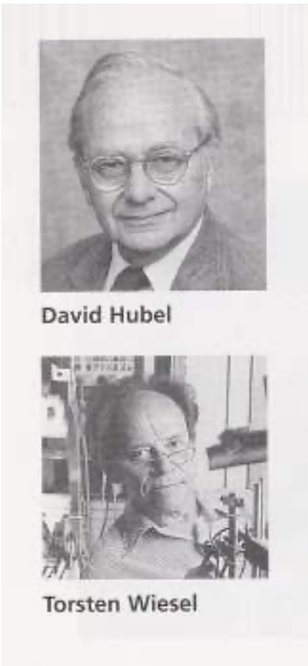
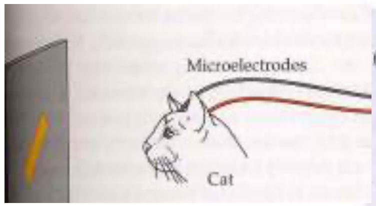
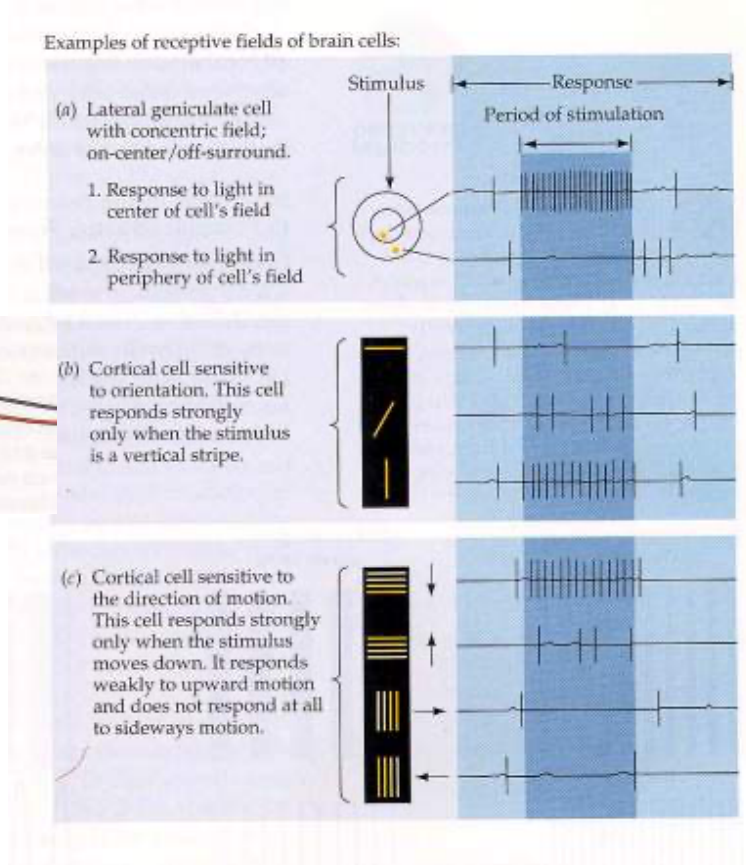
Lecture 14: Convolutional Neural Networks

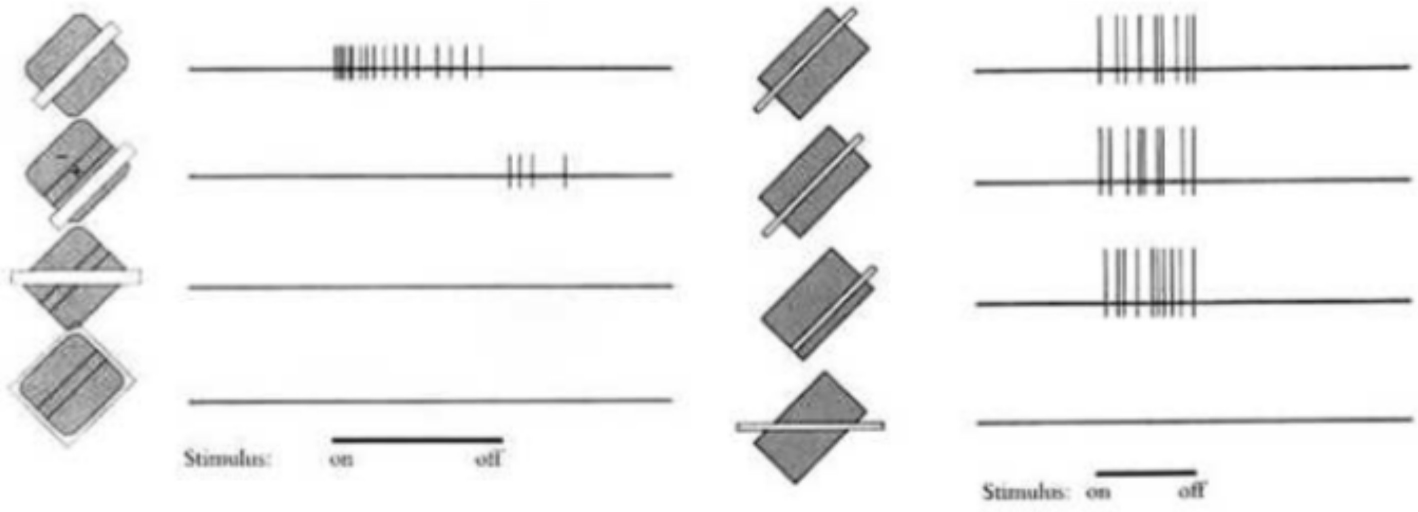
Andreas Wichert
Department of Computer Science and Engineering
Técnico Lisboa

Receptive Fields of Lateral Geniculate and Primary Visual Cortex









- Simple cell (left) and complex Cell (right) illustrative responses in primary visual cortex (from: [Hubel et al., 1988])

- The visual cortex is composed essentially as an hierarchy of cells
  - Layers of simple and complex cells are arranged in a hierachical way
  - The input of a layer is the output of the previous layer

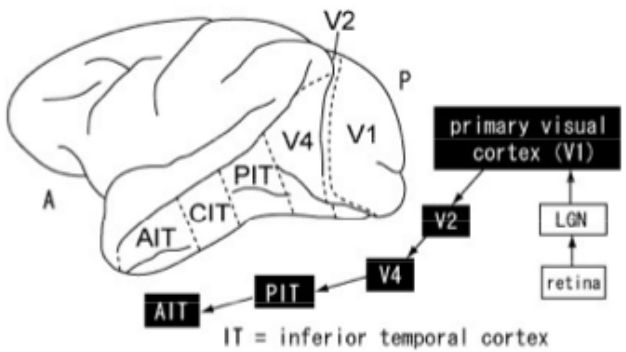


Fig. – Visual pathway [nips.ac.jp]

- Throughout the visual cortex there is a gradual increase in the complexity of the preferred stimulus
- The receptive field sizes and invariance properties also increase gradually

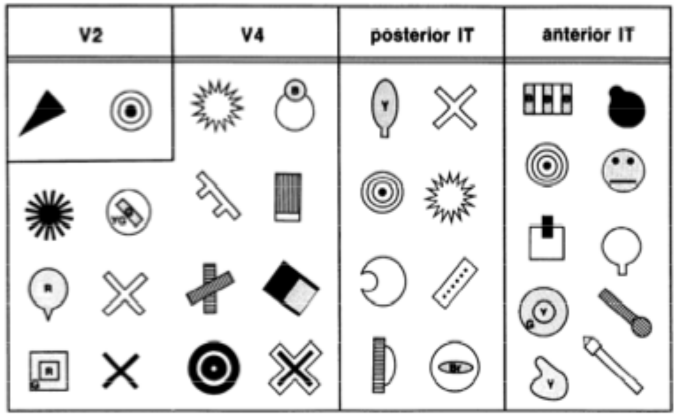


Fig. – Increasing Complexity in prefered stimulus [Kobatake et al. 94]

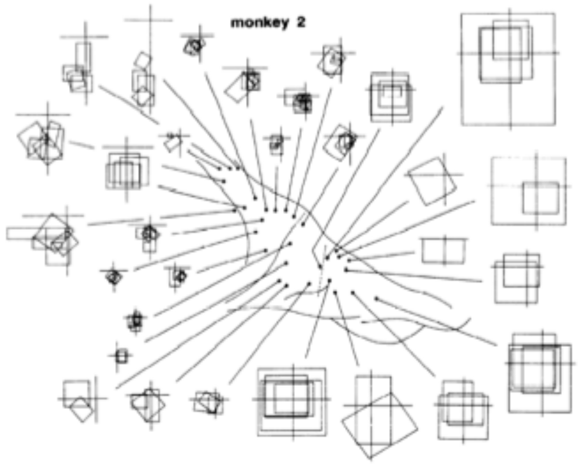
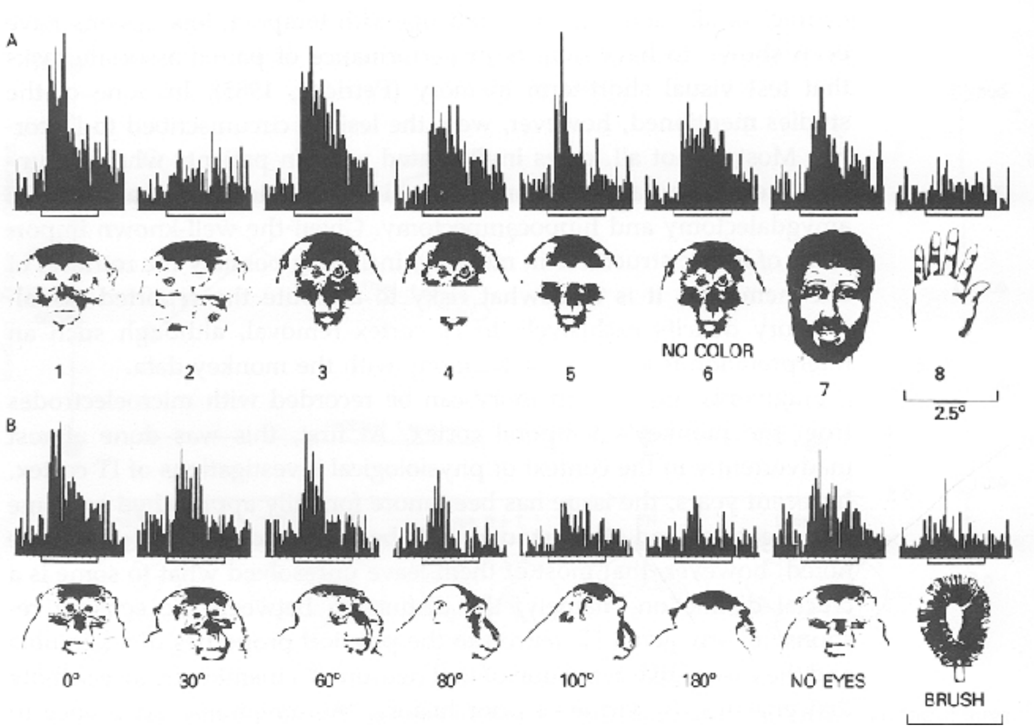
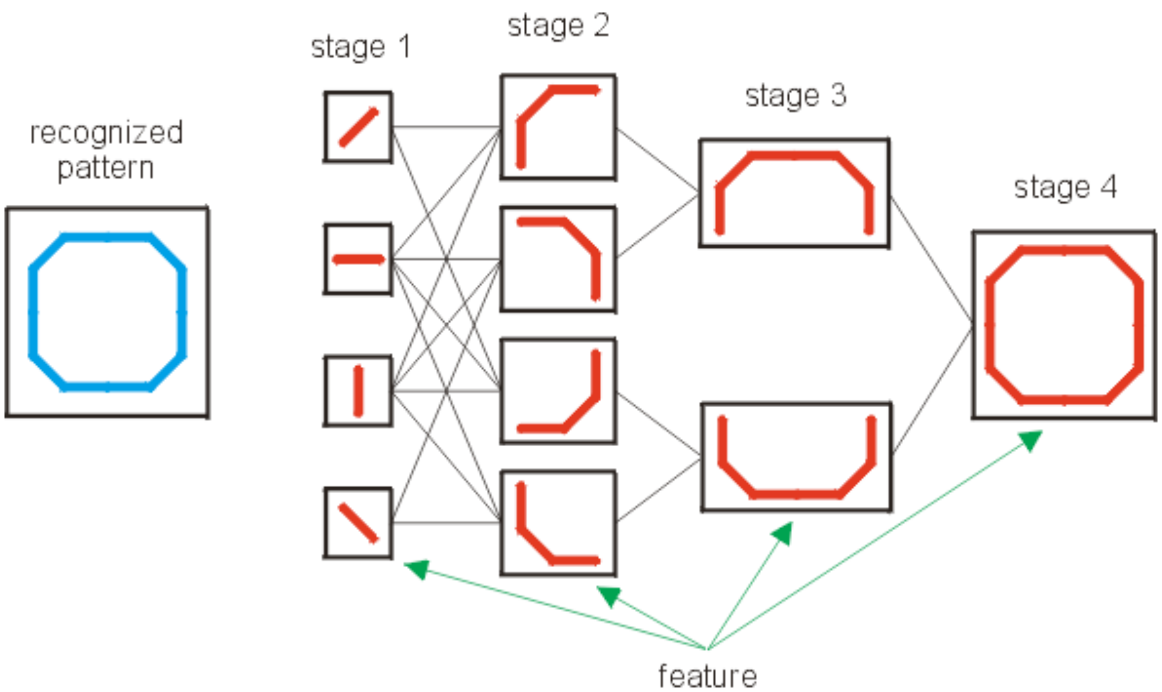


Fig. – Receptive fields from a region including V4 and IT [Kobatake et al. 94]

Face Cells in Monkey

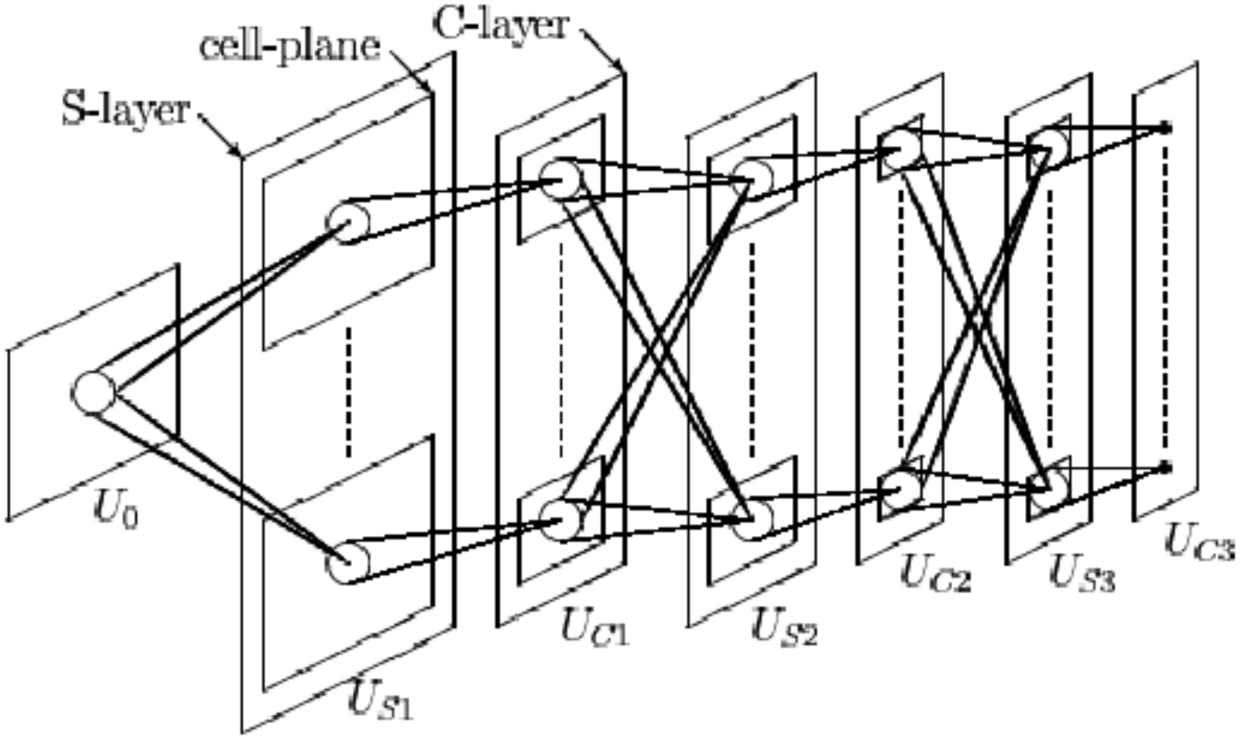




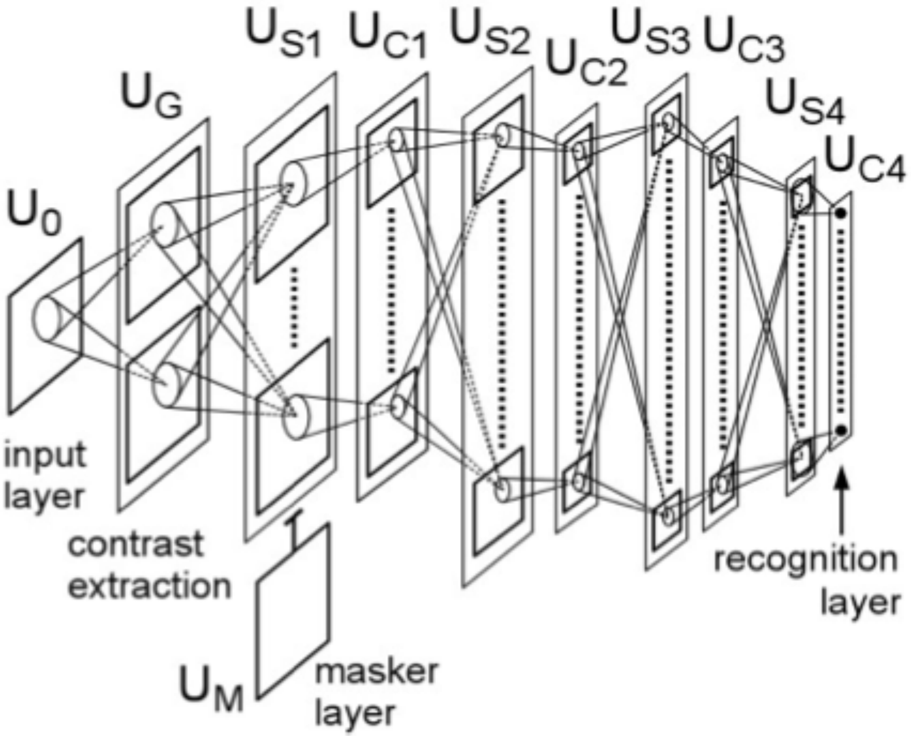
- Image passed through layers of units with progressively more complex features at progressively less specific locations.
- Hierarchical in that features at one stage are built from features at earlier stages

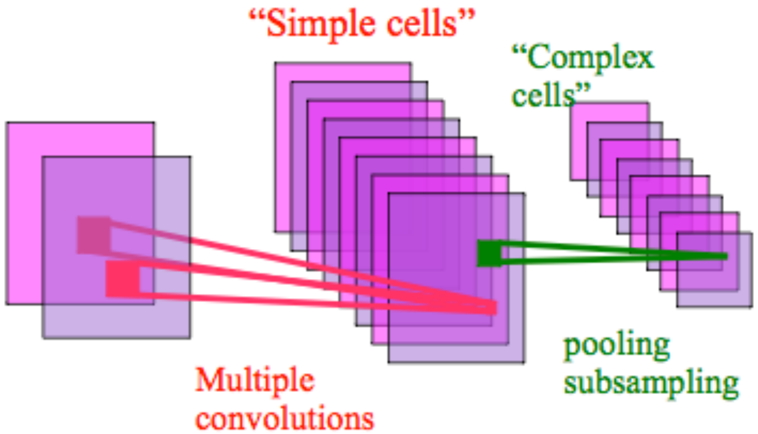
Hierarchical Template Matching:

Fukushima & Miyake (1982)'s Neocognitron



- [Hubel & Wiesel 1962]:
  - simple cells detect local features
  - complex cells "pool" the outputs of simple cells within a retinotopic neighborhood.



