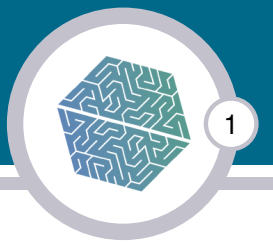
Deep Learning

Aurelien Lucchi

May 25, 2016

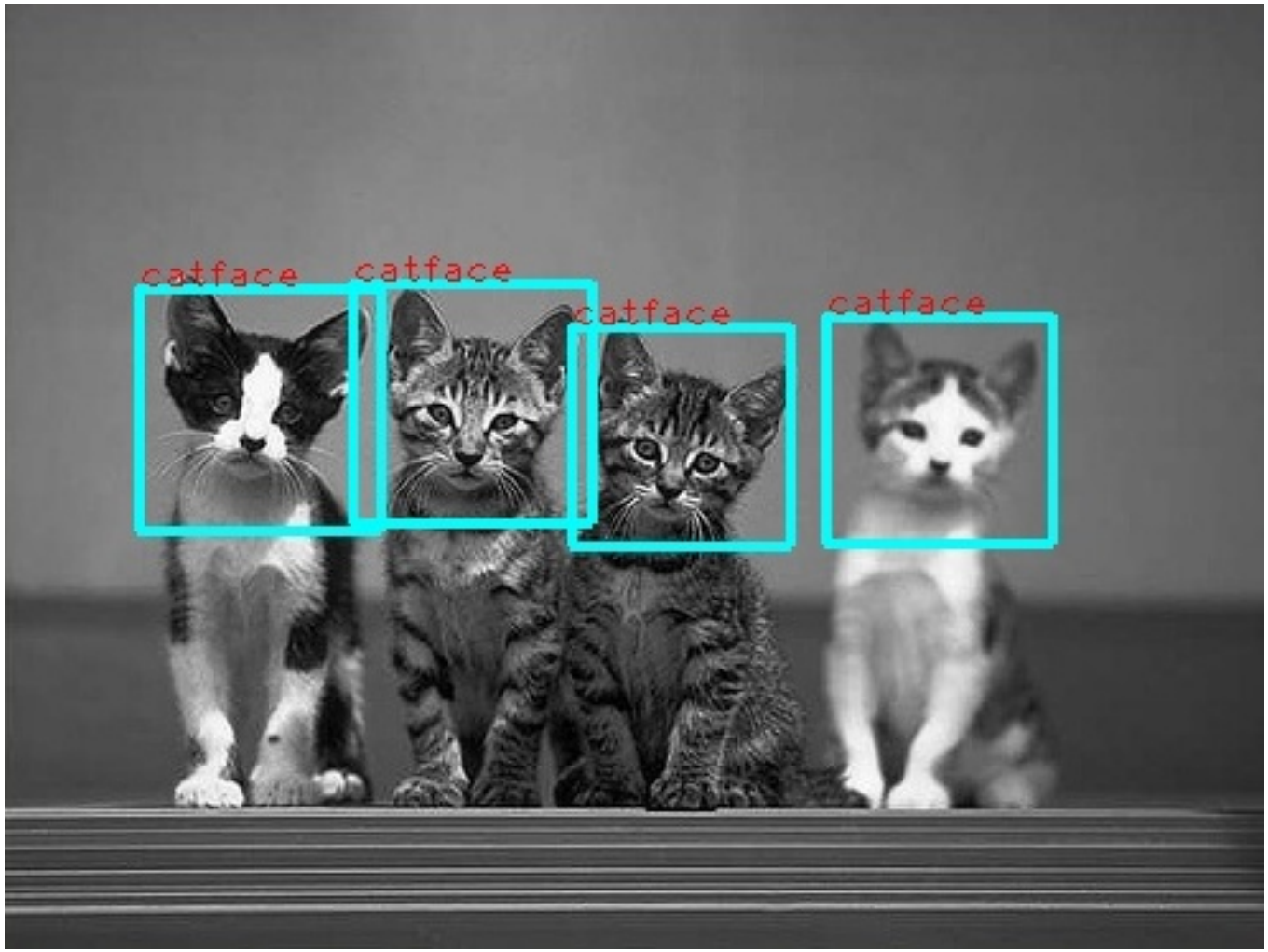
Outline



- Recall machine learning
- Support Vector Machine (SVM) with hand-designed image features
- Convolutional Neural Networks

