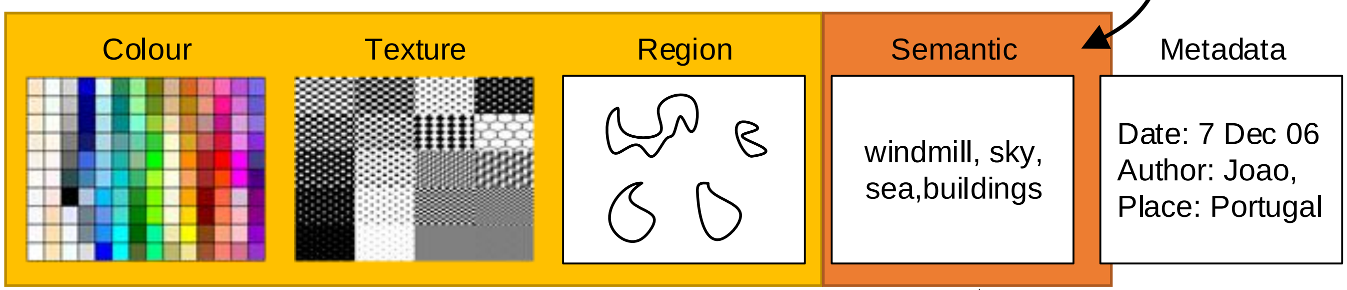
Learning Visual Data Representations

Convolutional Neural Networks

Web Data Mining and Search

Hand-crafted vs learned feature extraction

Approach B: end-to-end classifiers <u>learn task specific data representations</u>.



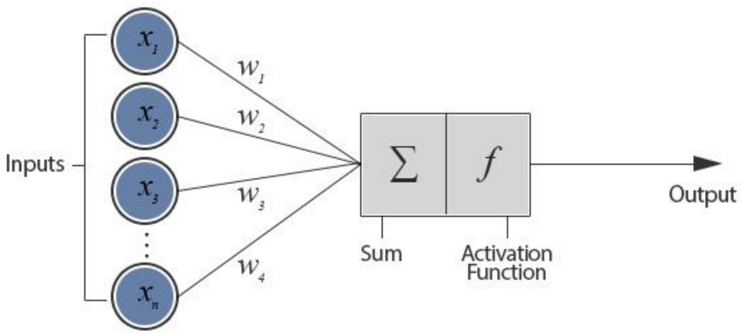
Approach A: classifiers learn with <u>hand-crafted methods for feature extraction</u>.

Perceptron: general formulation

- **Binary classification:**

$$z = w_0 + w_1 x_1 + \ \ldots + w_n x_n$$

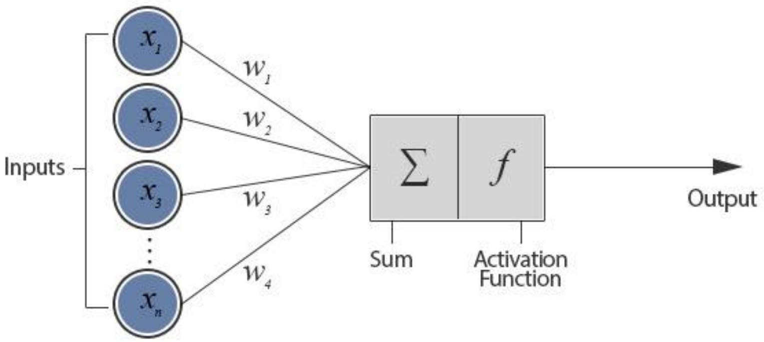
$$\hat{y} = f(z) = \begin{cases} +1 & , if\ z \geq 0 \\ -1 & , if\ z < 0 \end{cases}$$



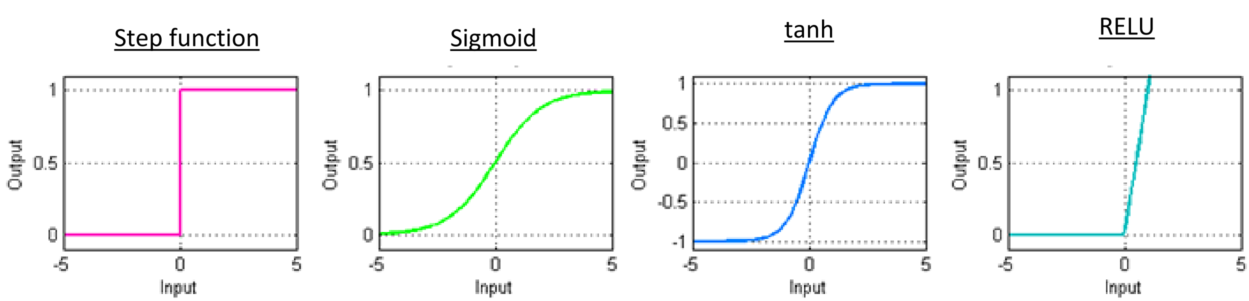
- **Input:** Vectors $\boldsymbol{x^{(j)}}$ and labels $\boldsymbol{y^{(j)}}$
  - Vectors $\boldsymbol{x^{(j)}}$ are real valued where $\|\boldsymbol{x}\|_2 = \mathbf{1}$
- **Goal:** Find vector $\boldsymbol{w} = (\boldsymbol{w_1}, \boldsymbol{w_2}, \ldots, \boldsymbol{w_d})$
  - Each $\boldsymbol{w_i}$ is a real number

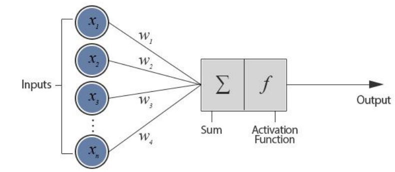
Activation functions

- The perceptron was initially proposed with the step function.
- Historically, other activation functions have been studied.
- The perceptron with the sigmoid activation function corresponds to the logistic regression model.



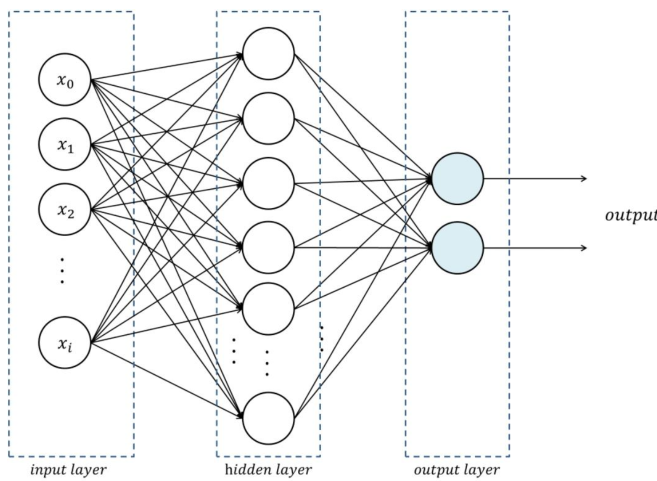
Activation functions



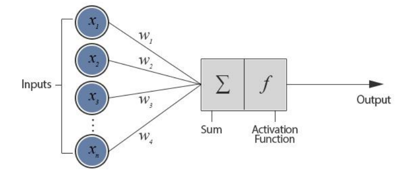


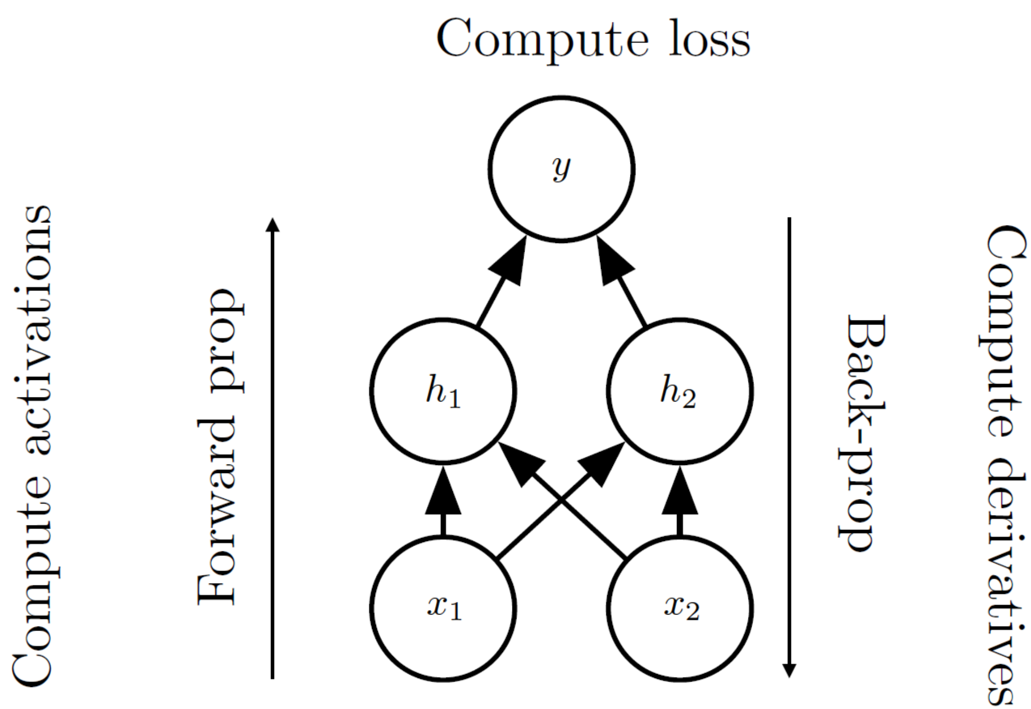
Multi-layer classifiers

- Multi-layer classifiers allow to learn non-linear relations, i.e. complex relationships such as exclusive-OR.
- Usually one to two hidden layers produce the best results.
- Trained with the back-propagation algorithm