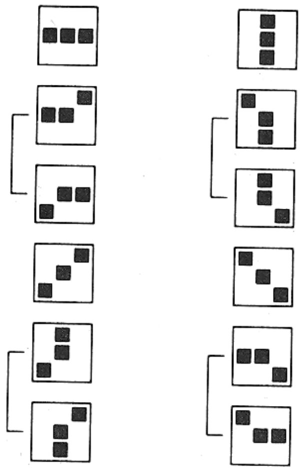


Cognitron & Neocognitron [Fukushima 1974-1982]

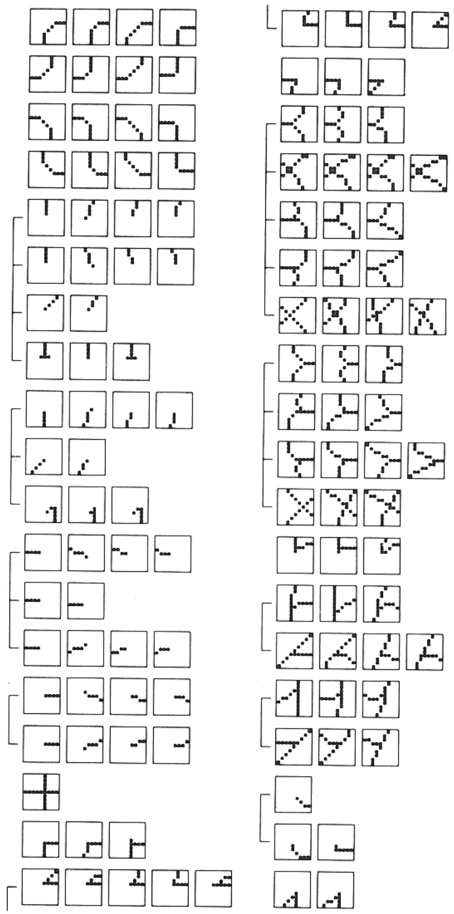
- S-cells
  - represent simple cells in the visual cortex
    - Extract features
  - Learn to form a template of particular feature in particular position
  - Share a weight-vector with all cells in their cell-plane
    - In a cell-plane all cells extract the same feature in different positions
- C-cells
  - Represent complex cells in the visual cortex
    - Allow positional shifts in features
  - It's output is a blurred version of their input

- C-cells resemble complex cells in the visual cortex
- Their purpose is to allow positional changes and distortions of the features
- They do this by blurring the stimulus they receive

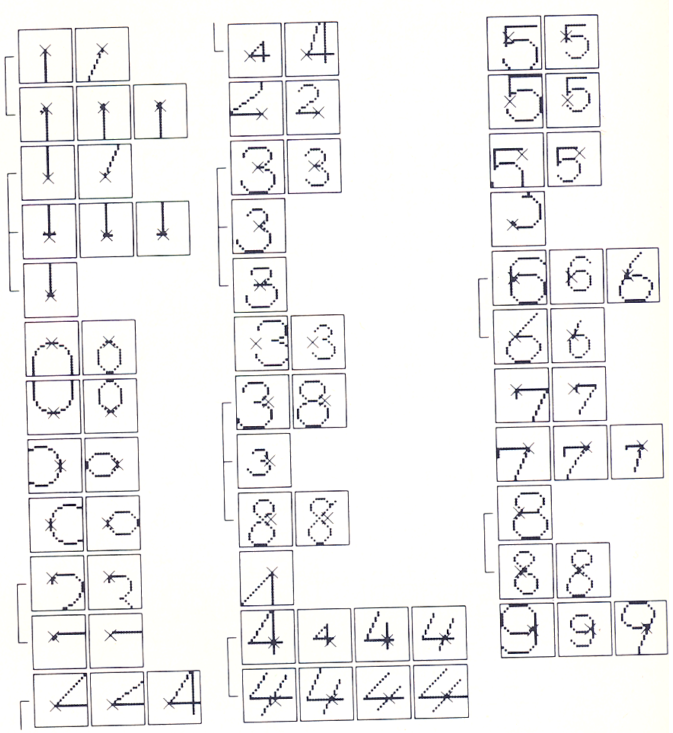
First S-layer after learning



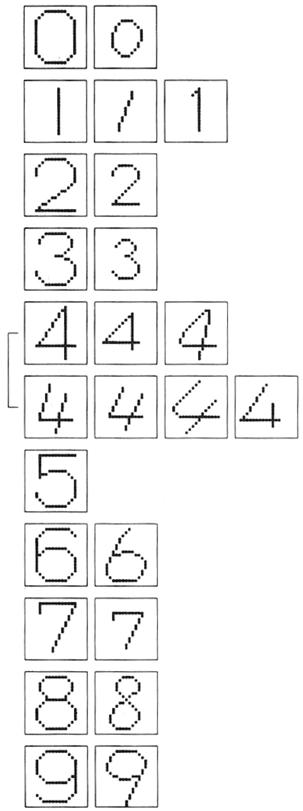
Second S-layer

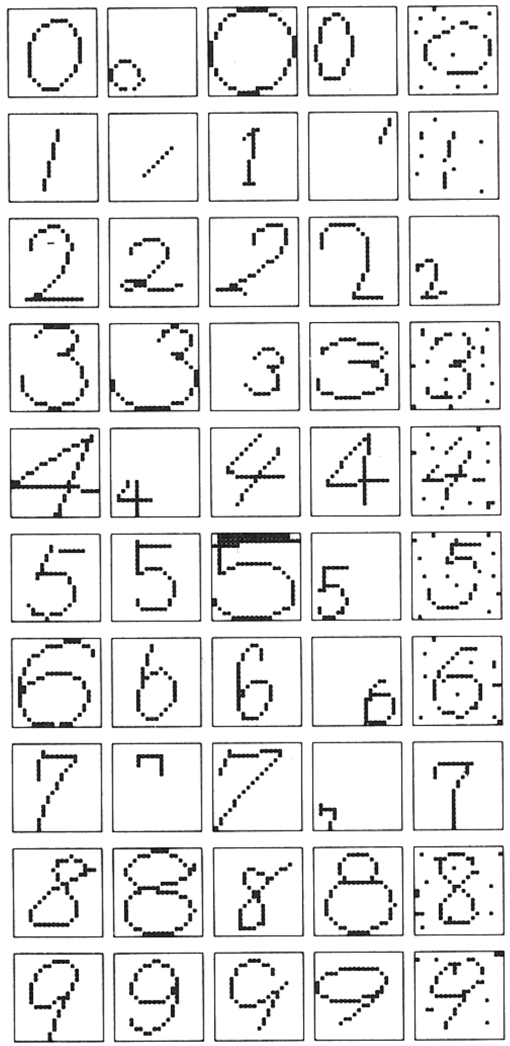


Third S-layer



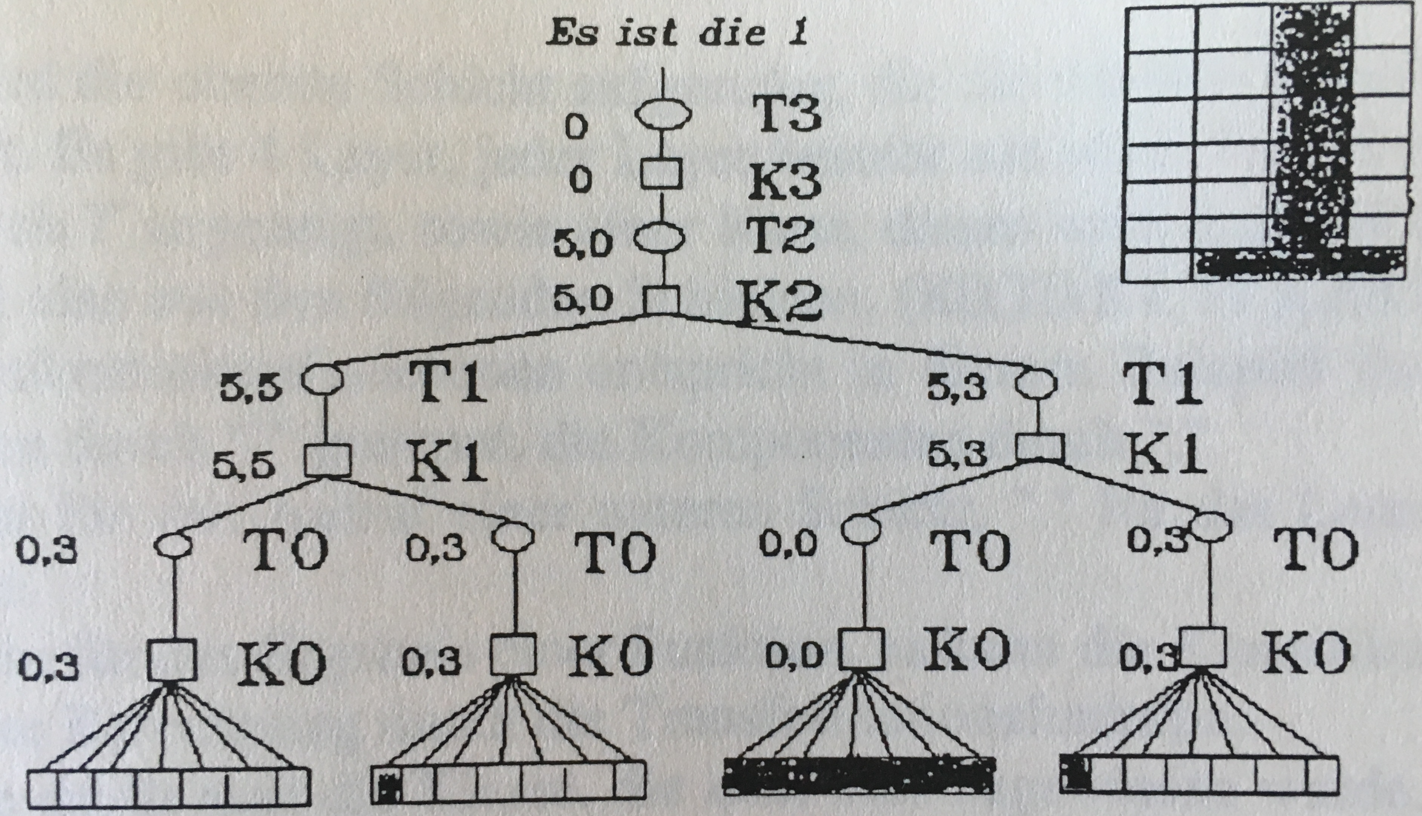
Fourth S-cell layer

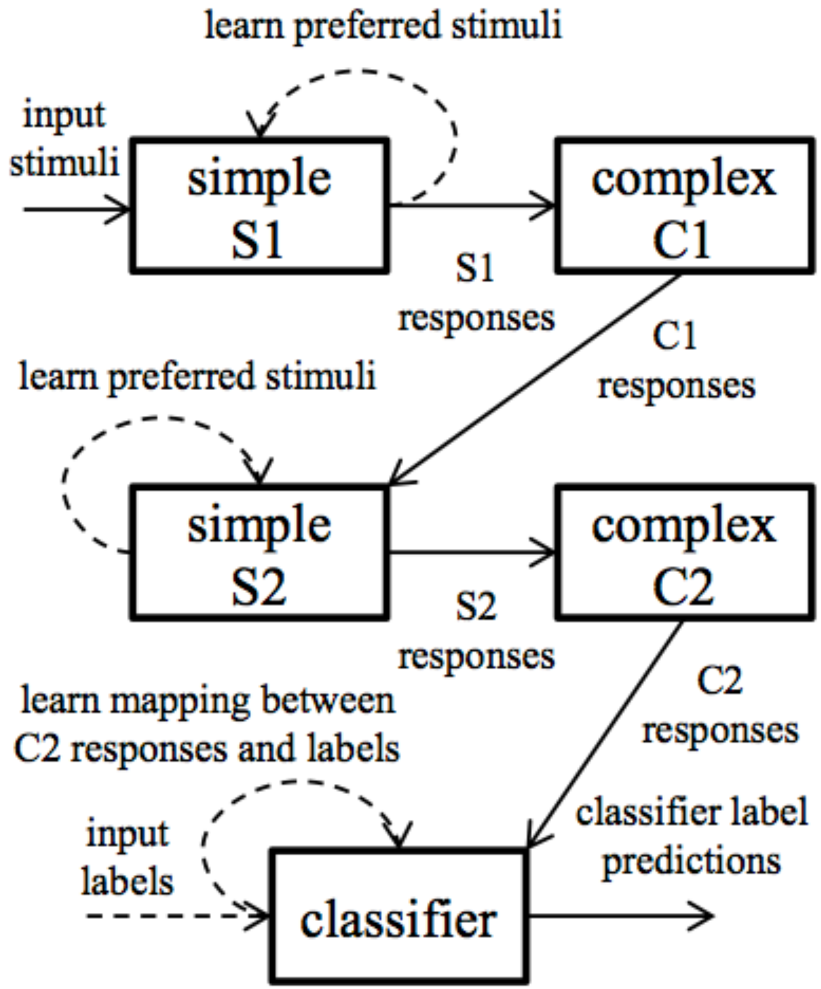




Map Transformation Cascade (MTC)

- A less complex description of the the Neocognitron is the hierarchical neural network called map transformation cascade (Wichert 1992, 1993)
  - Wichert, A.: MTCn-Nets. Proceeding World Congres on Neural Networks 1993, Vol.IV, pp.59-62, Lawrence Erlbaum, 1993
- The information is processed sequentially, each layer only processes information after the previous layer is finished.
- The input is tiled with a squared mask, where each sub-pattern is replaced by a number indicating a corresponding class. By doing so, we get a representation of the pattern in the class space.
- The mask has the same behavior in all different positions, resembling the weight-sharing mechanism in Neocognitron.


Es ist die 1
0
T3
0
K3
5,0
T2
5,0
K2
5,5
T1
5,5
K1
5,3
T1
5,3
K1
0,3
T0
0,3
K0
0,3
T0
0,3
K0
0,0
T0
0,0
K0
0,3
T0
0,3
K0


learn preferred stimuli
input stimuli
simple S1
complex C1
S1 responses
C1 responses
learn preferred stimuli
simple S2
complex C2
S2 responses
C2 responses
learn mapping between C2 responses and labels
input labels
classifier
classifier label predictions

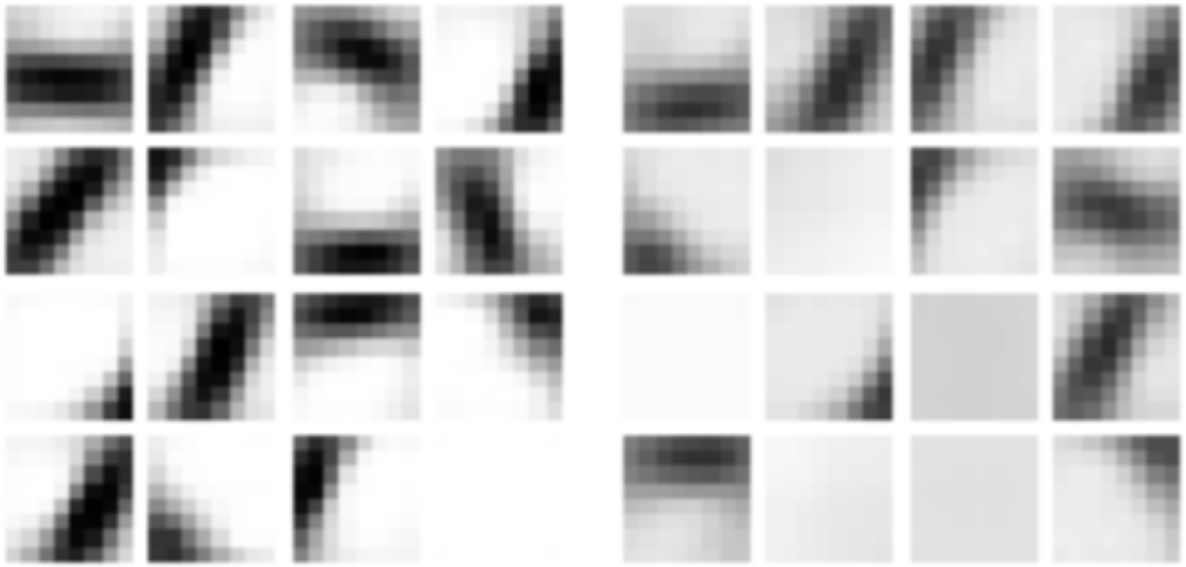
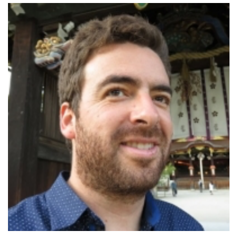
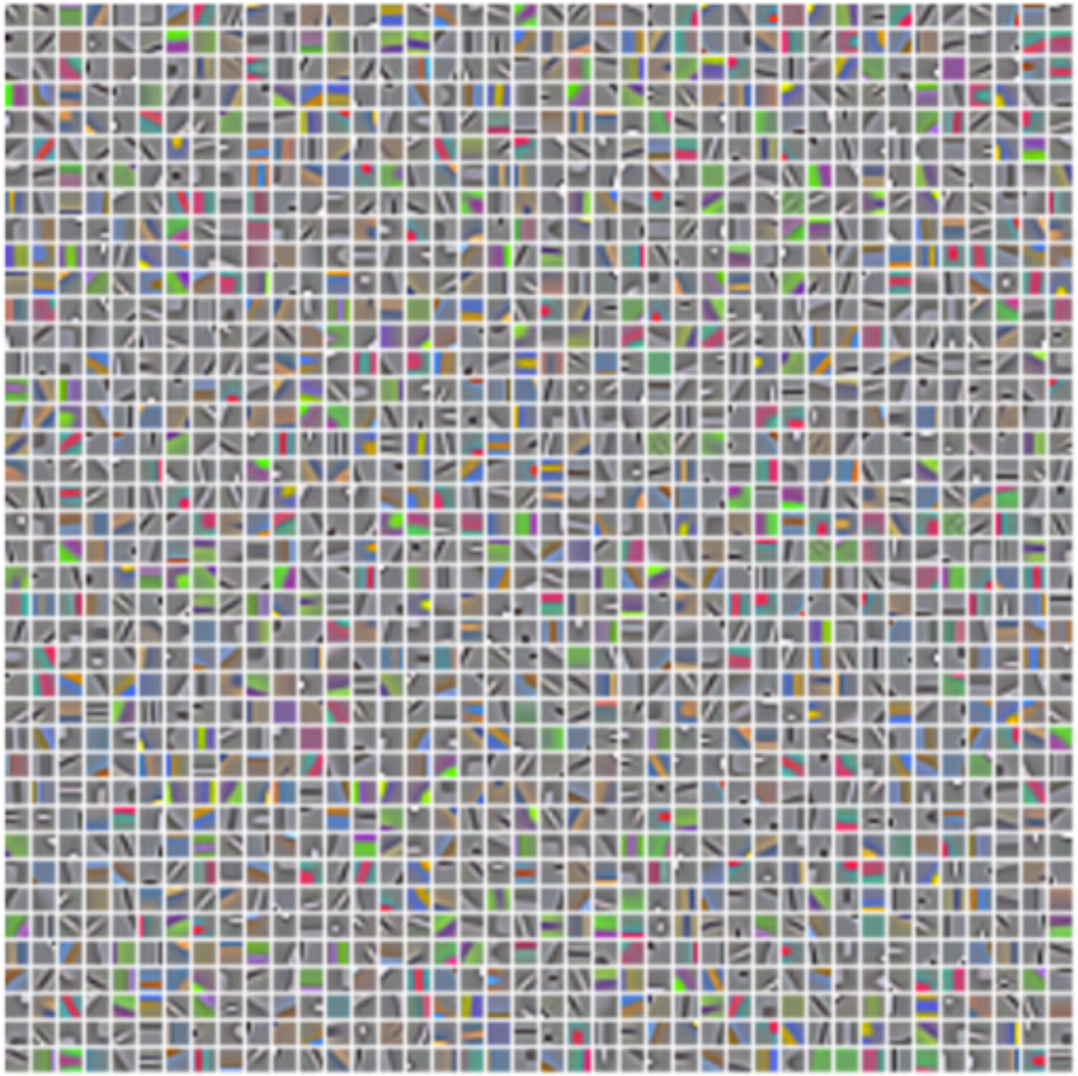


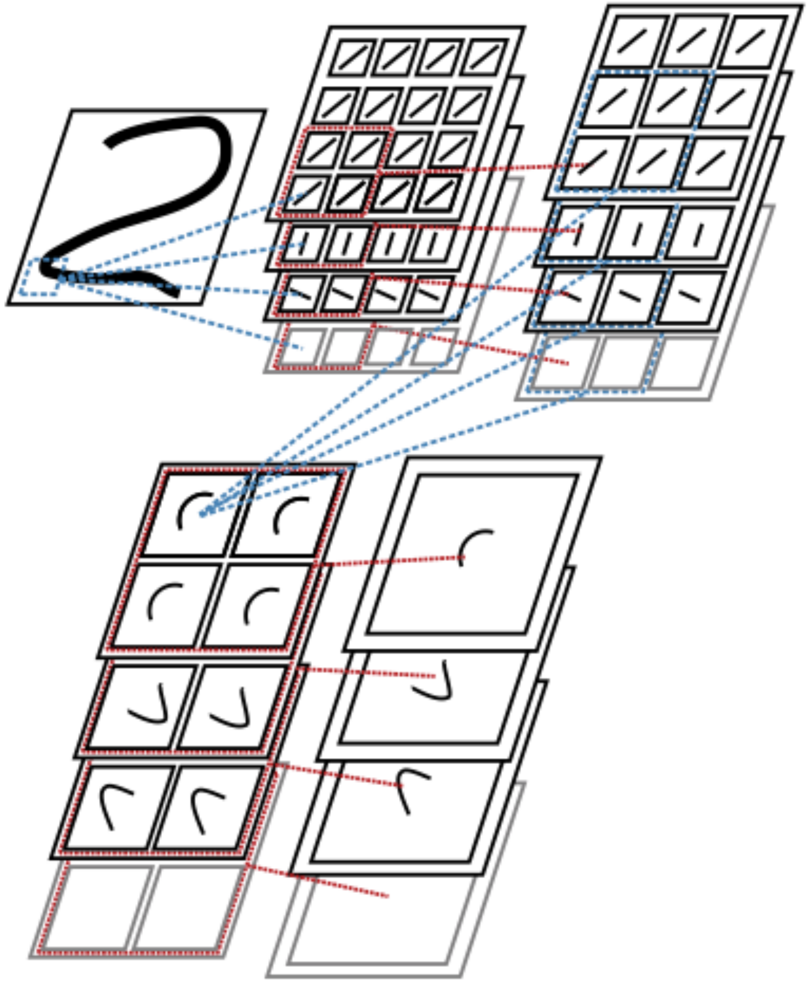
Figure 4.1: A set of 16 preferences learned using K-means on ETL1 for 10000 patches of size 8 × 8. On the left, the preferences using binary versions of th patterns. On the right, the preferences using grayscale versions. We can se that the grayscale versions of the patterns produce low-contrast preferences Several of the preferences are simply different shades of gray.

