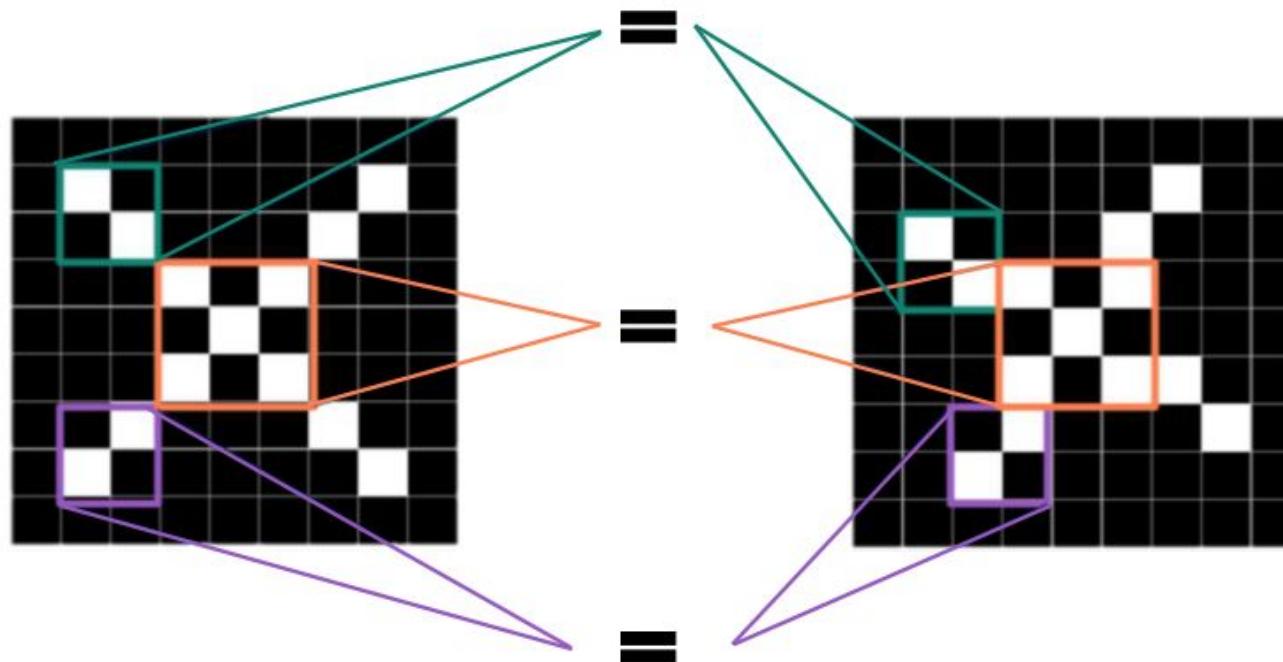
Motivation

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



Motivation

- Compare image features found in the images

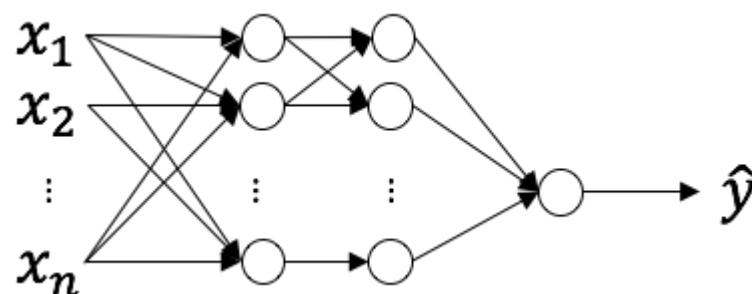


Problem with a fully connected network

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.

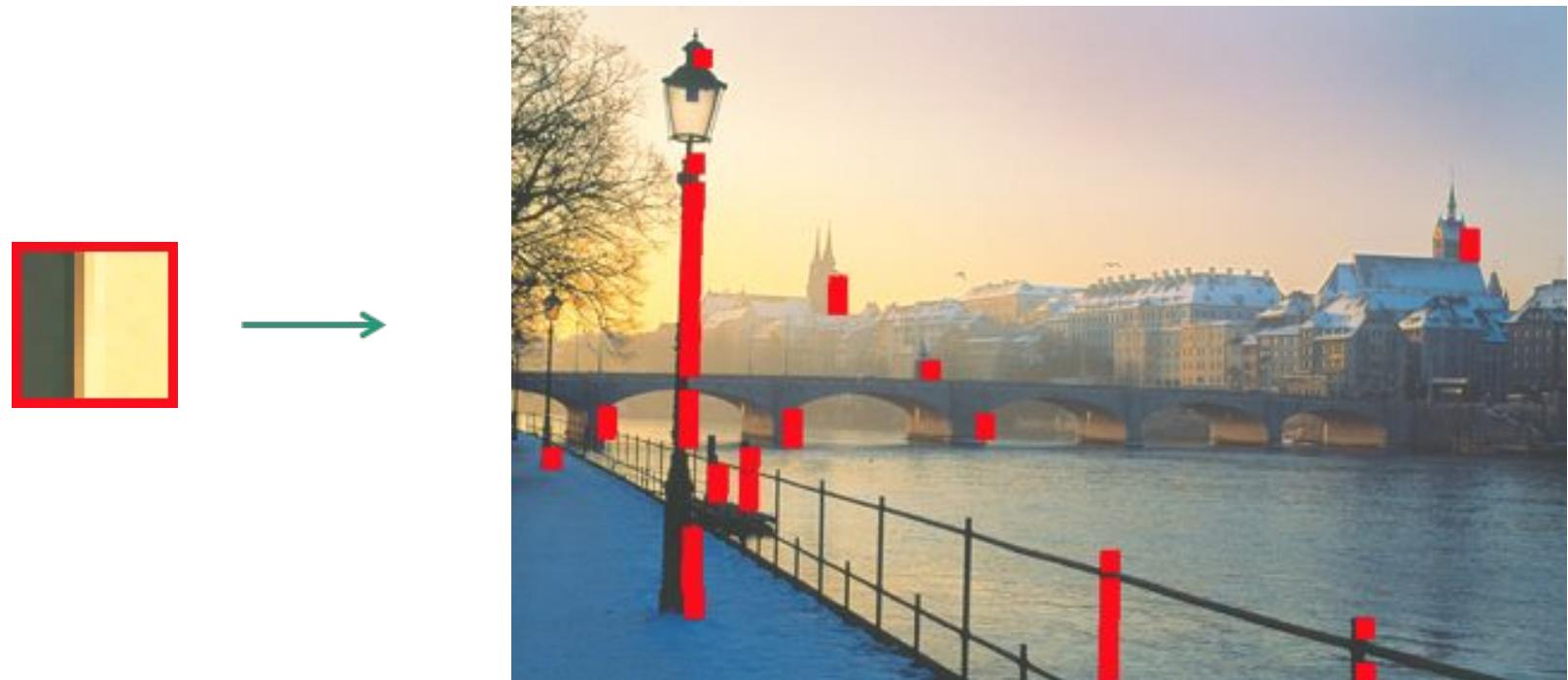
Fundamental Properties of Images

Property 1: Image statistics are locally correlated/structured.



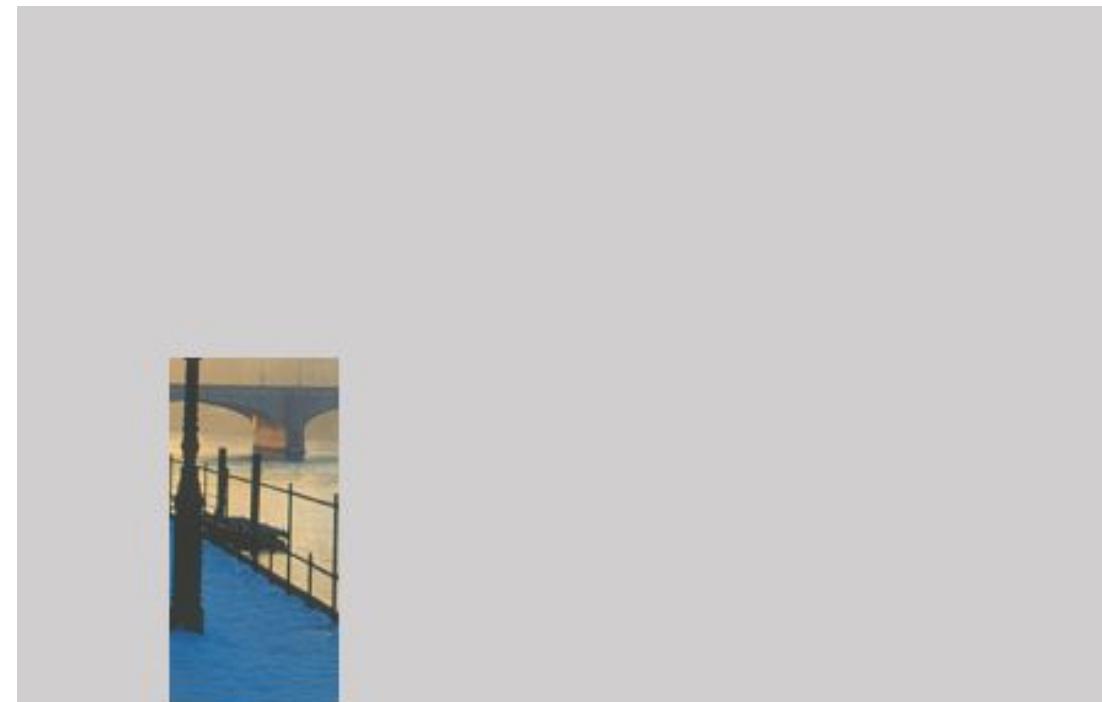
Fundamental Properties of Images

Property 2: Redundancy of structures.



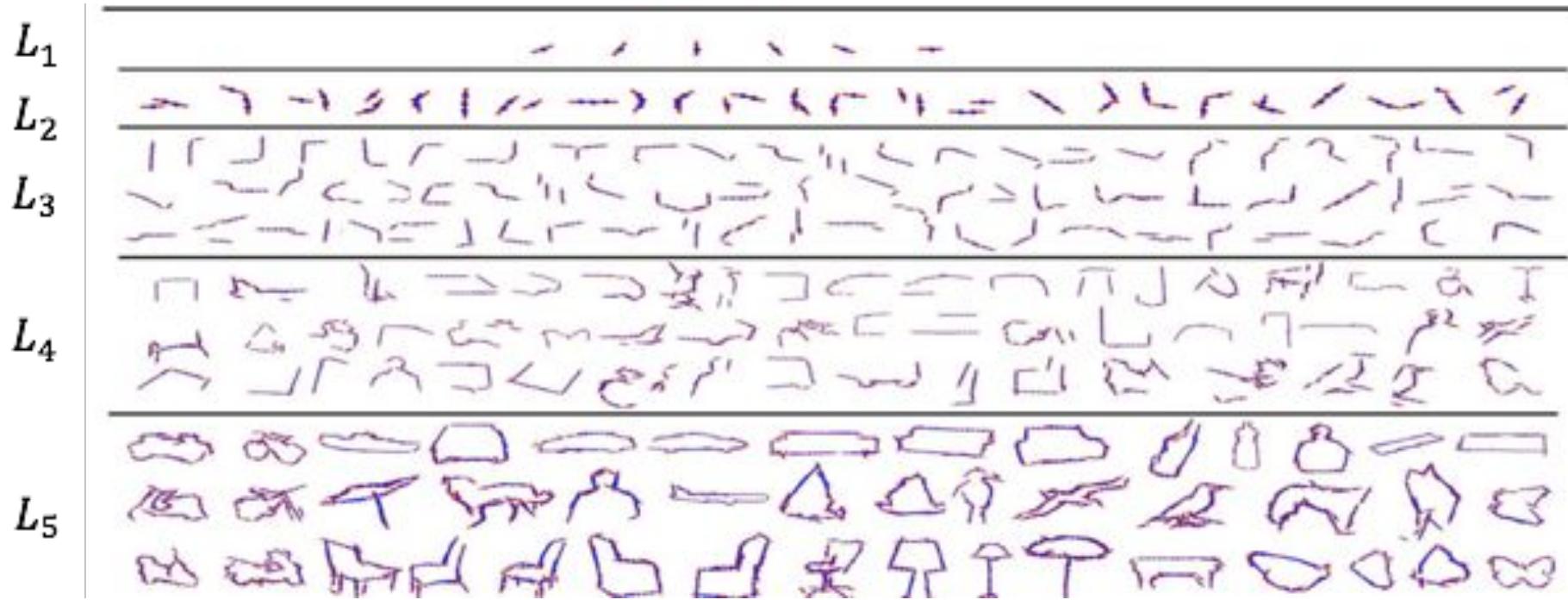
Fundamental Properties of Images

Property 3: Global Correlation.