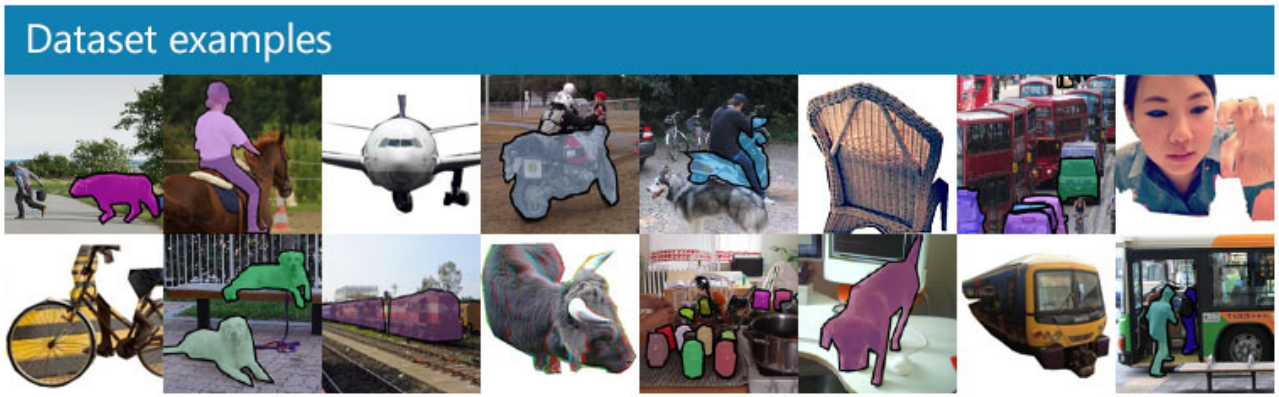
Machine learning

- Object detection



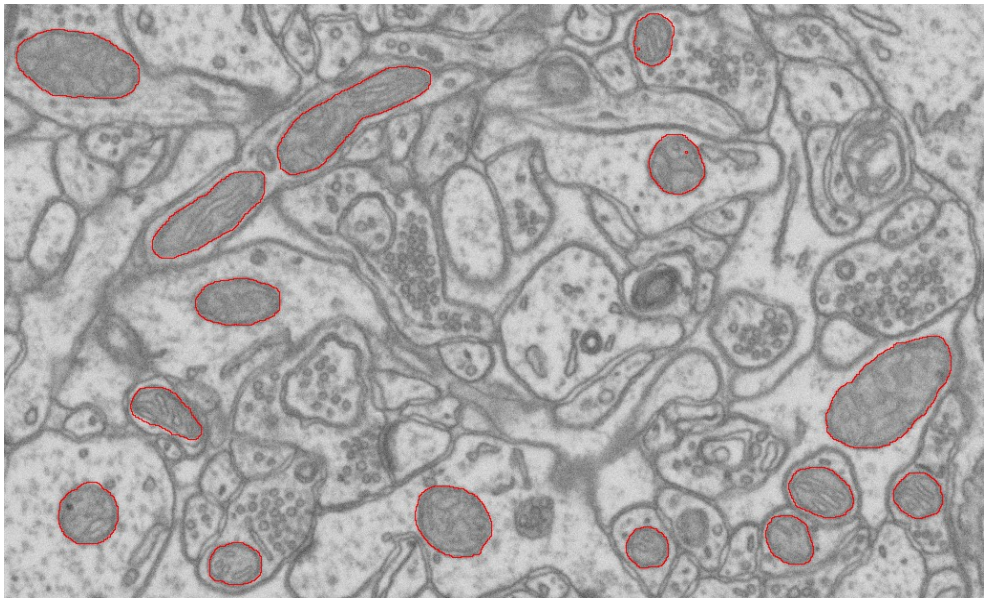
Machine learning

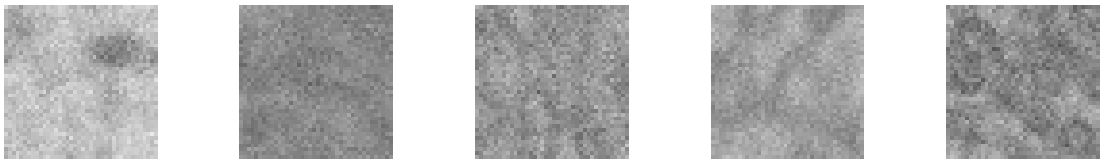
- Image segmentation



Machine learning

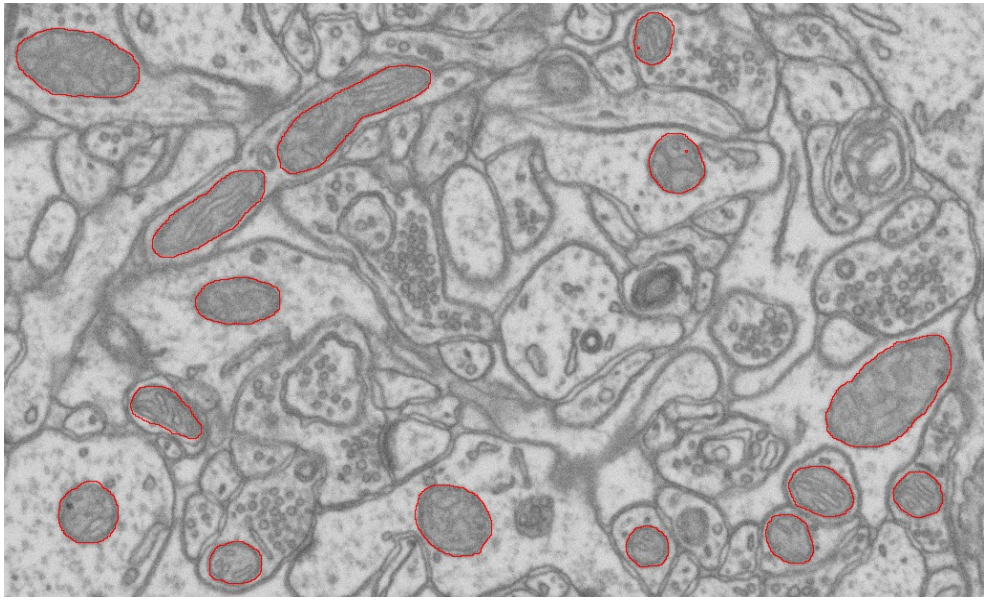
- Can you find the mitochondria in this image?

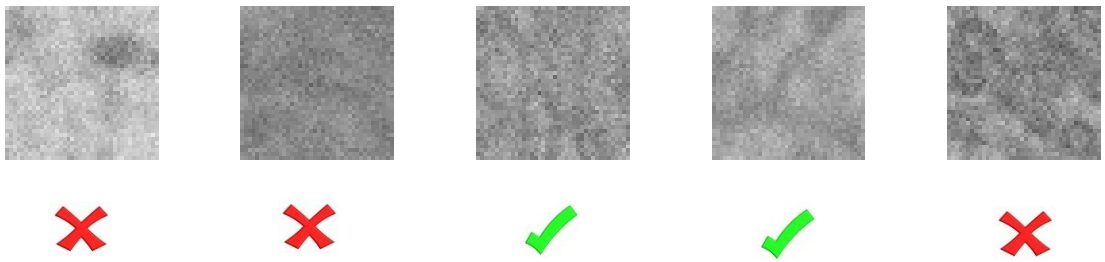




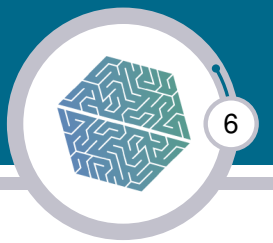
Machine learning

- Can you find the mitochondria in this image?





Machine learning - more formally



- Given a dataset $\{\boldsymbol{x}, \boldsymbol{y}\} = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)\}$
  - $\boldsymbol{x}_i \in \mathbb{R}^d$ = feature vector
  - $y_i \in \{0, 1\}$ = label (binary classification)
- Consider $\boldsymbol{w} \in \mathbb{R}^d$ to be a vector of parameters
- Seek a function $f(\boldsymbol{x}; \boldsymbol{w}) : \mathbb{R}^d \to \mathbb{R}$ to model the data

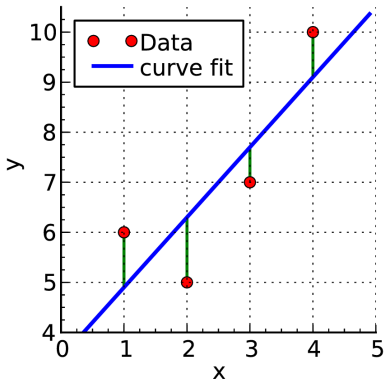
Machine learning - more formally

Open questions:

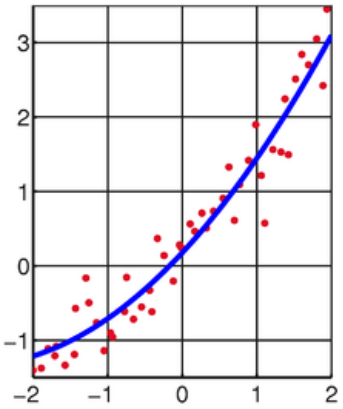
- What features $\boldsymbol{x}$ should I extract?
- What function $f(\boldsymbol{x}; \boldsymbol{w})$ should we pick?

Machine learning - more formally

- Function $f(\boldsymbol{x}; \boldsymbol{w})$ found by minimizing a loss function $L(\boldsymbol{x}; \boldsymbol{w})$
  - Simplest function: $f(\boldsymbol{x}; \boldsymbol{w}) = \boldsymbol{w}^\top \boldsymbol{x}$
  - Linear least-squares, $L(\boldsymbol{x}; \boldsymbol{w}) = \frac{1}{n} \sum_{i=1}^{n} (y_i - \boldsymbol{w}^\top \boldsymbol{x}_i)^2$



Linear fit

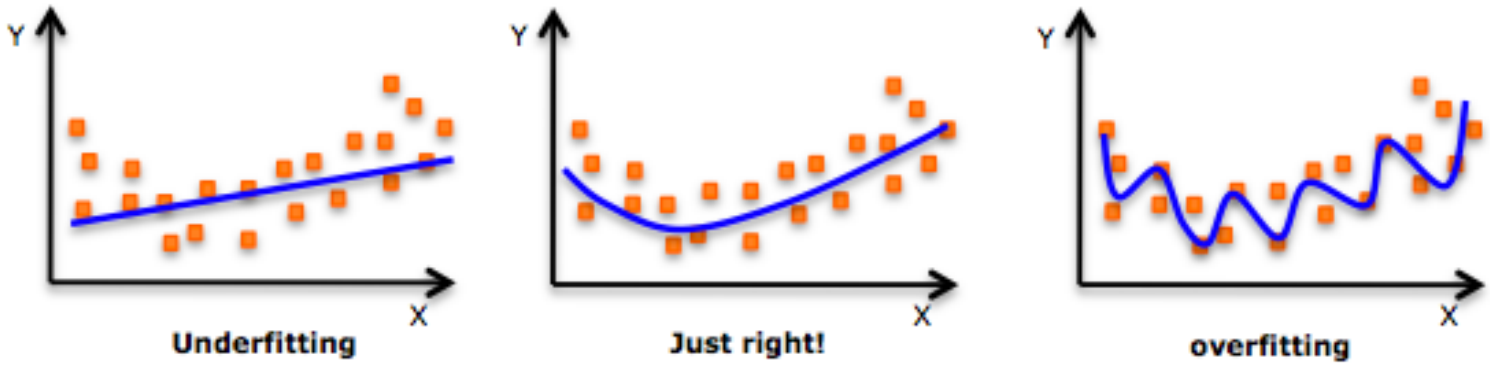


Quadratic fit

Machine learning - more formally

- Consider regularizing in order not to overfit, e.g. ℓ_2 norm

$$\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} L(\boldsymbol{x}; \boldsymbol{w}) + \frac{1}{2}\|\boldsymbol{w}\|^2$$