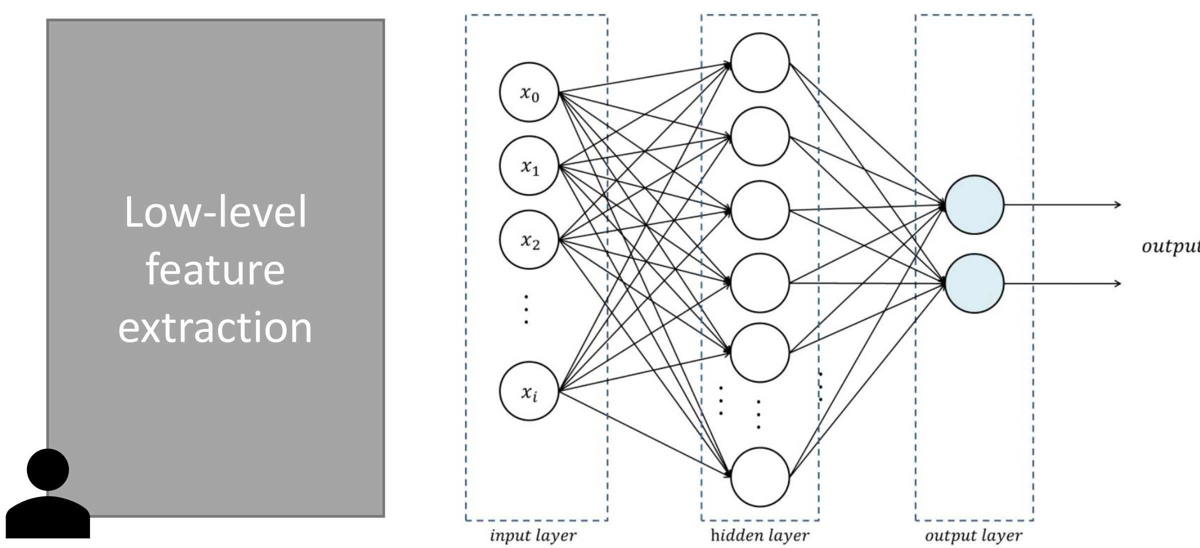
Simple back propagation





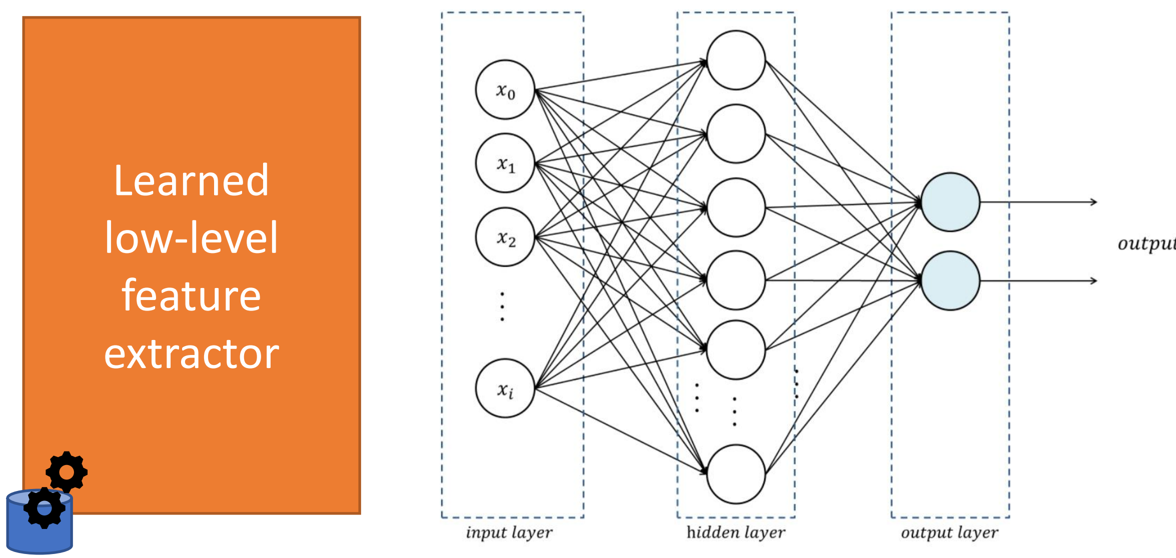
Traditional neural network architectures

- Traditionally, neural networks receive input features that are extracted from data (text, images, etc.) and are task independent.
- This creates a bottleneck: only so much can you learn from those task independente features.



Low-level data representations

- Deep architectures were introduced to learn data representations that were better suited to each task.
- Deep architectures look at the most basic data element, i.e., an image pixel or a text character, to learn new data representations.

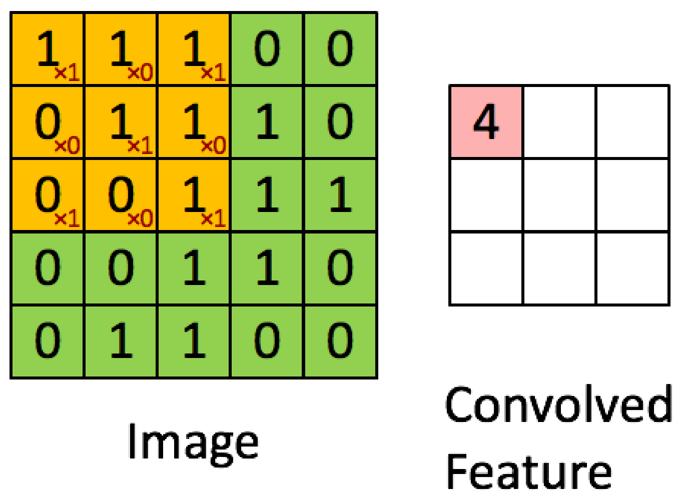


Convolution filters

- A convolution filter applies a kernel to the all image by performing the convolution operation.

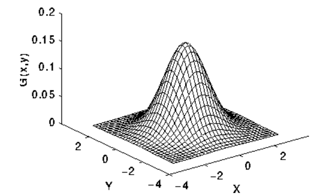
$$h * A = g(x,y) = \sum_{j=-M}^{M} \sum_{i=-M}^{M} h(i,j) \cdot A(x+i, y+j)$$

$$h(i,j) = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$



Low-pass convolution filters

- The low-pass convolution filter applies a gaussian filter to the input image.



- The Gaussian filter is approximated by a kernel with a given width.

$$h_0(x,y) = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

- Example:

Input image

$$A(x,y) = \begin{bmatrix} 255 & 255 & 0 & 0 \\ 255 & 255 & 0 & 0 \\ 255 & 255 & 0 & 0 \\ 255 & 255 & 0 & 0 \\ 255 & 255 & 0 & 0 \end{bmatrix}$$

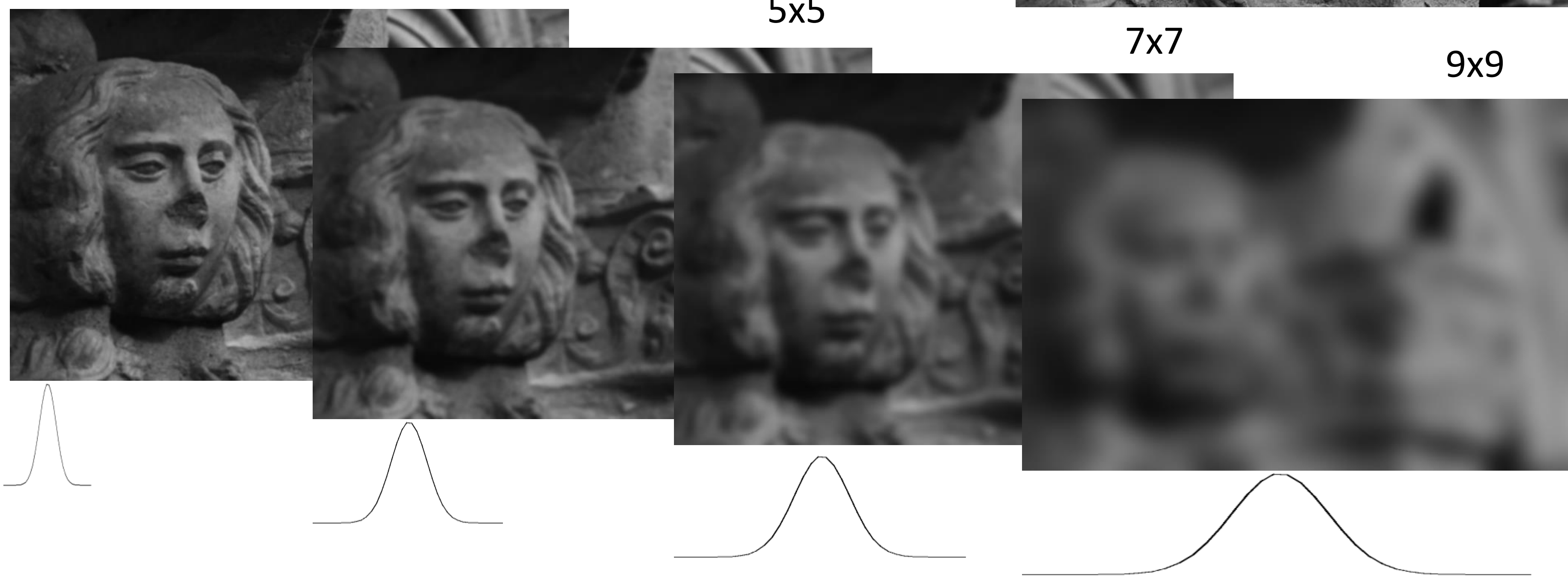
Output image

$$g(x,y) = \begin{bmatrix} 255 & 191 & 64 & 0 \\ 255 & 191 & 64 & 0 \\ 255 & 191 & 64 & 0 \\ 255 & 191 & 64 & 0 \\ 255 & 191 & 64 & 0 \end{bmatrix}$$

Example

$$h_0(x,y) = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

3x3



High-pass convolution filters

- High pass filters aim to detect the image edges

$$h_h(i,j) = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

- Different kernels are used to detect such edges at diferent scales and orientations.

$$h_v(i,j) = \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$$

Input image

$$A(x,y) = \begin{bmatrix} 255 & 255 & 0 & 0 \\ 255 & 255 & 0 & 0 \\ 255 & 255 & 0 & 0 \\ 255 & 255 & 0 & 0 \\ 255 & 255 & 0 & 0 \end{bmatrix}$$

Output image after applying horizontal filter

$$g_h(x,y) = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Output image after applying vertical filter

$$g_v(x,y) = \begin{bmatrix} 0 & 255 & 255 & 0 \\ 0 & 255 & 255 & 0 \\ 0 & 255 & 255 & 0 \\ 0 & 255 & 255 & 0 \\ 0 & 255 & 255 & 0 \end{bmatrix}$$

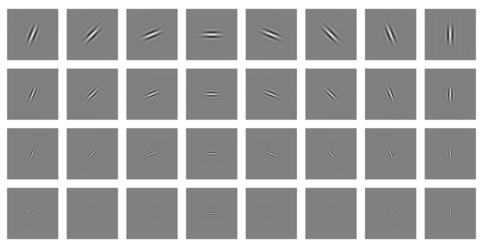
Example

$$h_v(i,j) = \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$$

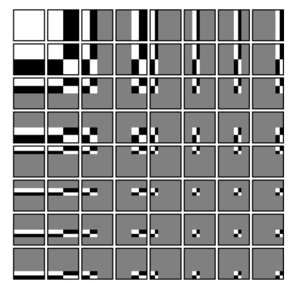
$$h_h(i,j) = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

Convolution filter kernels

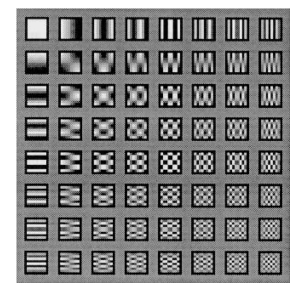
There are many diferent convolution filter kernels
that were studied over decades in the past.



Gabor



Haar



DCT

Can we learn the convolution kernels?

Yes, we can!

Convolutional Networks

- Scale up neural networks to process very large images / video sequences
  - Sparse connections
  - Parameter sharing

- Automatically generalize across spatial translations of inputs

- Applicable to any input that is laid out on a grid (1-D, 2-D, 3-D, …)