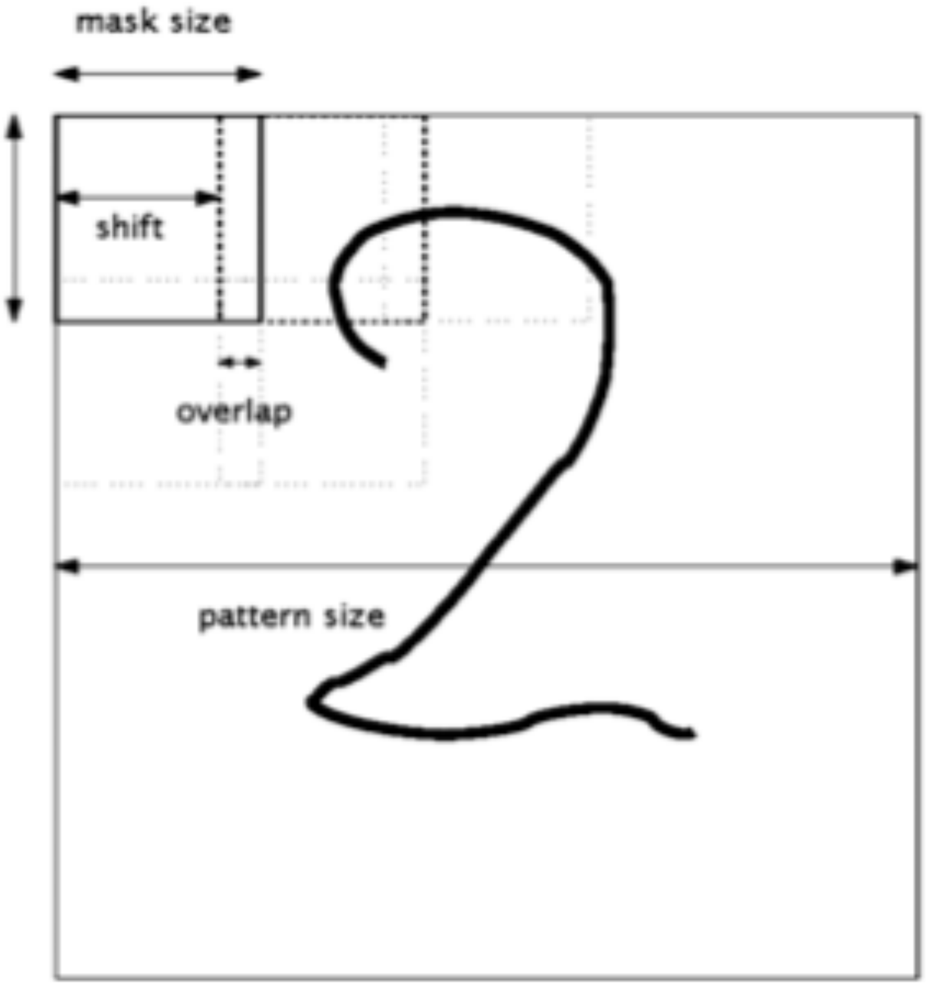


Ângelo Cardoso

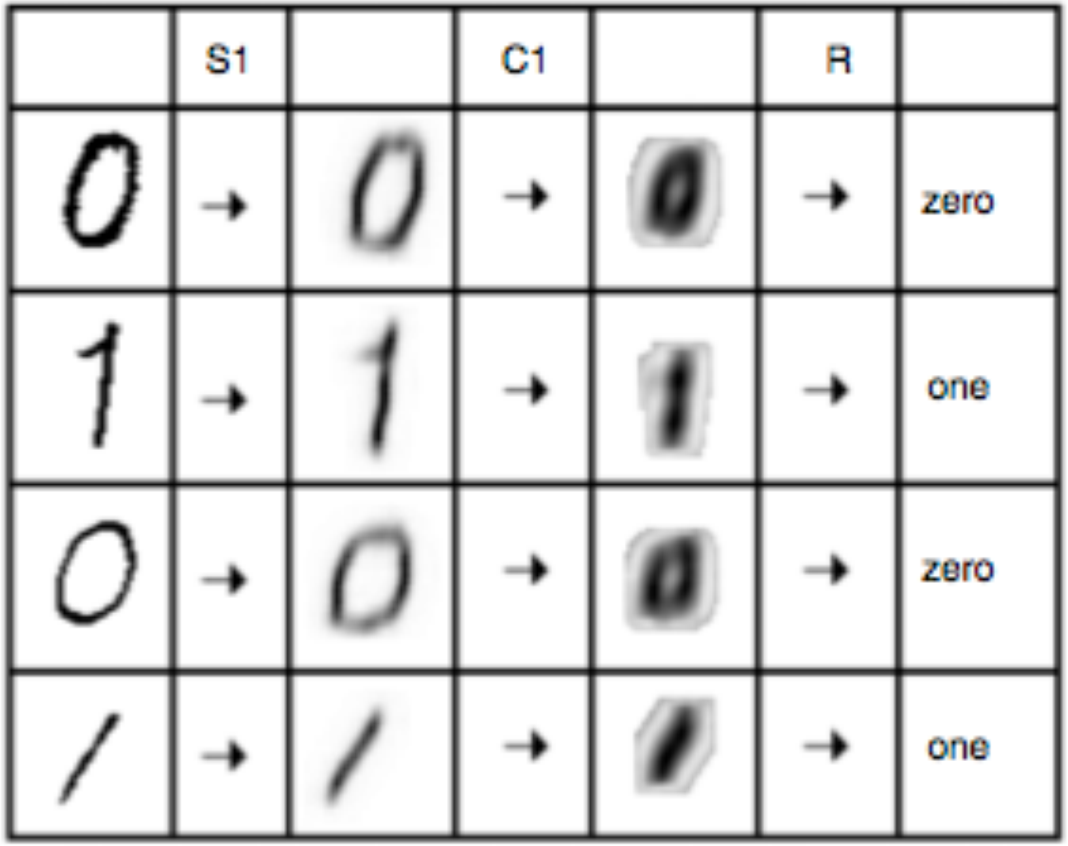
- The S-layer learning is performed by a clustering algorithm like k-Means

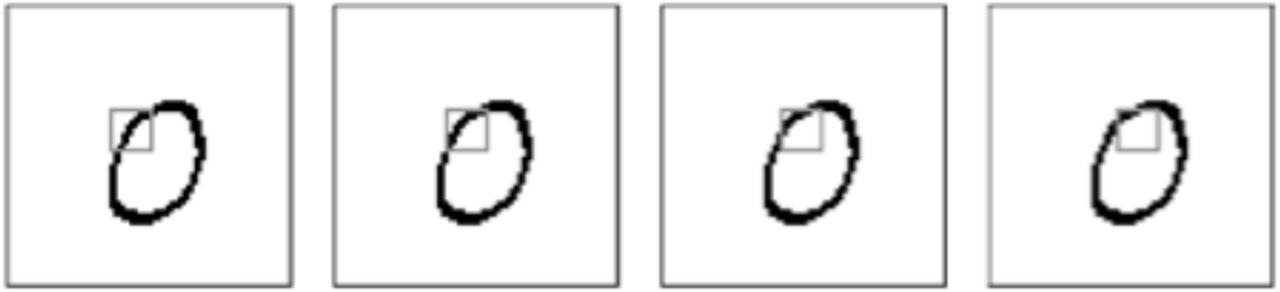




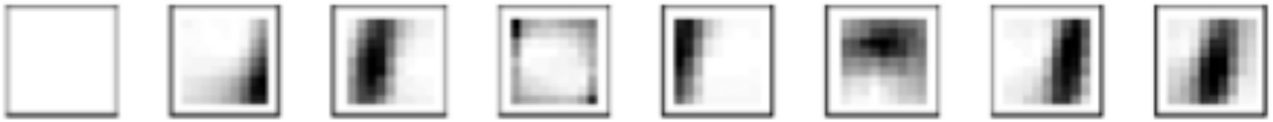

mask size
shift
overlap
pattern size

- The C-layer, which corresponds to a layer of complex cells in the visual cortex, transforms the input it receives from the S-layer. The transformation performed by the C-layer is fixed and can be not modified. Its purpose is to allow positional shifts, thus giving the model shift invariance.


S1
C1
R
zero
one
zero
one



(a) Output: 7. (b) Output: 2. (c) Output: 4. (d) Output: 3.



(a) 0 (b) 1 (c) 2 (c) 3 (c) 4 (c) 5 (c) 6 (c) 7

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

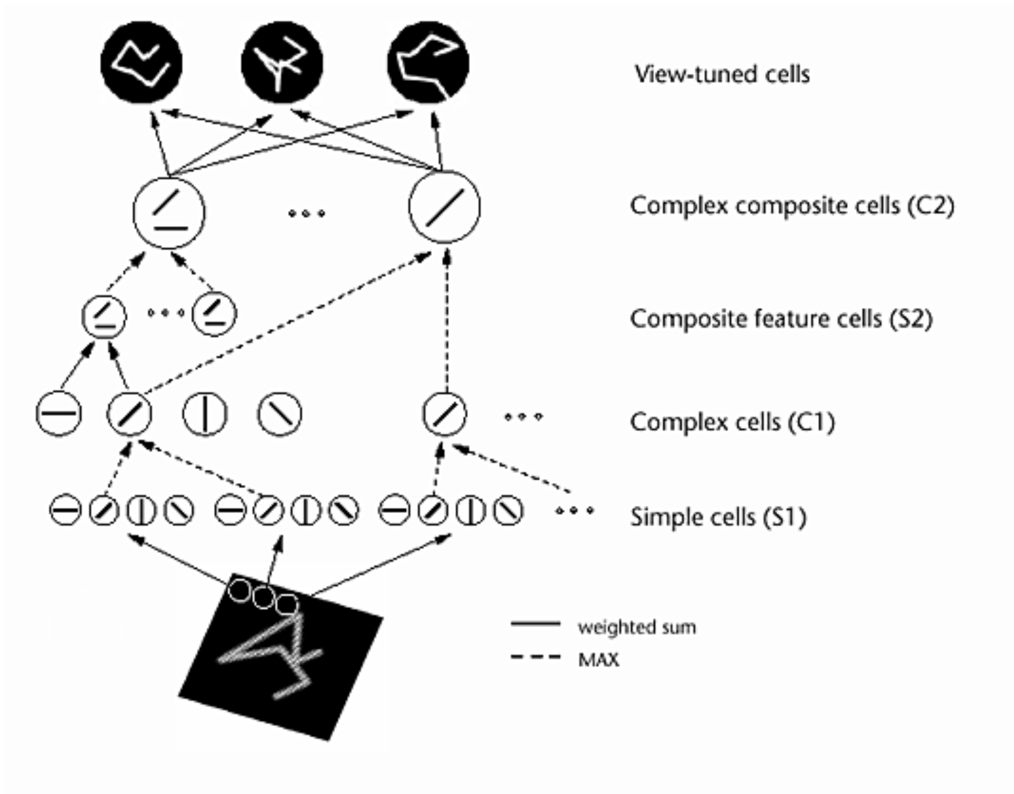
0	0	0	1	6
0	0	0	3	6
0	0	0	3	3
0	0	0	0	3
0	0	0	0	0

Fig. 8. C-Layer mask input in a given position when scanning Fig. 7, its output is {1, 6, 3}. It indicates the presence of these classes.

- The layers of a Map Transformation Cascade can be seen as filters, since they have a clear and interpretable output, which is a modification of the input information.

- Several filters transform and map the input pattern into a space where pat- terns of the same class are close. The output of the filters is then passed to a simple classifier, which produces a classification for the input pattern.

Computational Model of Object Recognition
(Riesenhuber and Poggio, 1999)



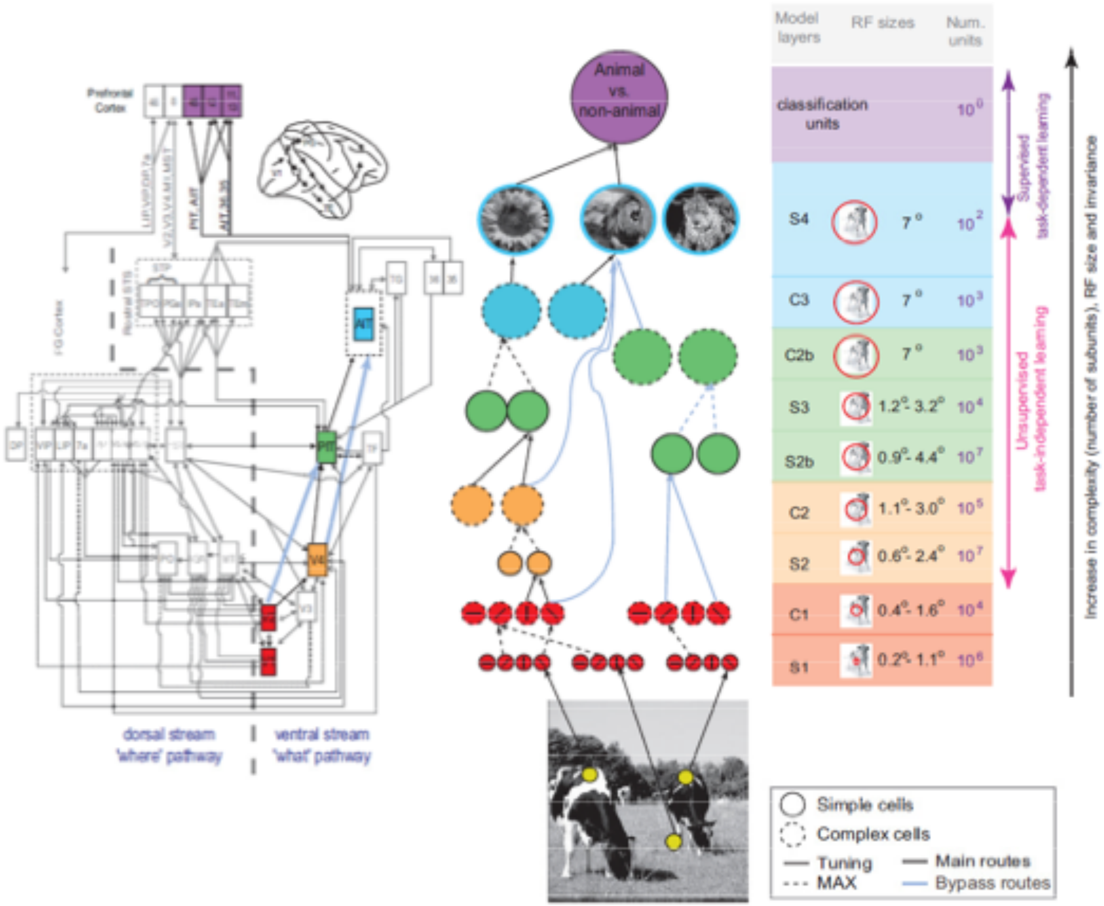
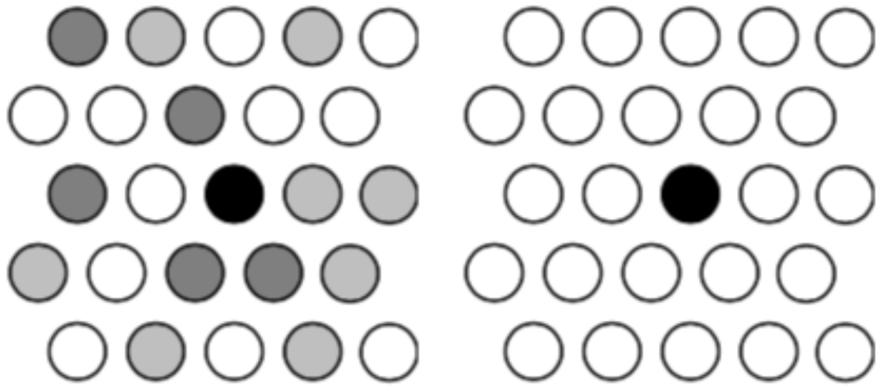


Fig. – HMAX Schematic [Serre et al. 07]

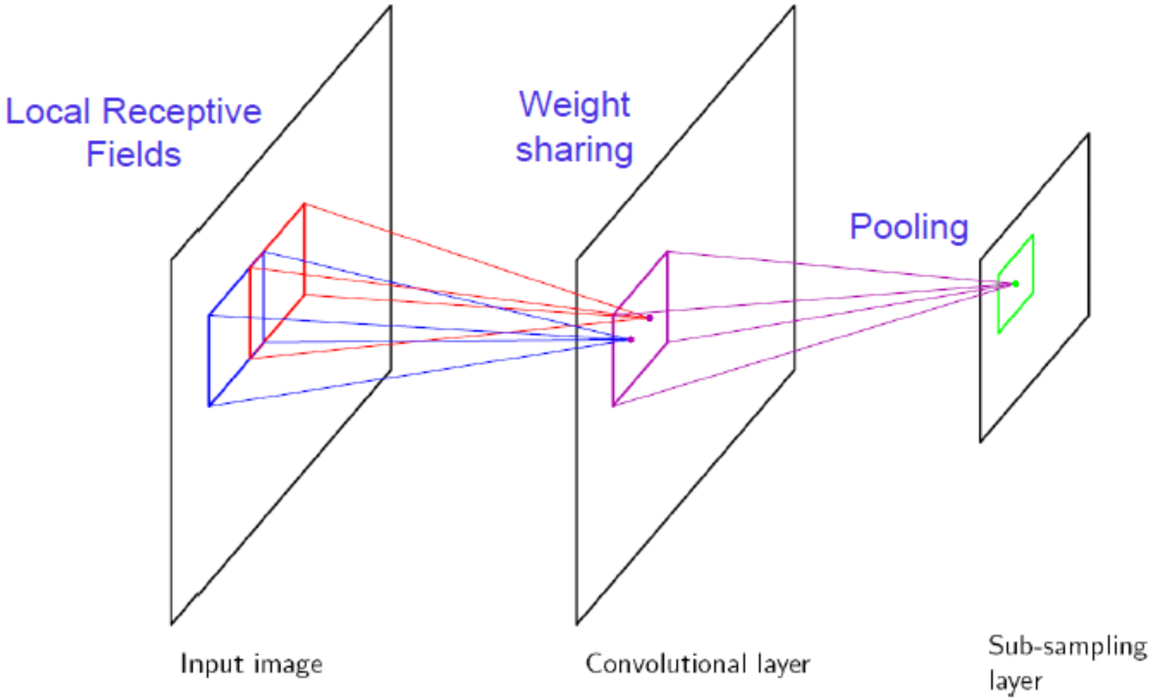


(a) Sum (b) Winner-take-all

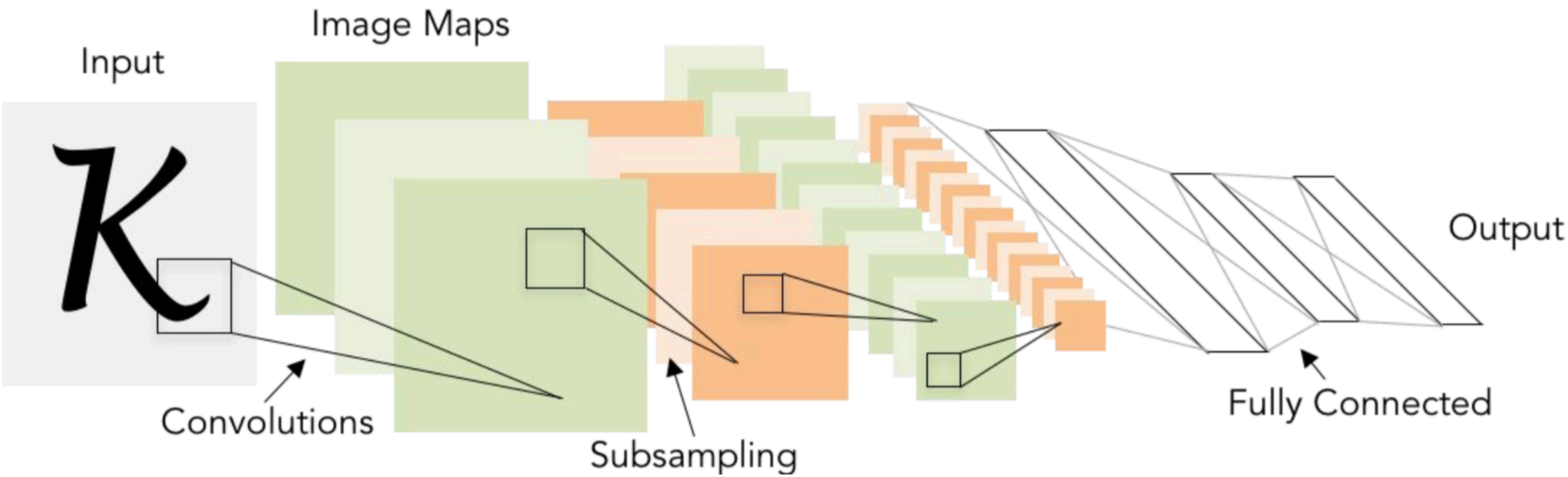
MNIST	no noise	white	salt & pepper
this work	0.71%[a]	1.17%	1.98%
HMAX [b]	2.9%[c]	50%[c]	55%[c]

Convolutional Neural Networks

(LeCun et al., 1989)

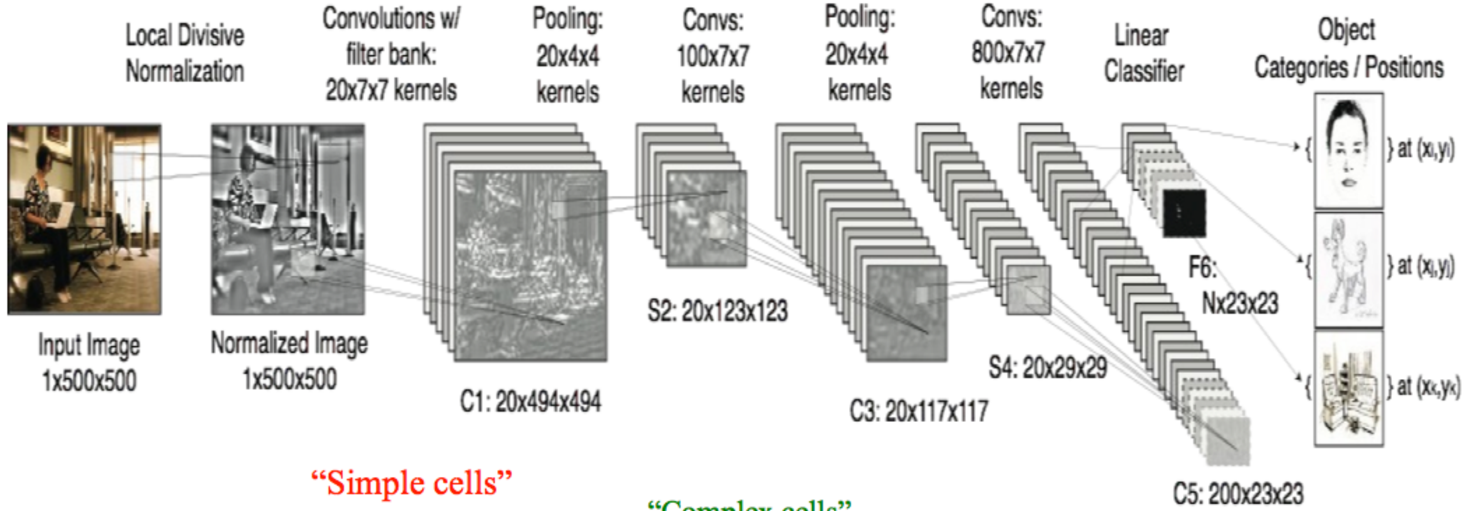


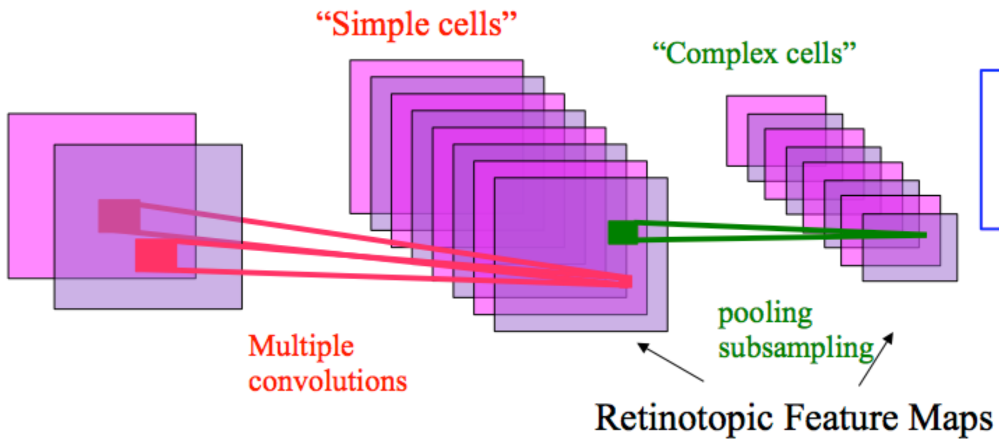
- Convolutional Neural Networks



MNIST Data Set

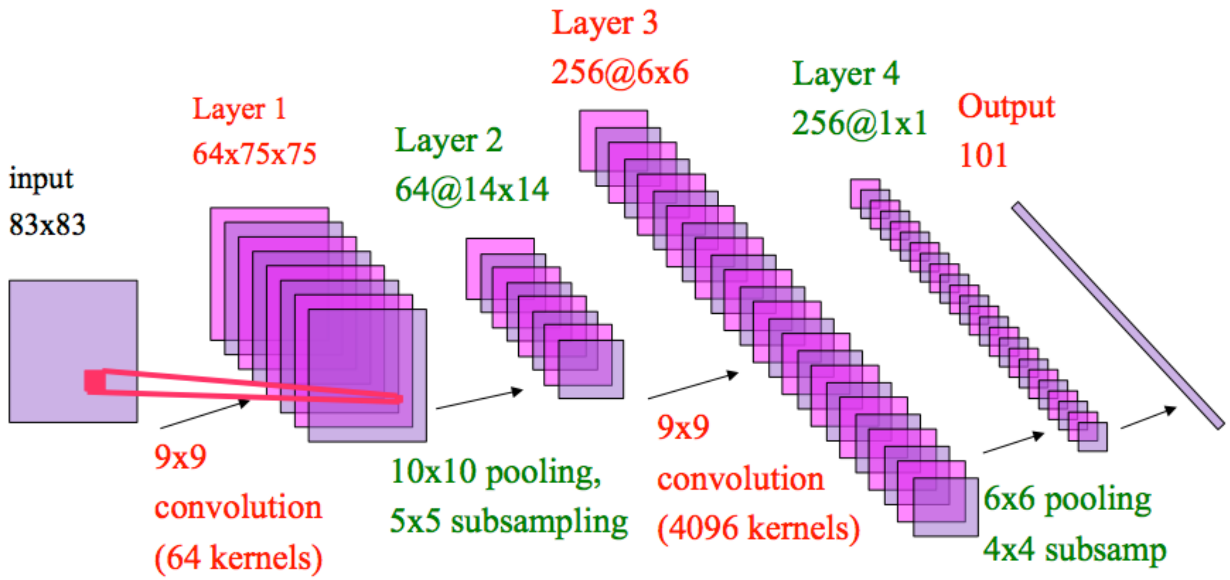
- The MNIST database contains 60, 000 training im- ages and 10, 000 testing images. The images of the digits contain grey levels represented by a 28 × 28 matrix resulting in 784 dimensional input vector