"Old" pipeline

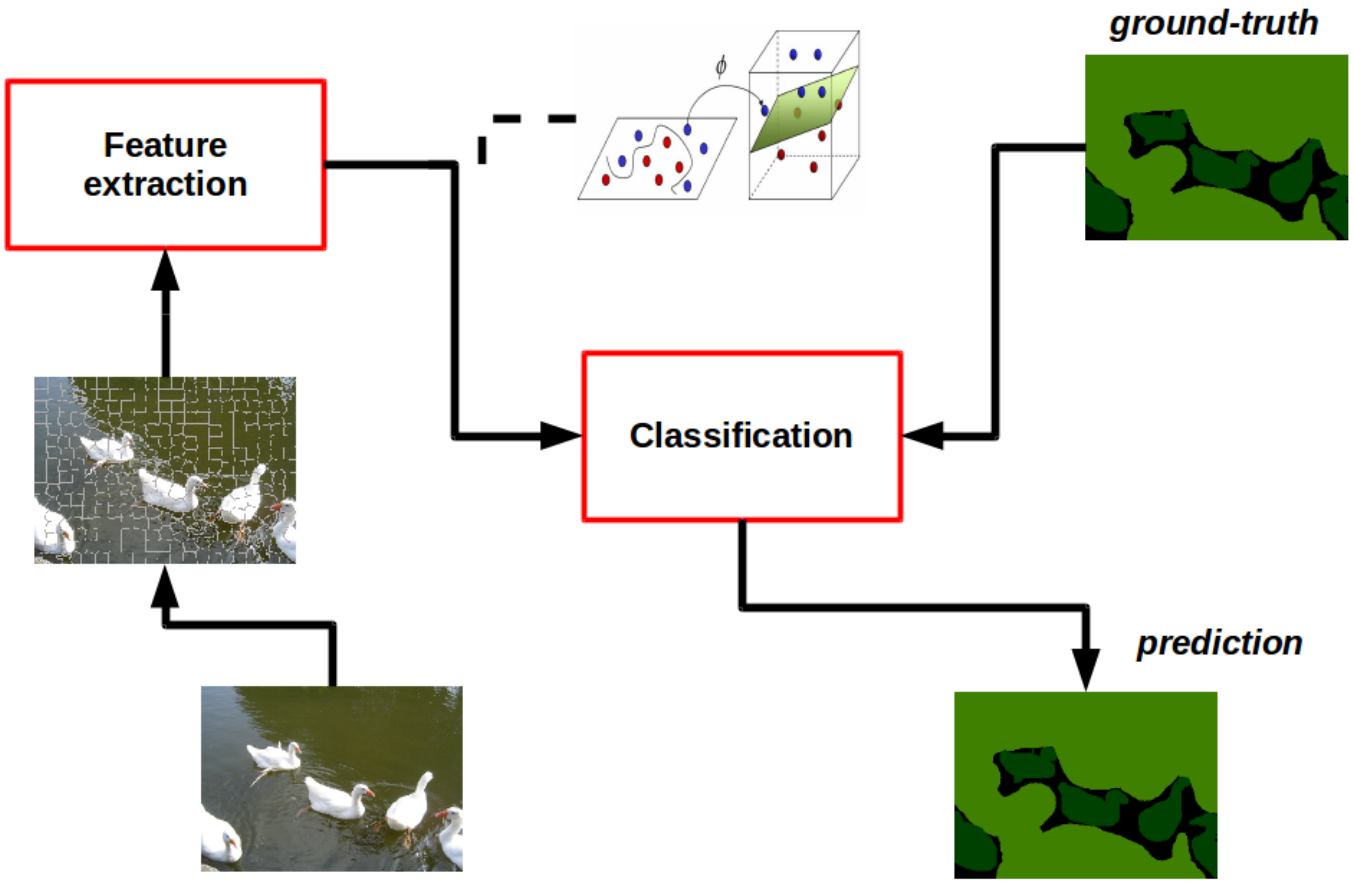


Image features

- **Color histogram**

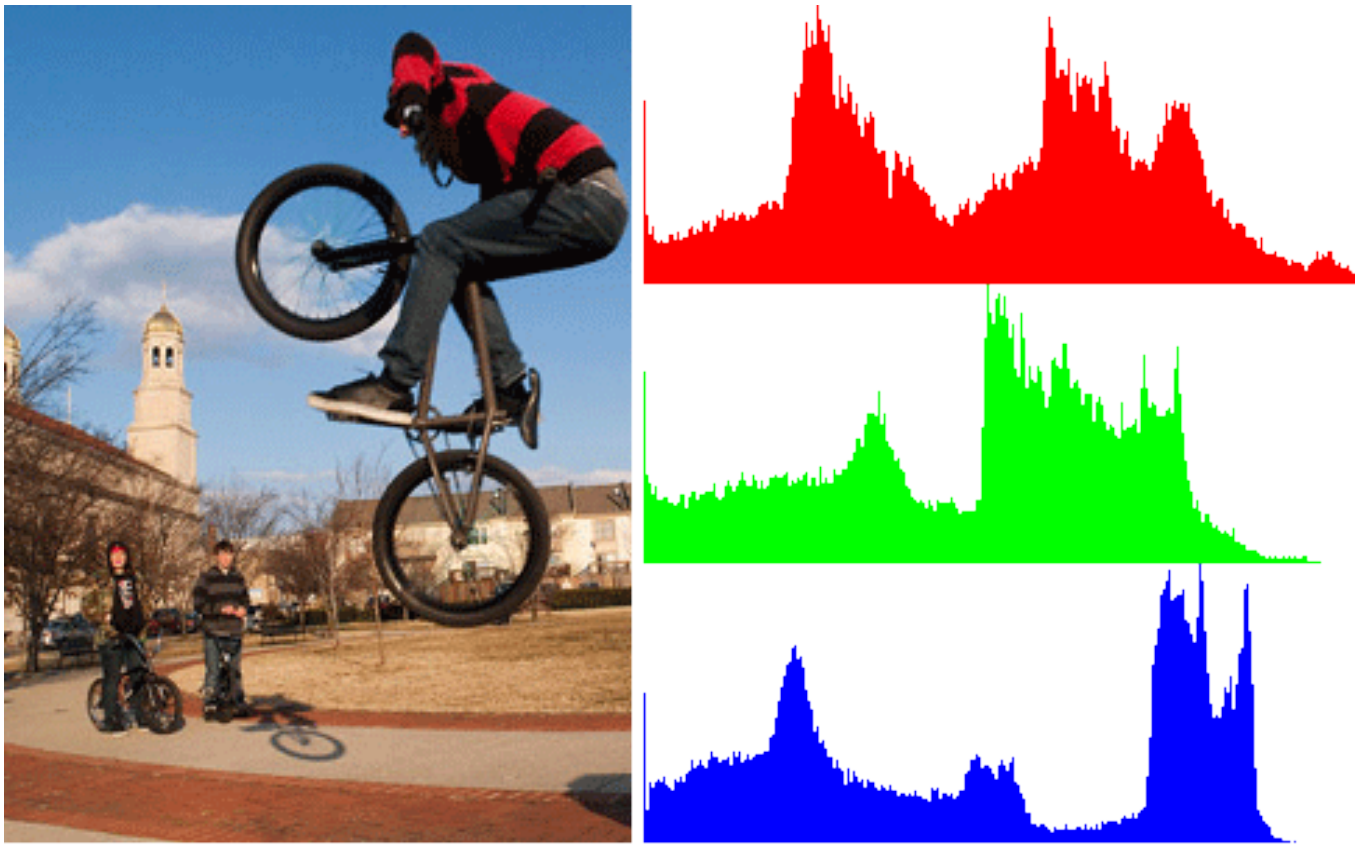