Fundamental Properties of Images

Property 4: Compositionality of objects - a small set of building blocks (L_1) is enough to build complex objects (L_5) via 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

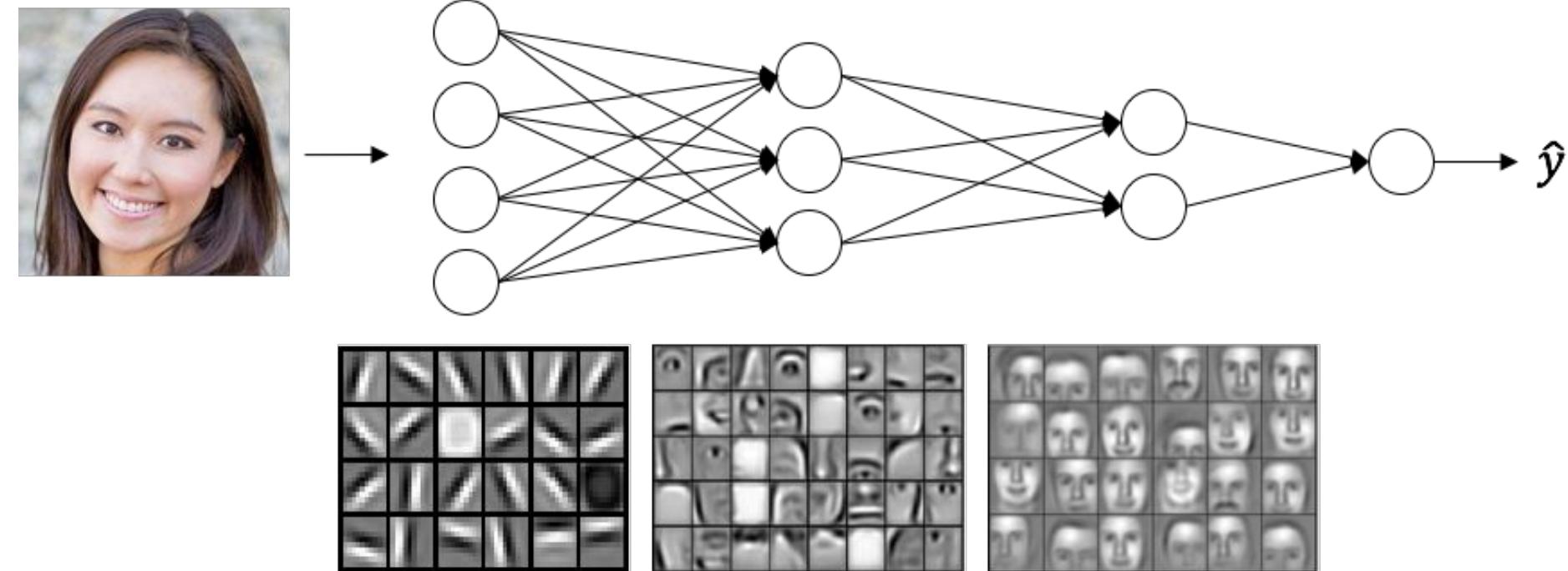


Image Filtering (linear)

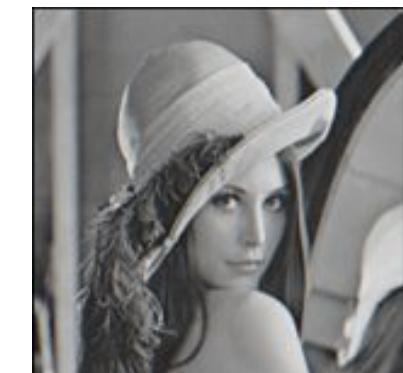
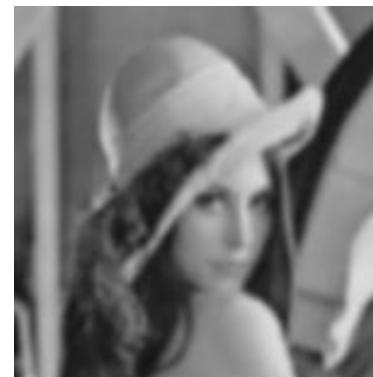
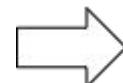
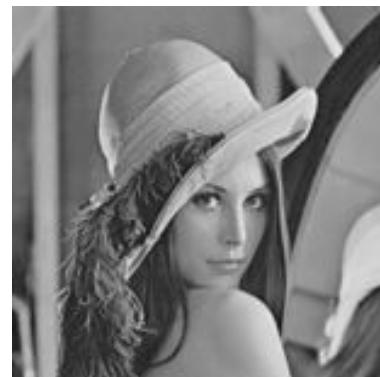


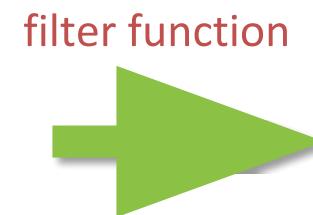
Image Filtering (linear)

- Each novel output pixel value $O(x,y)$ is a 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)

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

10	5	3
4	5	1
1	1	7

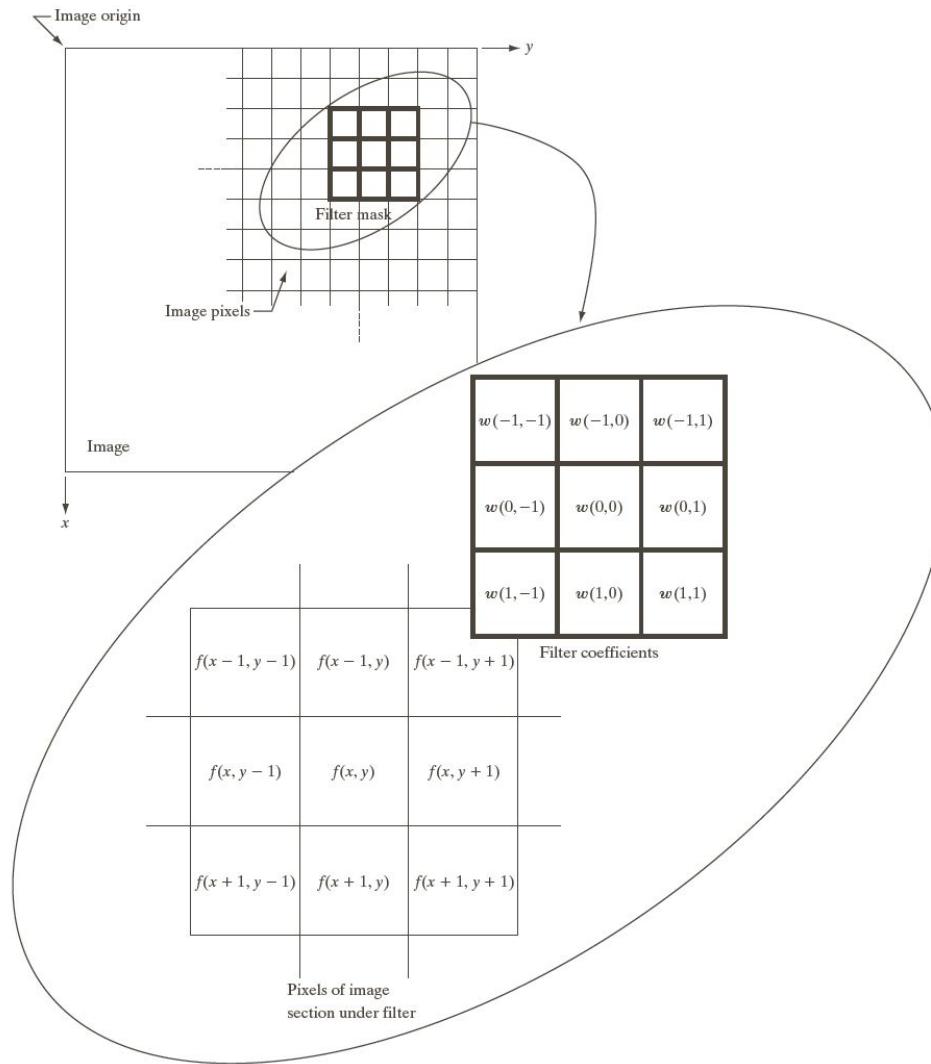
I Input Image



	7	

O Output image

Spatial Filtering



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: $O = k \otimes I$

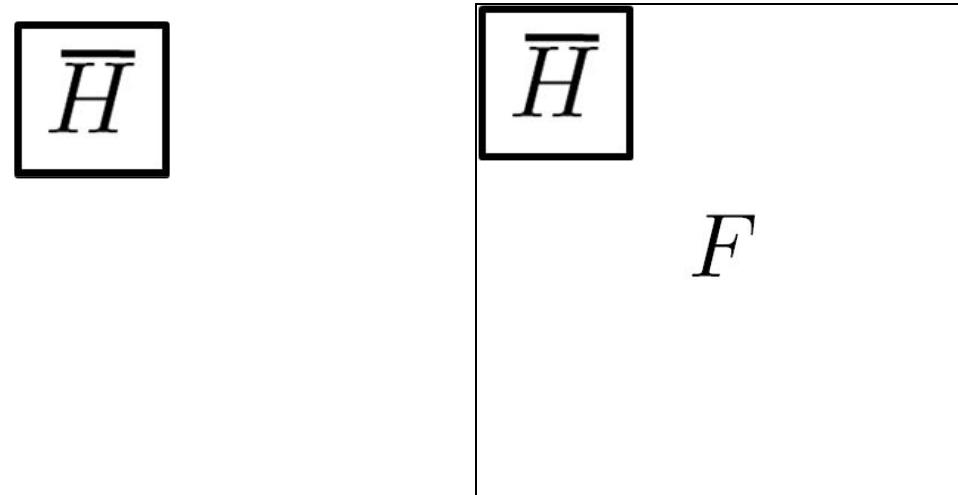
- Convolution:
$$O[x, y] = \sum_{s=-a}^a \sum_{t=-b}^b k[s, t] I[x-s, y-t]$$

Symbol: $O = k * I$

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

For symmetric kernels there is no difference !!!

Convolution



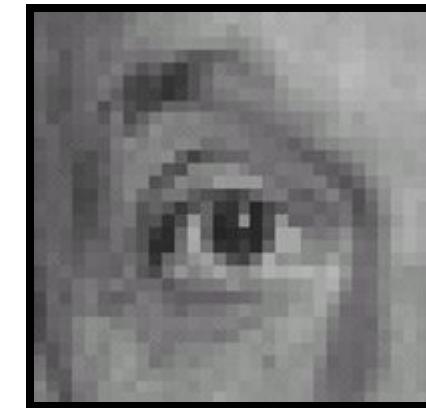
		Padded f								
		0	0	0	0	0	0	0	0	0
Origin $f(x, y)$		0	0	0	0	0	0	0	0	0
0 0 0 0 0		0	0	0	0	1	0	0	0	0
0 0 0 0 0		$w(x, y)$	0	0	0	0	0	0	0	0
0 0 1 0 0		1 2 3	0	0	0	0	0	0	0	0
0 0 0 0 0		4 5 6	0	0	0	0	0	0	0	0
0 0 0 0 0		7 8 9	0	0	0	0	0	0	0	0
(a)		(b)								
Initial position for w		Full correlation result				Cropped correlation result				
1 2 3		0	0	0	0	0	0	0	0	0
4 5 6		0	0	0	0	0	0	0	0	0
7 8 9		0	0	0	0	0	0	0	0	0
0 0 0 0 0 0		0	0	0	9	8	7	0	0	0
0 0 0 0 1 0		0	0	0	6	5	4	0	0	0
0 0 0 0 0 0		0	0	0	3	2	1	0	0	0
0 0 0 0 0 0		0	0	0	0	0	0	0	0	0
0 0 0 0 0 0		0	0	0	0	0	0	0	0	0
0 0 0 0 0 0		0	0	0	0	0	0	0	0	0
(c)		(d)				(e)				
Rotated w		Full convolution result				Cropped convolution result				
9 8 7		0	0	0	0	0	0	0	0	0
6 5 4		0	0	0	0	0	0	0	0	0
3 2 1		0	0	0	0	0	0	0	0	0
0 0 0 0 0 0		0	0	0	1	2	3	0	0	0
0 0 0 0 1 0		0	0	0	4	5	6	0	0	0
0 0 0 0 0 0		0	0	0	7	8	9	0	0	0
0 0 0 0 0 0		0	0	0	0	0	0	0	0	0
0 0 0 0 0 0		0	0	0	0	0	0	0	0	0
0 0 0 0 0 0		0	0	0	0	0	0	0	0	0
(f)		(g)				(h)				

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

Linear filters: examples

 \ast

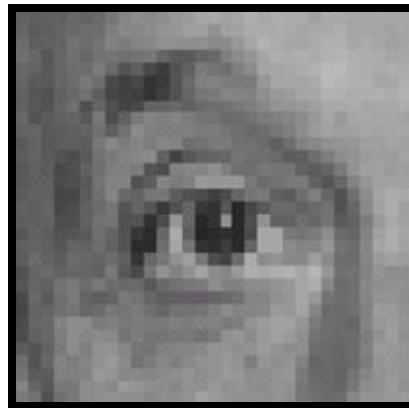
0	0	0
0	1	0
0	0	0

 $=$ 

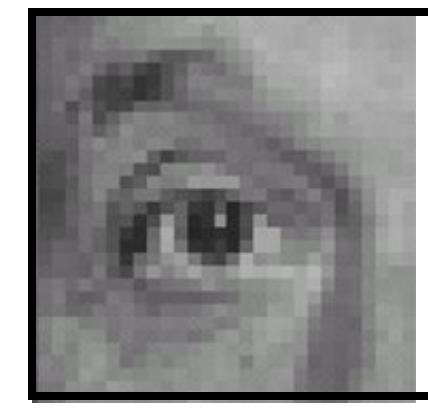
Original

Identical image

Linear filters: examples

 \ast

0	0	0
1	0	0
0	0	0

 $=$ 

Original

Shifted left
By 1 pixel

Linear filters: examples

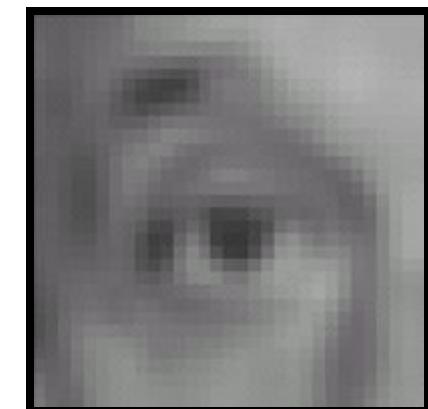


Original



$$\frac{1}{9}$$

1	1	1
1	1	1
1	1	1



Blur (with a mean filter)

Linear filters: examples



Original

$$\text{Original} \quad * \left(\begin{array}{ccc} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{array} \right) - \frac{1}{9} \left(\begin{array}{ccc} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{array} \right) = \text{Result}$$



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: $(n \times n) * (f \times f) \rightarrow (n - f + 1) \times (n - f + 1)$
- "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:

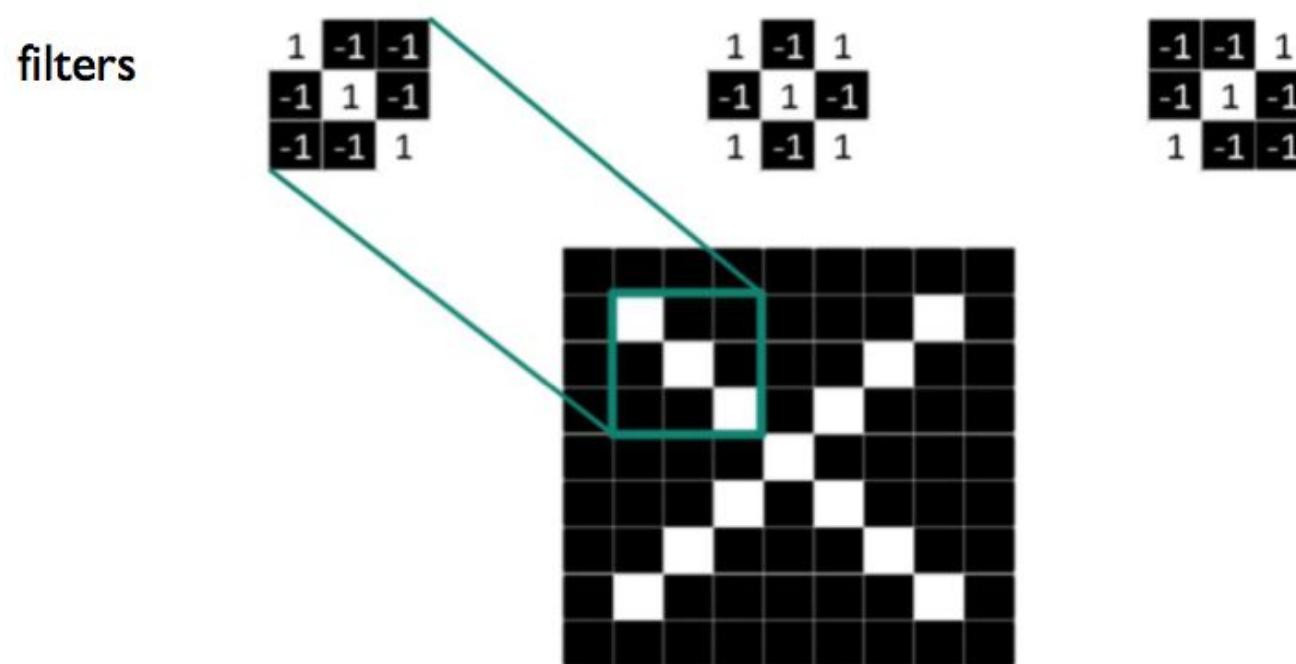
$$(n \times n) * (f \times f) \longrightarrow \left(\frac{n+2p-f}{s} + 1\right) \times \left(\frac{n+2p-f}{s} + 1\right)$$