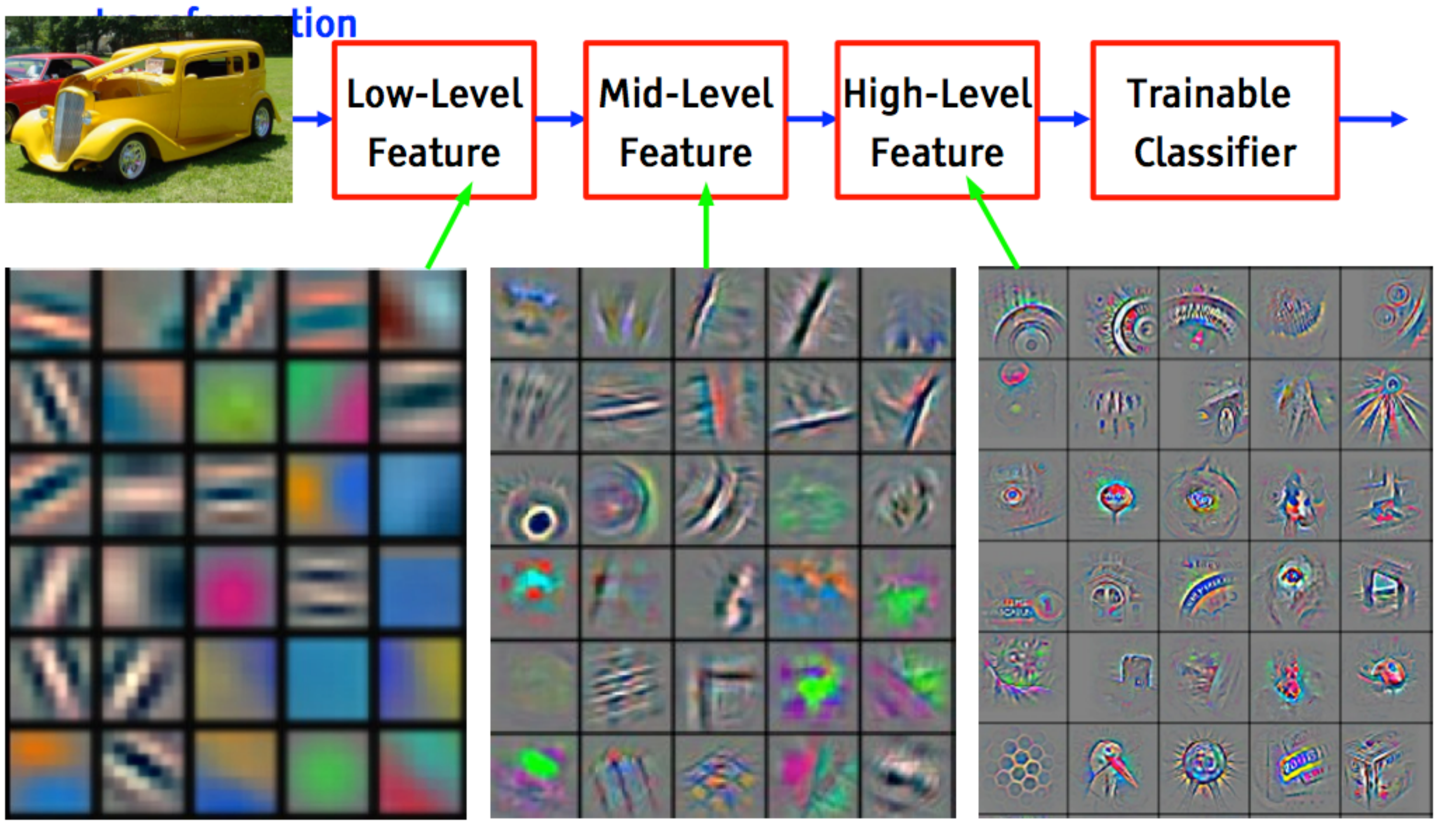


- Training is supervised
- With stochastic gradient descent

[LeCun et al. 89]
[LeCun et al. 98]



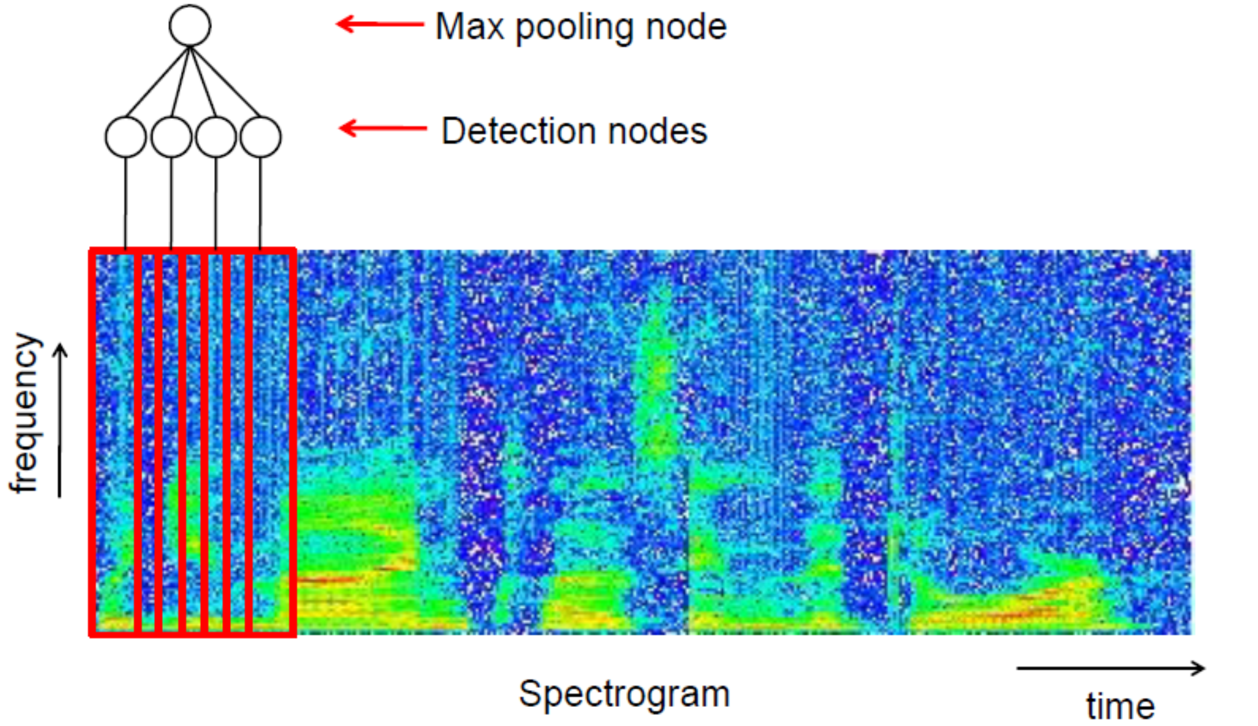
- Non-Linearity: half-wave rectification, shrinkage function, sigmoid
- Pooling: average, L1, L2, max
- Training: Supervised (1988-2006), Unsupervised+Supervised (2006-now)



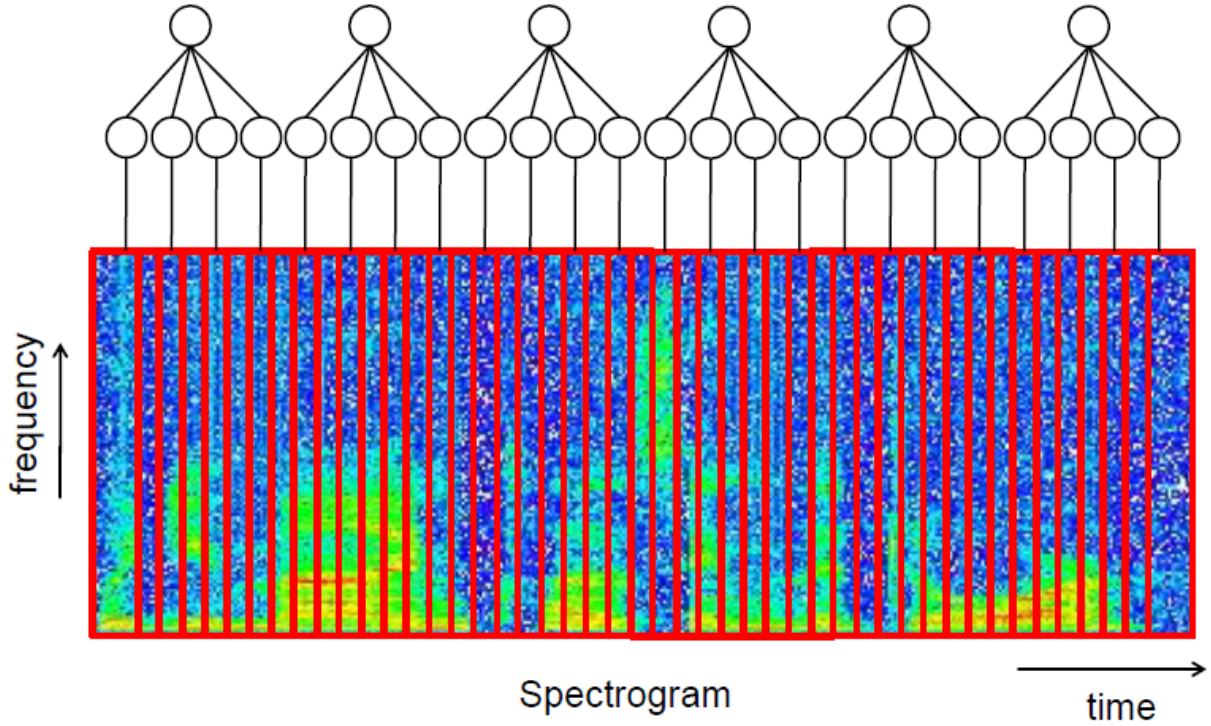
Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

Convolutional DBN for audio

(Lee et al., 2009)



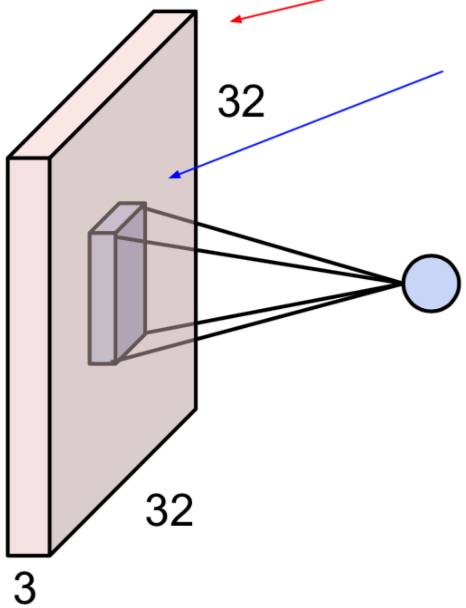
Convolutional DBN for audio



Convolution Layer

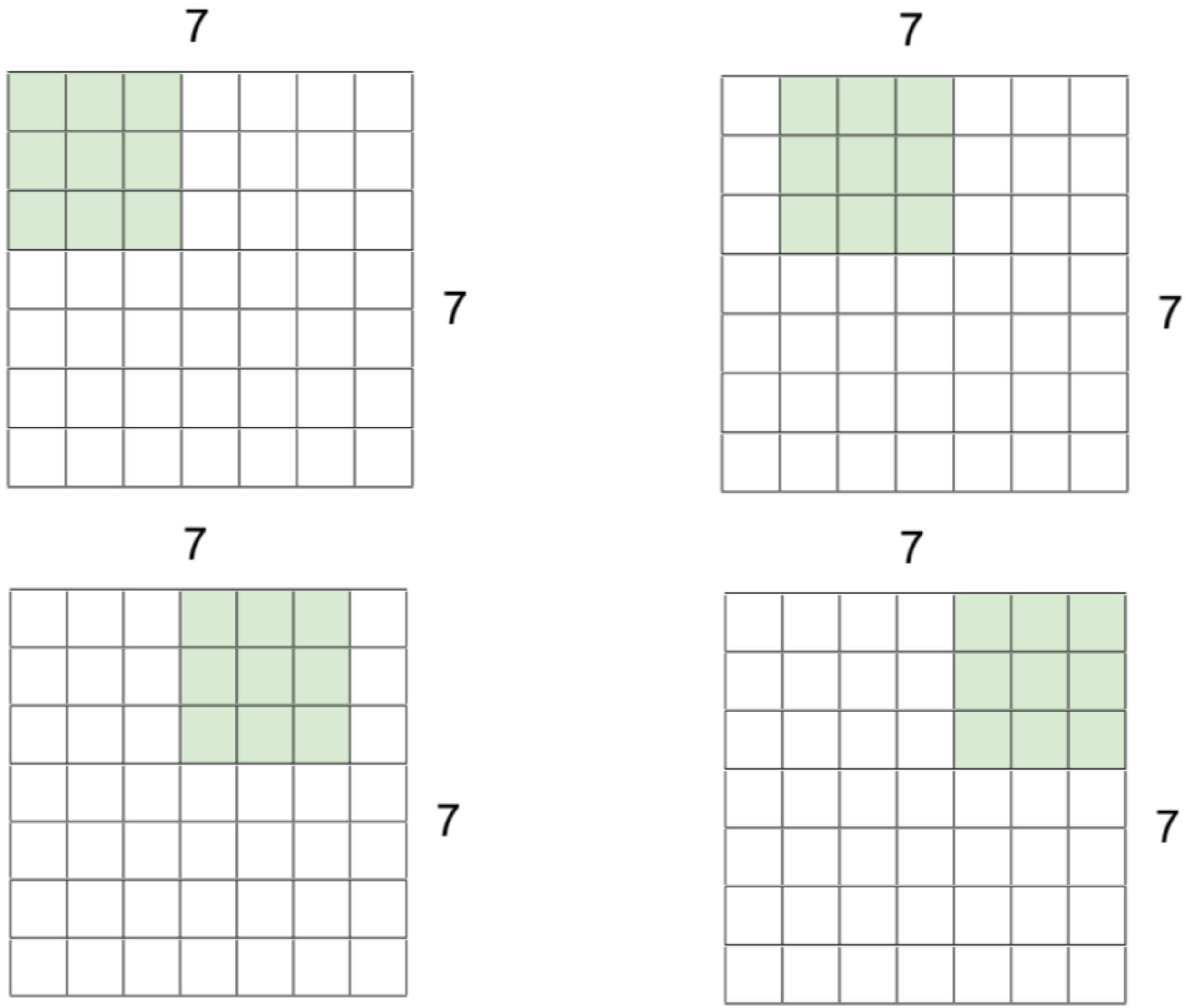
32x32x3 image

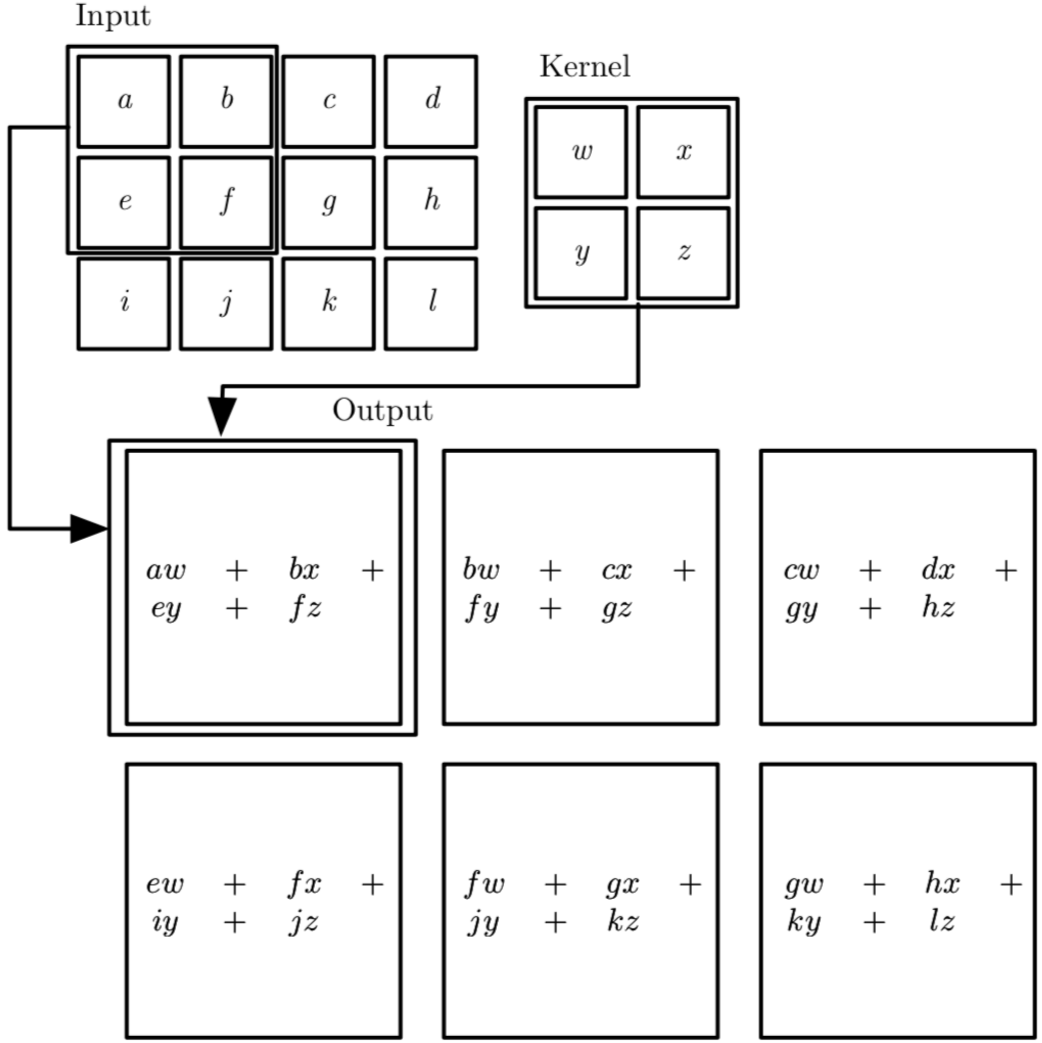
5x5x3 filter w



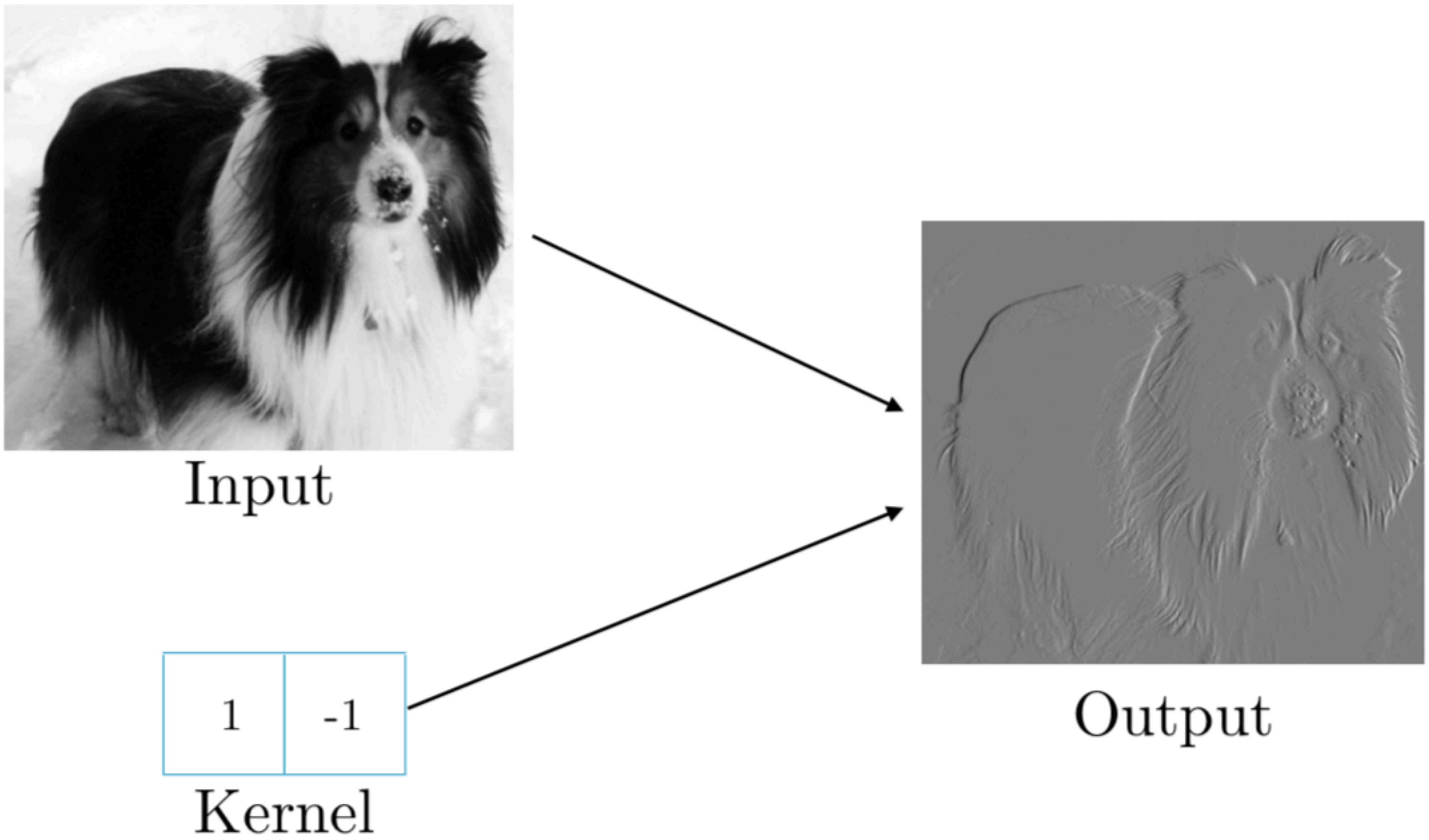
1 number:
the result of taking a dot product between the filter and a small 5x5x3 chunk of the image
(i.e. 5*5*3 = 75-dimensional dot product + bias)

$$w^T x + b$$


7
7
7
7
7
7
7
7


Input
a b c d
e f g h
i j k l
Kernel
w x
y z
Output
$aw + bx + ey + fz$
$bw + cx + fy + gz$
$cw + dx + gy + hz$
$ew + fx + iy + jz$
$fw + gx + jy + kz$
$gw + hx + ky + lz$