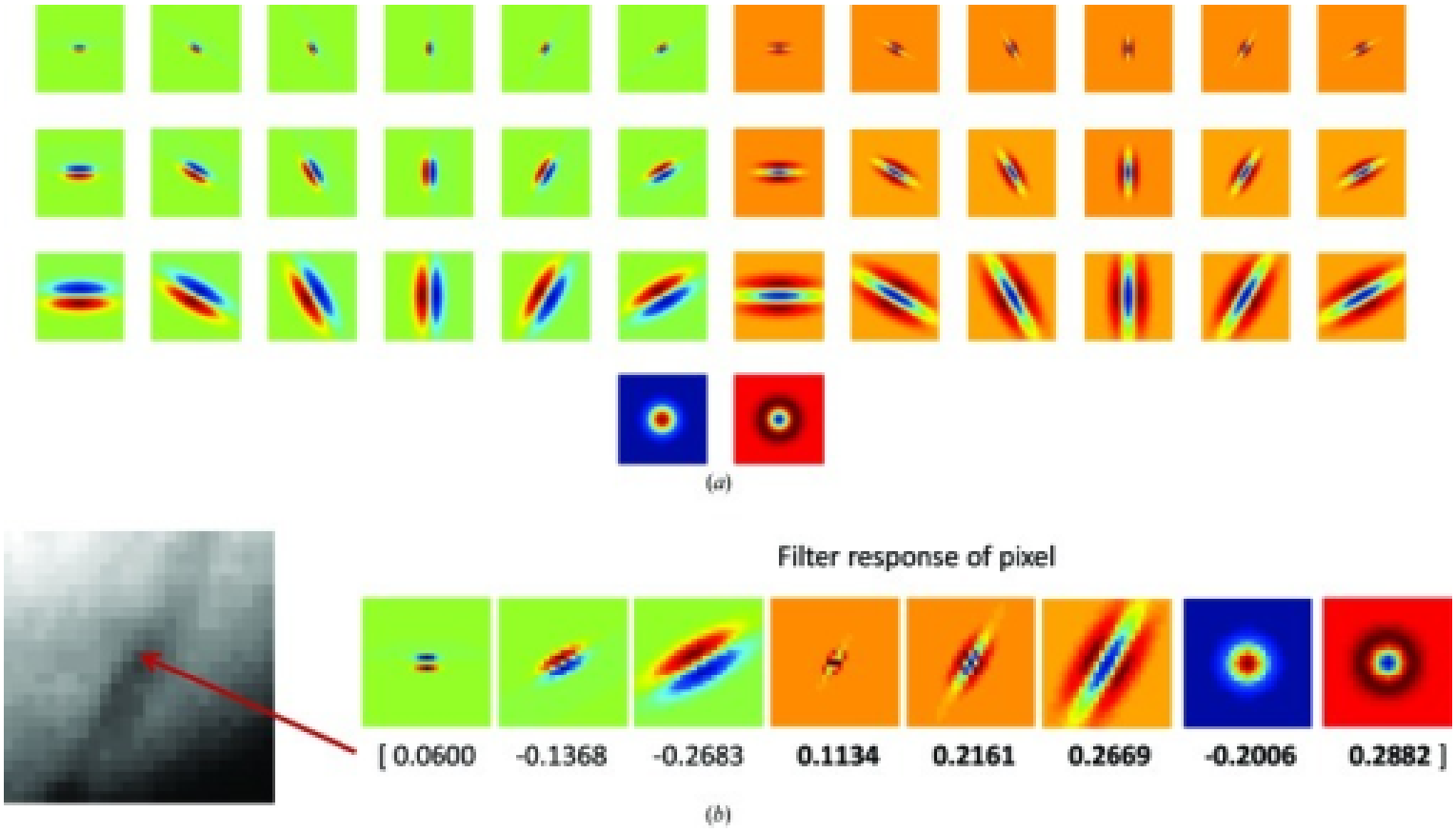


Image features

- **Edge filters**

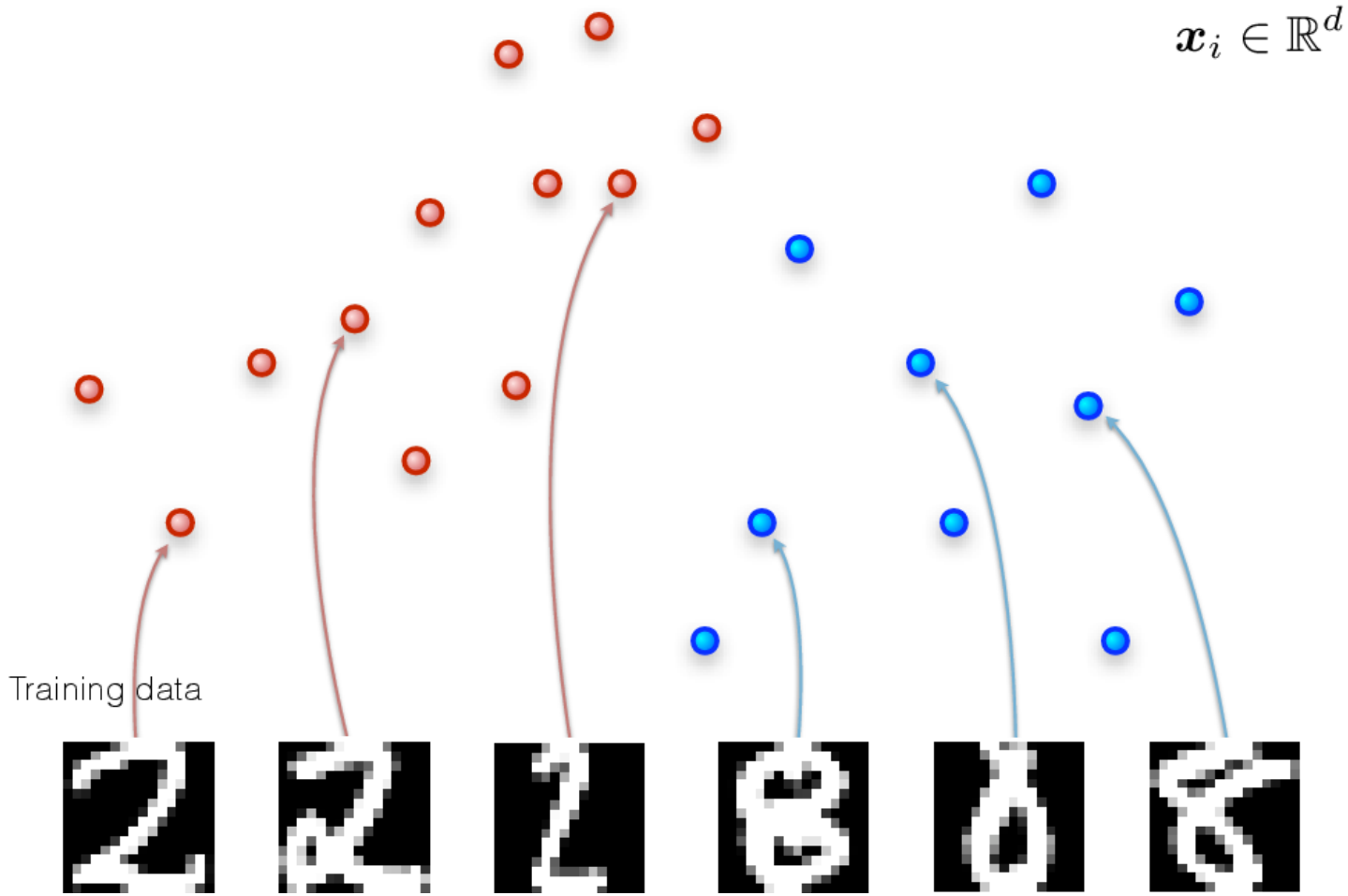


SVM (Support Vector Machine)