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

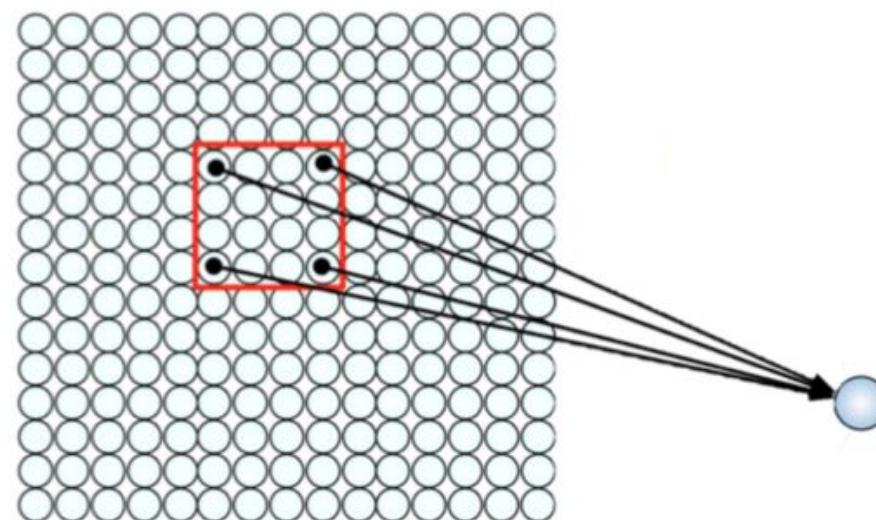
Motivation

- Compare image features found in the images



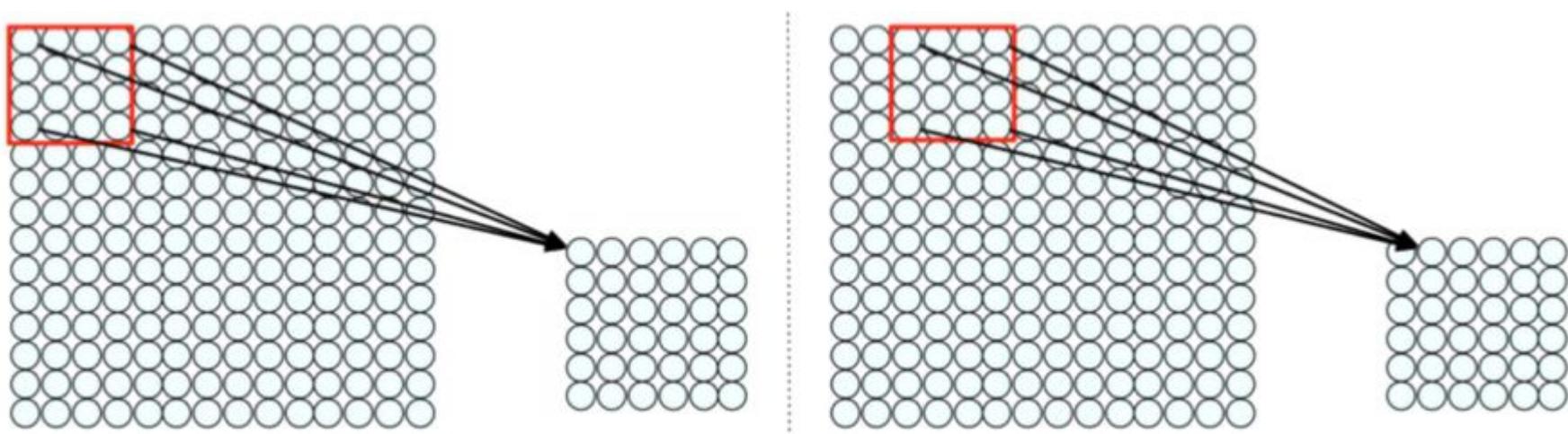
Use Spatial Structure

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



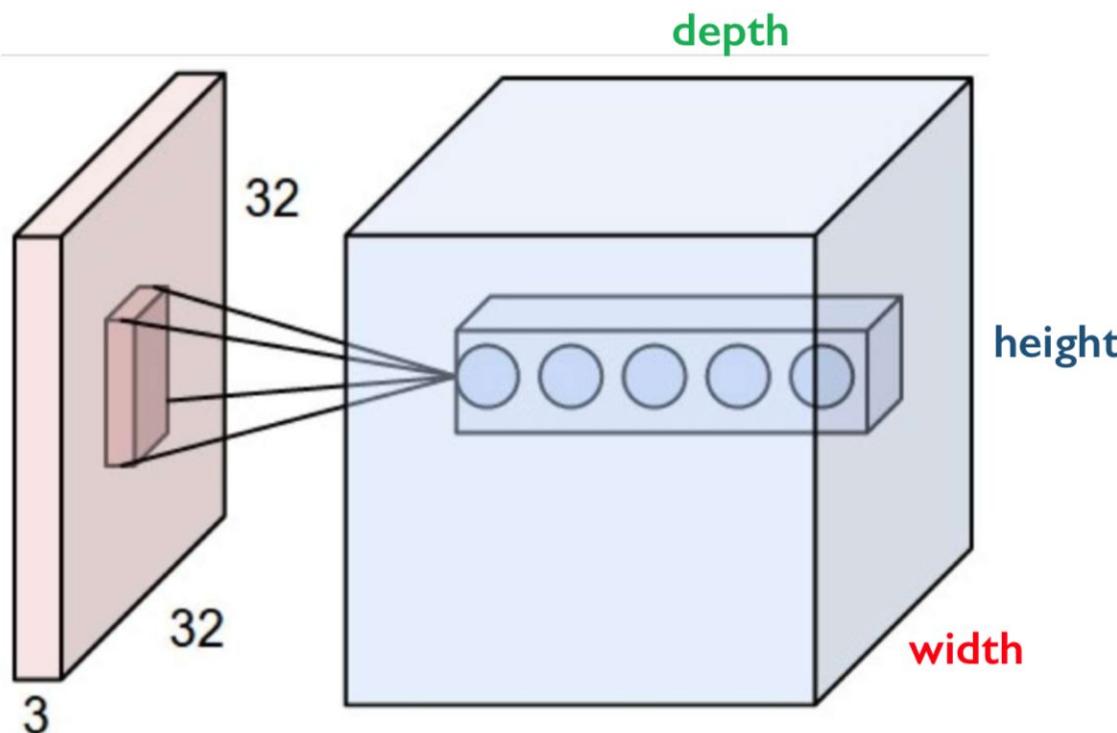
Use Spatial Structure

- 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.

Spatial Arrangement of Output Volume



Layer Dimensions:

$h \times w \times d$

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

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.

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.

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		

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	

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.

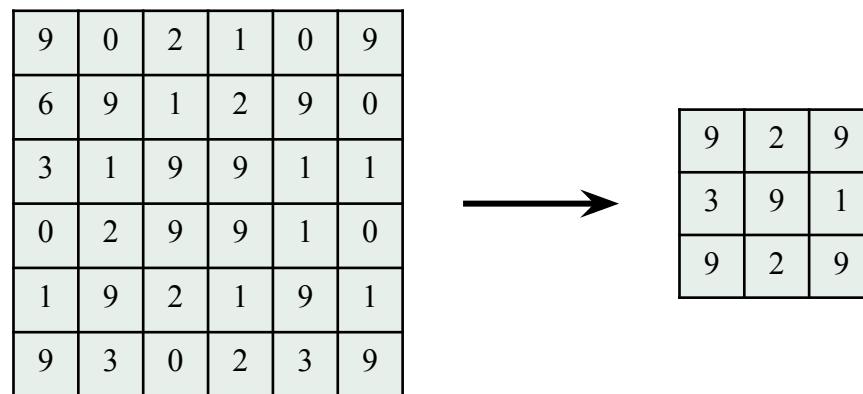
9	0	2	1	0	9
6	9	1	2	9	0
3	1	9	9	1	1
0	2	9	9	1	0
1	9	2	1	9	1
9	3	0	2	3	9



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



Other used pooling strategies:

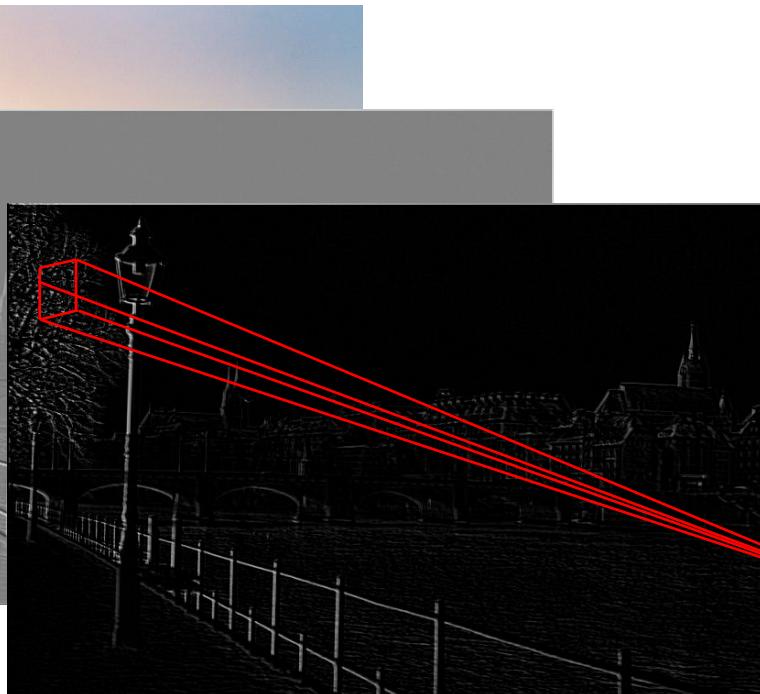
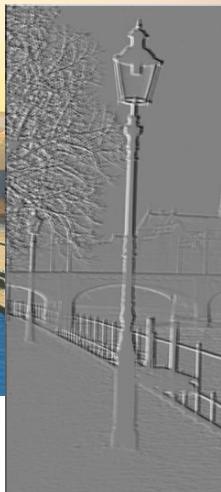
- Average Pooling