Convolutional Network Components

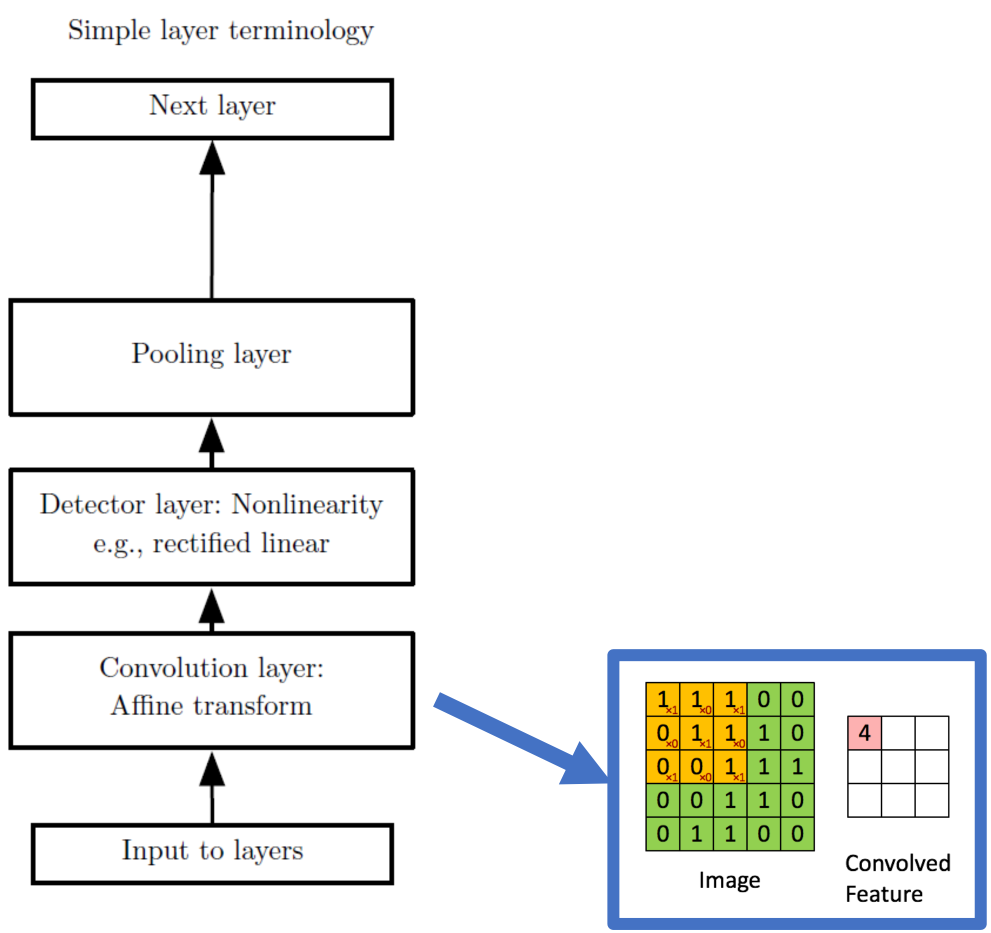
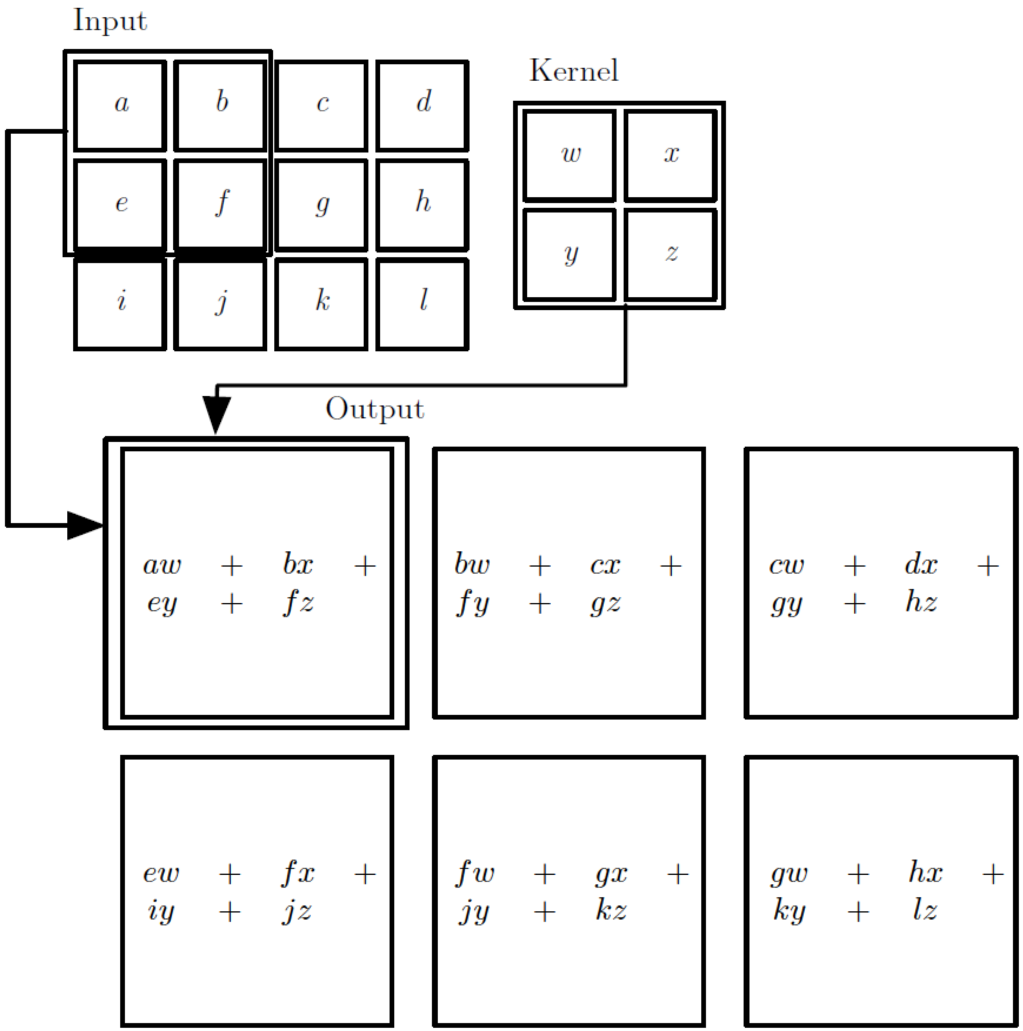


Figure 9.7

2D Convolution



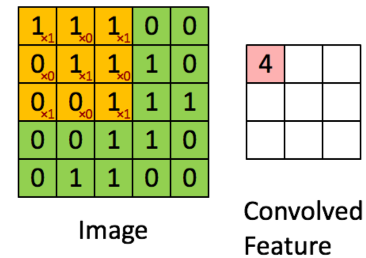


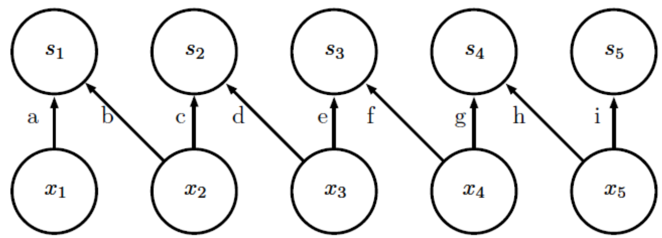
Figure 9.1

Types of connectivity

Kernel:

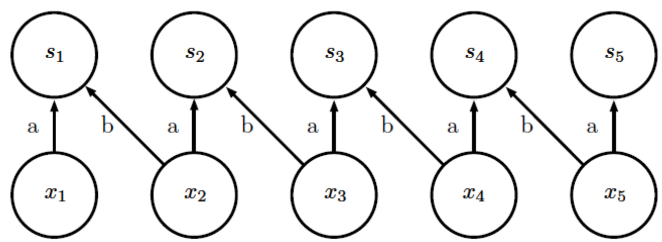
1	-1

a,b,c,d=?

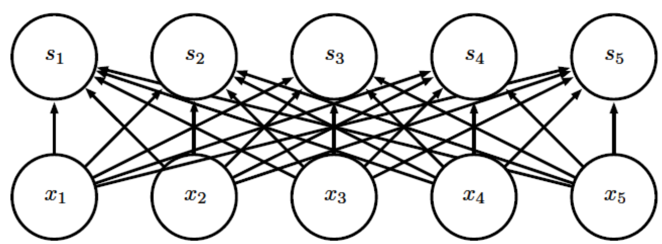


Local connection: like convolution, but no sharing

a,b=?