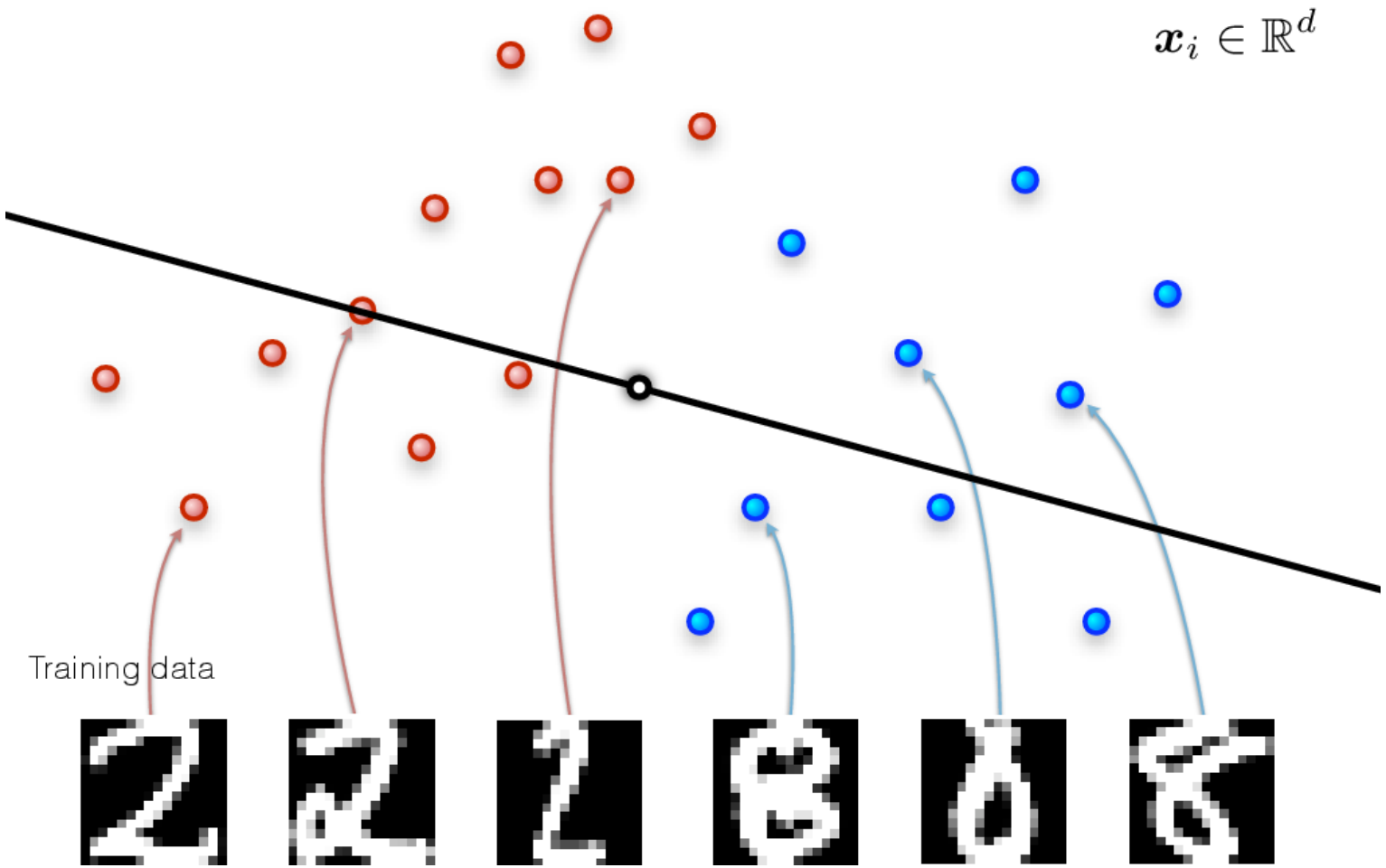
$$\boldsymbol{x}_i \in \mathbb{R}^d$$

Training data

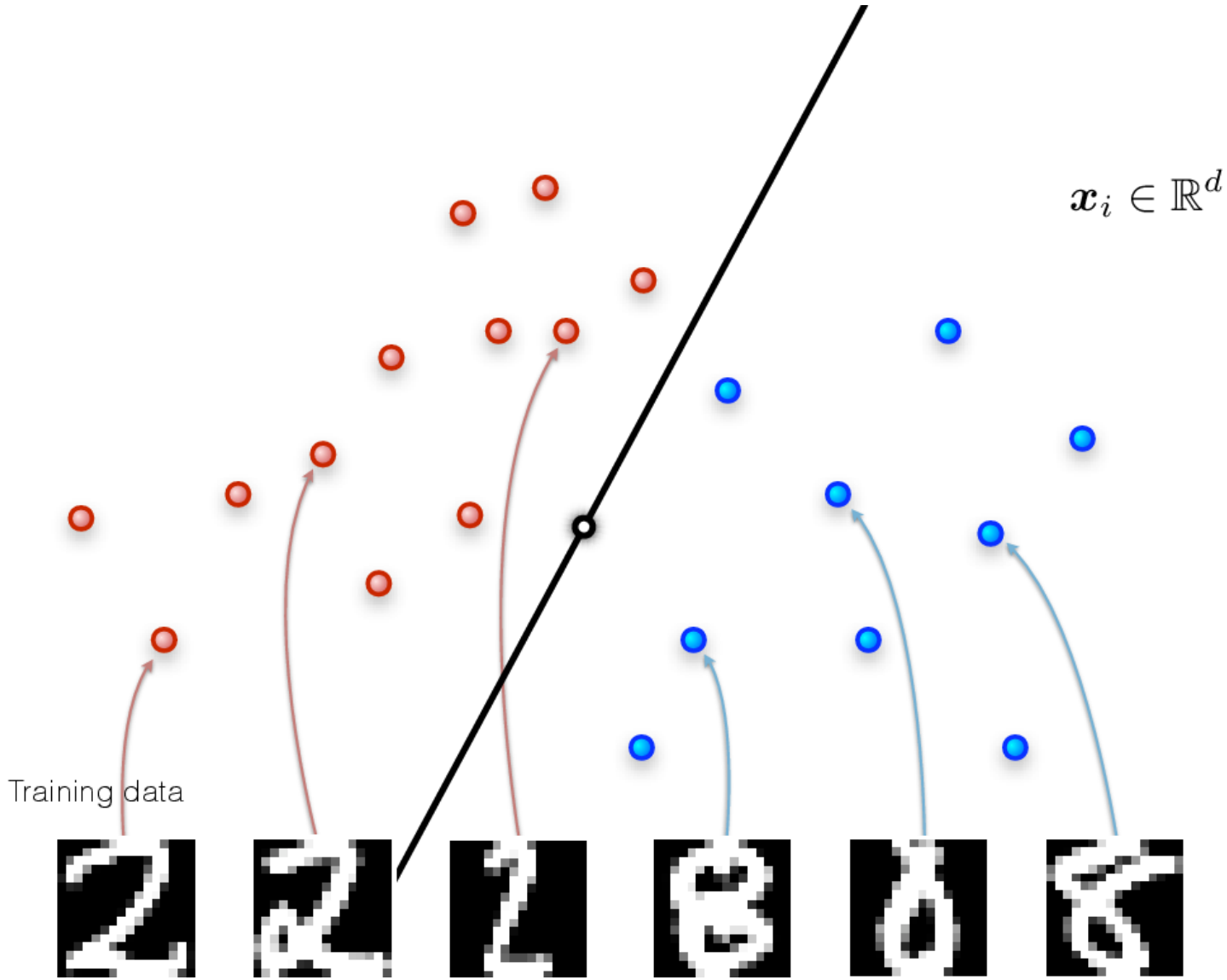
SVM



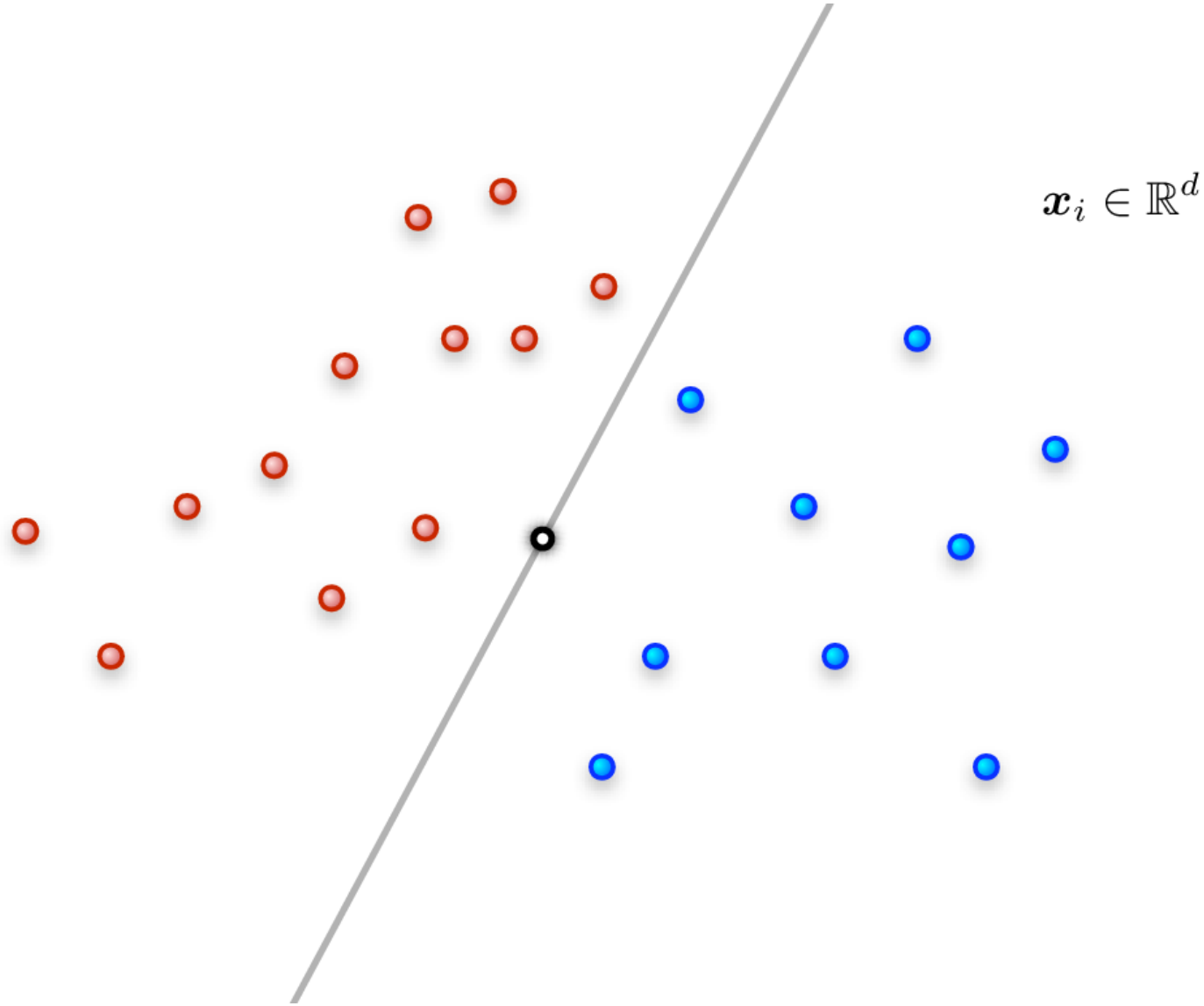
SVM



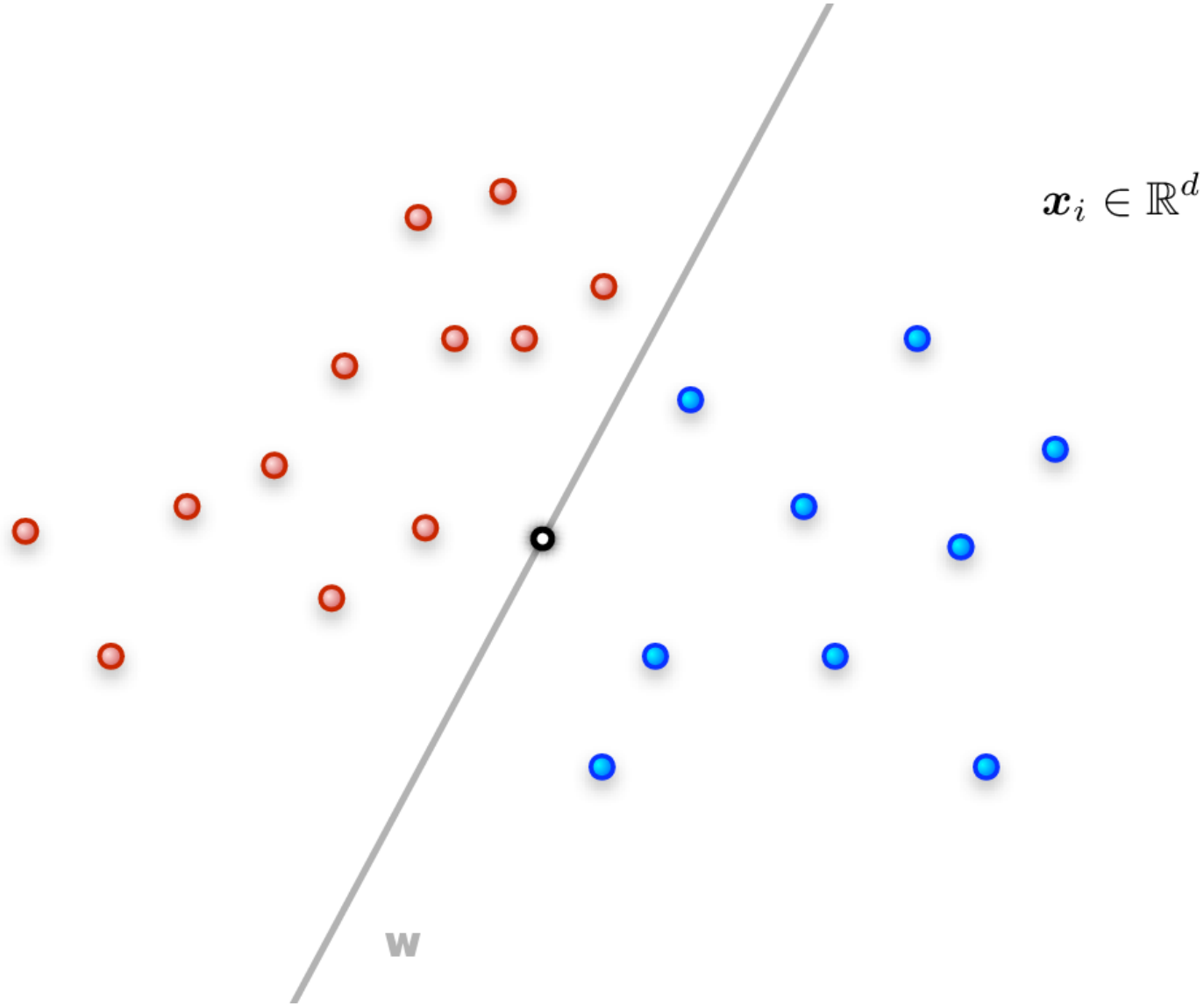
SVM



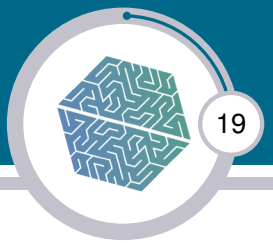
SVM



SVM



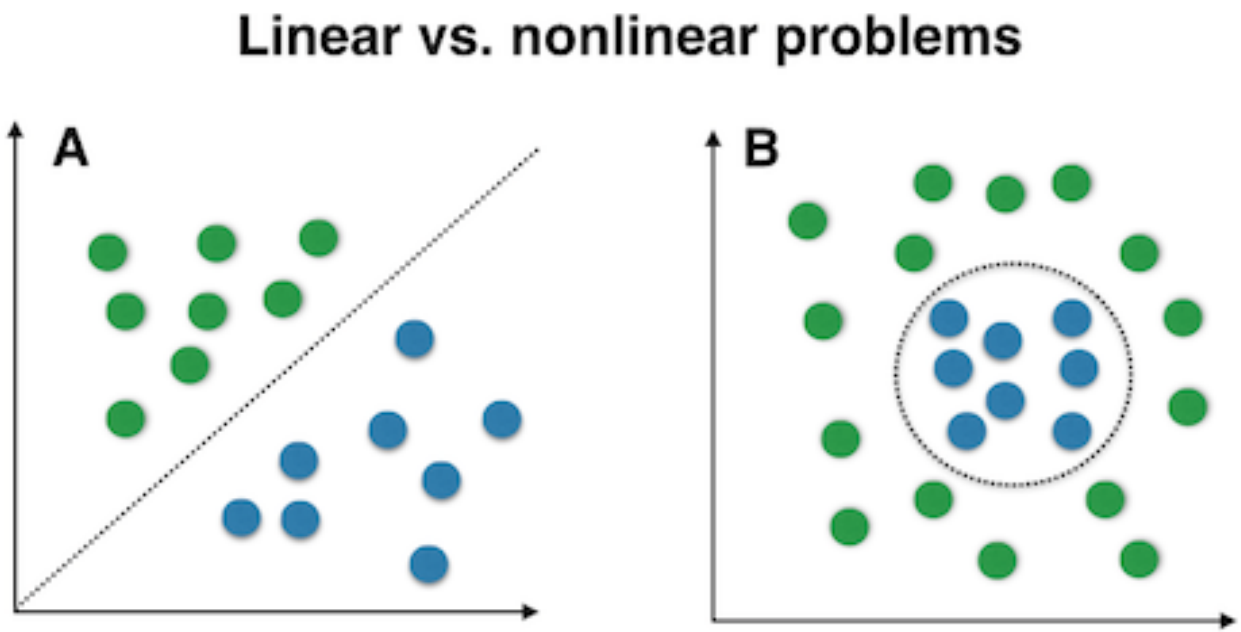
Questions?



- Why just using linear classifiers?