Pooling Layer

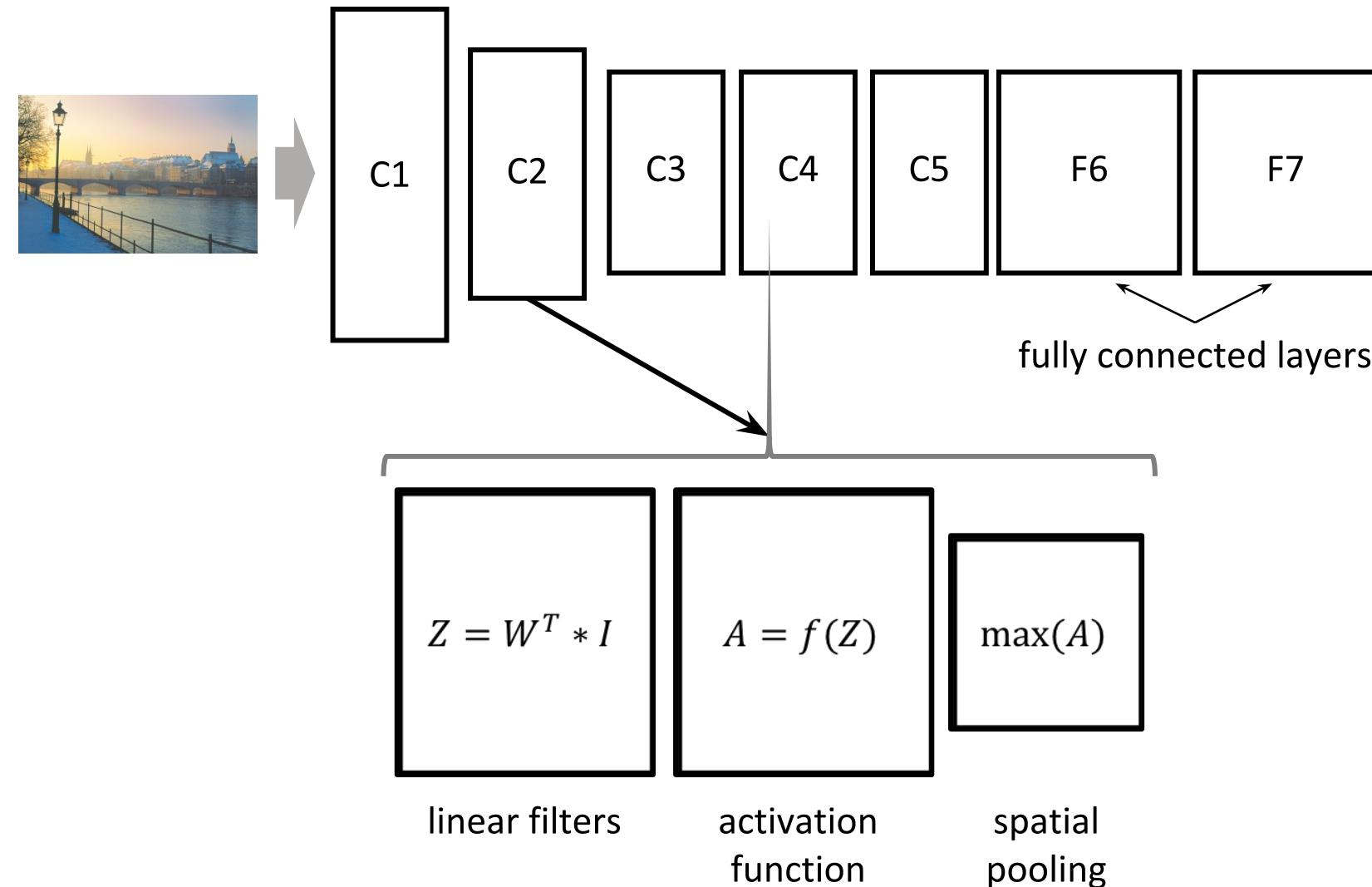
Input Image



Feature Maps

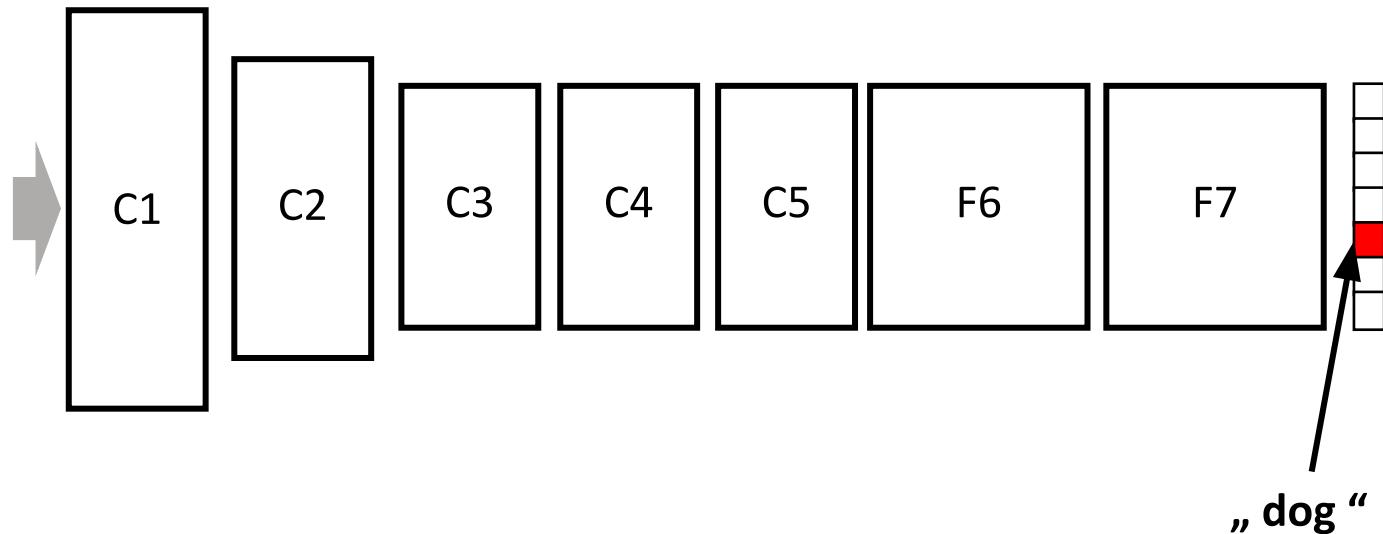


Layered CNN Architecture



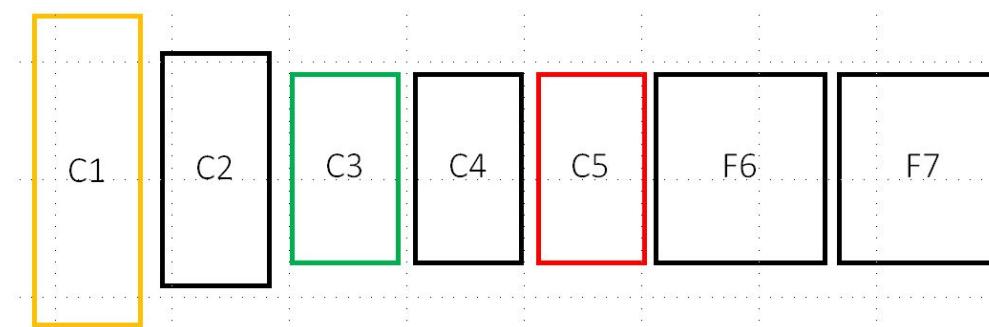
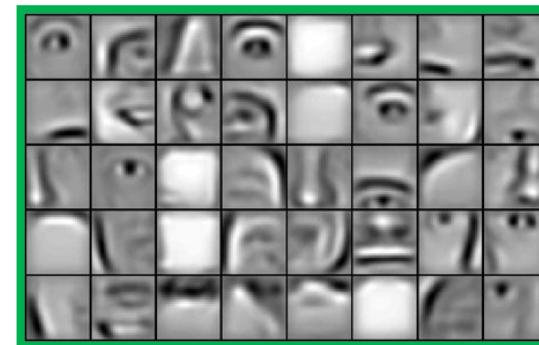
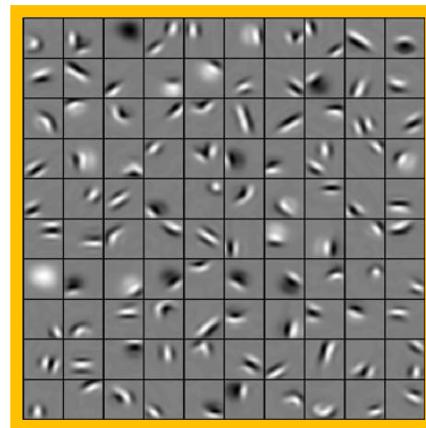
Classification

- Add an output layer and train the weights with backpropagation



Visualization of the learned weights

- When trained for face detection:



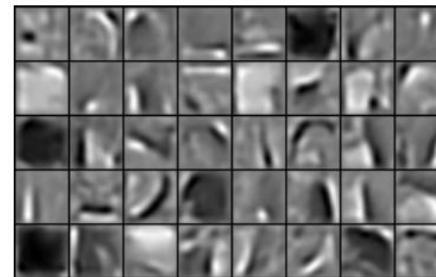
Visualization of the learned weights

- When trained for different object classes:

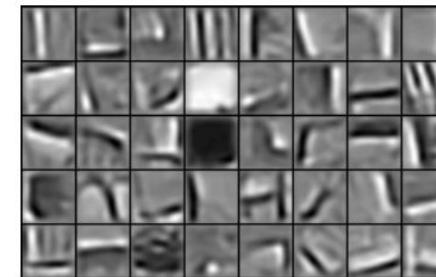
cars



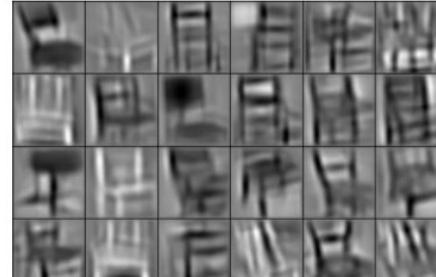
elephants



chairs



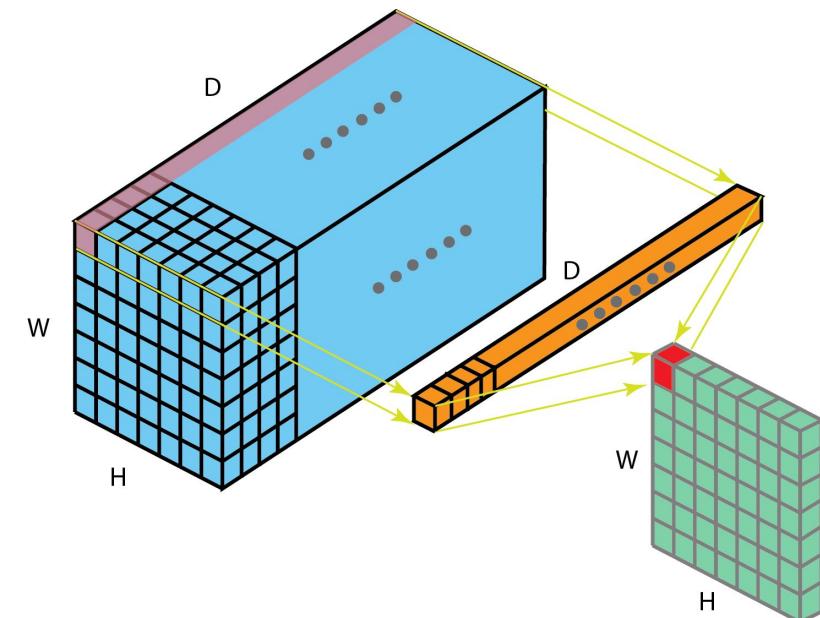
faces, cars, airplanes, motorbikes



1x1 Convolutions

Network in network

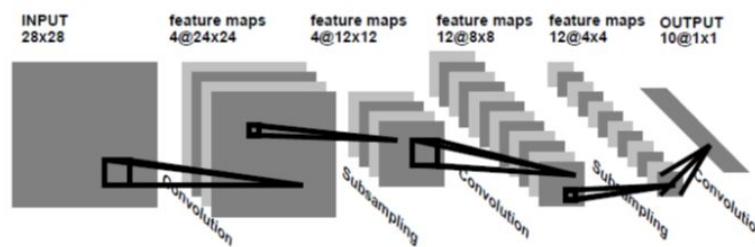
- Used for convolutions on the feature maps.
- Pooling shrinks the height and width of an image/feature map
- 1x1 Convolutions shrinks the number of filters
 - Ex. $28 \times 28 \times 192$ to $28 \times 28 \times 32$



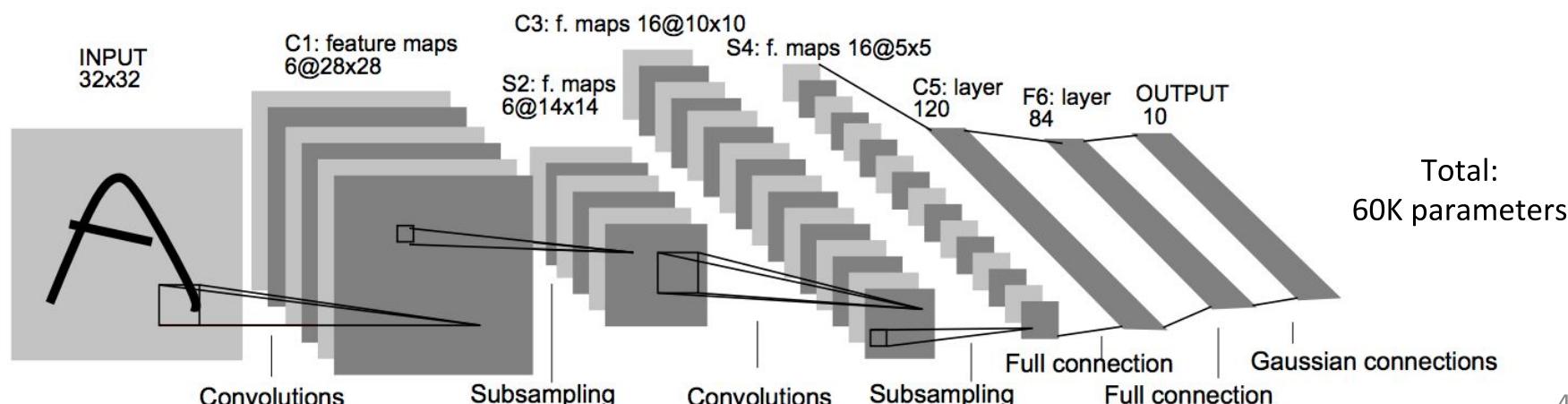
Popular/historical CNN Architectures

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.

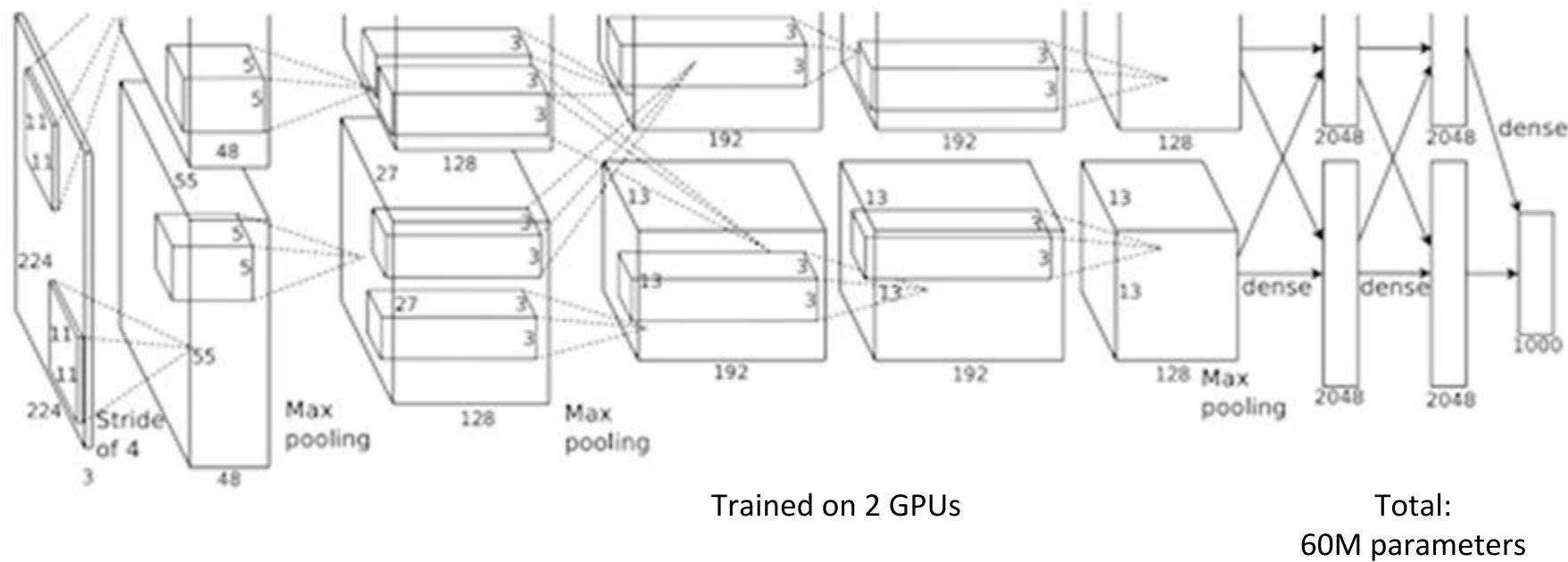


Today a softmax output is used.