Convolution



Fully connected

Sparse connectivity viewed from below

Sparse connections due to small convolution kernel

Dense connections

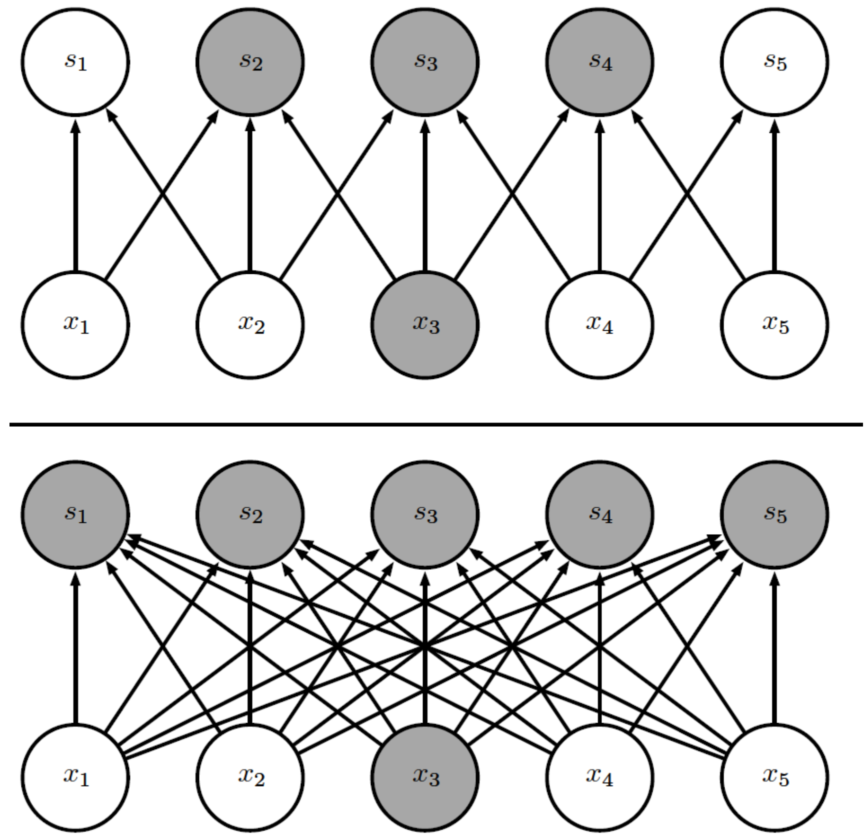


Figure 9.2

Sparse connectivity viewed from above

Sparse connections due to small convolution kernel

Dense connections

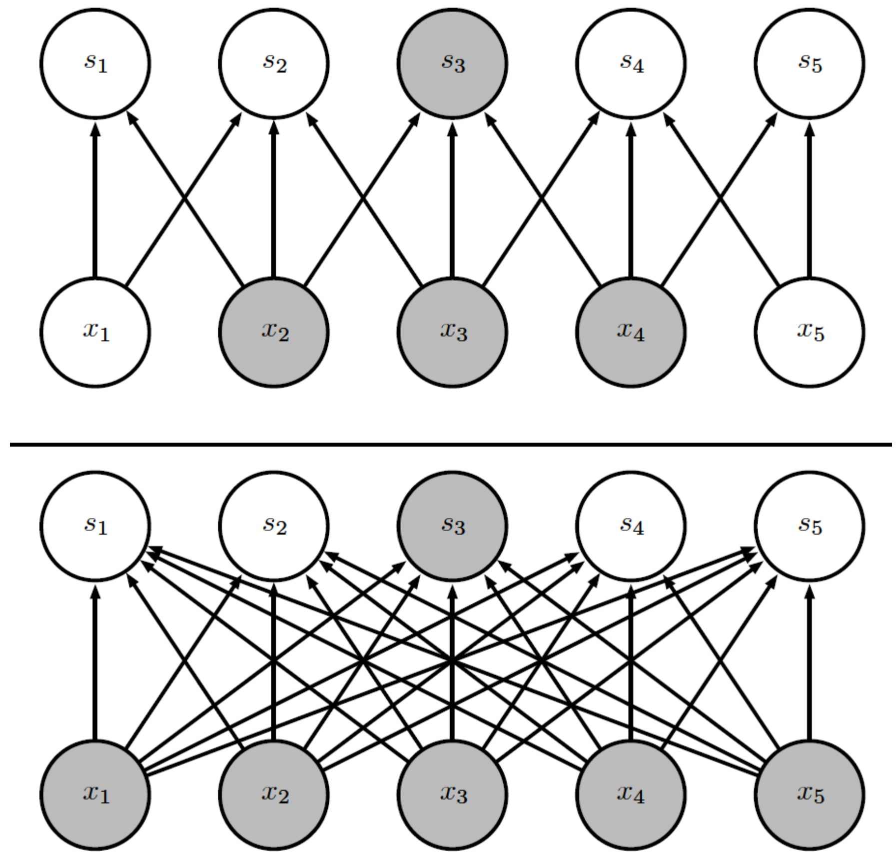


Figure 9.3

Parameter Sharing

Convolution shares the same parameters across all spatial locations

Traditional matrix multiplication does not share any parameters

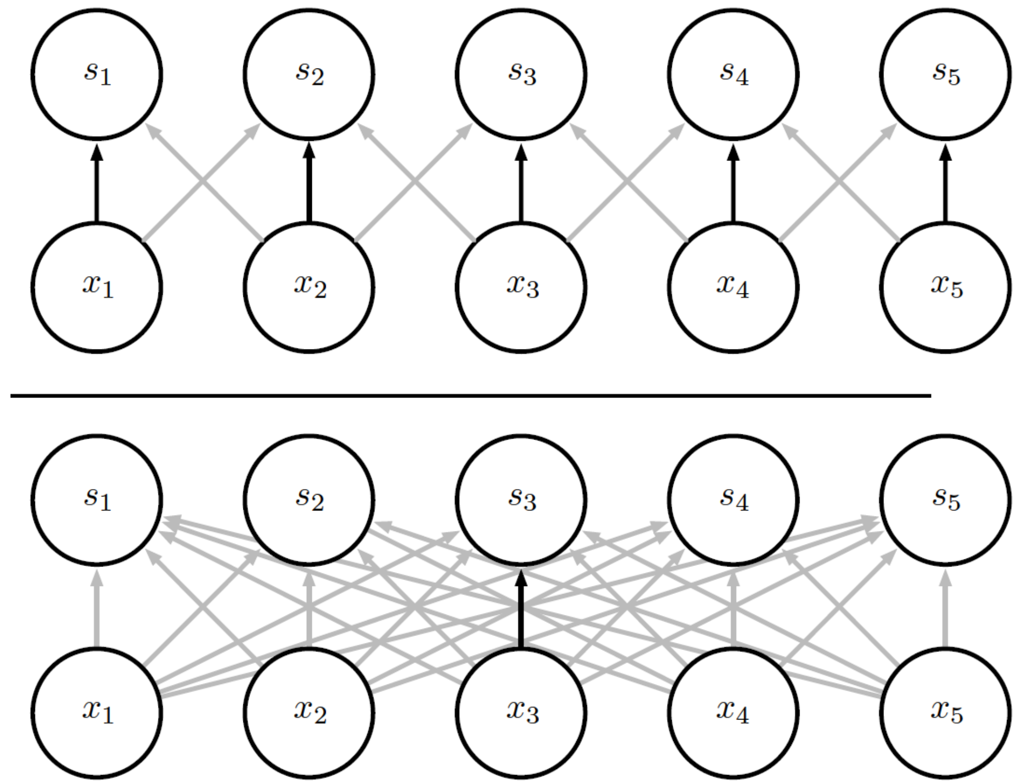


Figure 9.5

Convolution with Stride

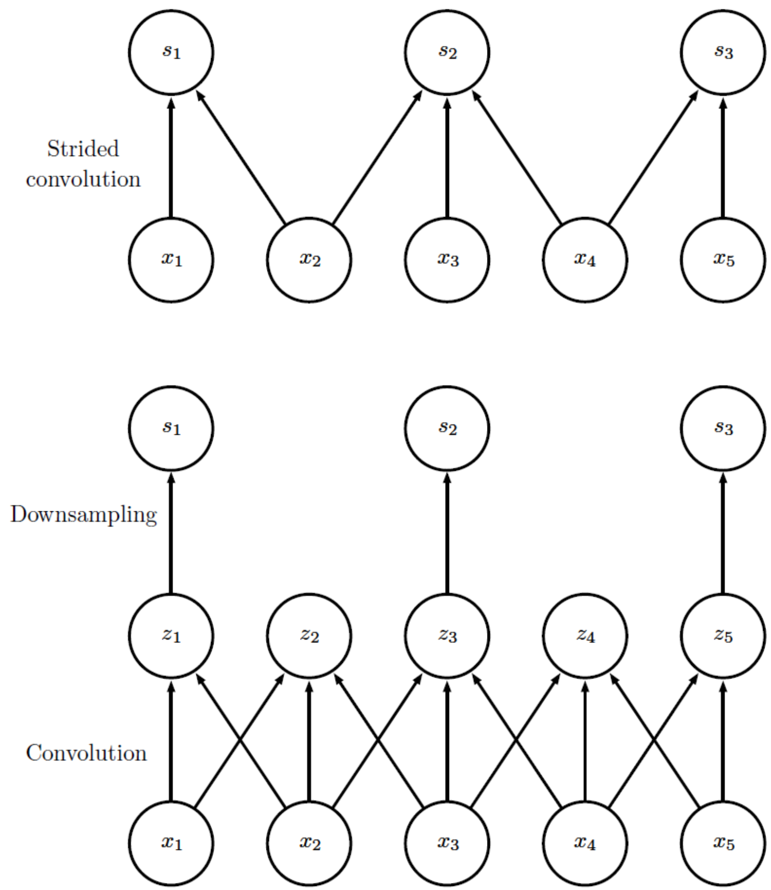
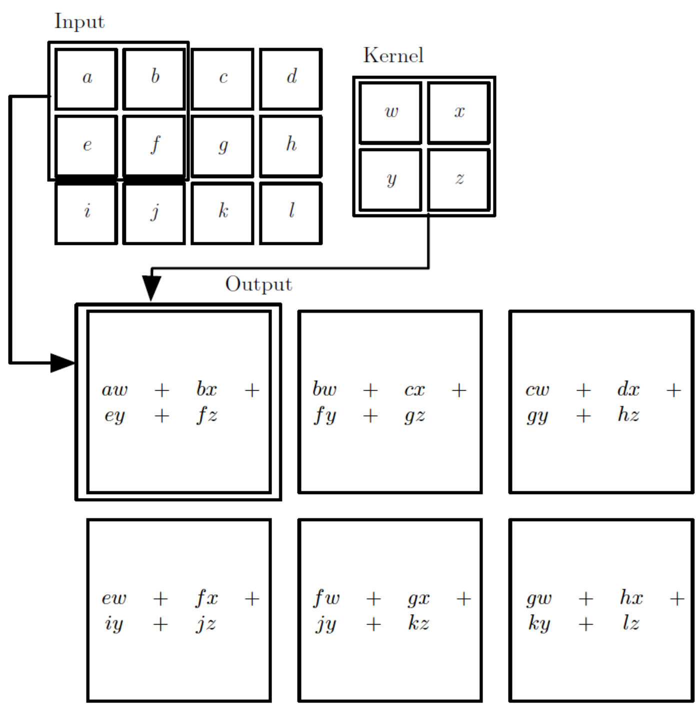


Figure 9.12

Exercise: draw the NN of this convolution



Convolutional Network Components

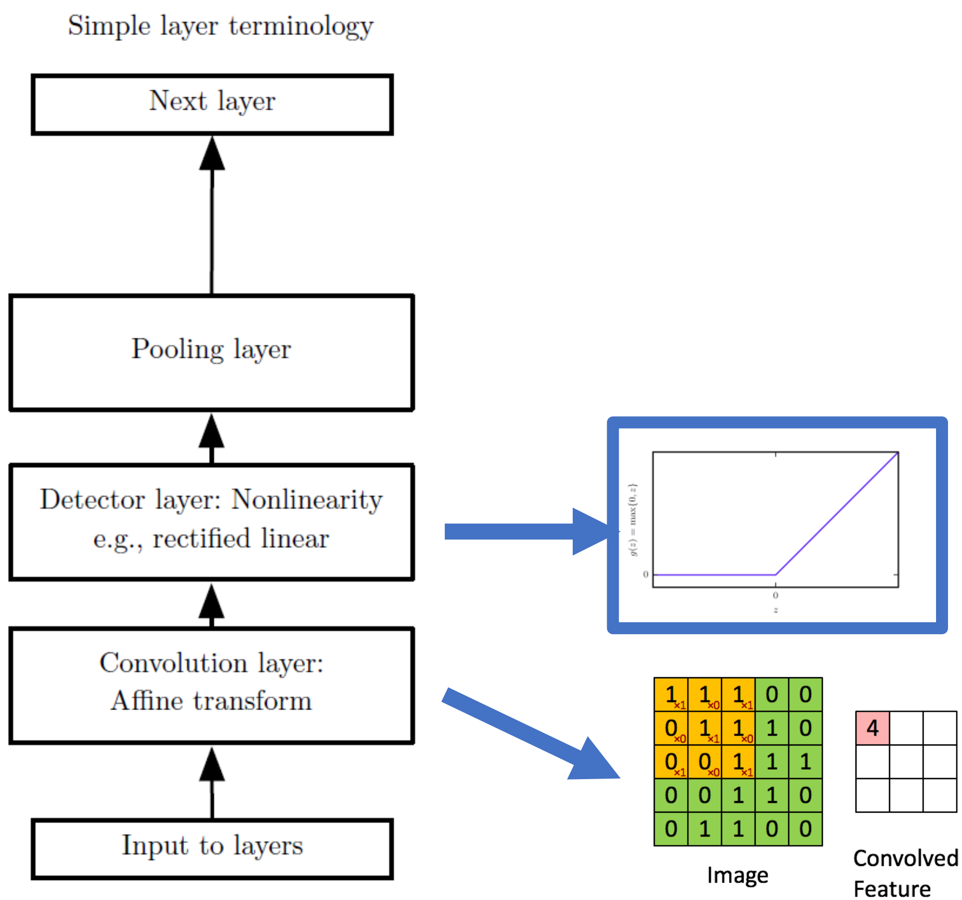


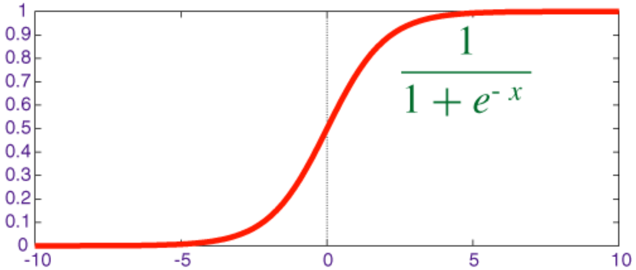
Figure 9.7

Softmax

- The softmax function was quite popular as the activation function of neural networks.

- It is diferentiable in all points
  - It is convenient from a mathematical point of view

- It can easily saturate for high values of inputs
  - Prevents passing information between layers

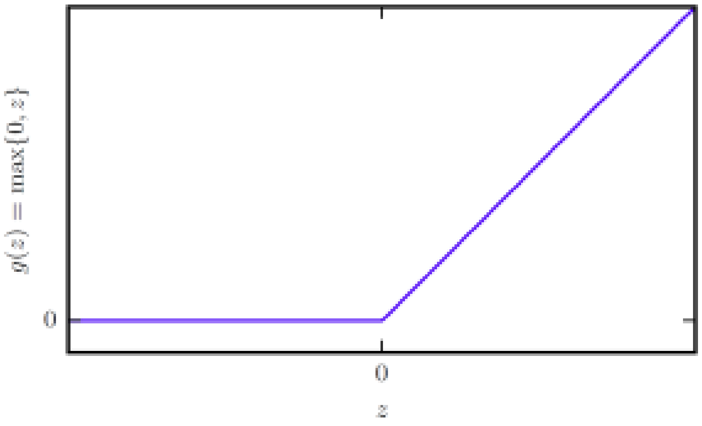


Rectified linear unit (ReLU)

- Rectified linear activation:

$$g(z) = \max\{0, z\}$$

- Brings several advantages over traditional softmax for hidden layers:
  - Never saturates, i.e. never looses information between layers
  - Gradient is constant, i.e. faster training
  - Forces sparsity, thus removes contribution from noisy units