Edge Detection by Convolution



Kernel in Image Processing

- In a convolutional network an adaptive kernel corresponding to ***n*** unit with an activation function that learns or a fixed kernel can be a part of convolution in a layer that acts as a filter.
  - Input:

$$\begin{pmatrix} f(x-1,y-1) & f(x-1,y) & f(x-1,y+1) \\ f(x,y-1) & f(x,y) & f(x,y+1) \\ f(x+11,y-1) & f(x+11,y) & f(x+1,y+1) \end{pmatrix}$$

  - Convolution Kernel:

$$\begin{pmatrix} w(-1,-1) & w(-1,0) & w(-1,-1) \\ w(0,-1) & w(0,0) & w(0,-1) \\ w(1,-1) & w(1,0) & w(1,-1) \end{pmatrix}$$

  - The value of the filter mask at the position (x, y)

$$g(x,y) = \sum_{s=-1}^{1} \sum_{t=-1}^{1} w(s,t) \cdot f(x+s, y+t)$$

Fixed Kernels

- In digital image processing, a kernel, convolution matrix, or mask is a small matrix. It is used for blurring, sharpening, embossing, edge detection, and more. This is accomplished by doing a convolution between a kernel and an image.

(a) 

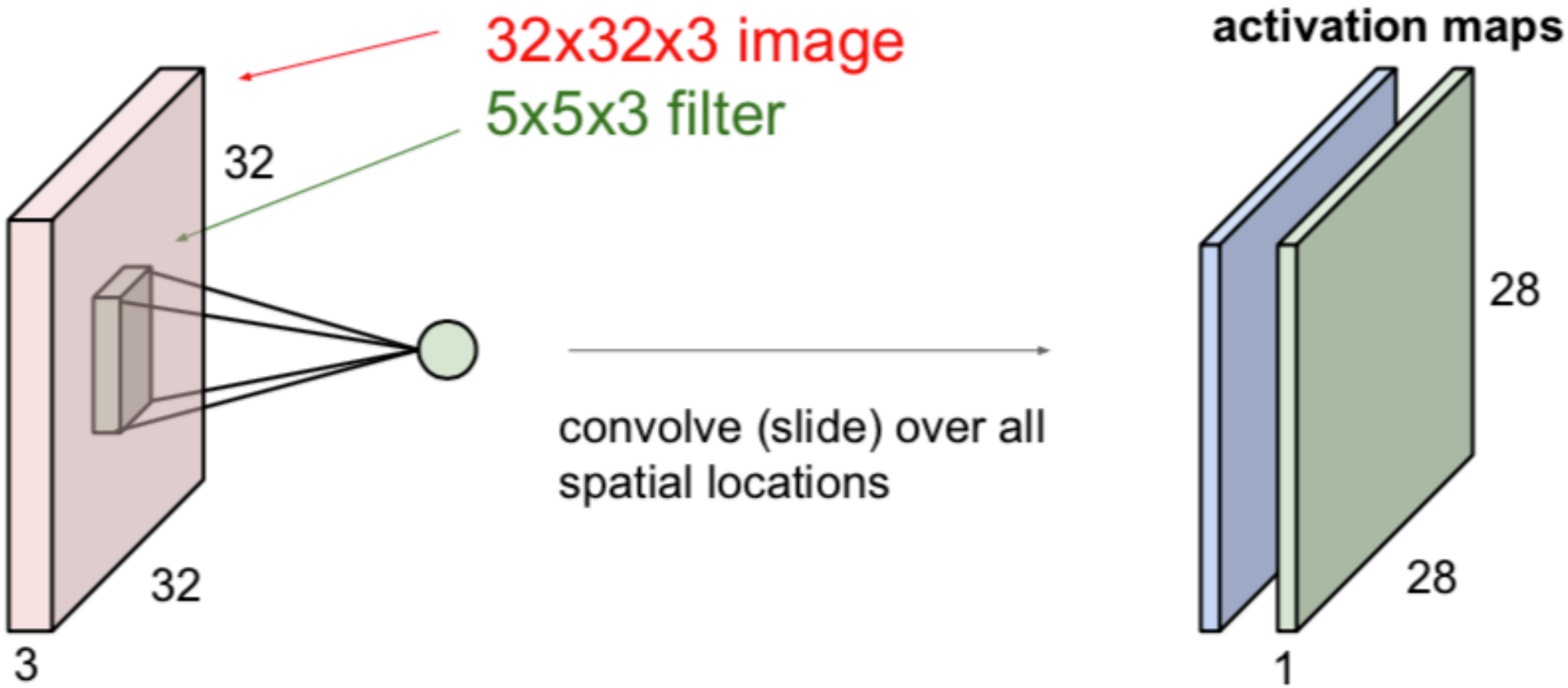
(b) 

$$\textit{edge detection kernel} = \begin{pmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{pmatrix}$$

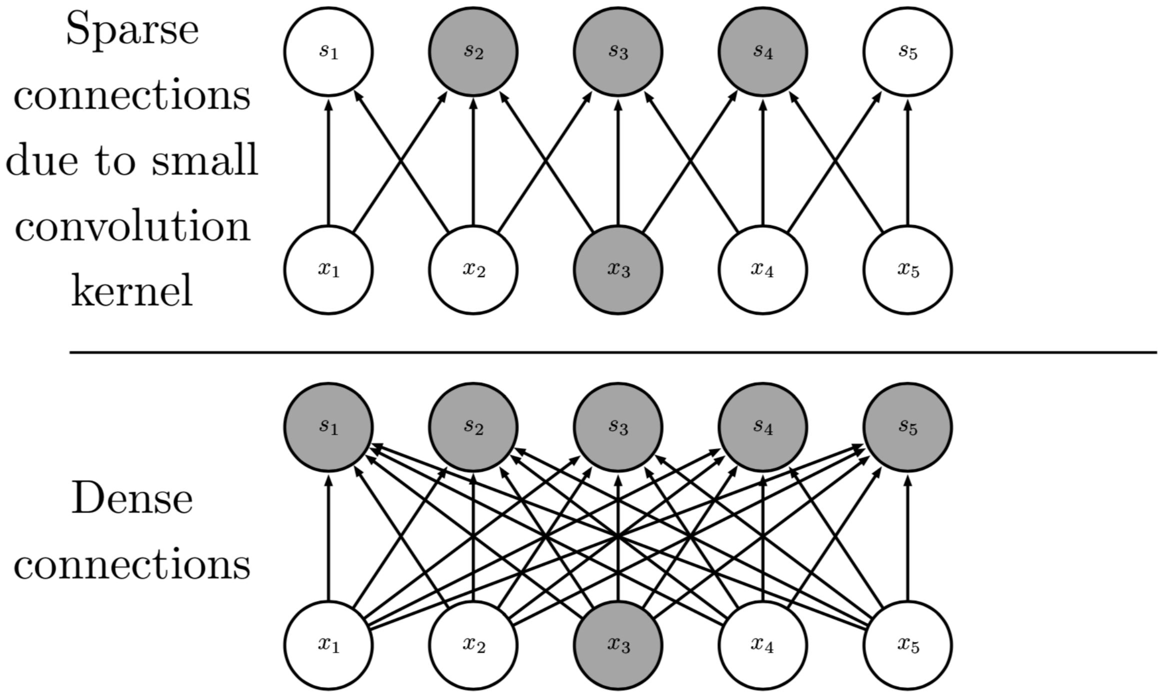
$$\textit{sharpen kernel} = \begin{pmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{pmatrix}.$$

$$\textit{Gaussian blur kernel} = \begin{pmatrix} \frac{1}{16} & \frac{2}{16} & \frac{1}{16} \\ \frac{2}{16} & \frac{4}{16} & \frac{2}{16} \\ \frac{1}{16} & \frac{2}{16} & \frac{1}{16} \end{pmatrix}.$$