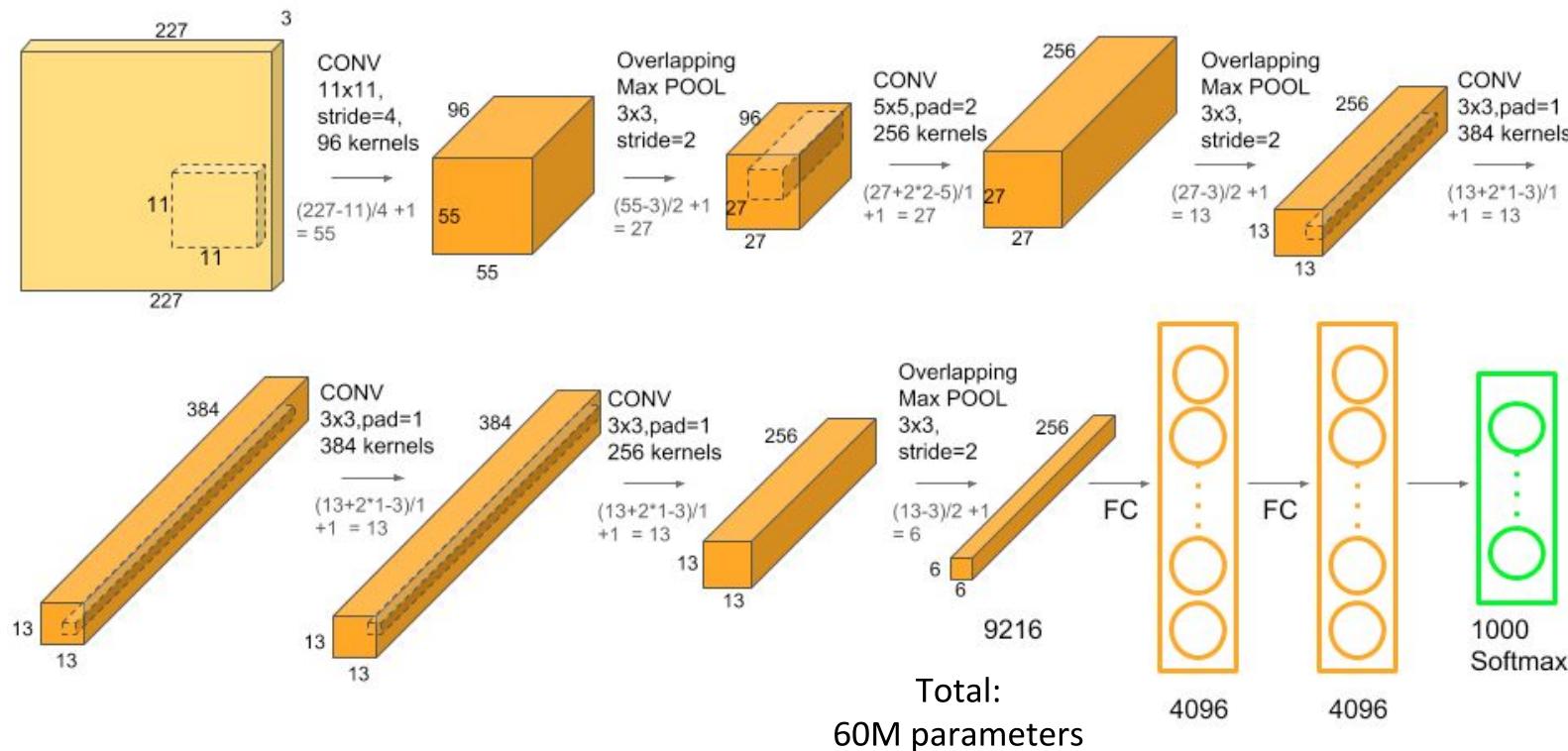
AlexNet - 2012

- 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).



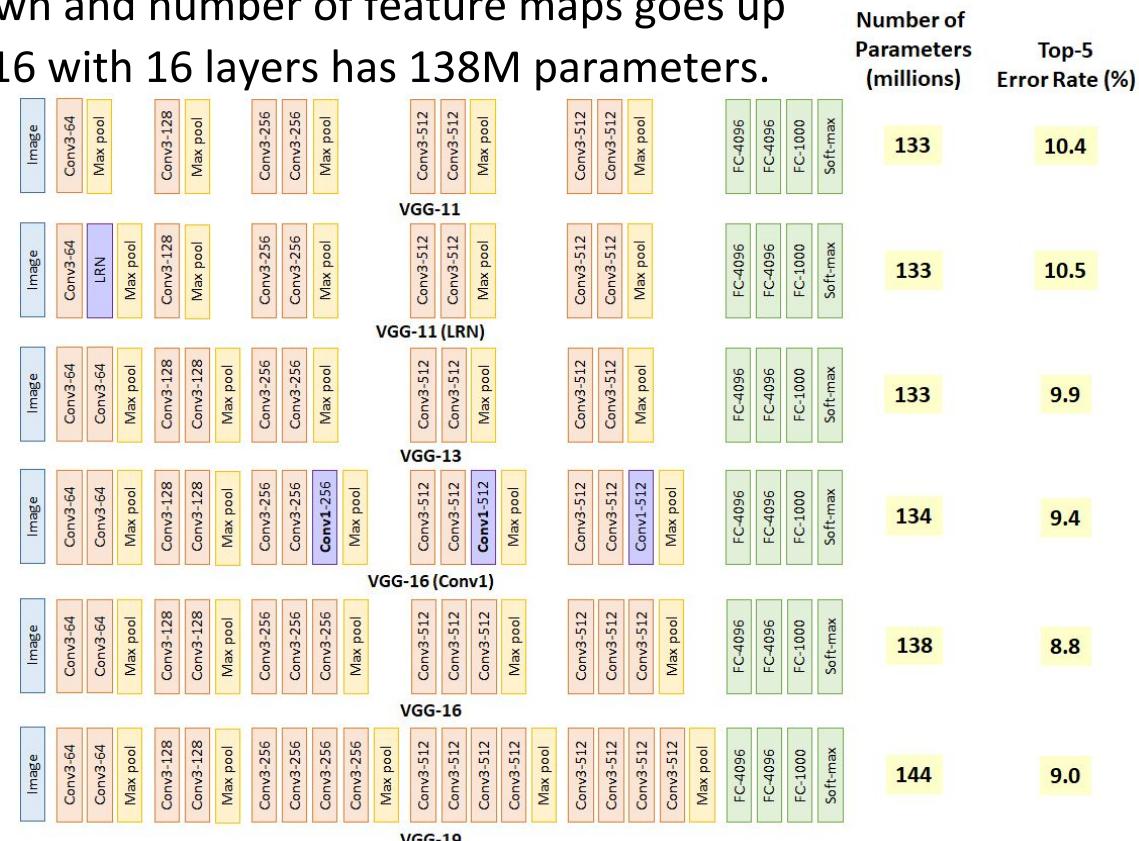
AlexNet - 2012

- 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).



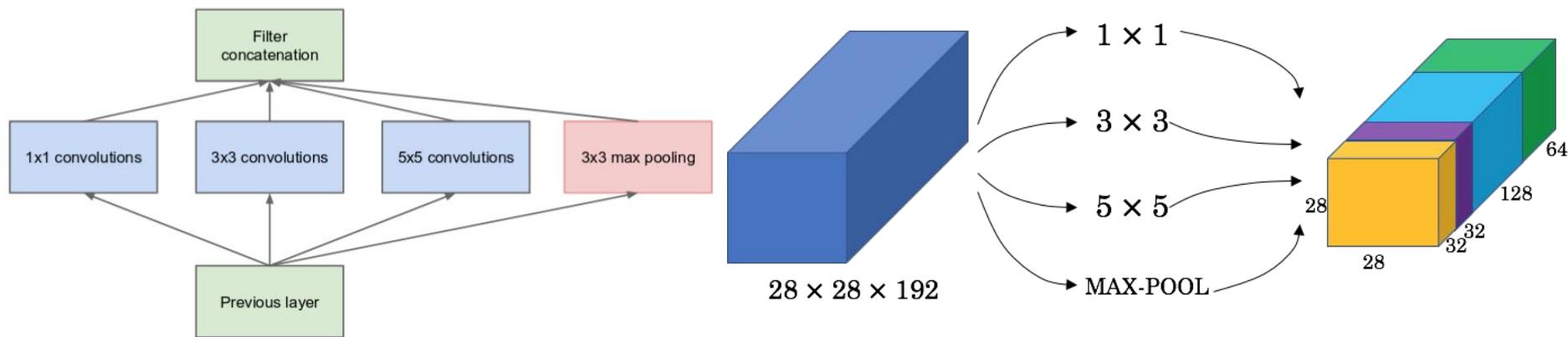
VGG (Visual Geometry Group) - 2014

- 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.

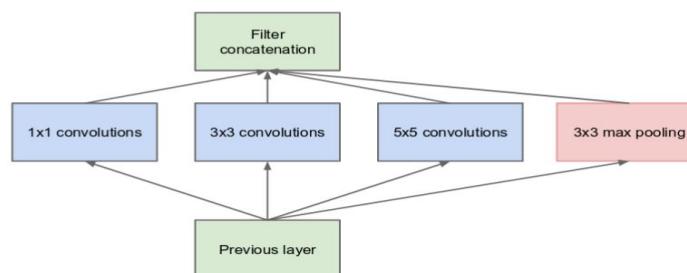


InceptionNet (GoogLeNet) - 2014

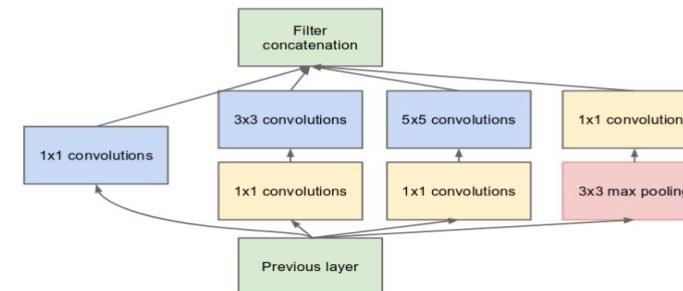
- Do not pick the operation to do - do them all.
 - 1×1 convolution
 - 3×3 convolution (same convolutions)
 - 5×3 convolution (same convolutions)
 - Pooling (with padding)
- Repeat the simple structure multiple times



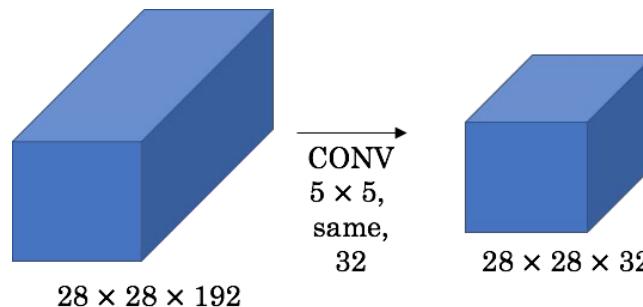
InceptionNet (GoogLeNet) - 2014



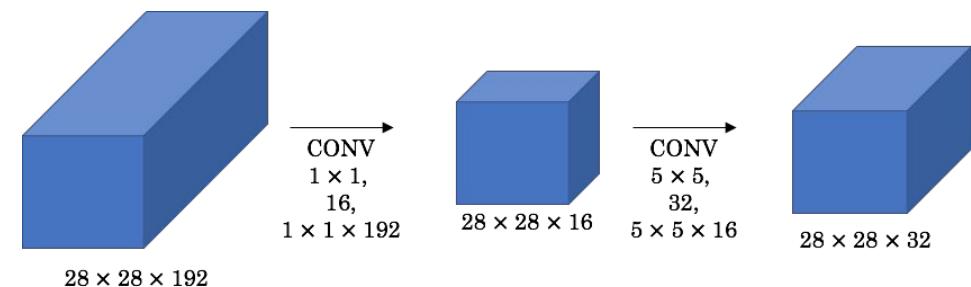
(a) Inception module, naïve version



(b) Inception module with dimension reductions



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



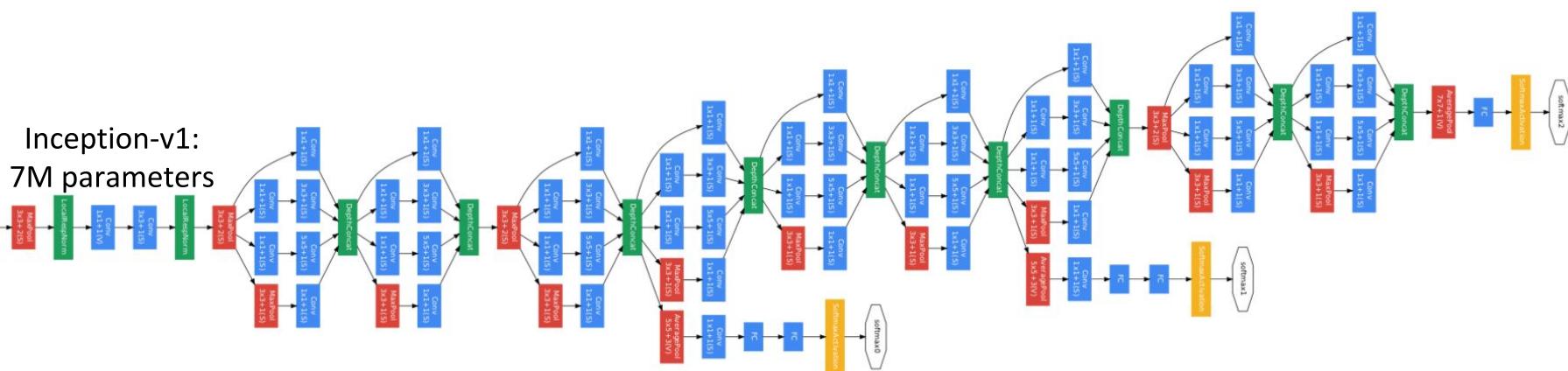
$$(28 \times 28 \times 16) \times 192 = 2.4M$$

$$(28 \times 28 \times 32) \times (5 \times 5 \times 16) = 10M$$

$$2.4 + 10M = 12.4M$$

InceptionNet (GoogLeNet) - 2014

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

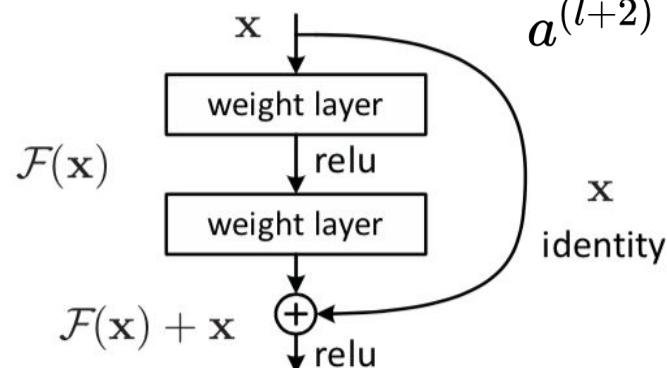


Meme citation in original paper:



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

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

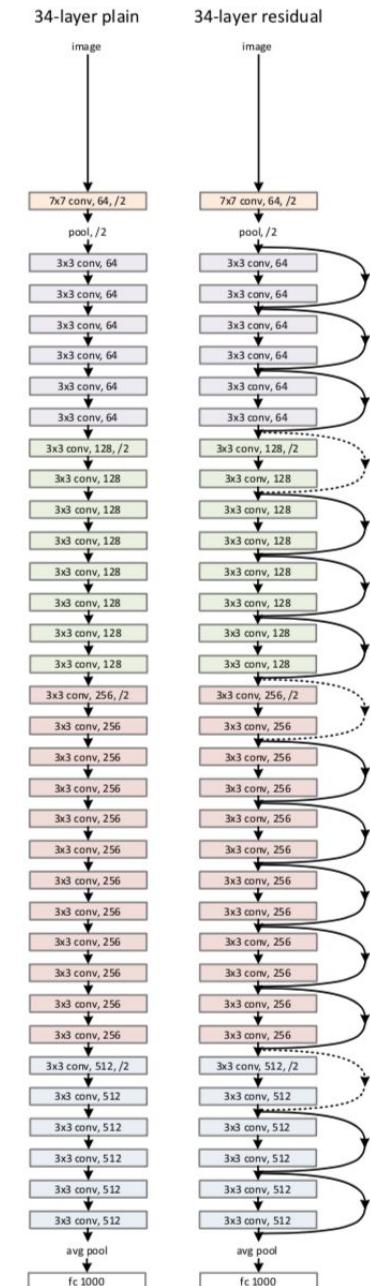


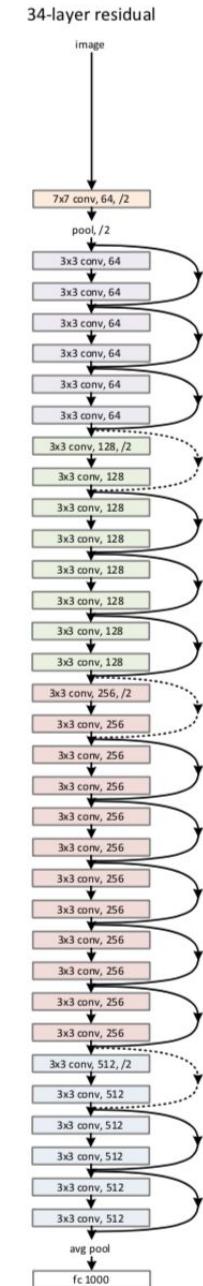
Figure 2. Residual learning: a building block.

Total (ResNet-152)
60M parameters

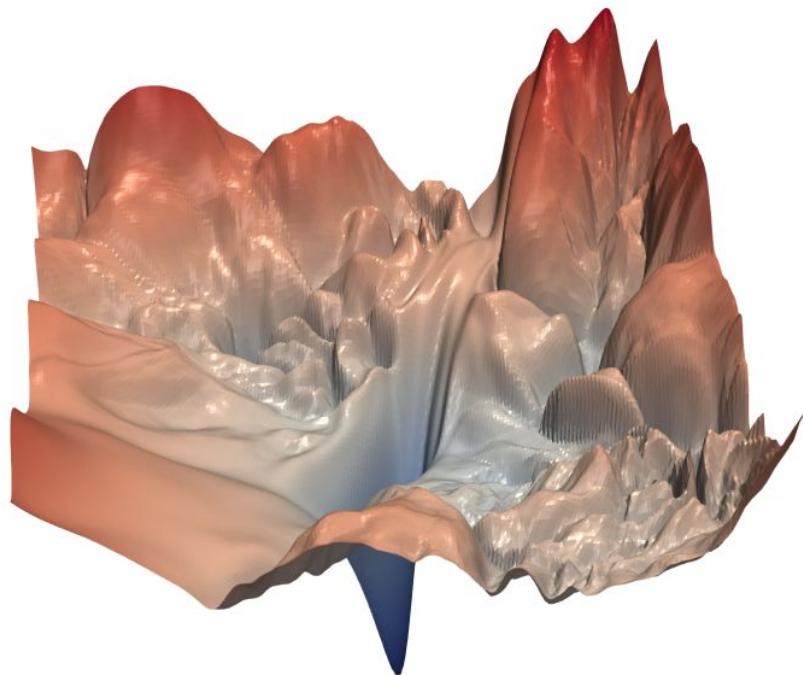
ResNet - 2015

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

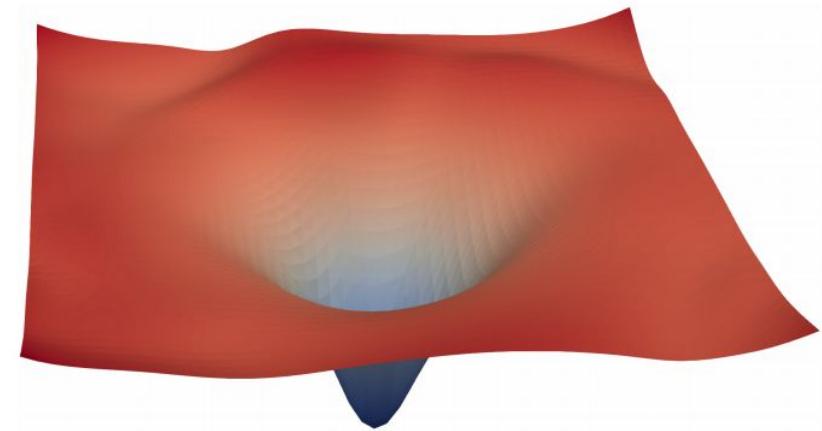
Table 4. Error rates (%) of **single-model** results on the ImageNet



Skip connections



(a) without skip connections

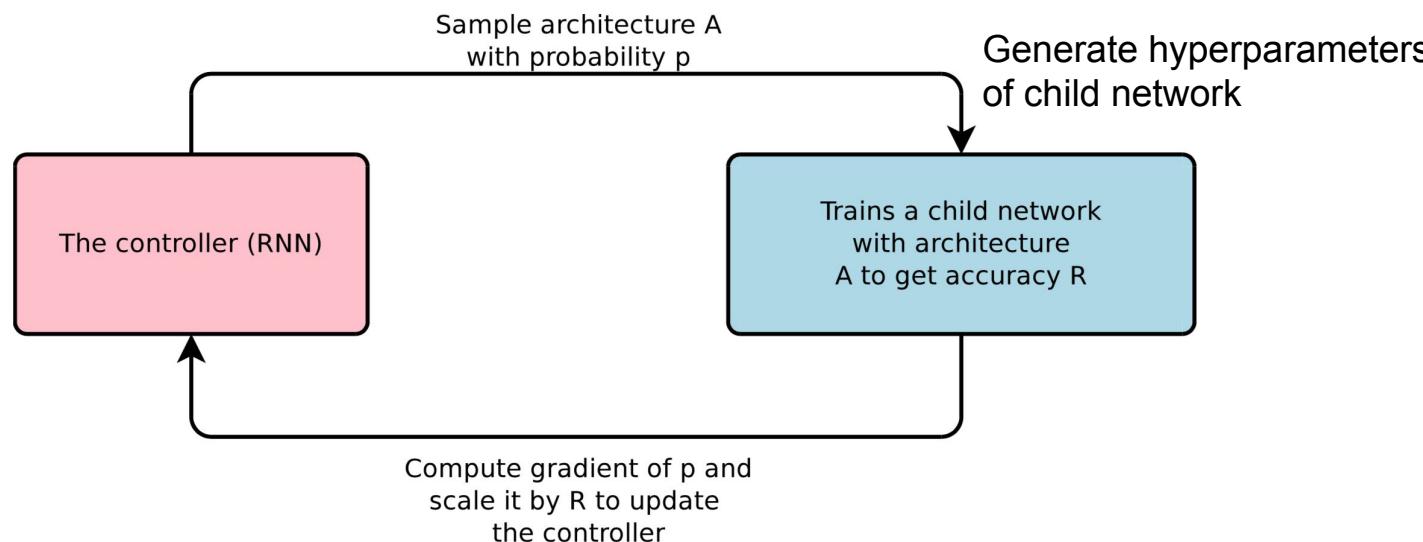


(b) with skip connections

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

Neural Architecture Search - 2017

NAS for finding good architectures with gradient-based search

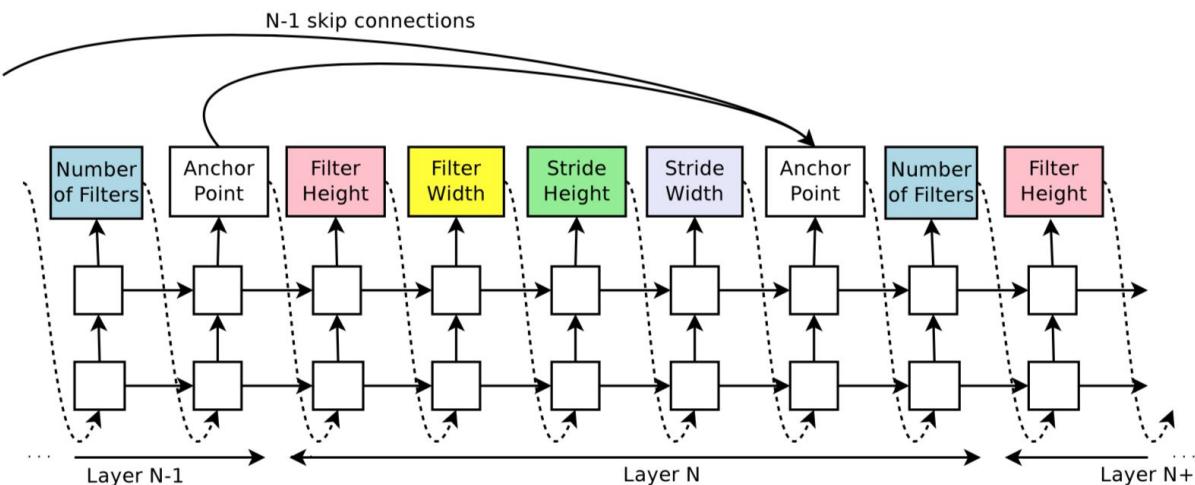


Learning for CIFAR-10 (image recognition) - Final test on ImageNet

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!

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.

U-Net - 2015

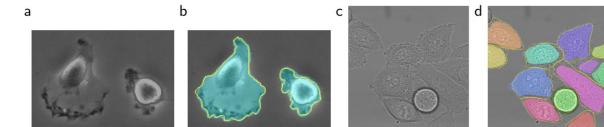
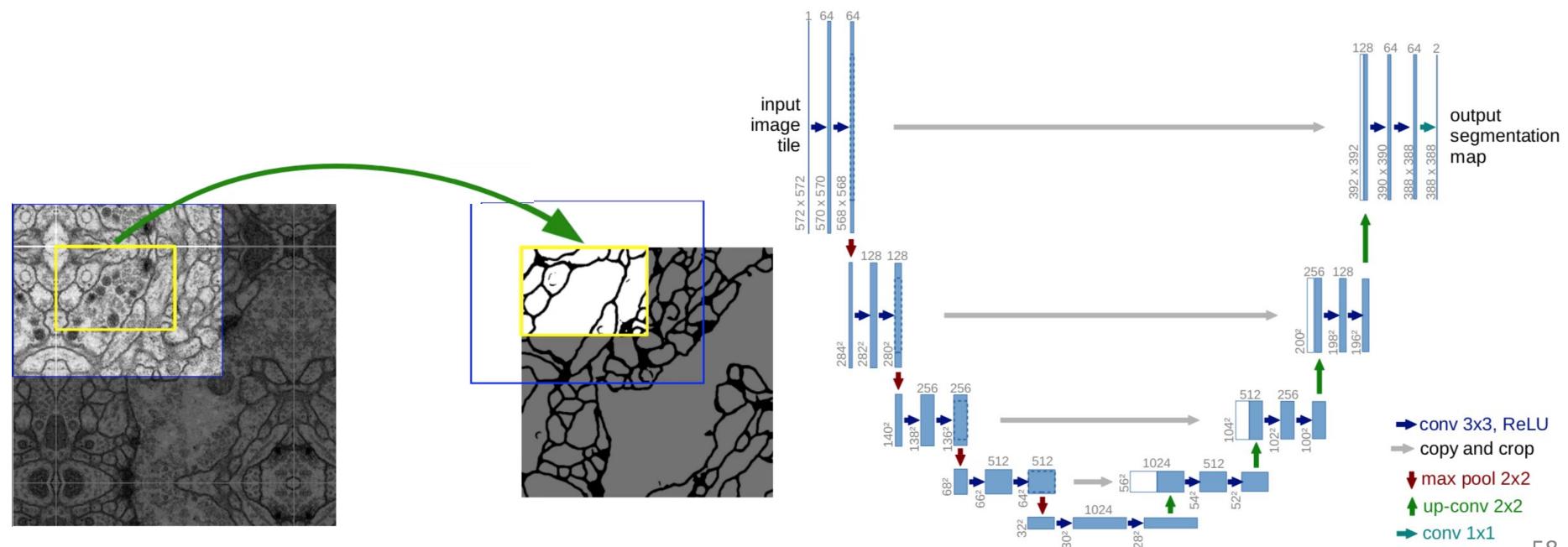