Convolutional Network Components

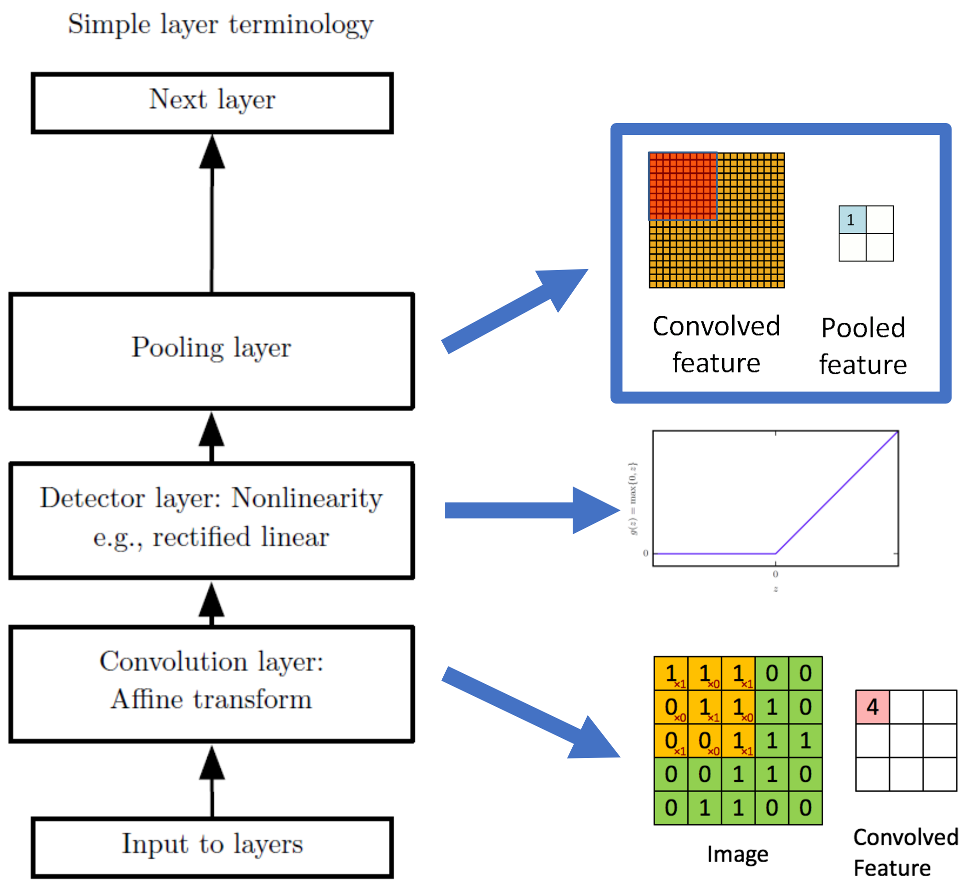
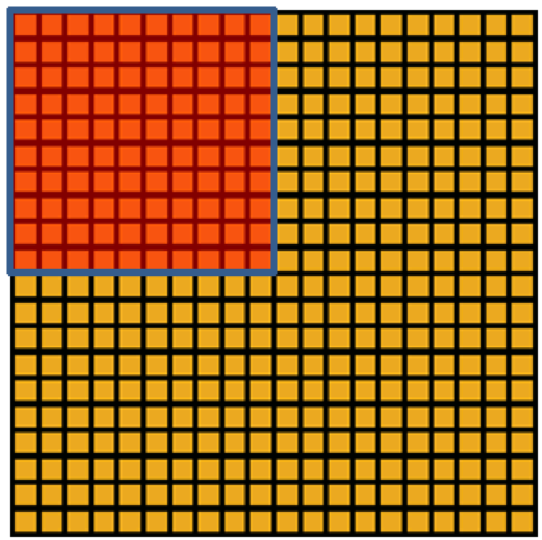
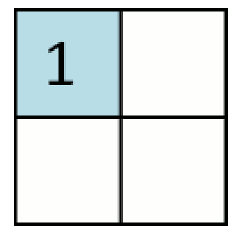


Figure 9.7

Pooling layers



Convolved feature



Pooled feature

Max Pooling and Invariance to Translation

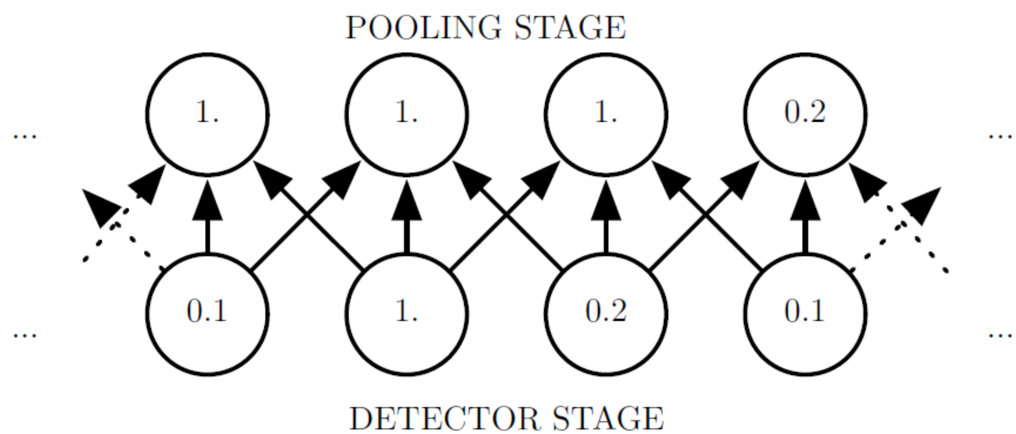


Figure 9.8

Pooling with Downsampling

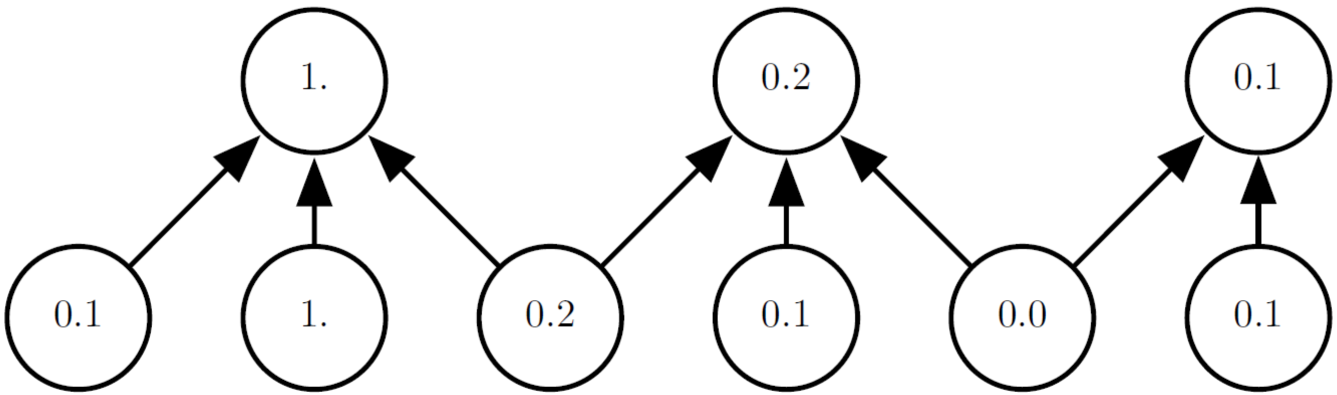


Figure 9.10

Convolutional Network Components

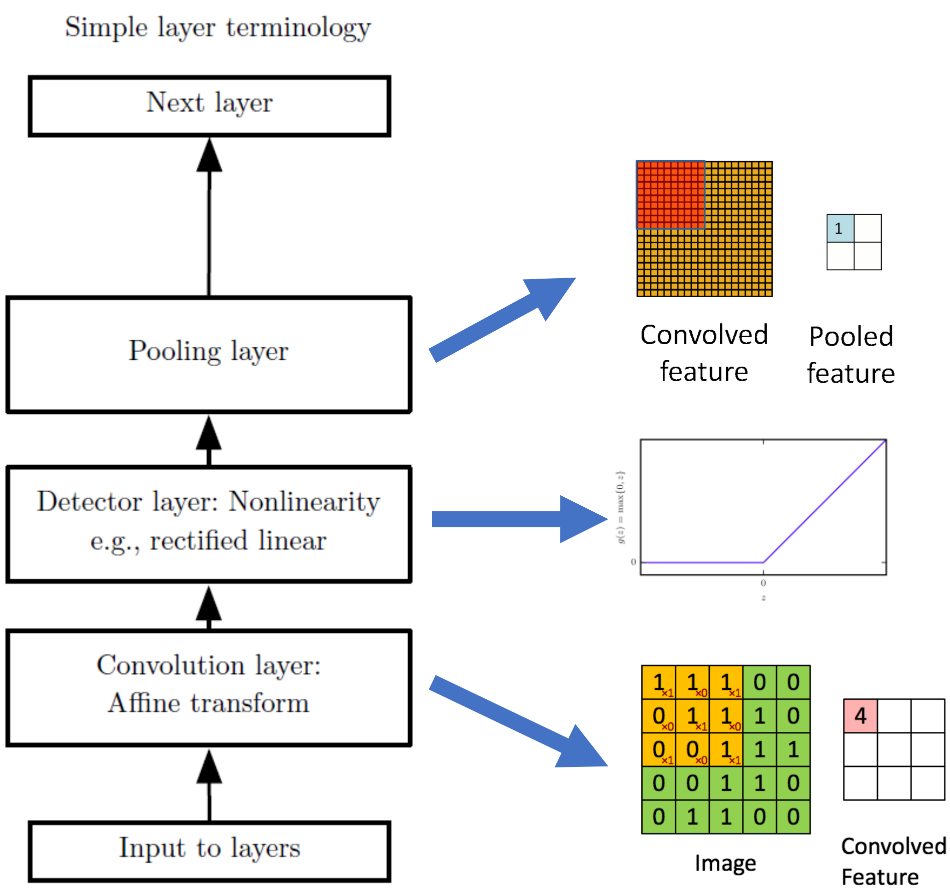
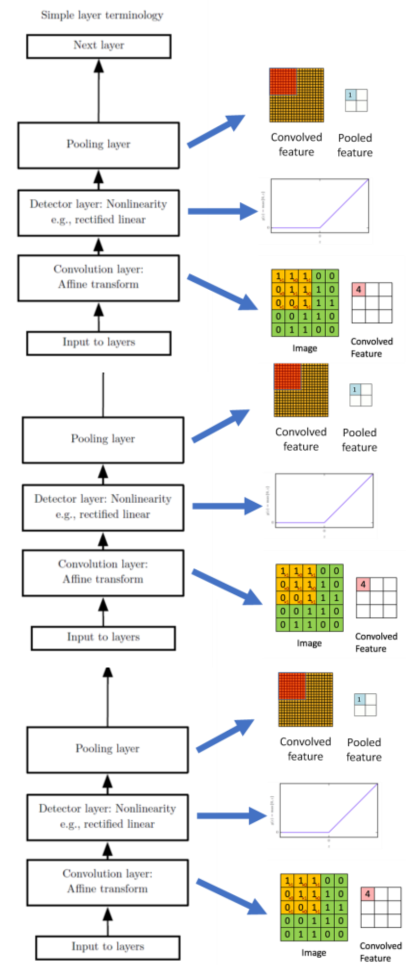


Figure 9.7

Learning deep data representations

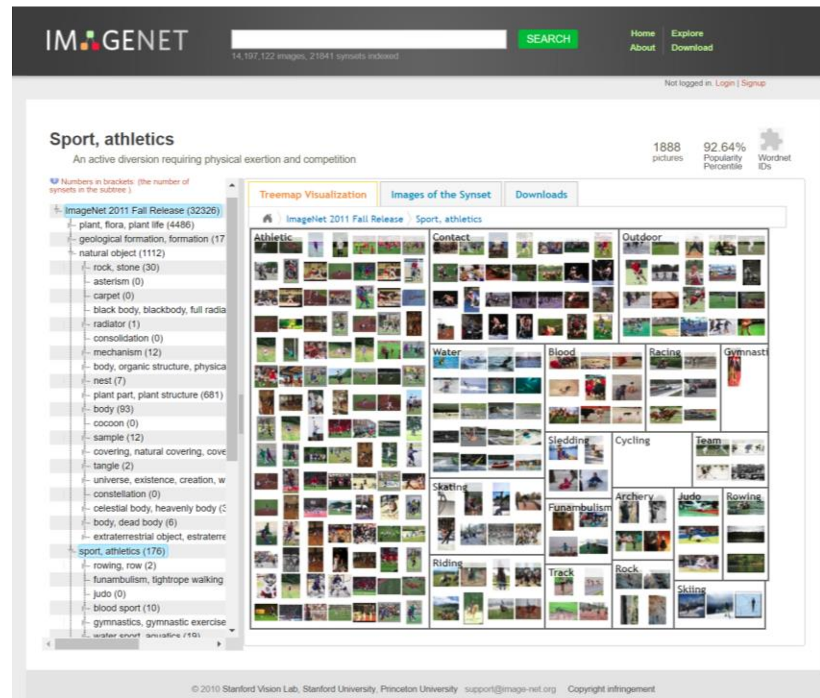
- Deep learning architectures stack multiple layers of convolutions.
- These architecures learn hierarchies of data representations
- Traditionally, training neural networks with many layers did not produce good results.
  - Some of the many hidden layers would force the model to get stuck in a local minima.