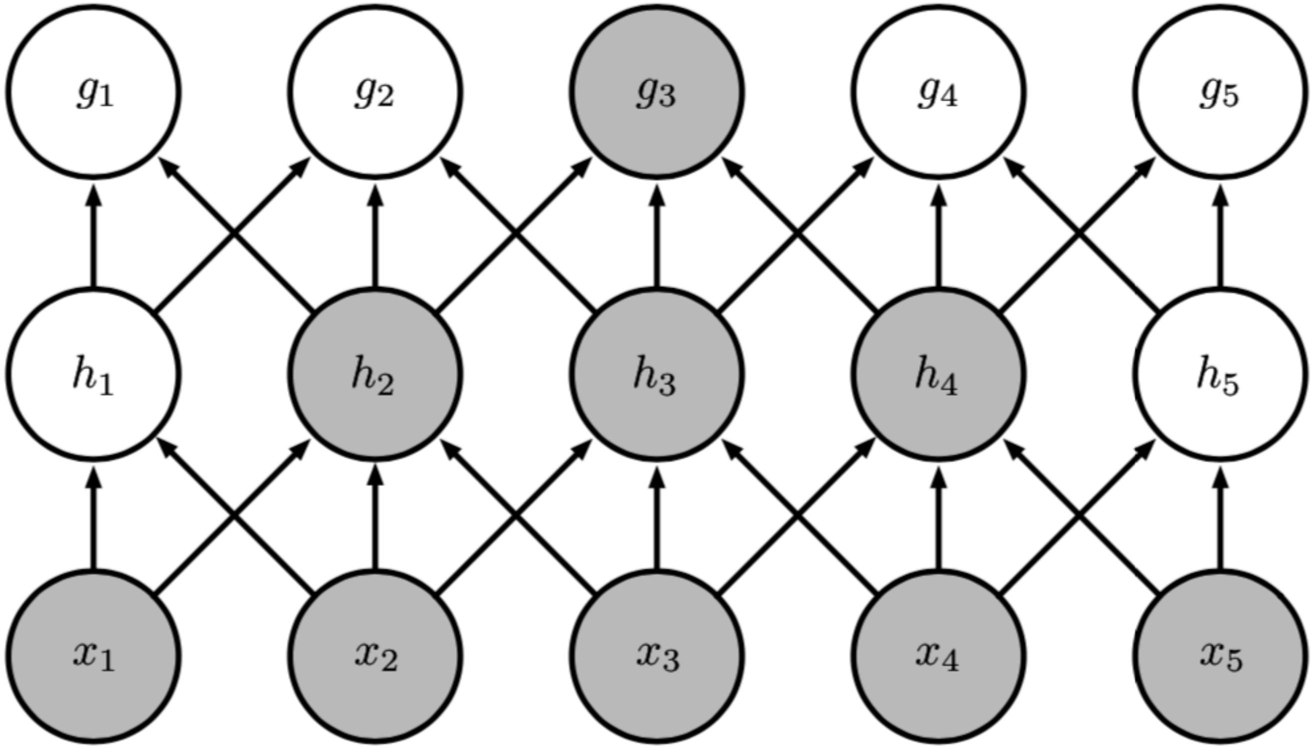
Convolution Layer



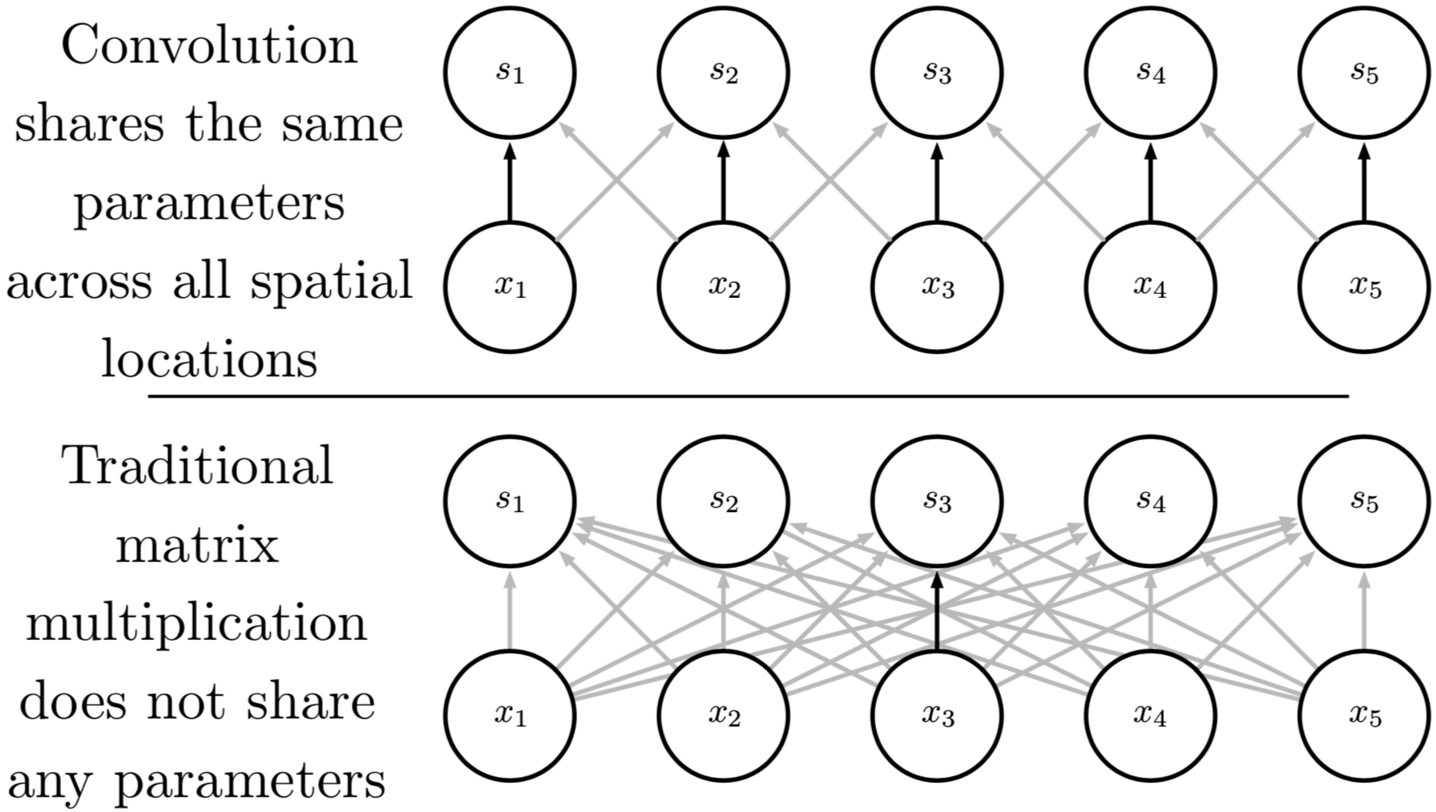
Sparse Connectivity



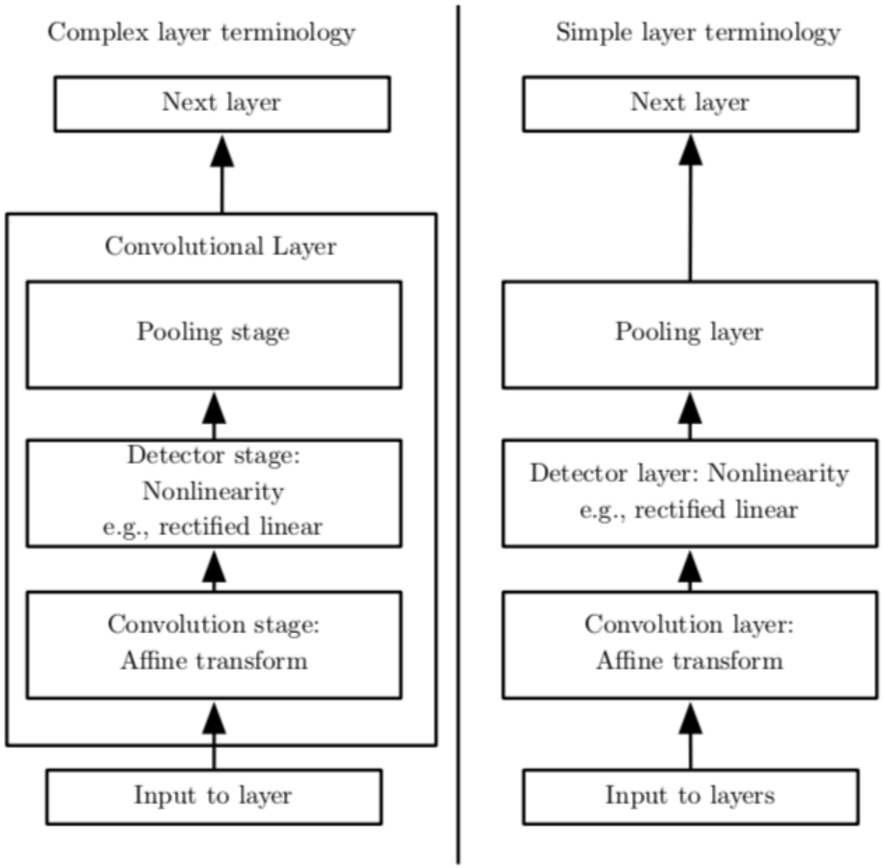
Growing Receptive Field



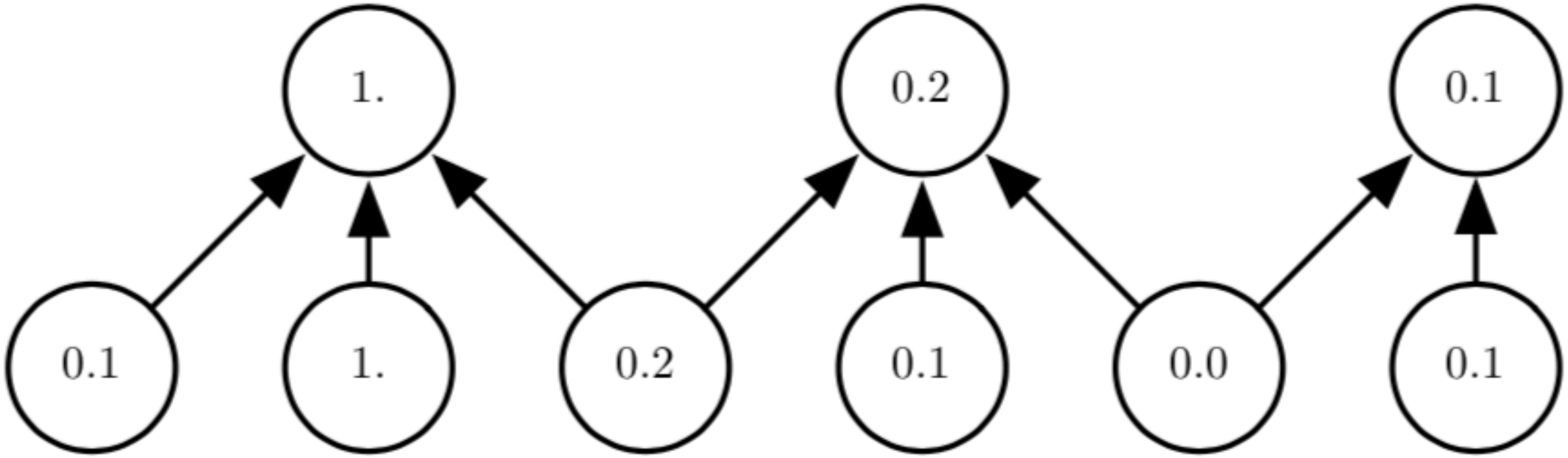
Parameter Sharing



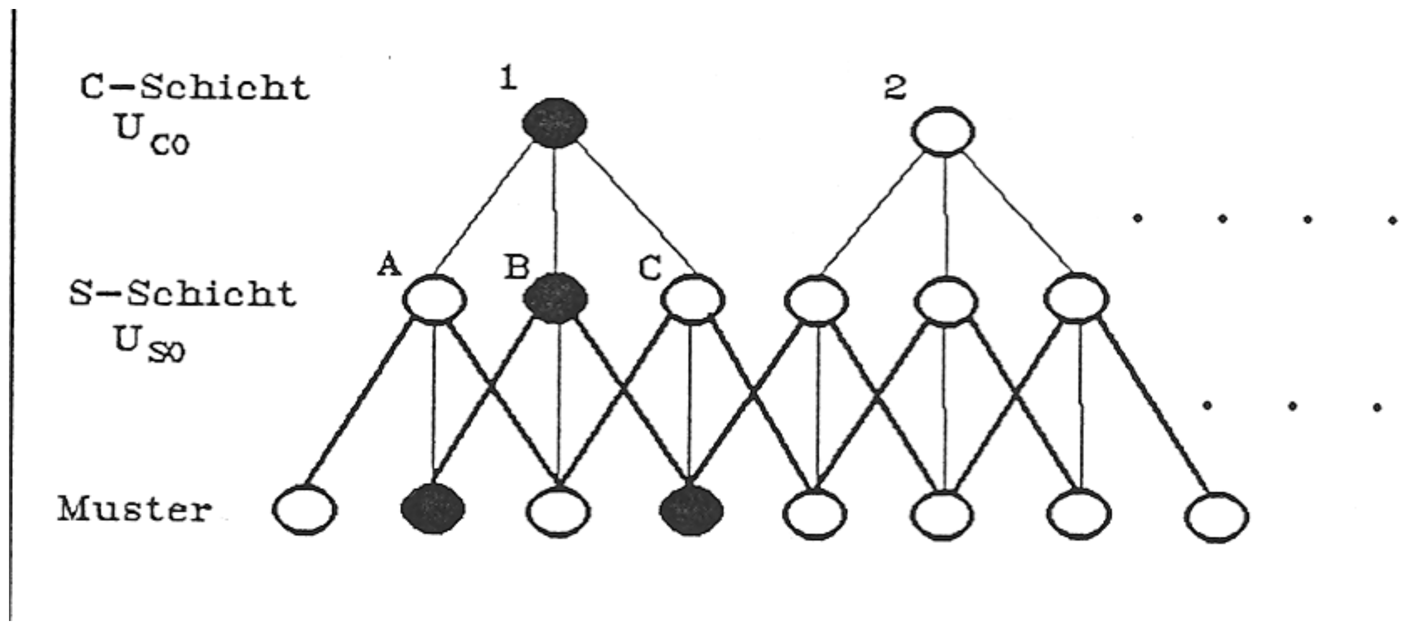
Convolutional Network Components



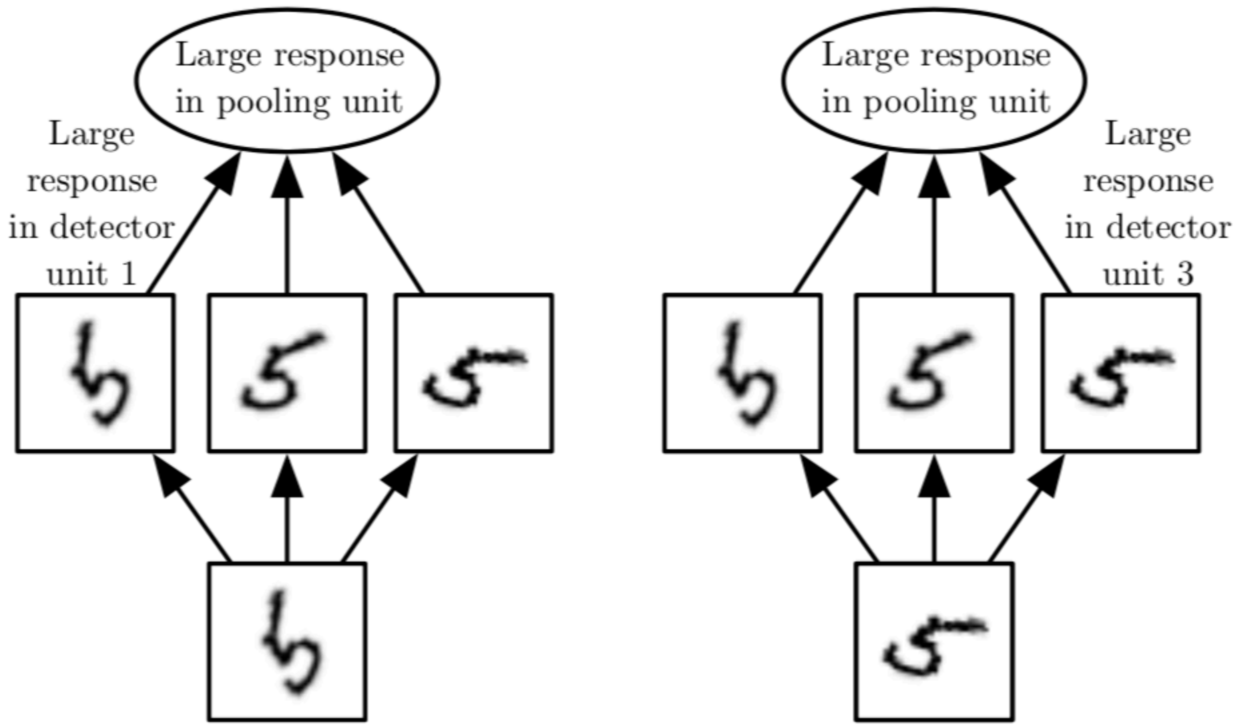
Pooling with Downsampling



Pooling

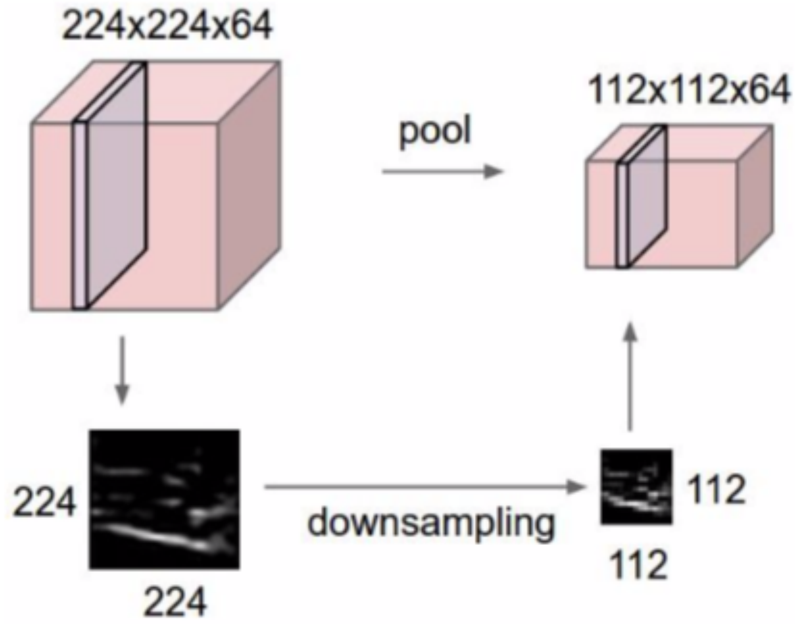


Cross-Channel Pooling and Invariance

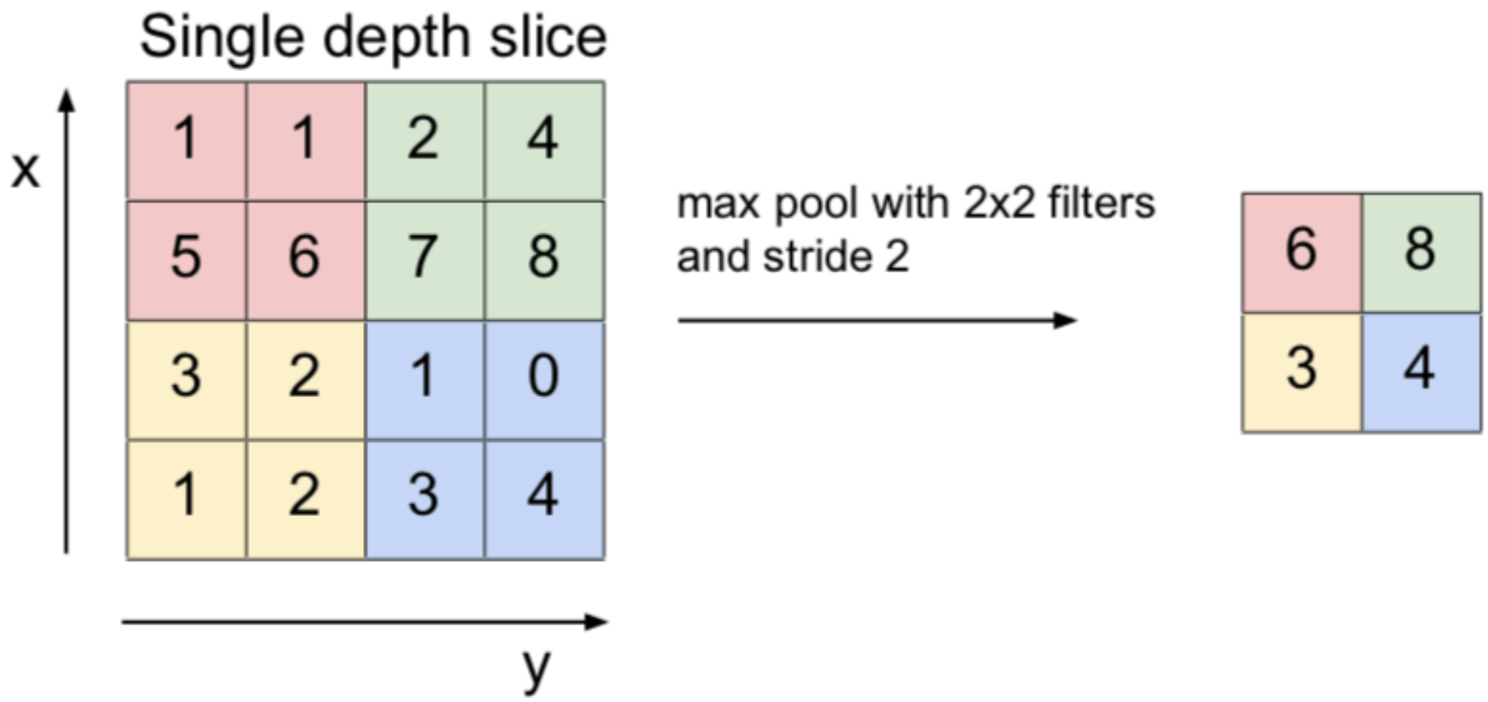


Pooling

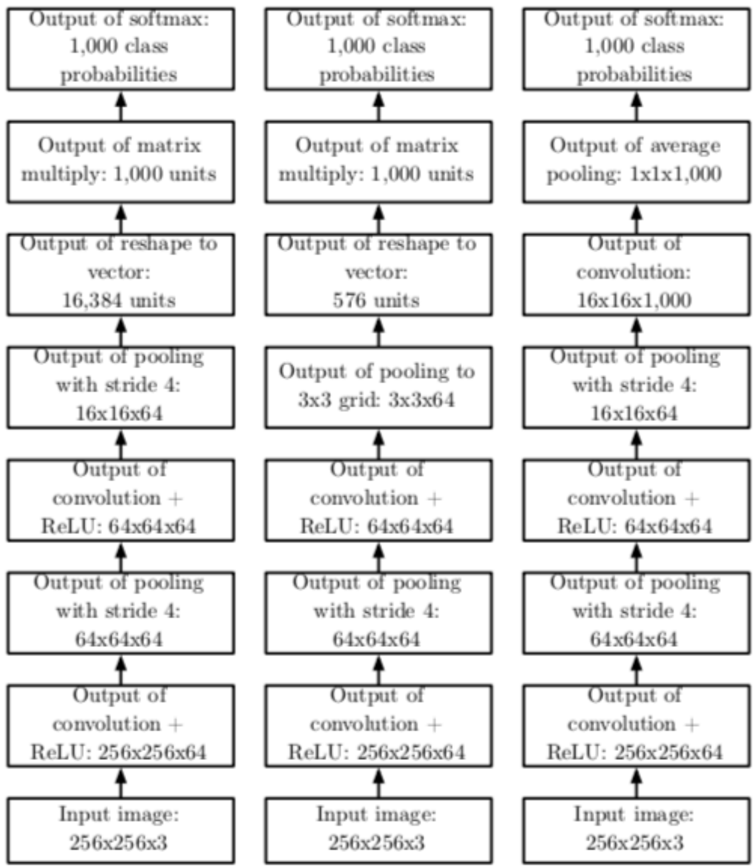
- makes the representations smaller and more manageable
- operates over each activation map independently:

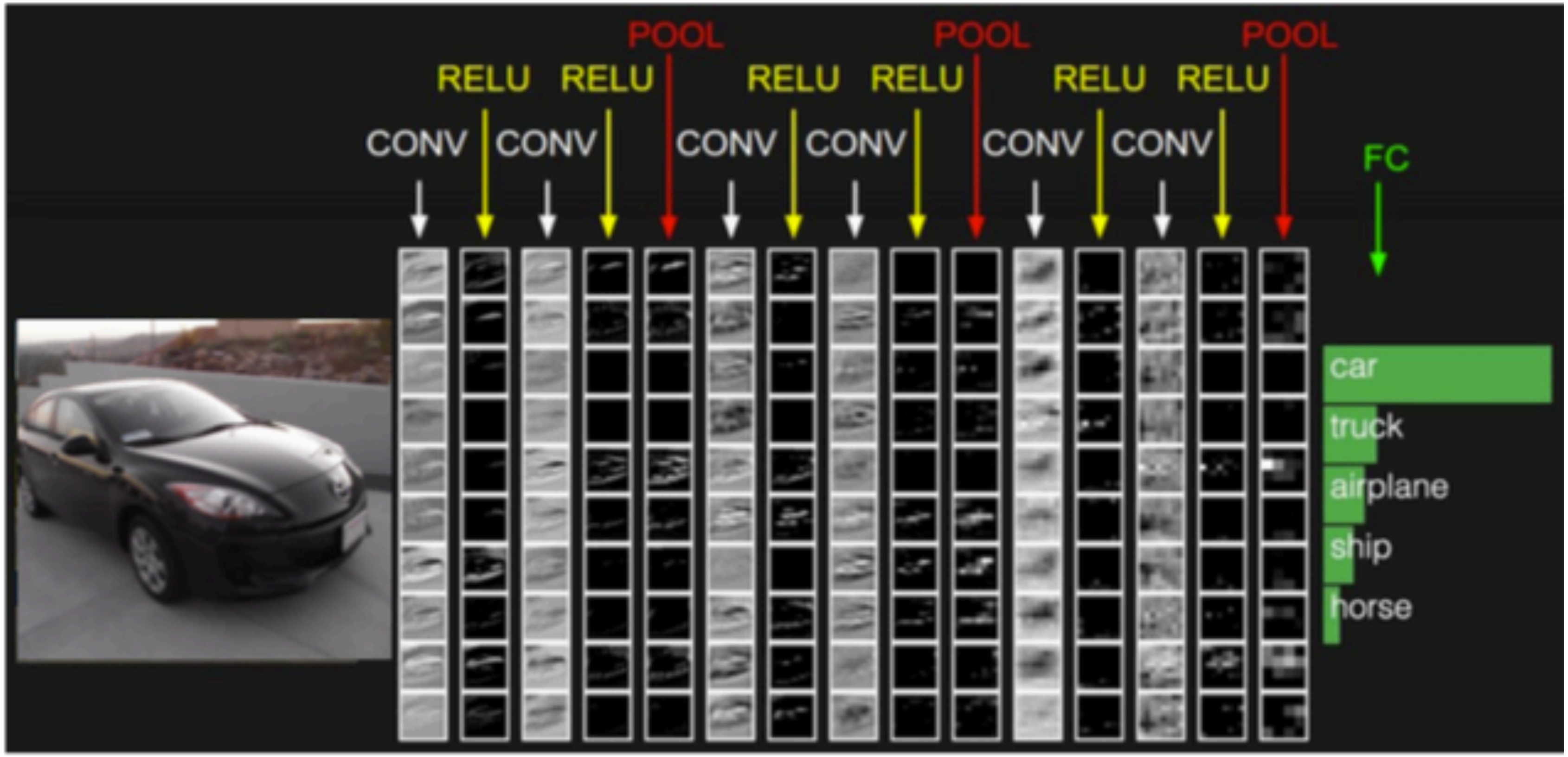


MAX POOLING


Single depth slice
x
y
1 1 2 4
5 6 7 8
3 2 1 0
1 2 3 4
max pool with 2x2 filters
and stride 2
6 8
3 4

Example Classification of Architectures

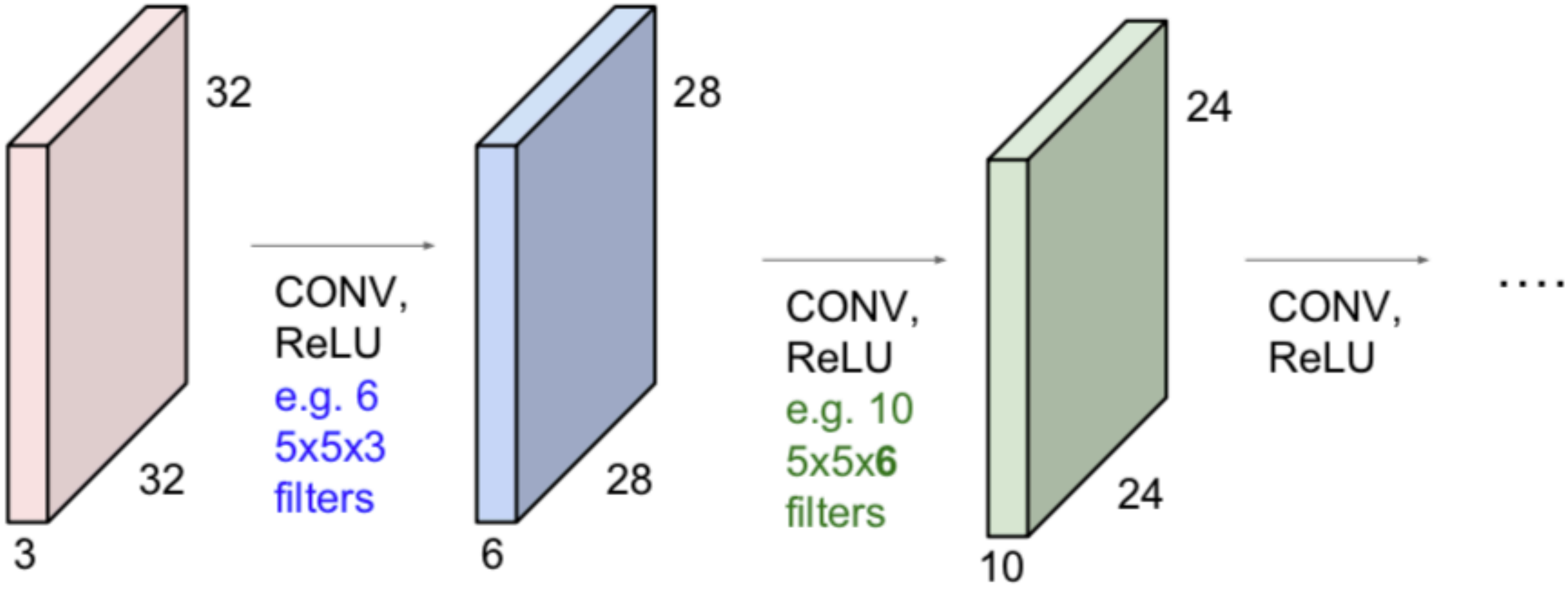


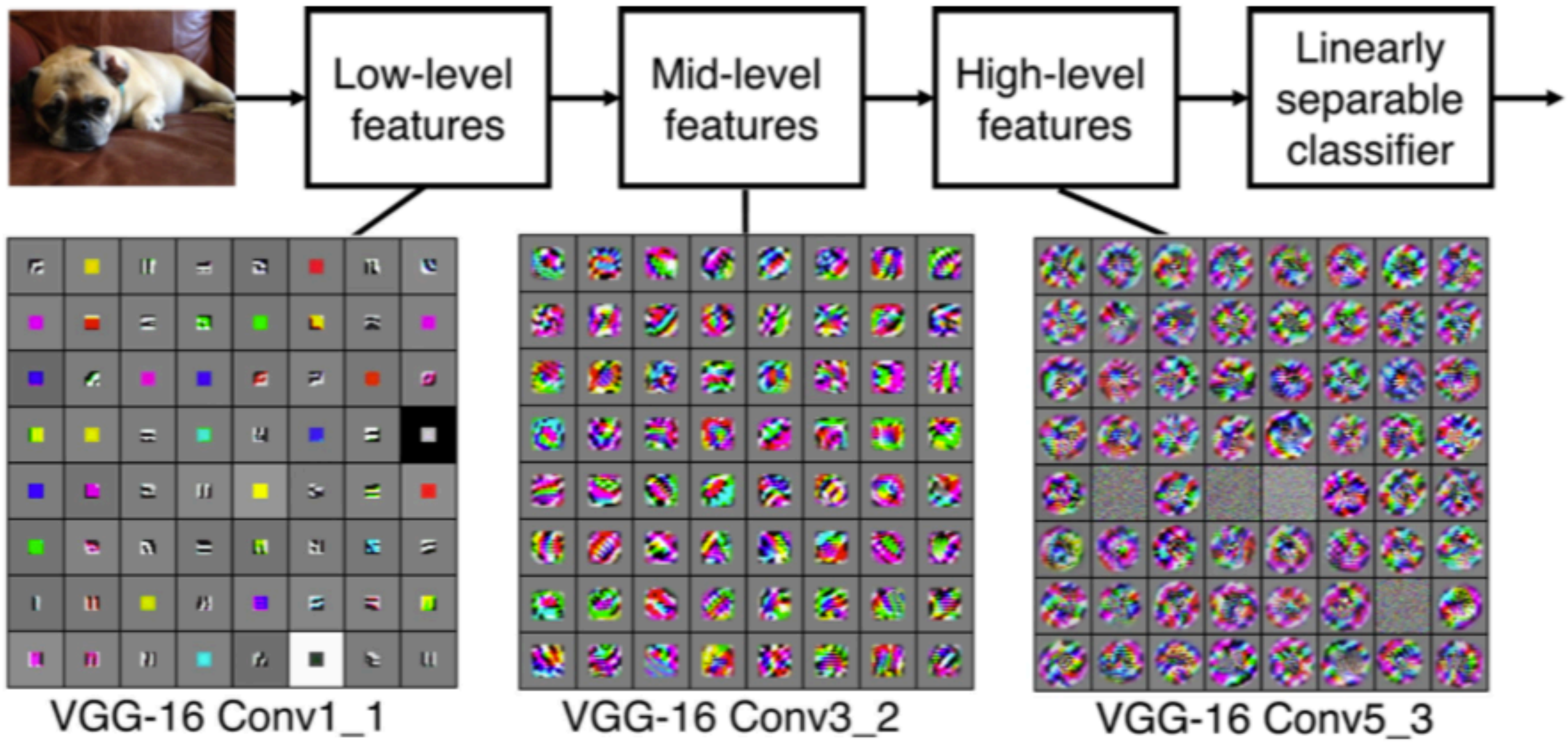

POOL
POOL
POOL
RELU
RELU
RELU
RELU
RELU
RELU
CONV
CONV
CONV
CONV
CONV
CONV
FC
car
truck
airplane
ship
horse