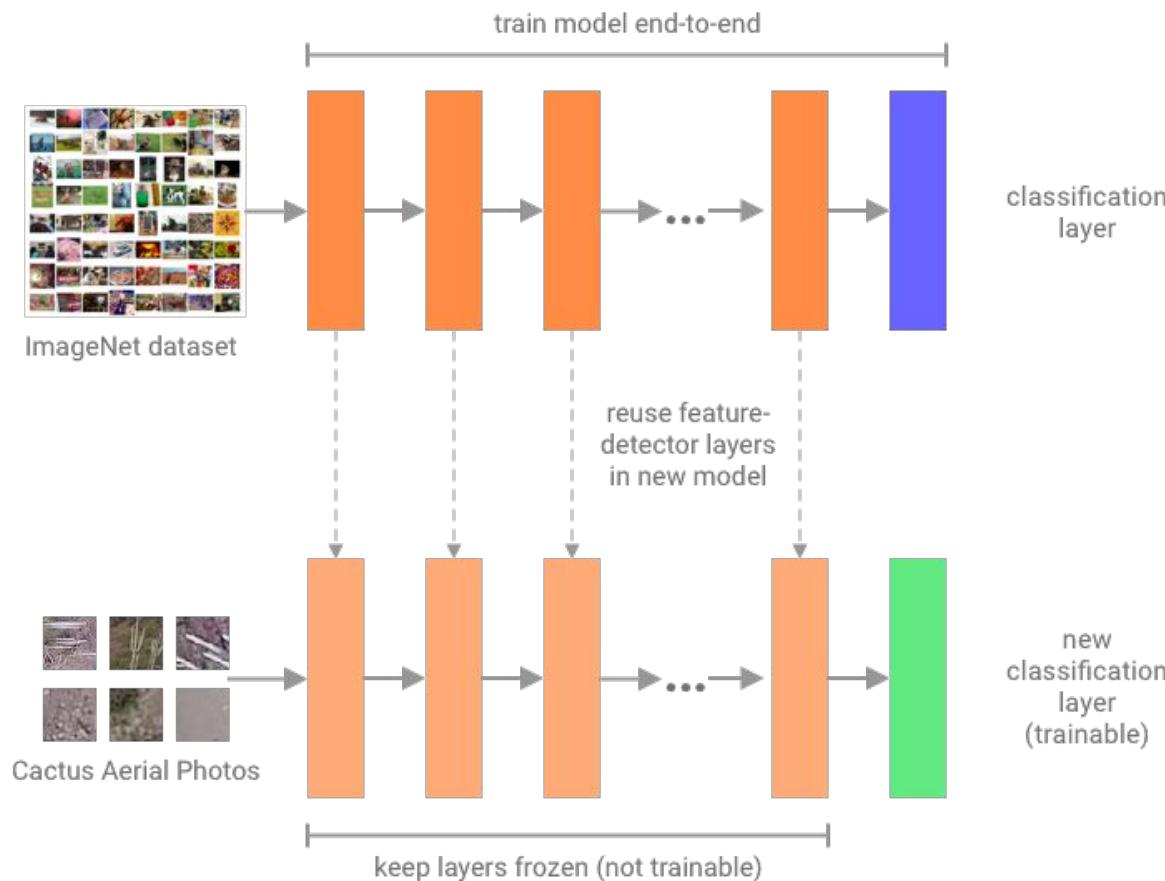
Fig. 4. Result on the ISBI cell tracking challenge. (a) part of an input image of the "PhC-U373" data set. (b) Segmentation result (cyan mask) with manual ground truth (yellow border) (c) input image of the "DIC-HeLa" data set. (d) Segmentation result (random colored masks) with manual ground truth (yellow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



Transfer learning

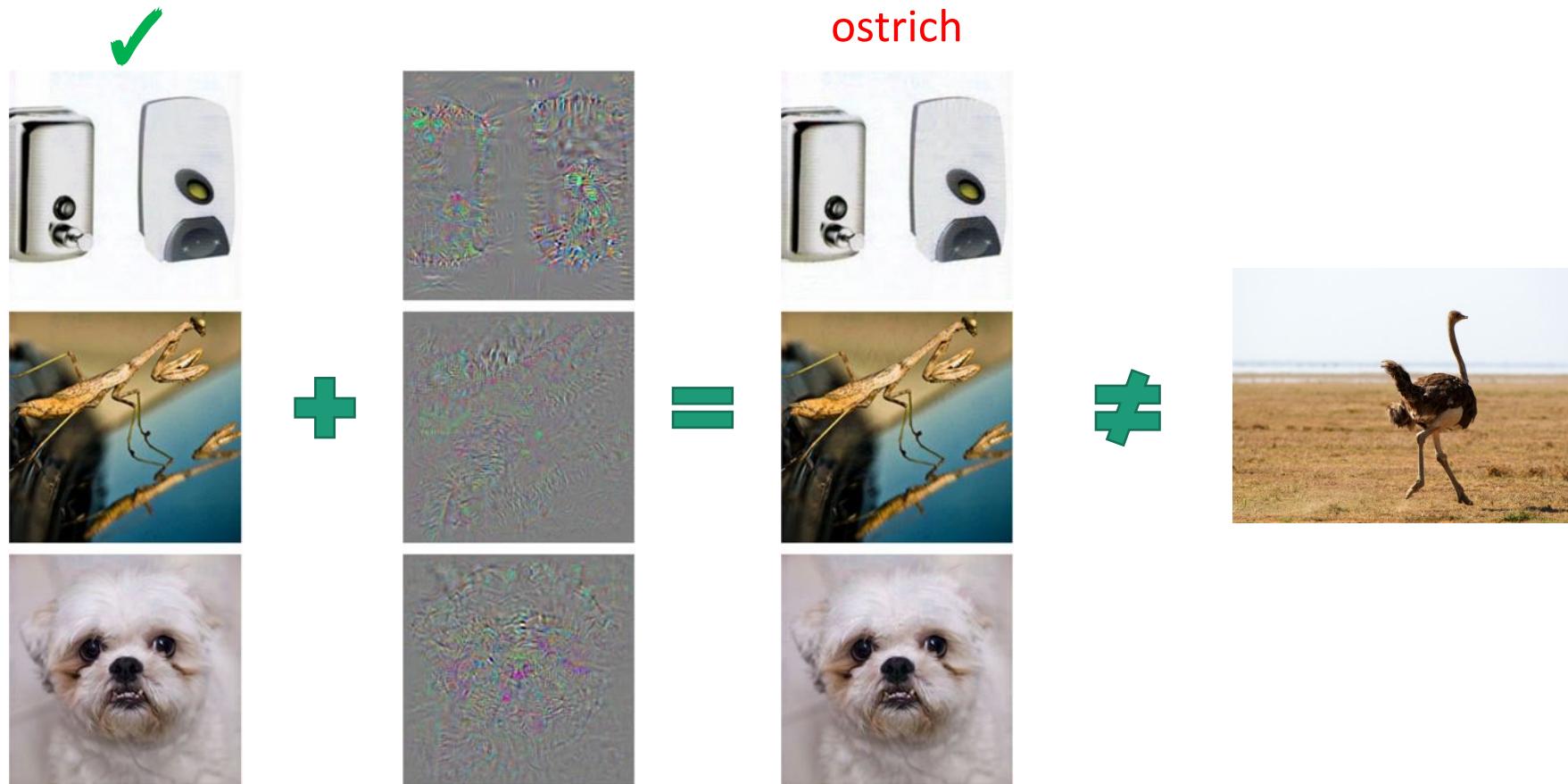
- Use a pretrained network on another tasks



See python notebooks for example on this.

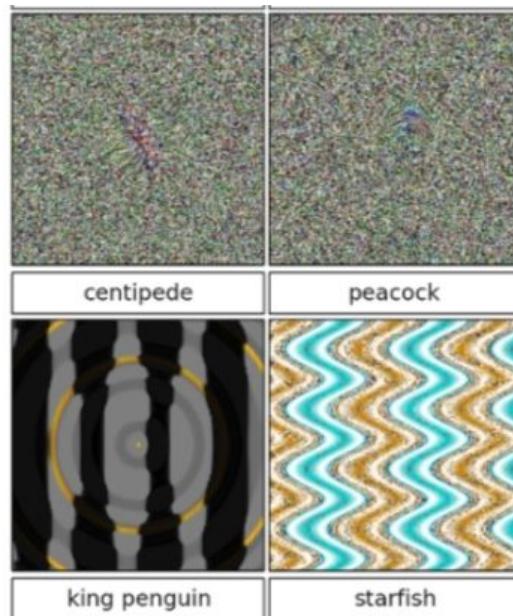
Adversarial attacks on images

- Adding the “right” noise induces miss-classification

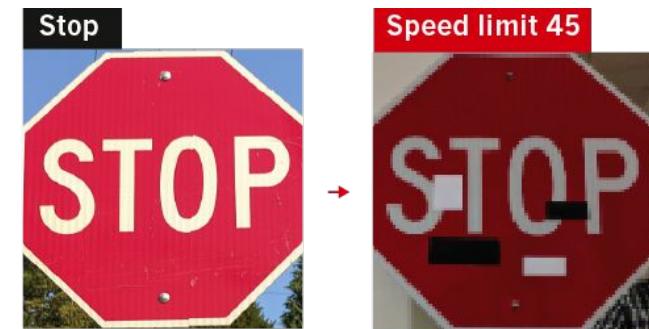


Adversarial attacks on images

- Generating “adversarial” examples – classification confidence > 99%



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



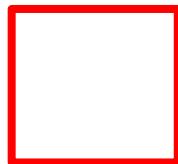
- Security risk for DNN systems: self-driving cars detecting a stop sign or a pedestrian.
- One does not even need to know the DNN and its weights. By sending enough requests to the model, the internal mechanism can be inferred.
- Deep learning systems still does not understand the world!

Beyond image classification

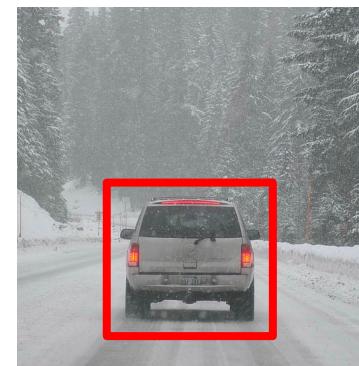
Object localization vs detection

Image classification

Naive method:
Sliding window



Classification with
localization



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

Detection



- 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?



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

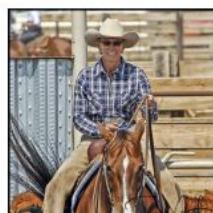
Region-CNN (R-CNN)

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



Nothing interesting!

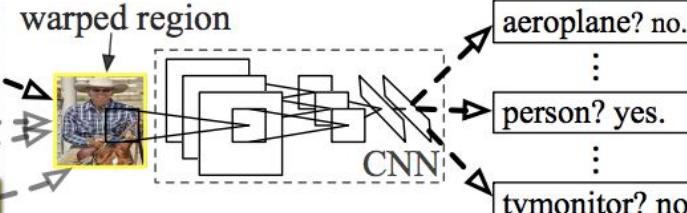
R-CNN: *Regions with CNN features*



1. Input image

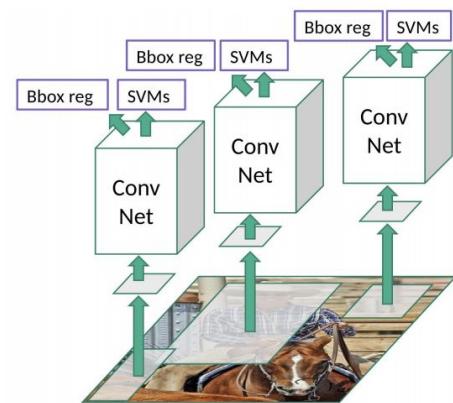


2. Extract region proposals (~2k)



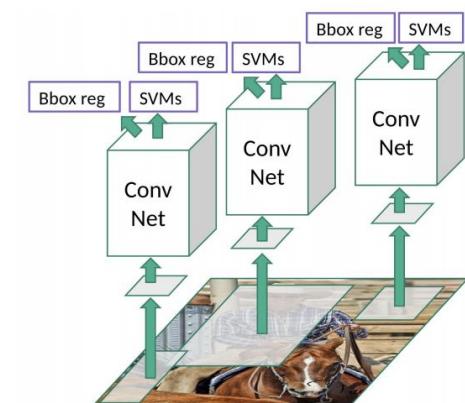
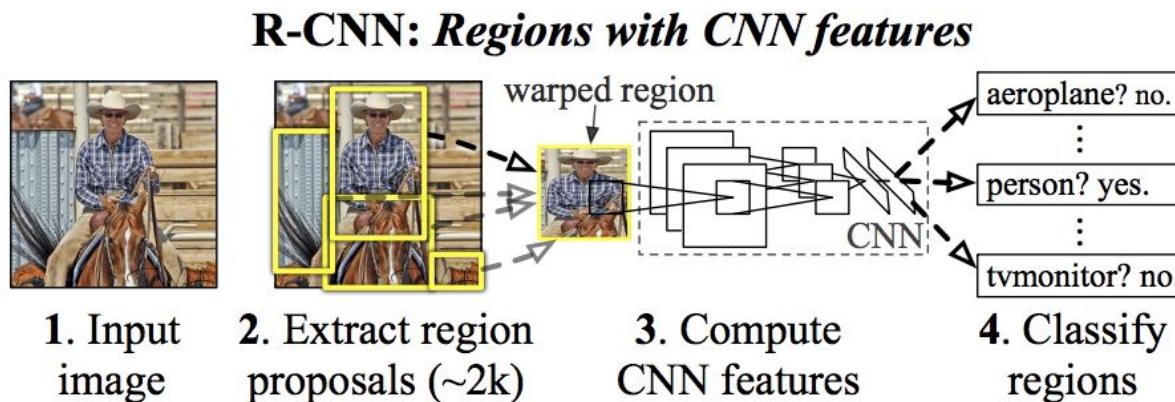
3. Compute CNN features

4. Classify regions

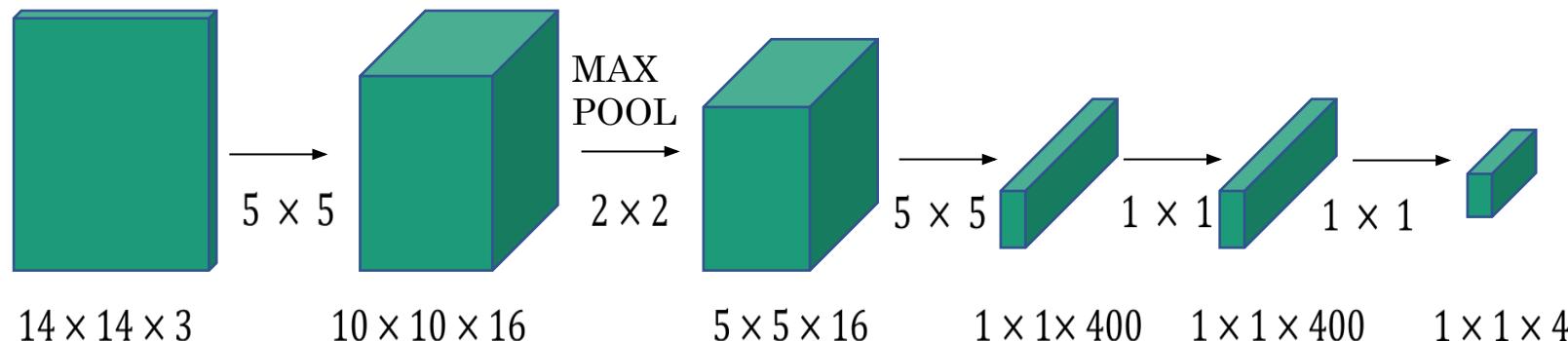
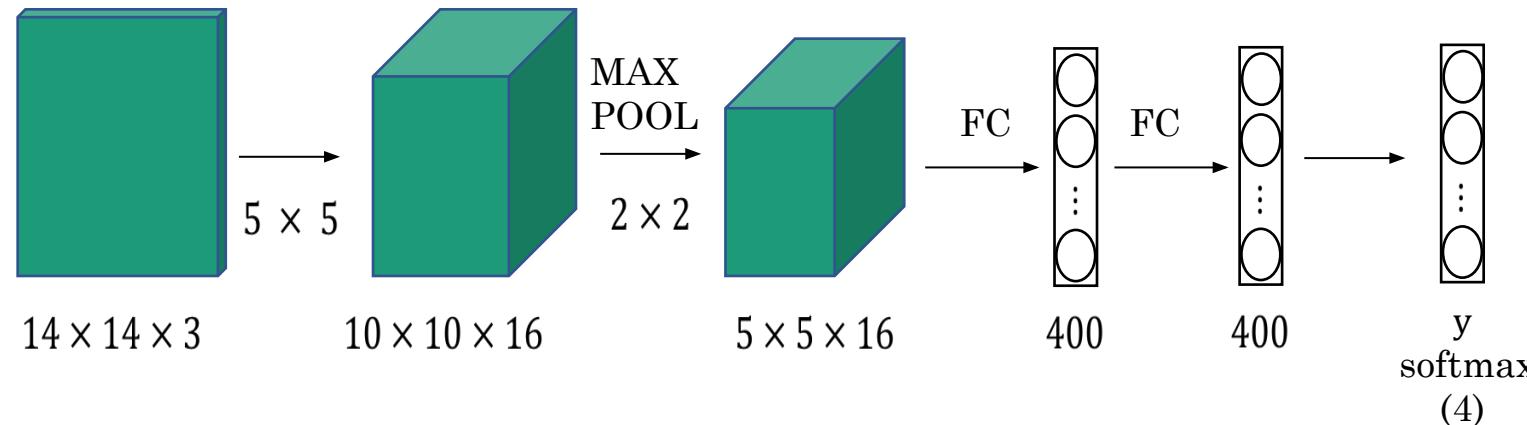