ImageNet competition

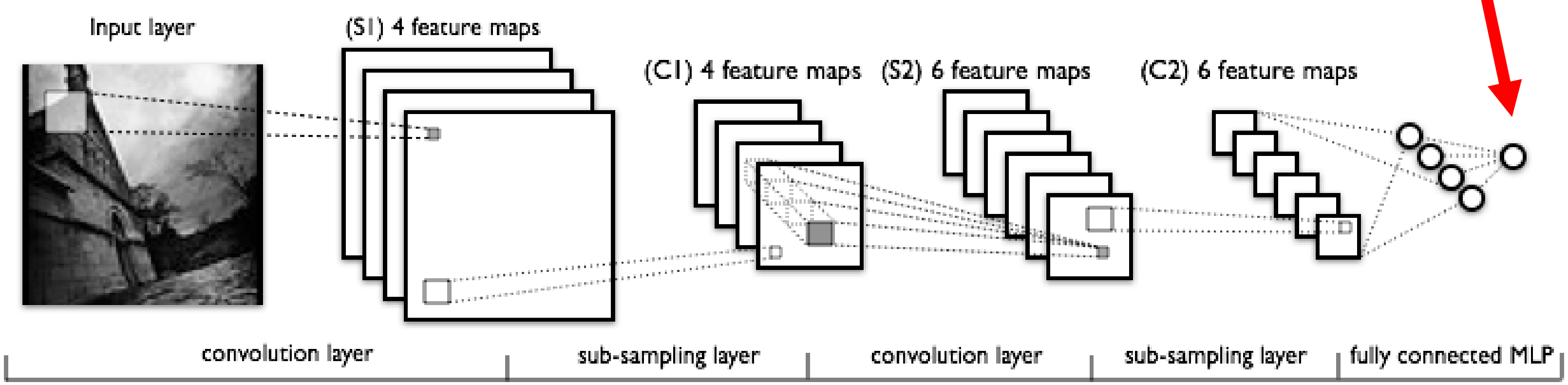
- A total of 1.43 million images annotated with 1.000 object classes
- The goal is to annotated a test sample and be as accurate as possible.
- Human error is 5.1%
- Great impact in advancing the state of the art.

http://image-net.org/explore.php



Examples of CNN architectures

Output neurons correspond to ImageNet concepts



Gabor-like Learned Kernels

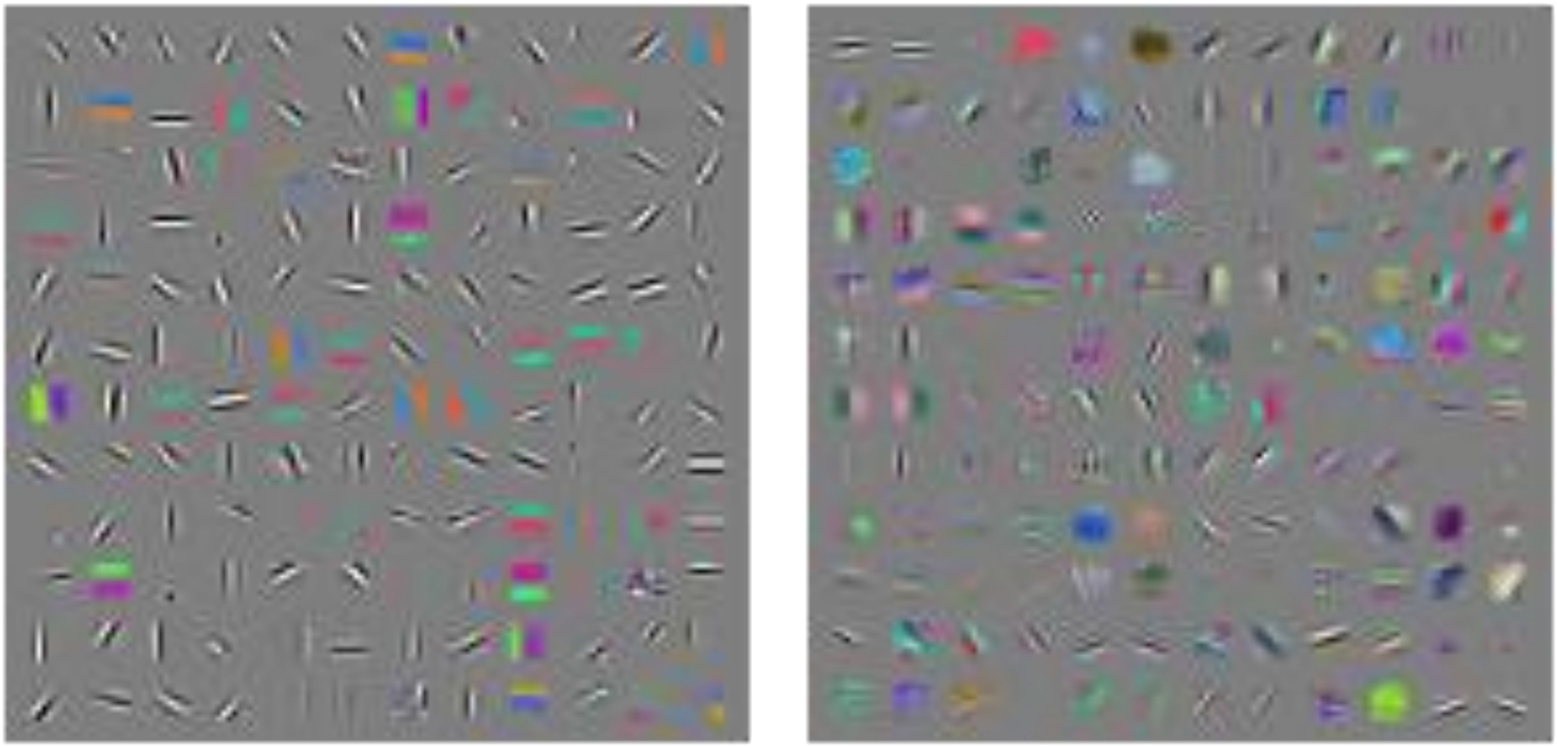


Figure 9.19

Gabor-like Learned Kernels

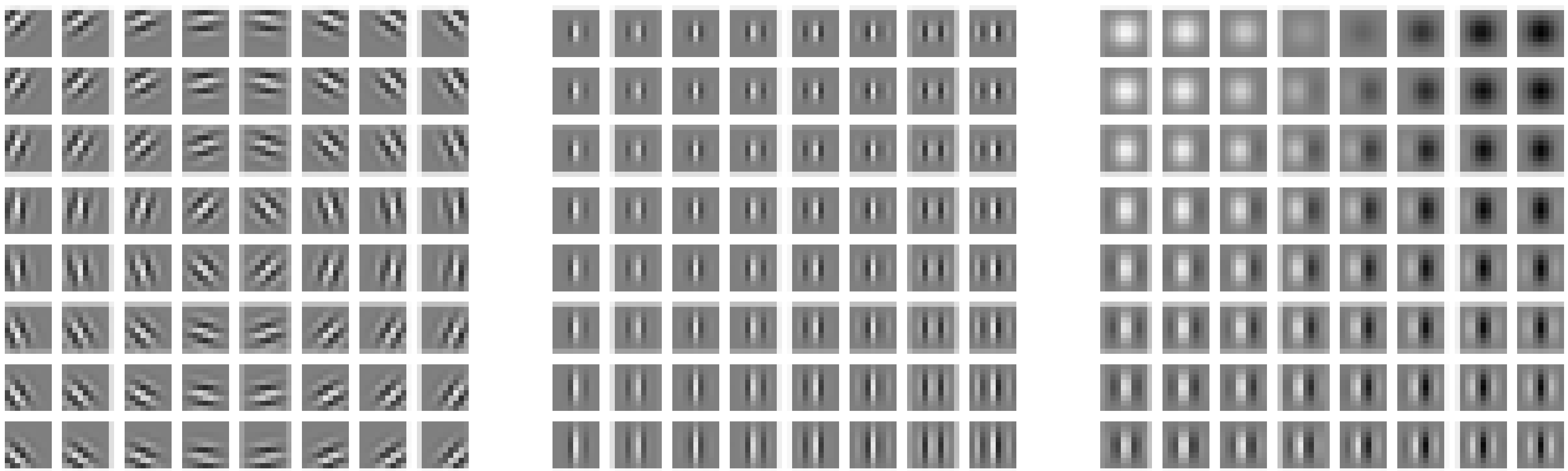
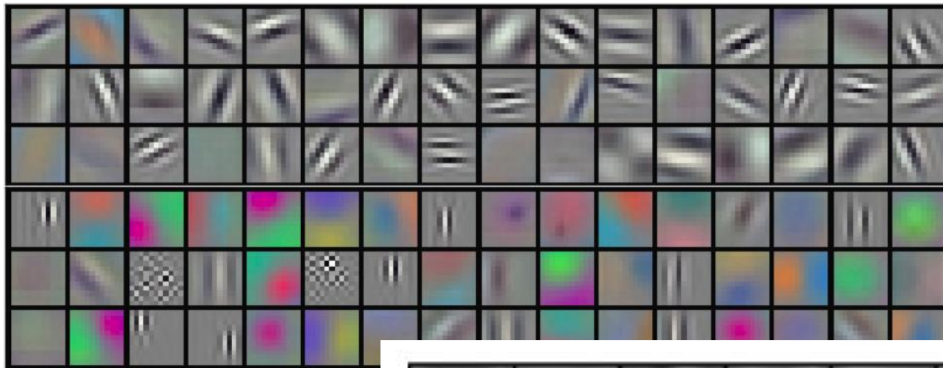