Architectures

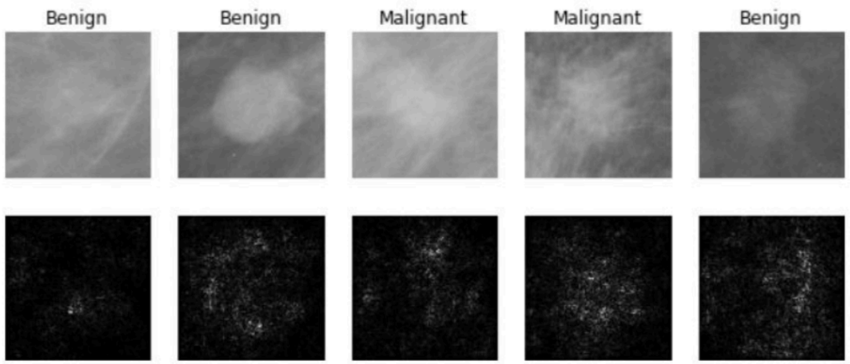
- Spatial Transducer Net: input size scales with output size, all layers are convolutional
- All Convolutional Net: no pooling layers, just use strided convolution to shrink representation size

ConvNet is a sequence of Convolution Layers, interspersed with activation functions




Low-level features
Mid-level features
High-level features
Linearly separable classifier
VGG-16 Conv1_1
VGG-16 Conv3_2
VGG-16 Conv5_3

Fast-forward to today: ConvNets are everywhere



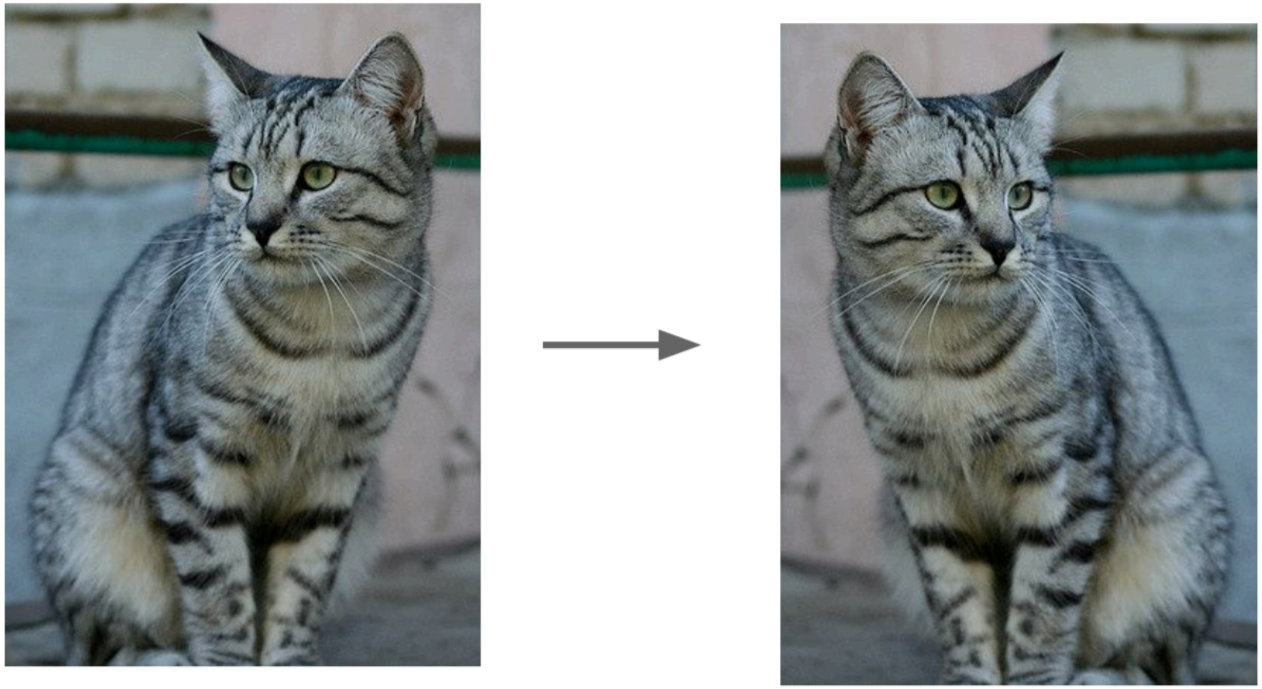
[Levy et al. 2016]

[Dieleman et al. 2014]

[Sermanet et al. 2011]
[Ciresan et al.]

Data Augmentation

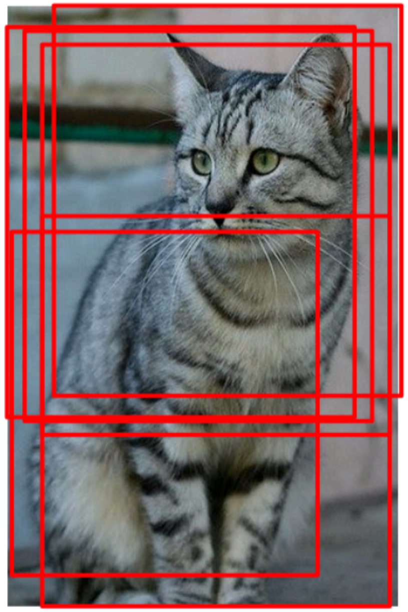
- Horizontal Flips to the original image



Data Augmentation

- **Training**: sample random crops / scales

- ResNet:
  - Pick random L in range [256, 480]
  - Resize training image, short side = L
  - Sample random 224 x 224 patch



Data Augmentation

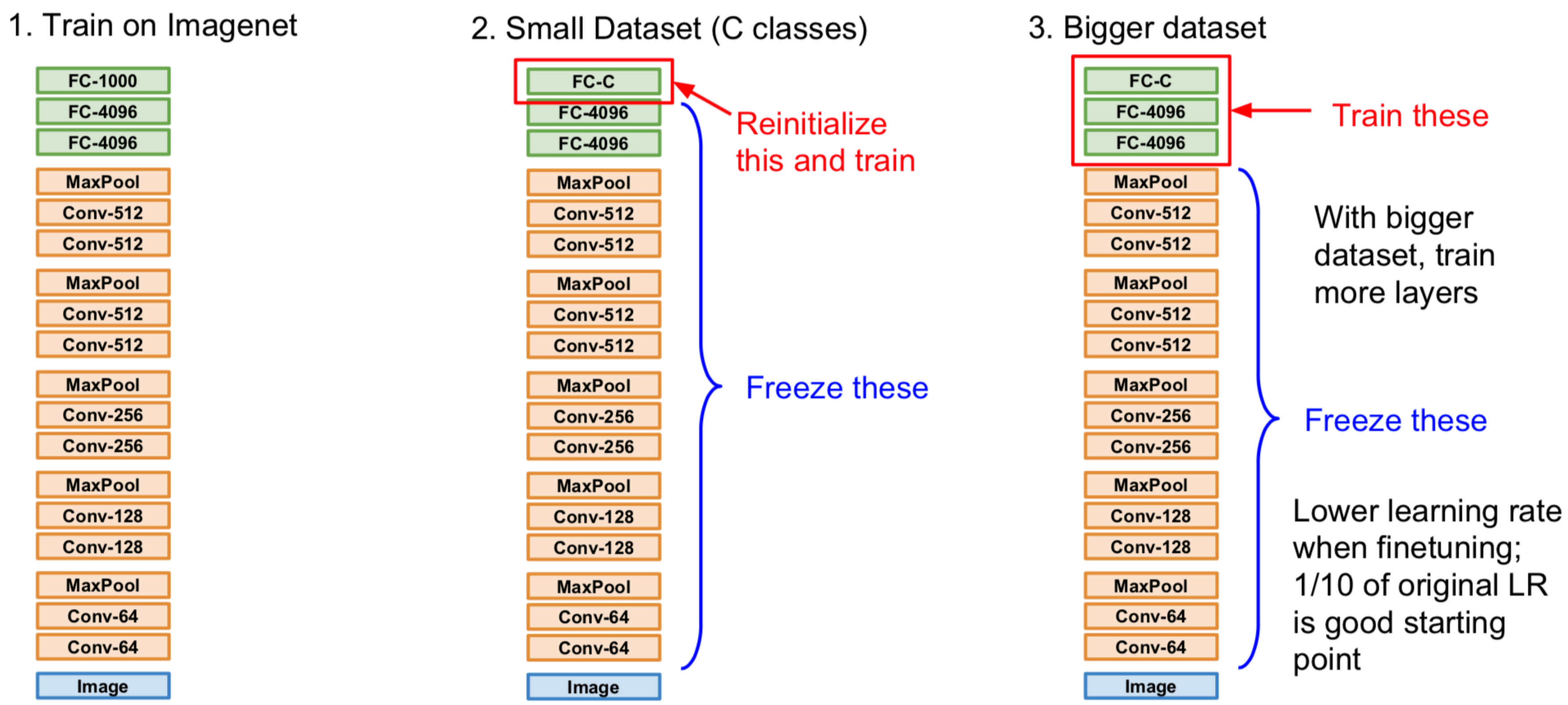
- Color Jitter
- Simple: Randomize contrast and brightness

- Apply PCA to all [R, G, B]
  - pixels in training set
  - Sample a “color offset” along principal component directions
  - Add offset to all pixels of a training image

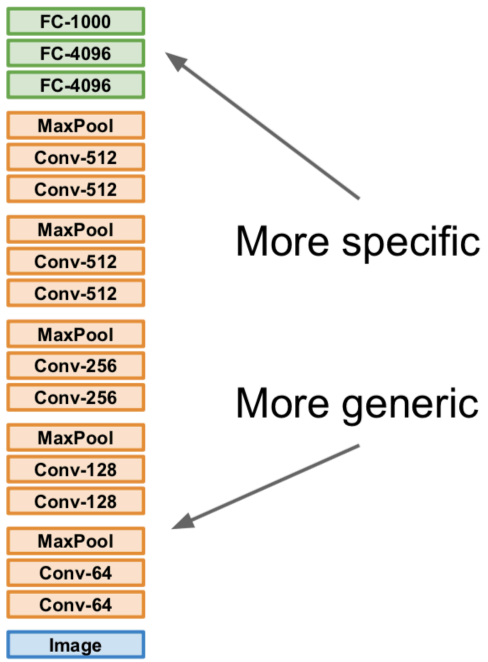
Transfer Learning

- You need a lot of a data if you want to train
- Transfer learning and domain adaptation refer to the situation where what has been learned in one setting (i.e., distribution P_1) is exploited to improve generalization in another setting (say distribution P_2).
- We assume that many of the factors that explain the variations in P_1 are relevant to the variations that need to be captured for learning P_2.

Transfer Learning with CNNs

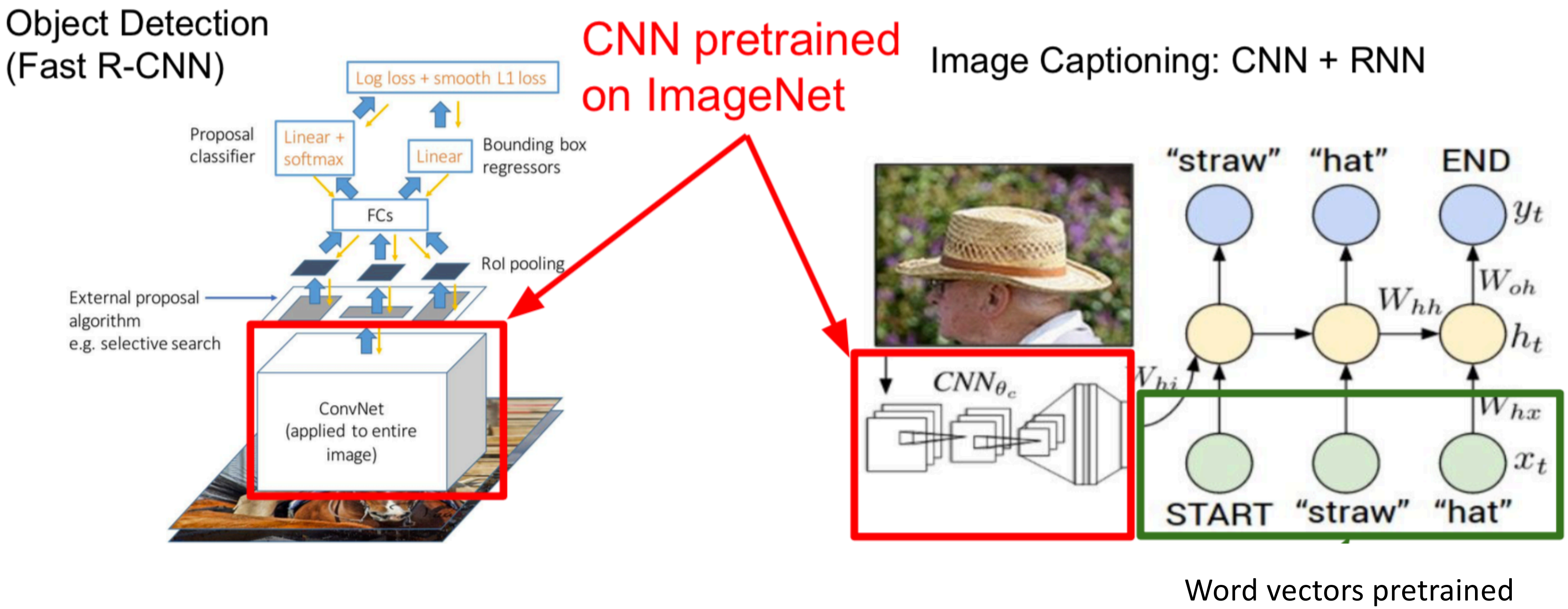


Transfer Learning with CNNs



	very similar dataset	**very different dataset**
very little data	Use Linear Classifier on top layer	You're in trouble… Try linear classifier from different stages
quite a lot of data	Finetune a few layers	Finetune a larger number of layers

Transfer learning with CNNs is common



TensorFlow: Pretrained Models

tf.keras:

https://www.tensorflow.org/api_docs/python/tf/keras/applications

TF-Slim:

https://github.com/tensorflow/models/tree/master/slim/nets

Tensorflow

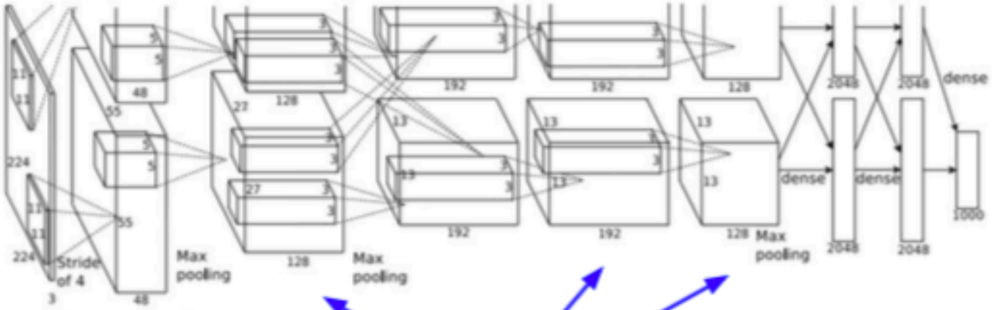
- Ships with Tensorflow

- tf.keras (https://www.tensorflow.org/api_docs/python/tf/keras)
- tf.layers (https://www.tensorflow.org/api_docs/python/tf/layers)
- tf.estimator (https://www.tensorflow.org/api_docs/python/tf/estimator)
- tf.contrib.estimator (https://www.tensorflow.org/api_docs/python/tf/contrib/estimator)
- tf.contrib.layers (https://www.tensorflow.org/api_docs/python/tf/contrib/layers)
- tf.contrib.slim (https://github.com/tensorflow/tensorflow/tree/master/tensorflow/contrib/slim)

- Third Party
- TFLearn (http://tflearn.org/)
- TensorLayer (http://tensorlayer.readthedocs.io/en/latest/)

Case Study: AlexNet

[Krizhevsky et al. 2012]



Full (simplified) AlexNet architecture:
[227x227x3] INPUT
[55x55x96] CONV1: 96 11x11 filters at stride 4, pad 0
[27x27x96] MAX POOL1: 3x3 filters at stride 2
[27x27x96] NORM1: Normalization layer
[27x27x256] CONV2: 256 5x5 filters at stride 1, pad 2
[13x13x256] MAX POOL2: 3x3 filters at stride 2
[13x13x256] NORM2: Normalization layer
[13x13x384] CONV3: 384 3x3 filters at stride 1, pad 1
[13x13x384] CONV4: 384 3x3 filters at stride 1, pad 1
[13x13x256] CONV5: 256 3x3 filters at stride 1, pad 1
[6x6x256] MAX POOL3: 3x3 filters at stride 2
[4096] FC6: 4096 neurons
[4096] FC7: 4096 neurons
[1000] FC8: 1000 neurons (class scores)

CONV1, CONV2, CONV4, CONV5:
Connections only with feature maps on same GPU

Details/Retrospectives:

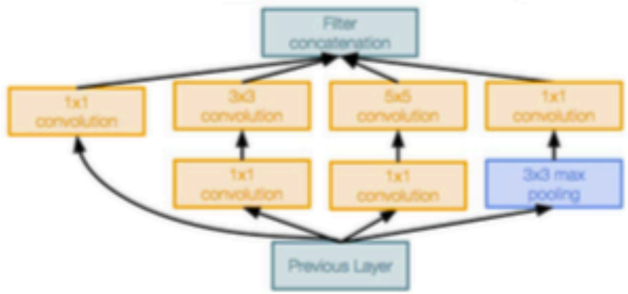
- first use of ReLU
- used Norm layers (not common anymore) - heavy data augmentation
- dropout 0.5
- batch size 128
- SGD Momentum 0.9
- Learning rate 1e-2, reduced by 10 manually when val accuracy plateaus
- L2 weight decay 5e-4
- 7 CNN ensemble: 18.2% -> 15.4%

Case Study: GoogLeNet

[Szegedy et al., 2014]

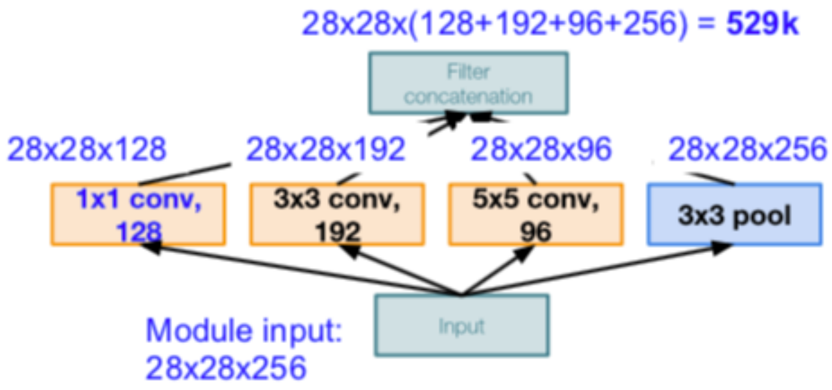
Deeper networks, with computational efficiency

- 22 layers
- Efficient “Inception” module
- No FC layers
- Only 5 million parameters!
  12x less than AlexNet
- ILSVRC’14 classification winner
  (6.7% top 5 error)

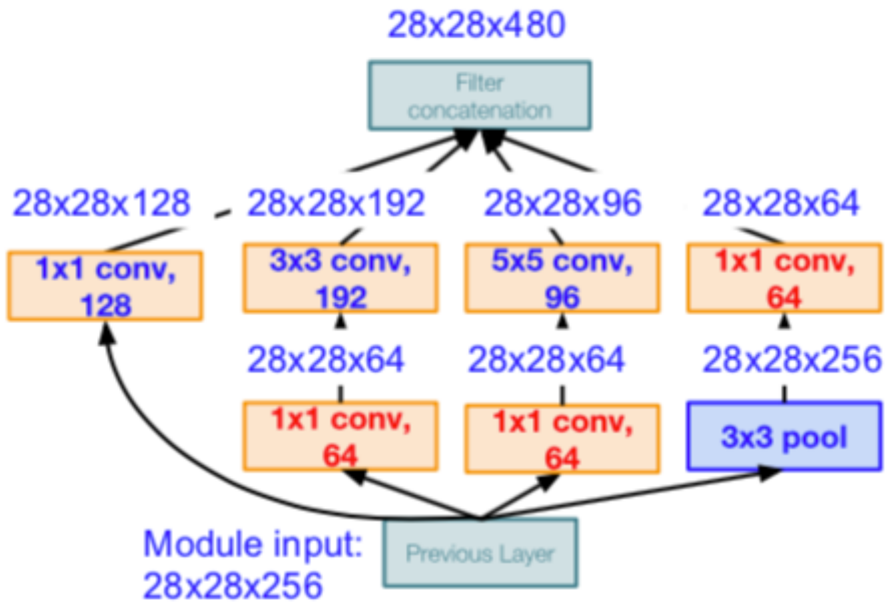


Inception module

Interception Module



Naive Inception module

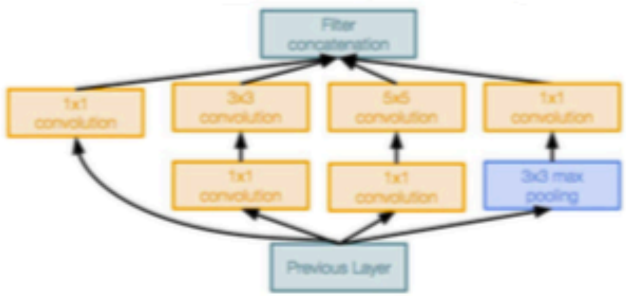


Inception module with dimension reduction

Case Study: GoogLeNet

[Szegedy et al., 2014]