- Can we learn features from data instead of hand-designing them?

Machine learning

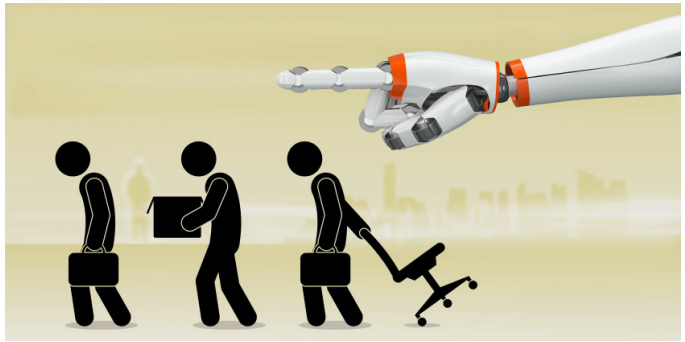
- **Linear classifiers**: Logistic regression, Support Vector Machine, ...
- **Non-linear classifiers**: Decision trees, (Deep) neural networks



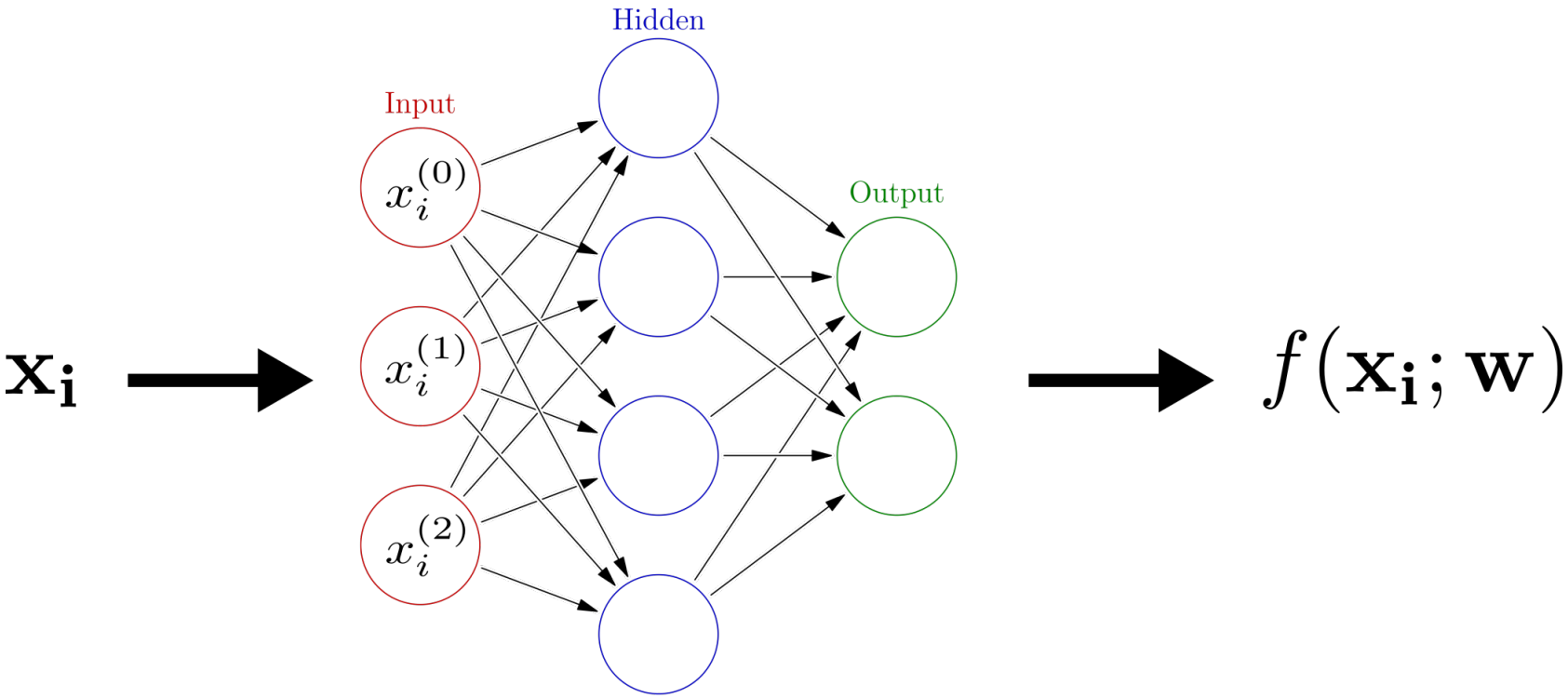
source: `http://sebastianraschka.com`

Neural networks

- Can we learn features from data instead of hand-designing them?