Region-CNN (R-CNN) problems

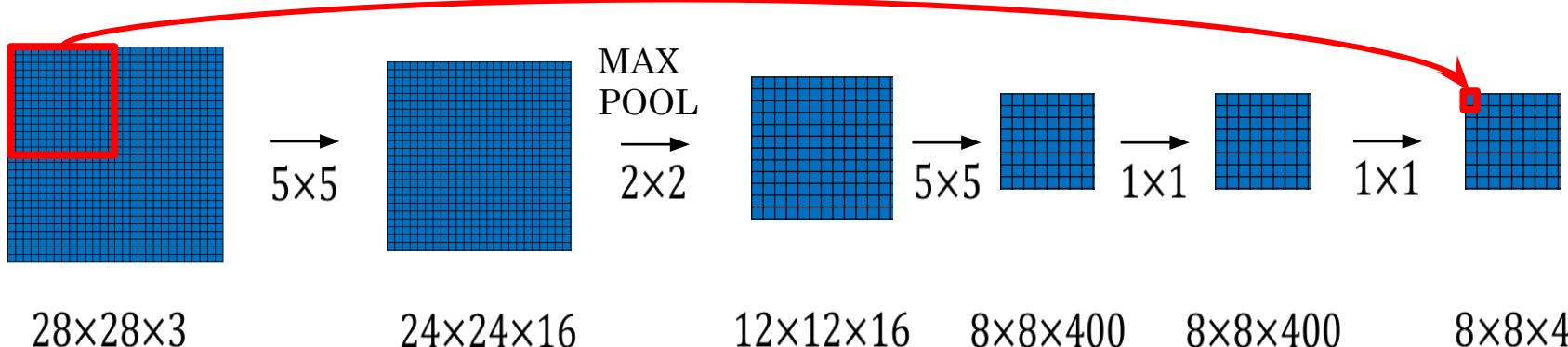
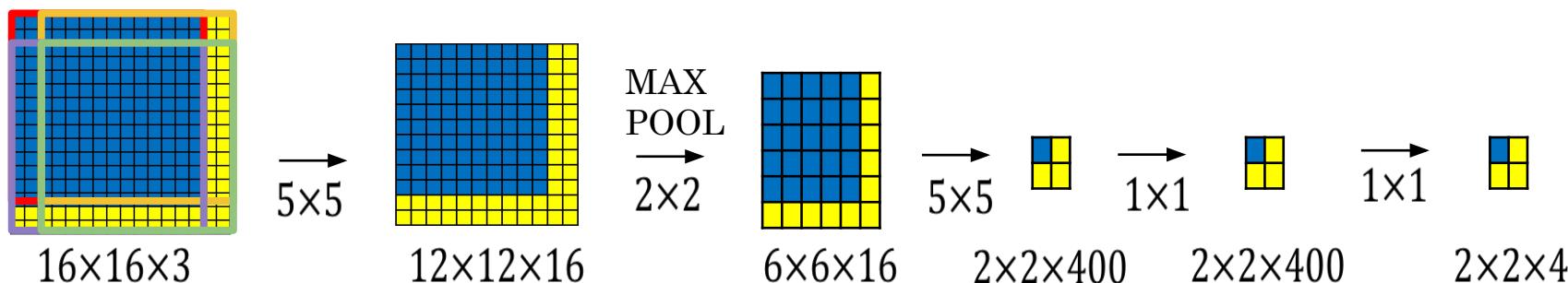
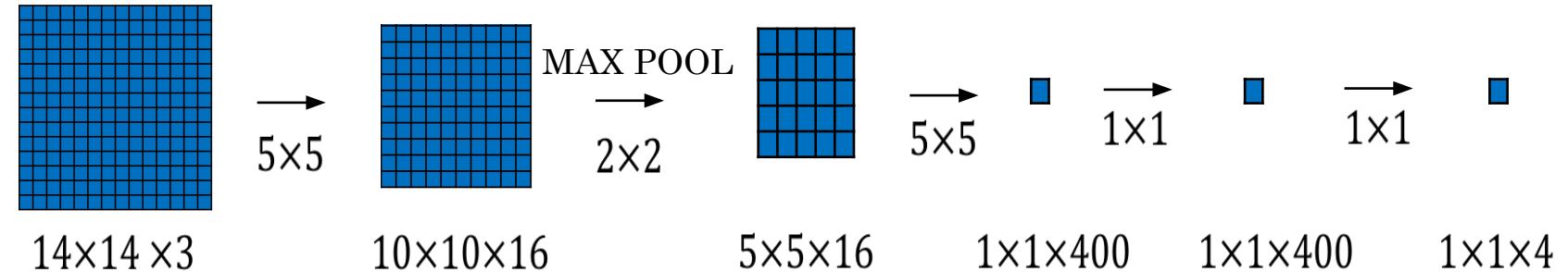
- It takes a long time to train the network where 2000 region proposals have to be classified.
- Not usable for real-time applications.
- No learning is happening for the region detections.



Fully connected to convolutional

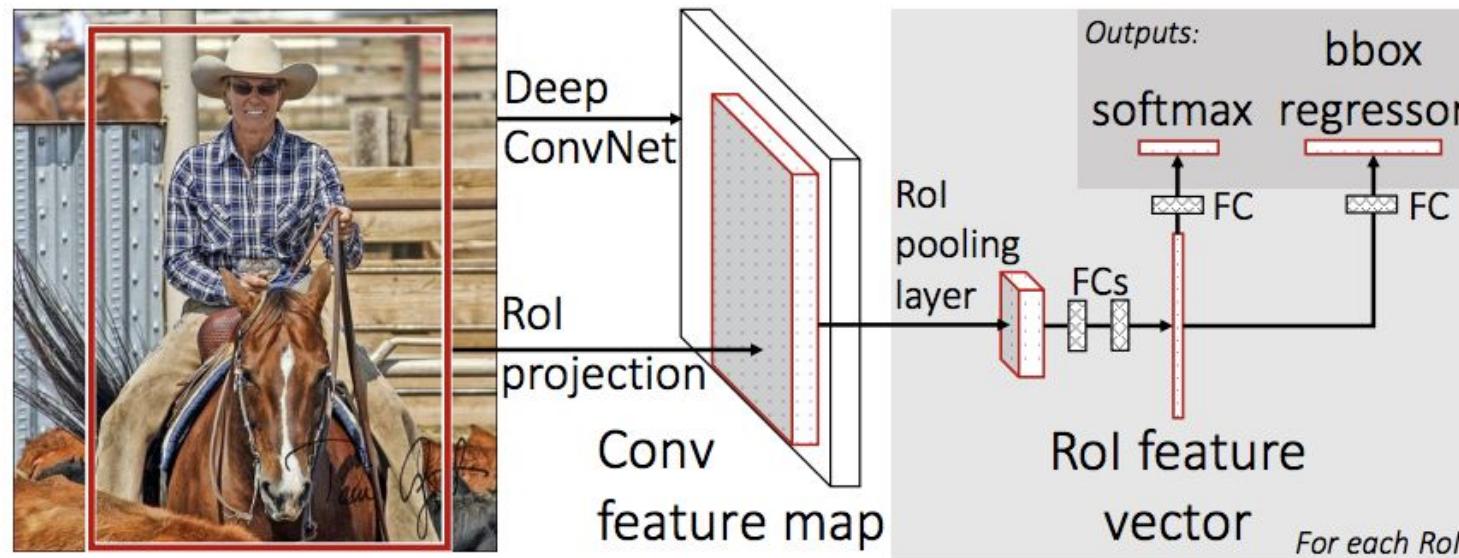


Convolutional sliding window



Fast R-CNN

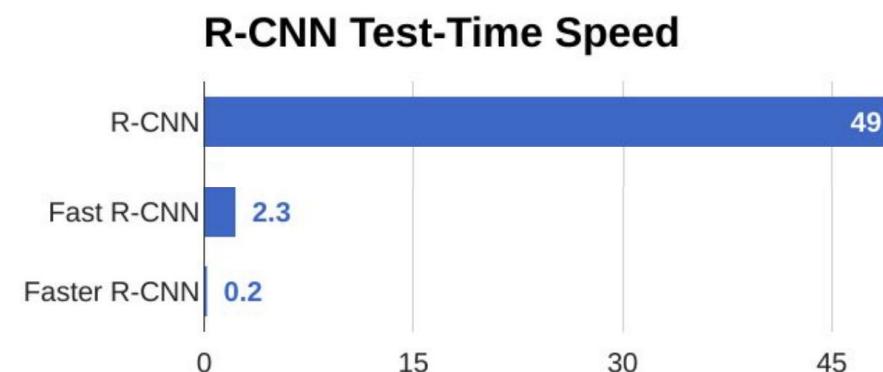
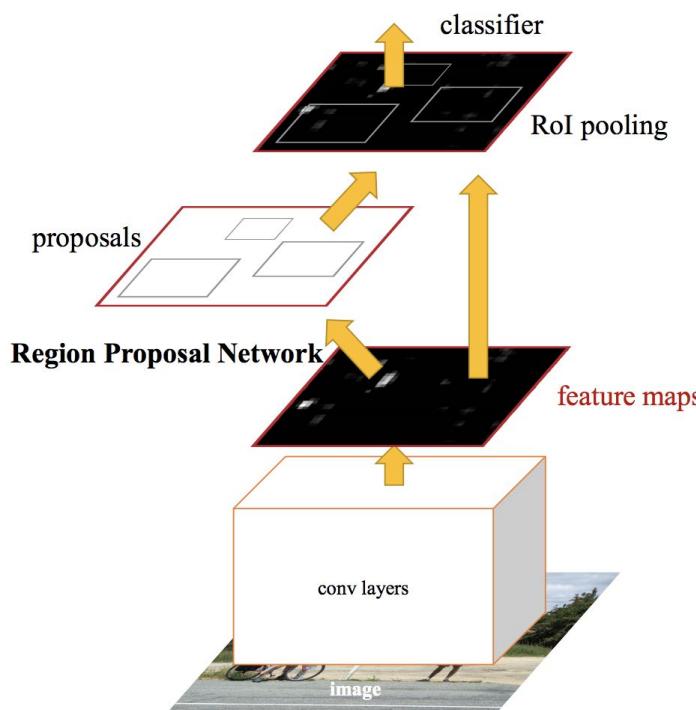
- "Fast" as the convolutional operation is implementing the sliding window to classify all the proposed regions.



- Slow to propose regions

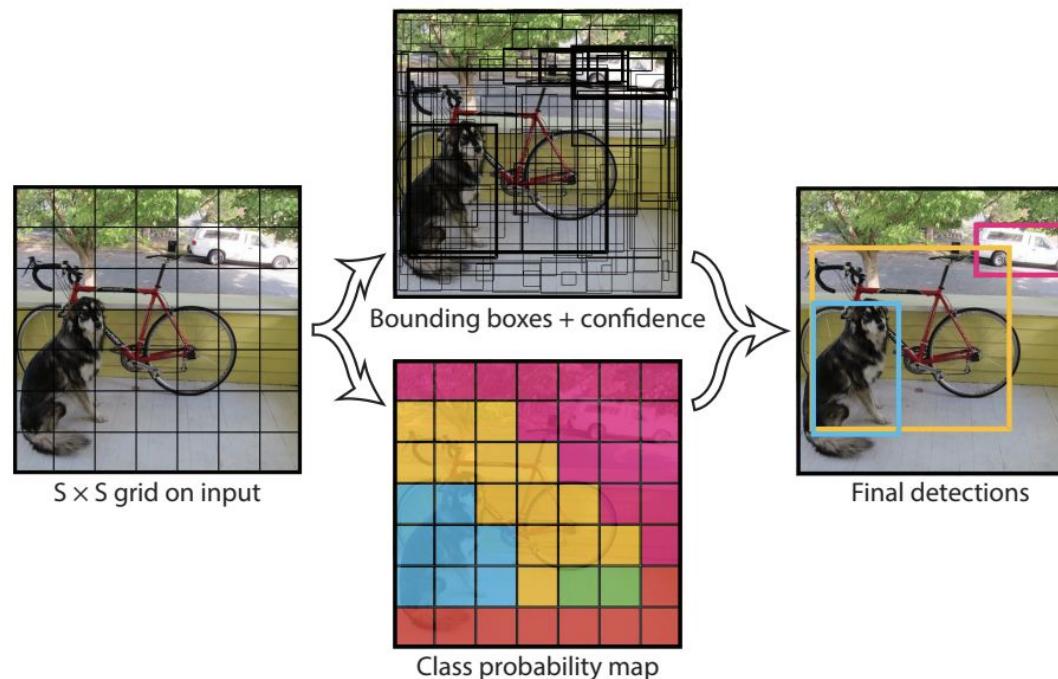
Faster-CNN

- Identify proposal regions from a convolutional neural network.



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.



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