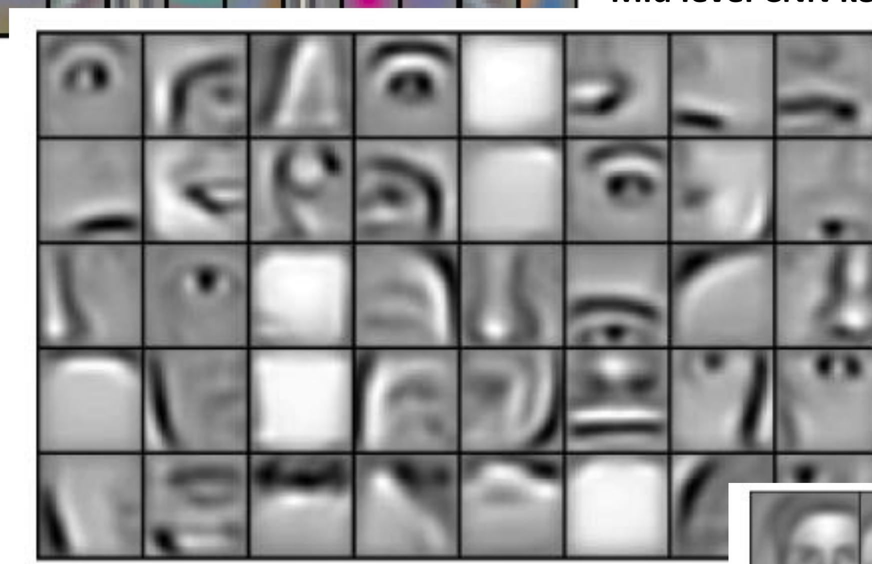
Figure 9.18

Example for face detection

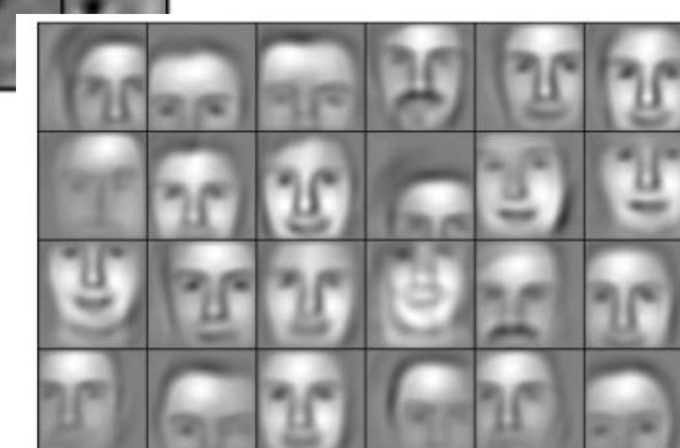
Low level CNN kernels



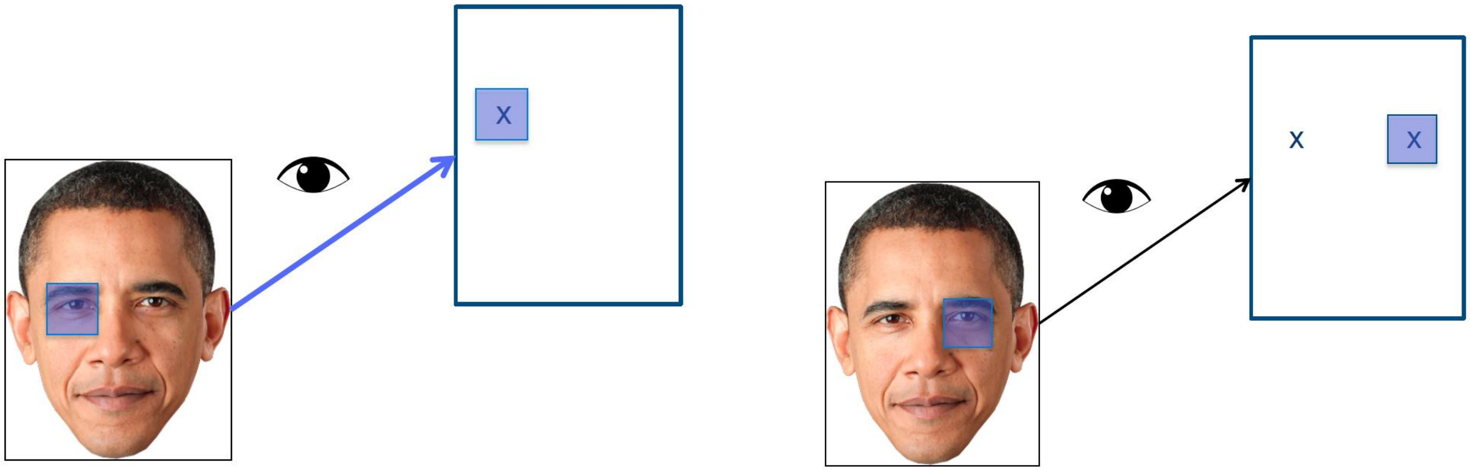
Mid level CNN kernels



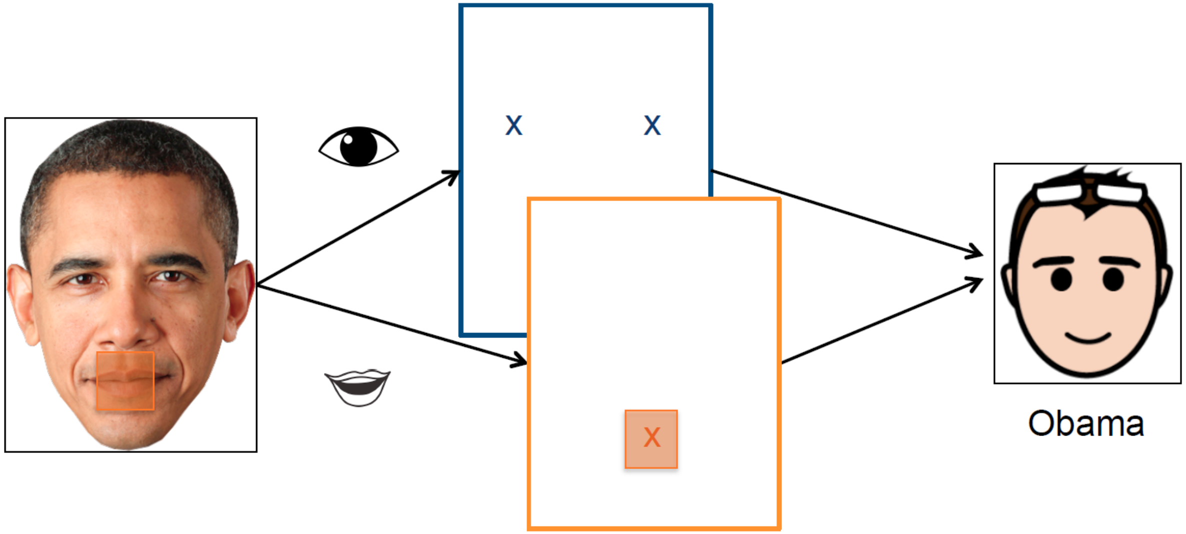
High level CNN kernels



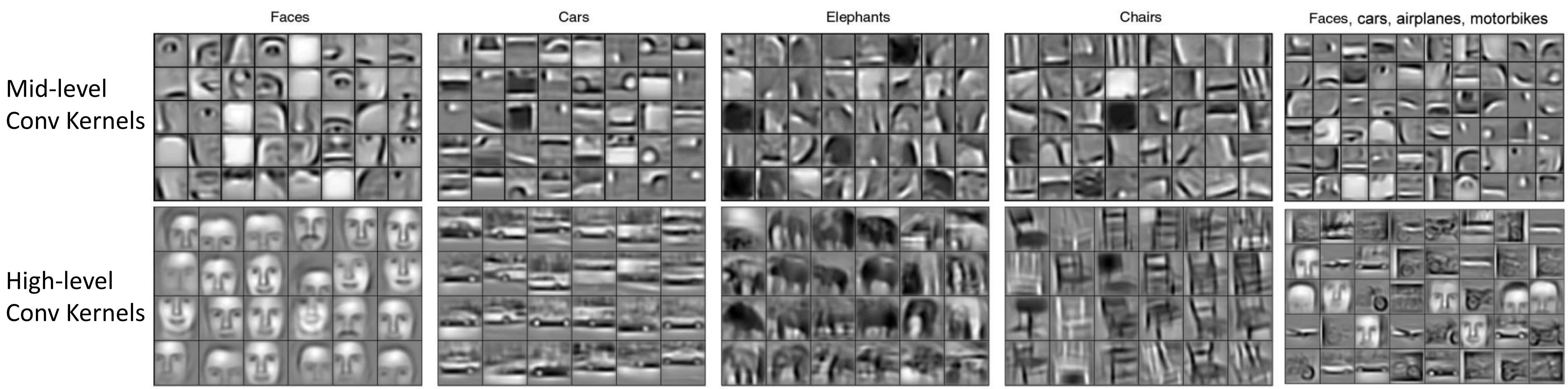
Features are translation invariant



High-level features are composed of low-level features

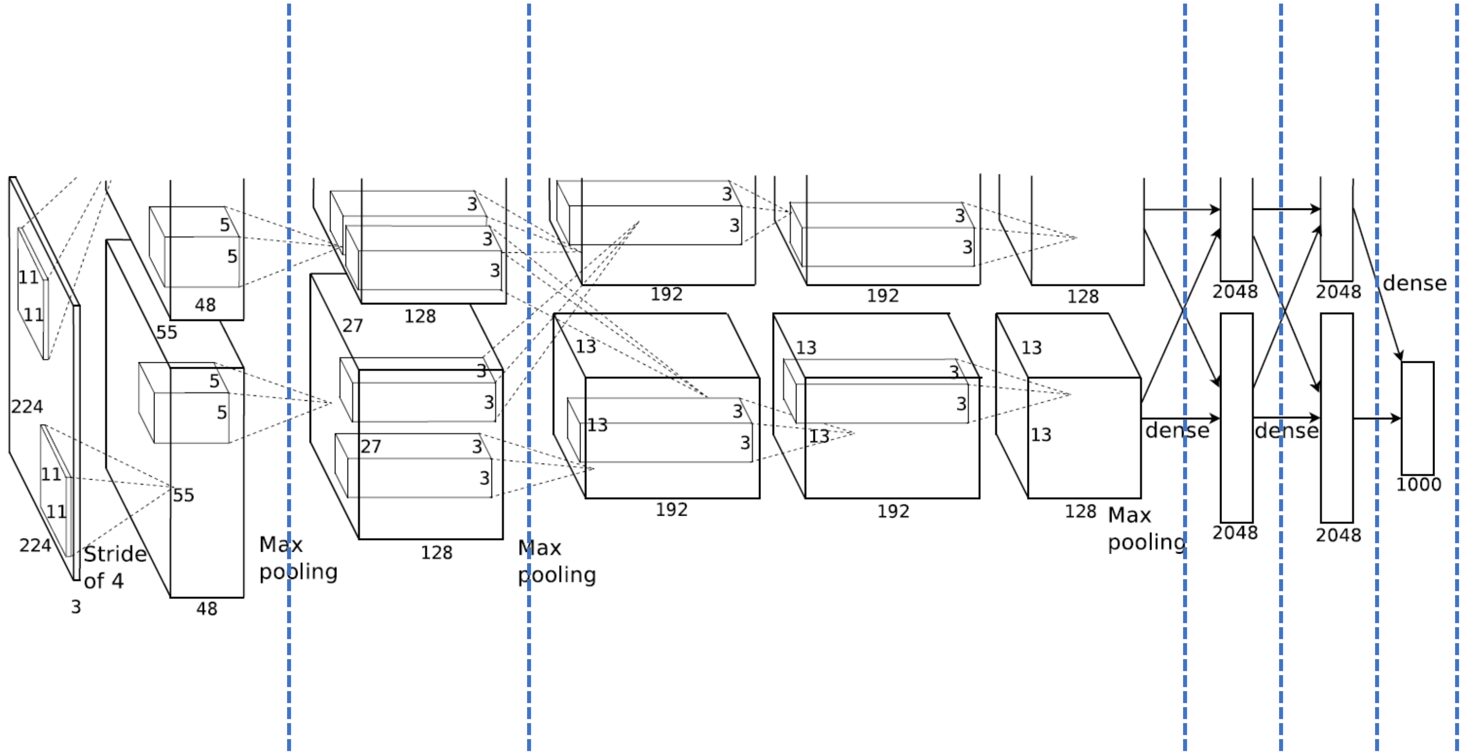


Example for multiple classes



Lee, H., Grosse, R., Ranganath, R., & Ng, A. Y. (2011). Unsupervised learning of hierarchical representations with convolutional deep belief networks. *Communications of the ACM*, *54*(10), 95-103.

AlexNet



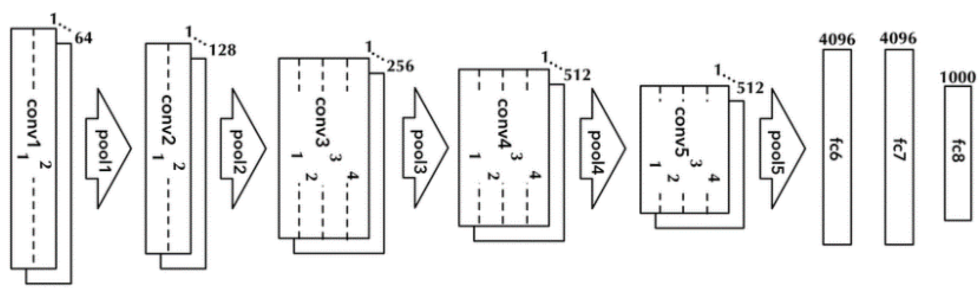
Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (pp. 1097-1105).