- Popular solution in the past few years = neural networks
  - Non-linear classifier
  - Neural Networks are modelled as collections of neurons that are connected in an acyclic graph



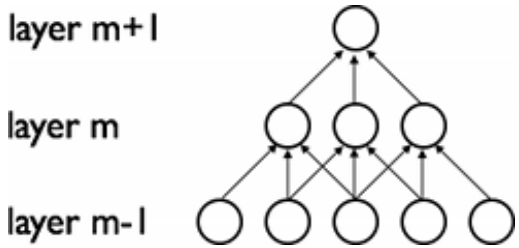
Neural networks



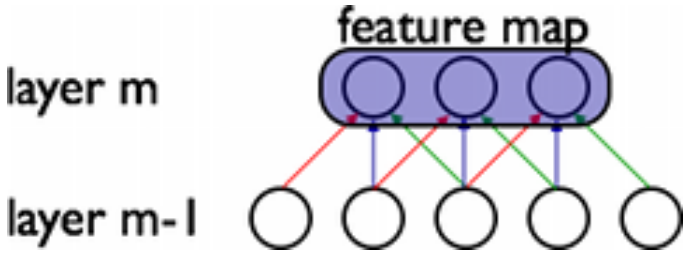
- The vector ***w*** is here represented by the edge weights
- Want to learn ***w*** by minimizing a certain loss function (more on this later)