Stack Inception modules
with dimension reduction
on top of each other

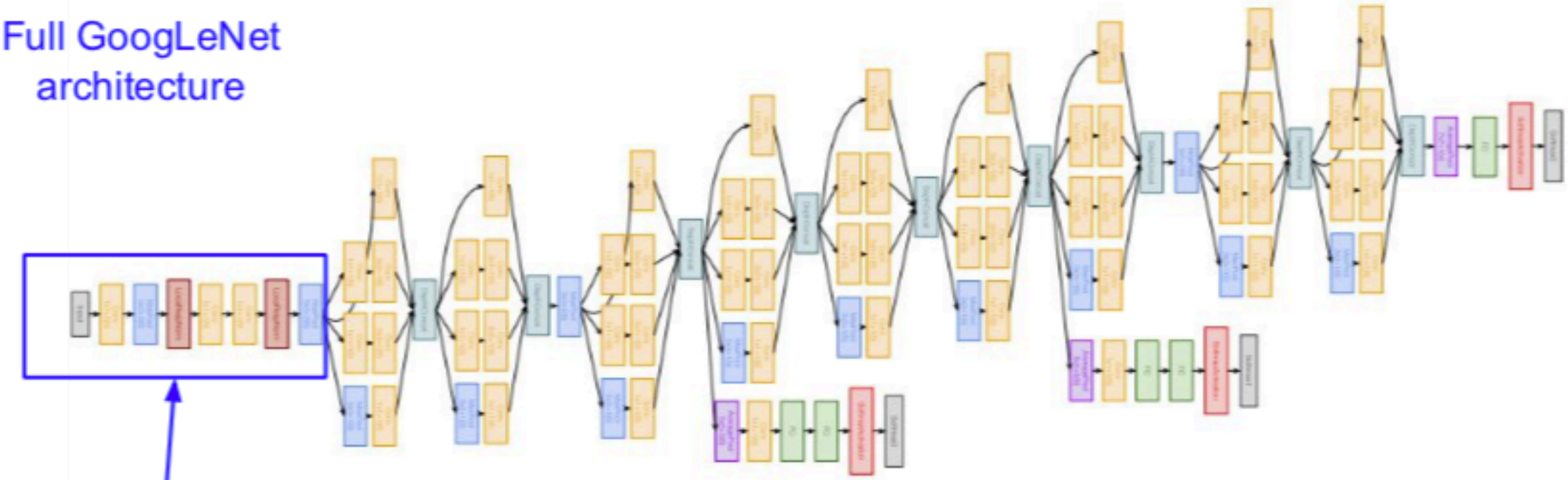


Inception module

Case Study: GoogLeNet

[Szegedy et al., 2014]

Full GoogLeNet architecture

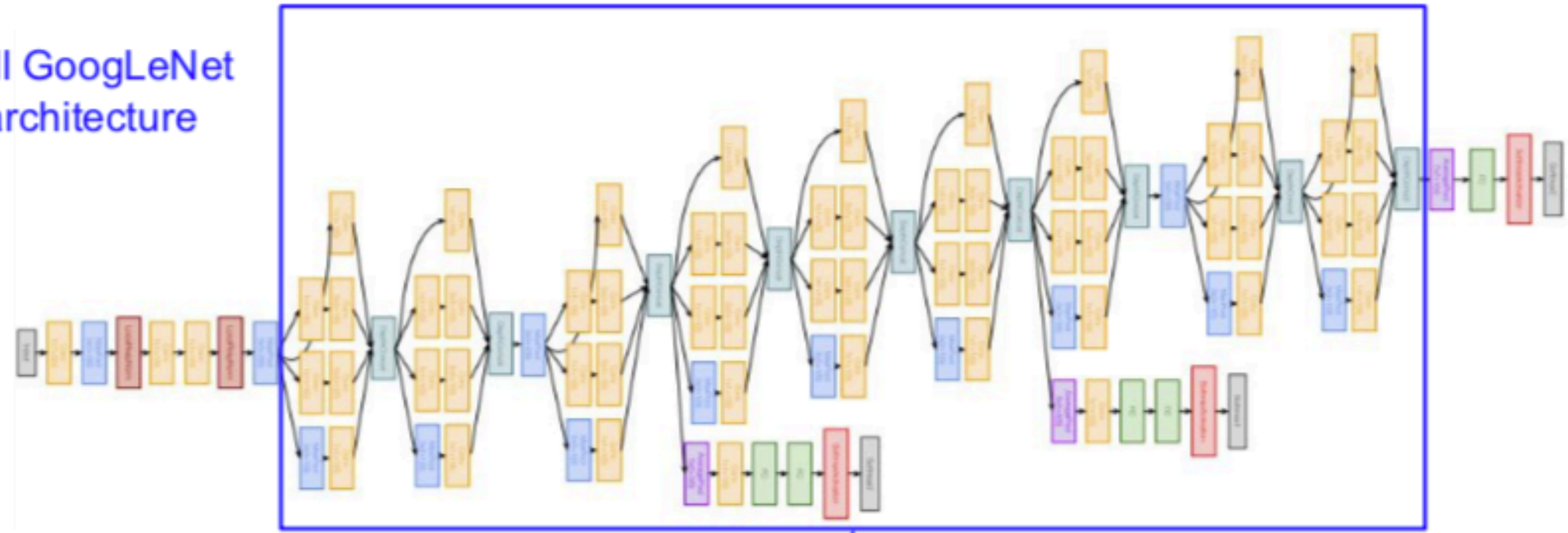


Stem Network:
Conv-Pool-
2x Conv-Pool

Case Study: GoogLeNet

[Szegedy et al., 2014]

Full GoogLeNet architecture

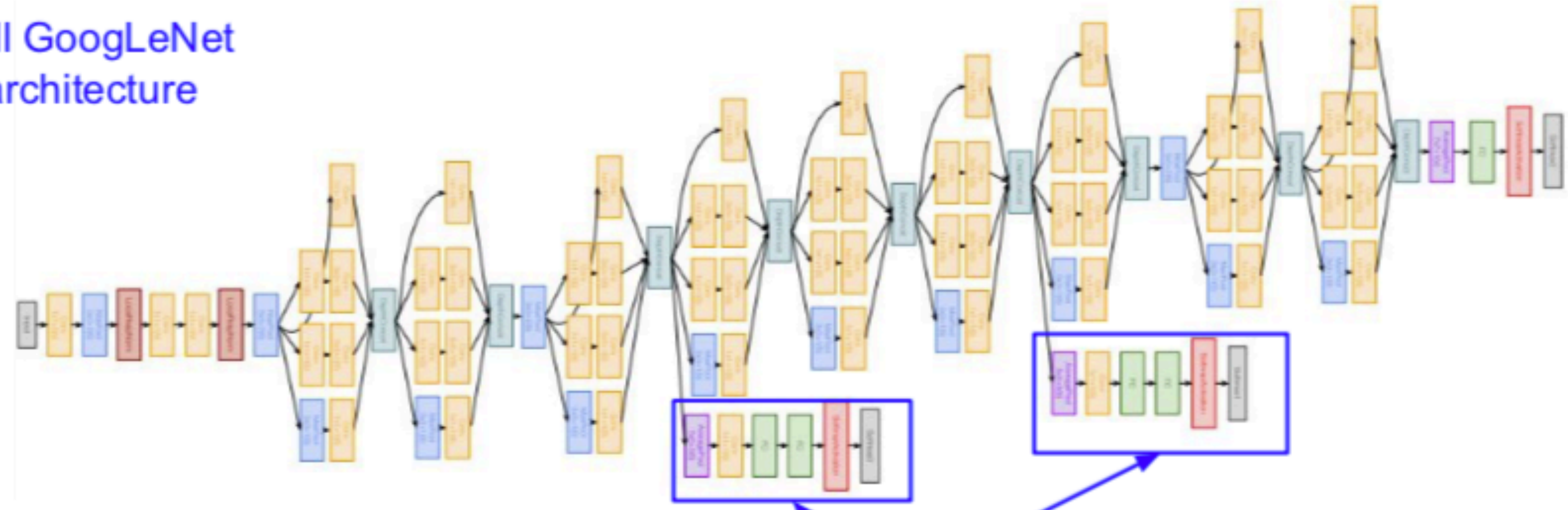


Stacked Inception Modules

Case Study: GoogLeNet

[Szegedy et al., 2014]

Full GoogLeNet architecture

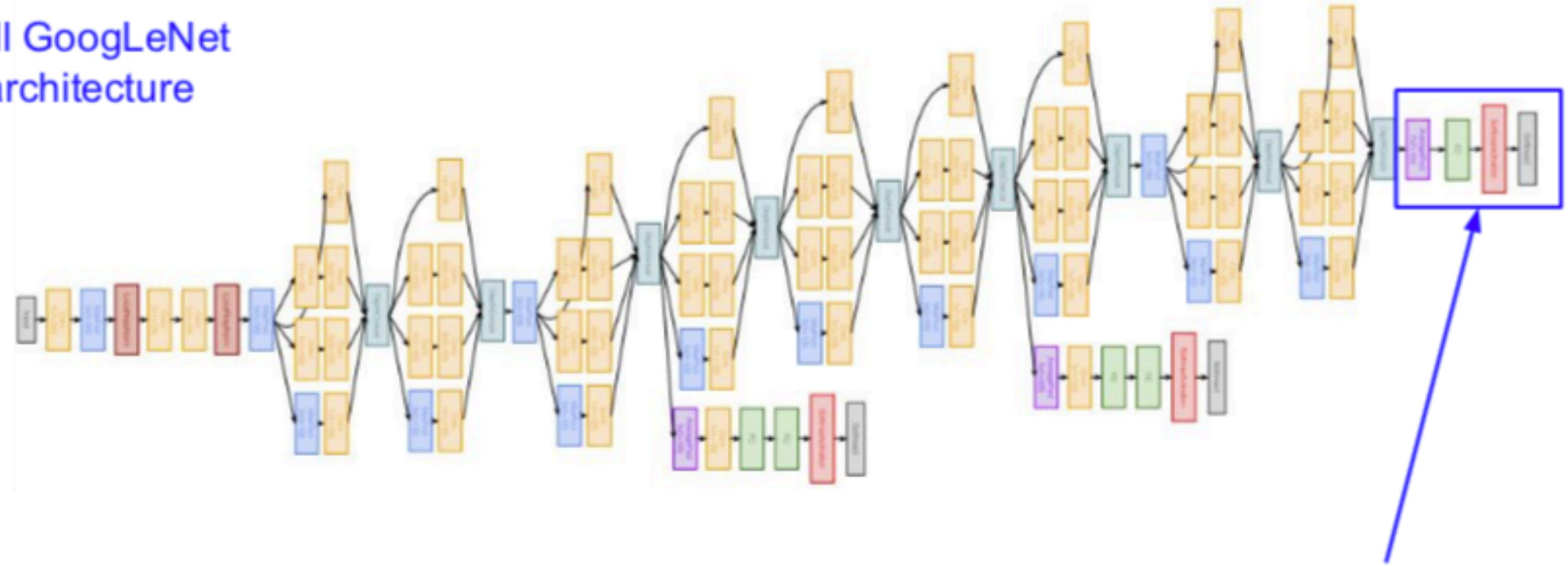


Auxiliary classification outputs to inject additional gradient at lower layers
(AvgPool-1x1Conv-FC-FC-Softmax)

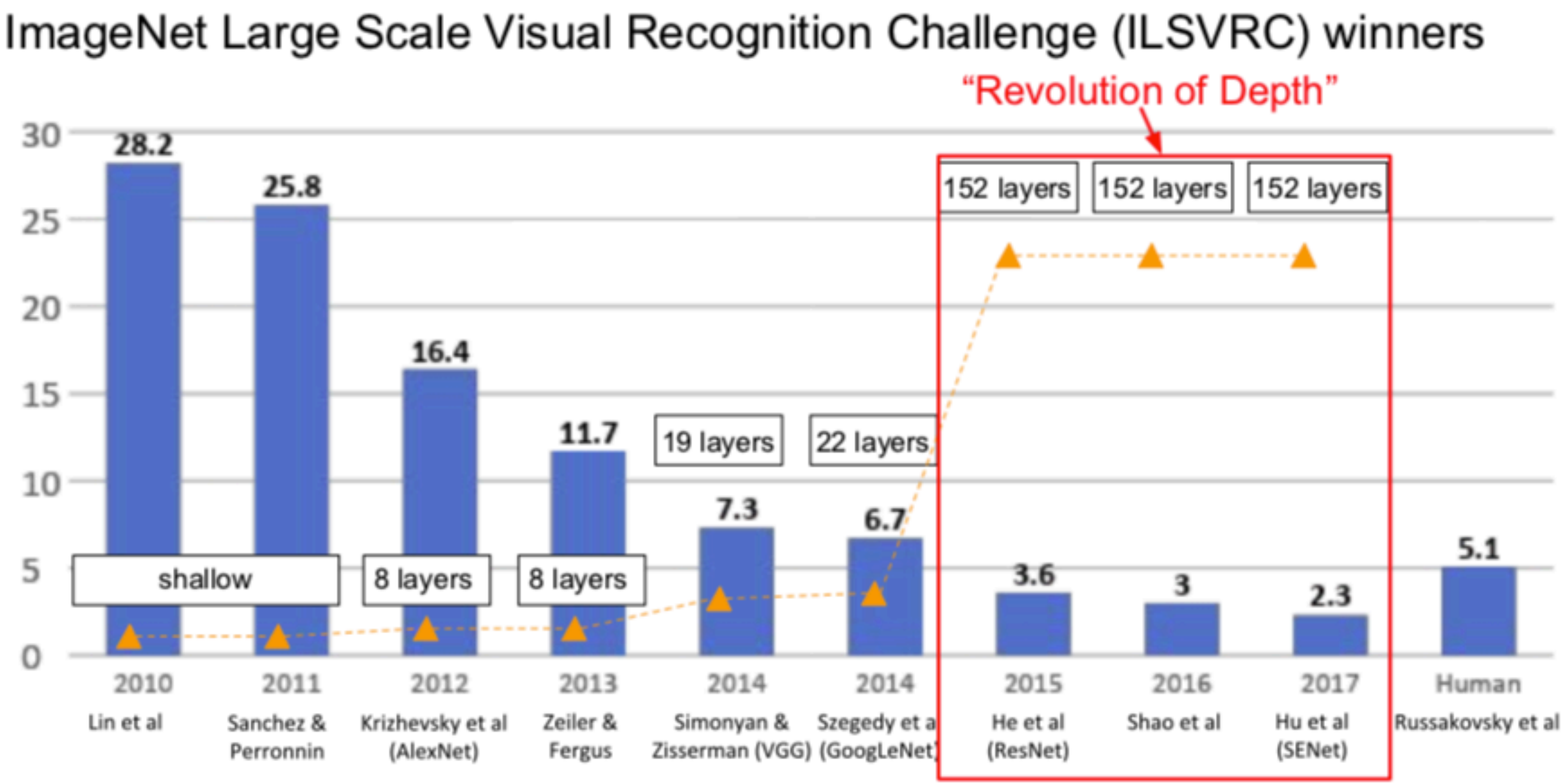
Case Study: GoogLeNet

[Szegedy et al., 2014]

Full GoogLeNet architecture

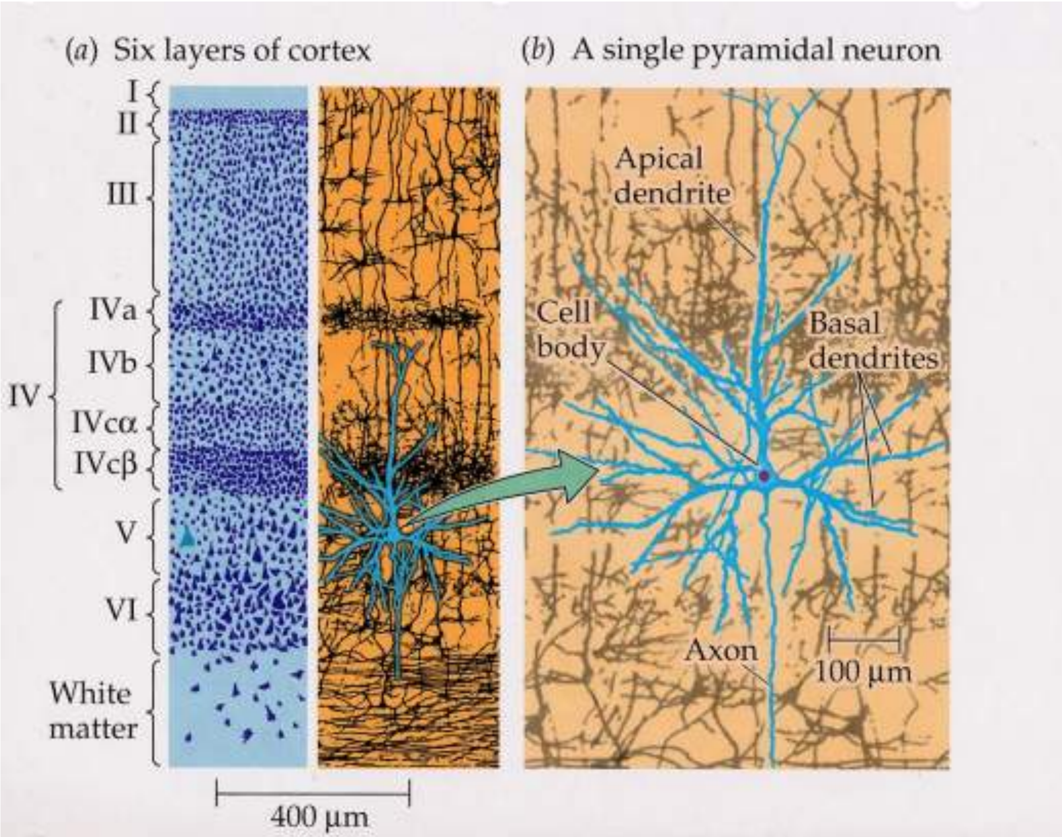


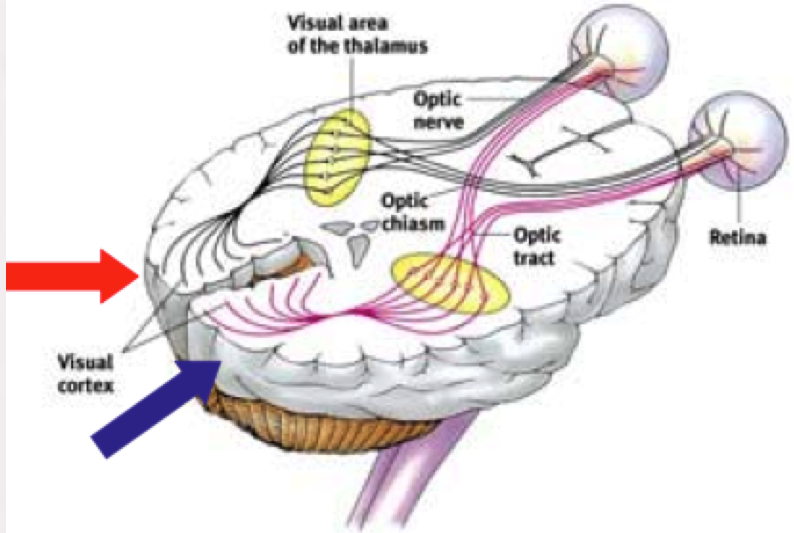
Classifier output
(removed expensive FC layers!)



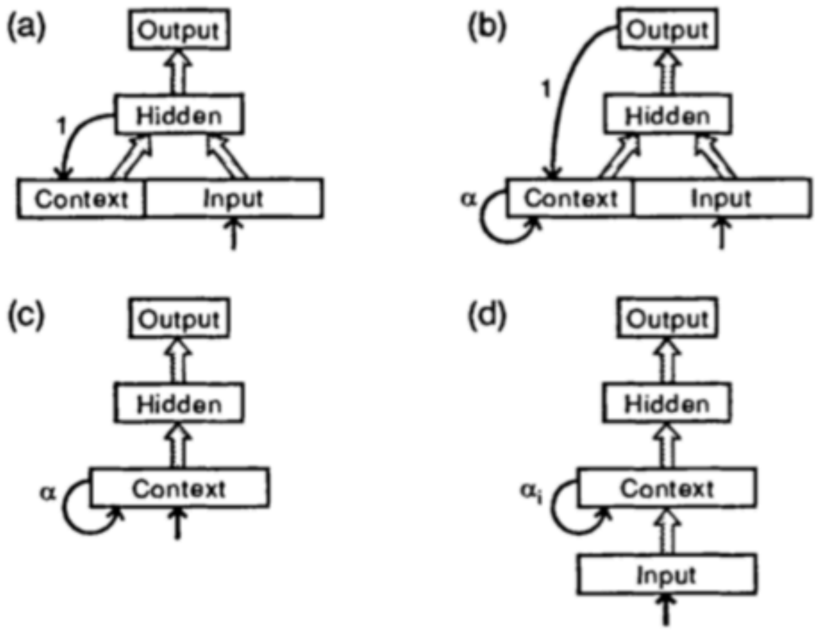
- This has **nothing** to do with Brain or Visual Cortex

Cortex

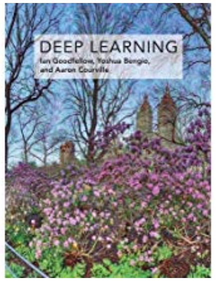

(a) Six layers of cortex
I
II
III
IVa
IVb
IV
IVcα
IVcβ
V
VI
White matter
400 μm
(b) A single pyramidal neuron
Apical dendrite
Cell body
Basal dendrites
Axon
100 μm


Visual area of the thalamus
Optic nerve
Optic chiasm
Optic tract
Retina
Visual cortex

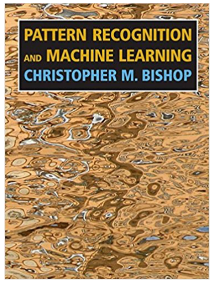
- Next: Recurrent Neural Networks



Literature

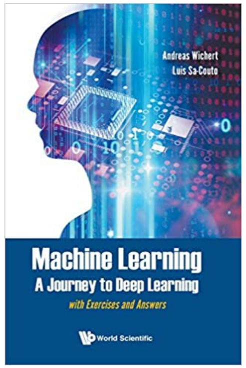


- Deep Learning, I. Goodfellow, Y. Bengio, A. Courville MIT Press 2016
  - Chapter 9



- Christopher M. Bishop, Pattern Recognition and Machine Learning (Information Science and Statistics), Springer 2006
  - Section 5.5.6

Literature



- Machine Learning - A Journey to Deep Learning, A. Wichert, Luis Sa-Couto, World Scientific, 2021
  - Chapter 13