VGG 16 architecture

Output neurons correspond to ImageNet concepts

Output neurons correspond to ImageNet concepts

VGG16

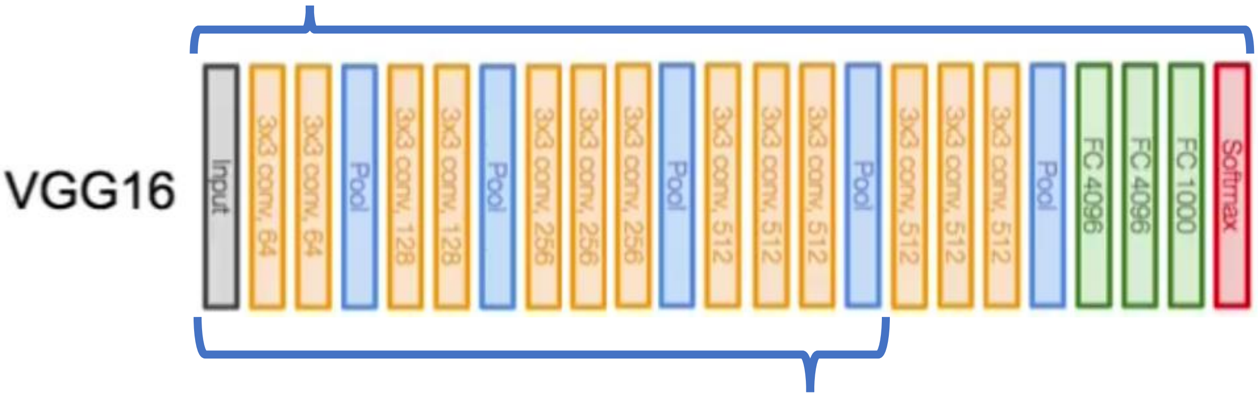


Example

```python
from tensorflow.keras.applications.vgg16 import VGG16
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.vgg16 import preprocess_input
from tensorflow.keras.models import Model
import numpy as np

base_model = VGG16(weights='imagenet')
base_model.summary()
```

Output neurons correspond to ImageNet concepts

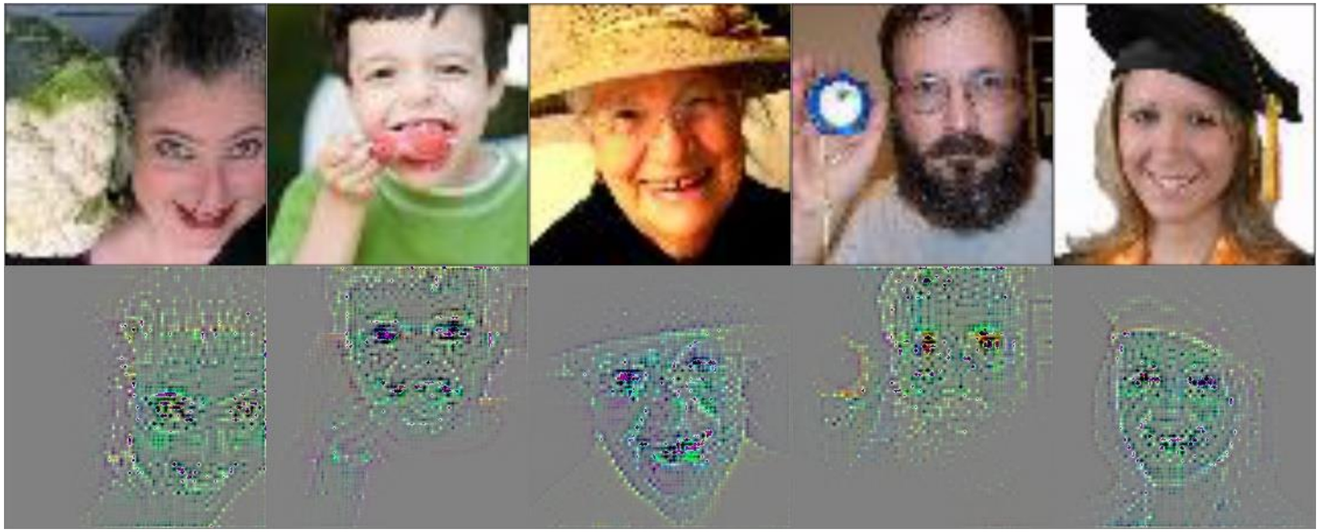



```python
model = Model(inputs=base_model.input, outputs=base_model.get_layer('block4_pool').output)
model.summary()
```

Visualizing VGG16

https://github.com/yosuah/vgg_deconv_vis

High level neuron from the fifth convolution block



Other major architectures

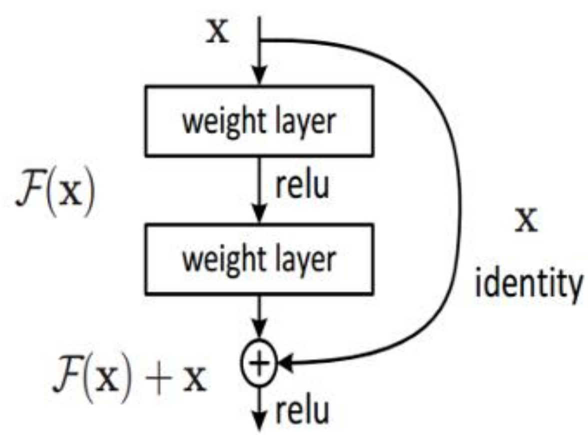
- Spatial Transducer Net: input size scales with output size, all layers are convolutional

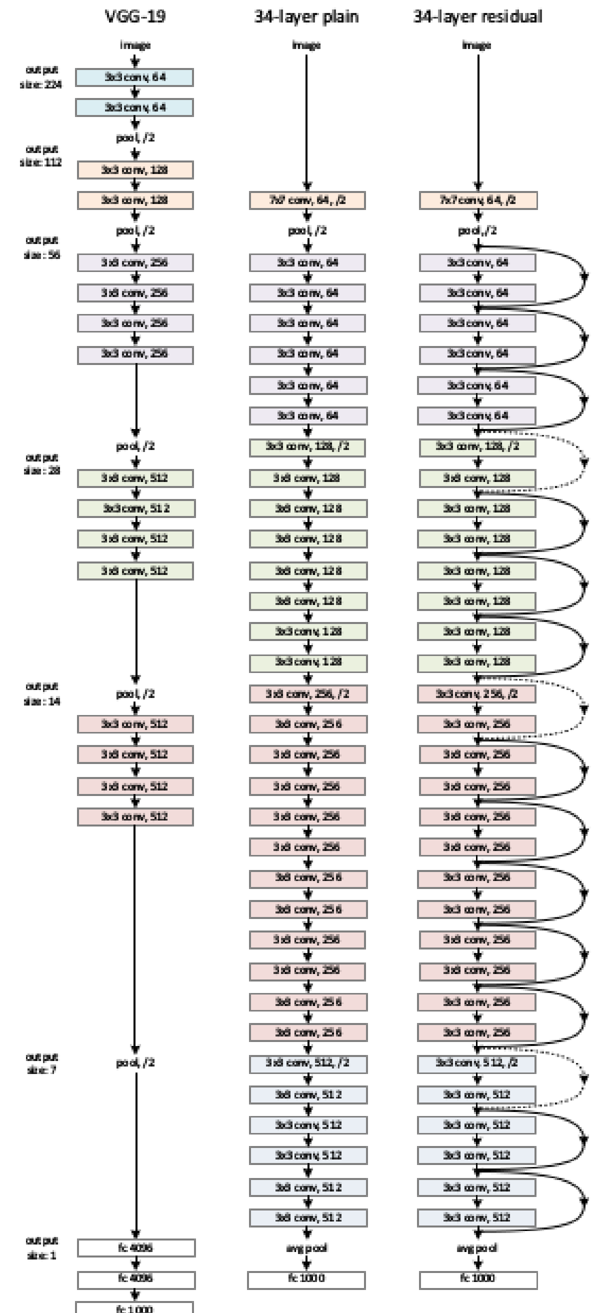
- All Convolutional Net: no pooling layers, just use strided convolution to shrink representation size

- Inception: complicated architecture designed to achieve high accuracy with low computational cost

- ResNet: blocks of layers with same spatial size, with each layer's output added to the same buffer that is repeatedly updated. Very many updates = very deep net, but without vanishing gradient.

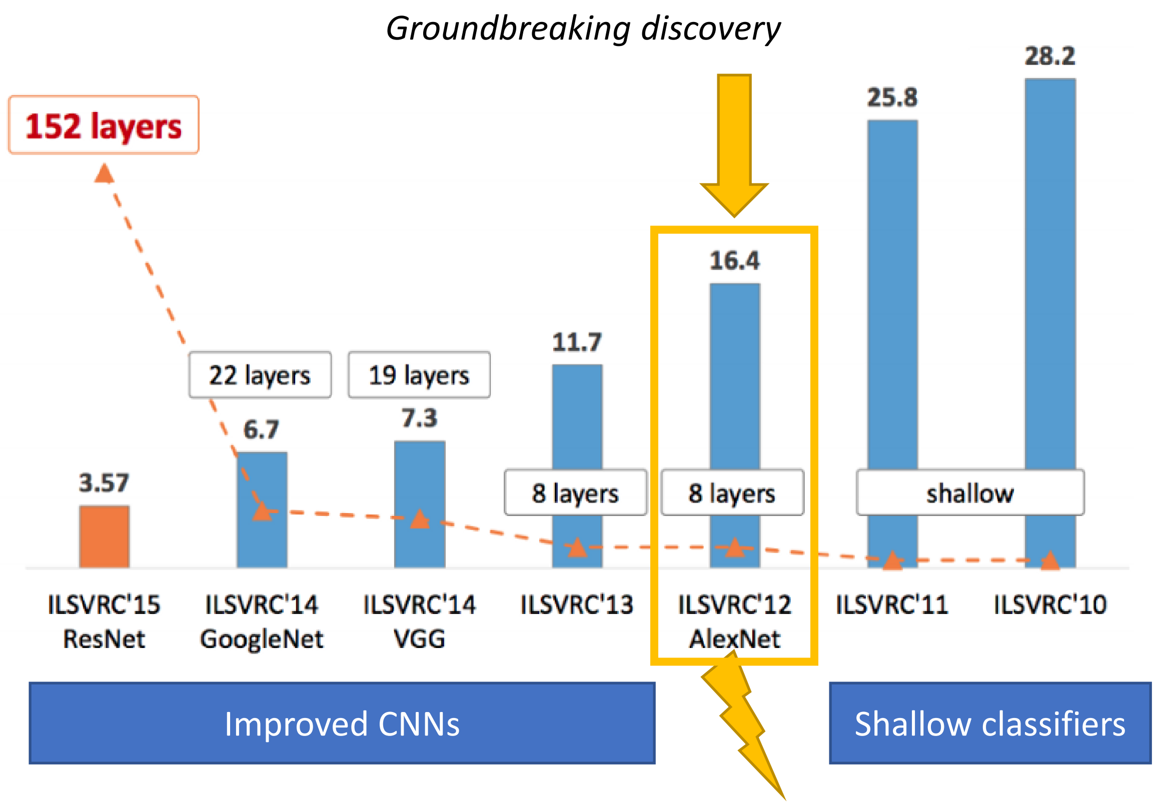
Residual Networks





He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778). https://arxiv.org/pdf/1512.03385.pdf

ImageNet Challenge top-5 error



Summary and readings

- Learning data representations
  - Convolution operation
  - ReLU activation
  - Pooling
  - Residual Networks

- Understand visual data representations:
  - low-level layers, mid-level layers and high-level layers

- Bibliography:
  - http://d2l.ai/chapter_convolutional-neural-networks/index.html
  - http://d2l.ai/chapter_convolutional-modern/index.html