Neural Networks

- **Sparse connectivity**: exploit spatially-local correlation by enforcing a local connectivity pattern between neurons of adjacent layers
- **Weight sharing**



Sparse connectivity

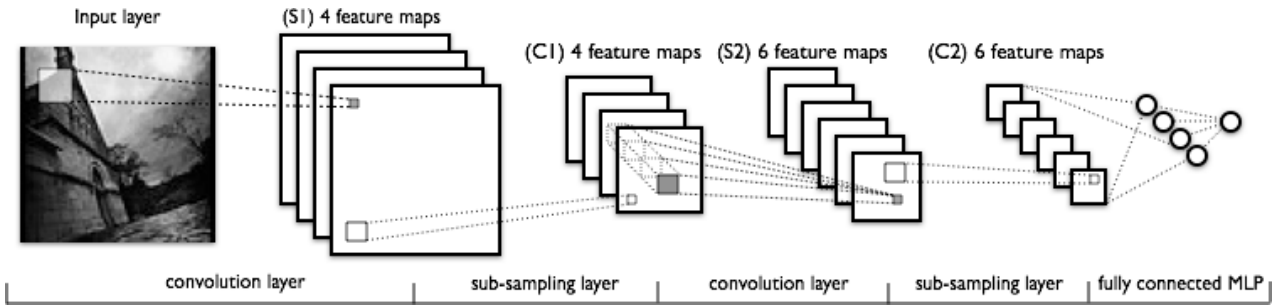


Weight sharing

source: `http://deeplearning.net/tutorial/lenet.html`

Neural Networks for Images

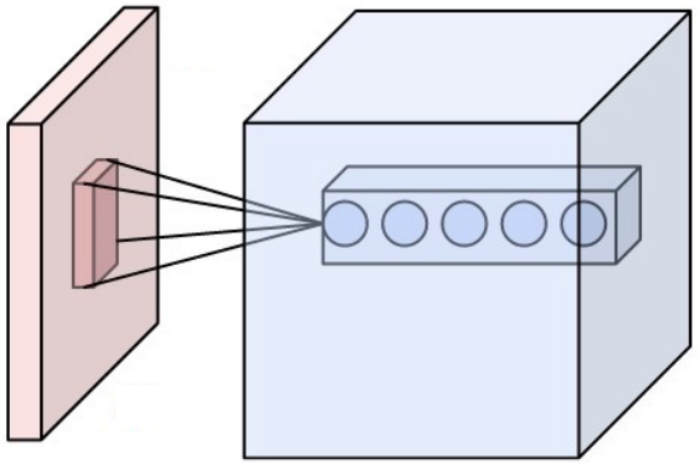
- Convolutional Neural Network = Feed-forward neural network in which the connectivity pattern between its neurons is inspired by the organization of the animal visual cortex.
- Three building blocks (more on this later)



Neural Networks for Images: Receptive Fields

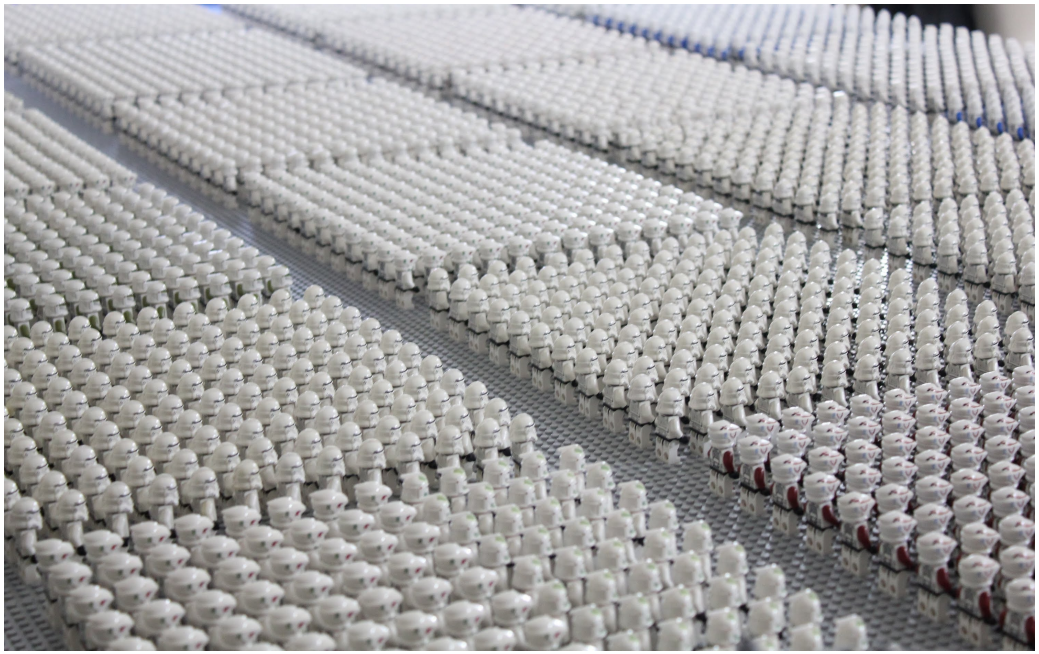
- Topological connectivity
  - encourage network to first extract localized features
  - subsequent layers: less and less localized features

- Receptive field
  - inputs that can affect a neuron (other weights = 0)
  - small images patches as receptive fields
  - can have multiple channels (in figure: 5)

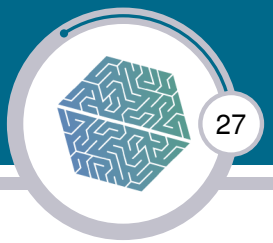


Neural Networks for Images: Translation Invariance

- Translation invariance of images
  - image patches look the same, irrespective of their location
  - idea: extract translation invariant features
  - what does that mean for a neural network?

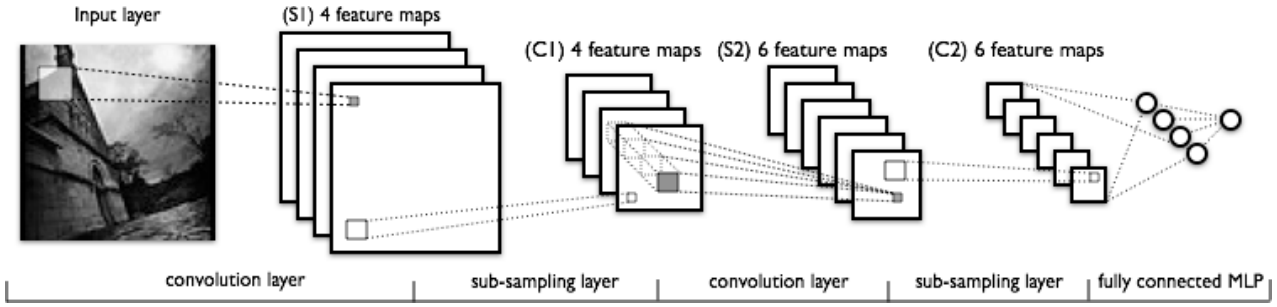


Neural Networks for Images: Weight Sharing



- Weight Sharing
  - neurons share the same weights = compute same function
  - differ in location of their receptive field = different input
  - mirrors what has been done in image processing (manually)
- Shift-invariant Filters
  - layers learn shift-invariant filters
  - weights define a filter mask (e.g. 3x3 or 5x5)
  - typically as many neurons as inputs (border padding etc.)
    - e.g. 64x64 pixel per image $\Rightarrow$ 64x64 neurons per channel
  - color images: 3 color channels, 3-dimensional filter mask
  - nice demo at `cs231n.github.io/assets/conv-demo/index.html`

CNN: Buildings blocks



- **Three building blocks:**
  - Convolutional layer
  - Pooling layer
  - Fully-connected layer

Convolutional Layers

- **Convolution:**
  - Mathematical operation on two functions (f and g)
  - It produces a third function that is typically viewed as a modified version of one of the original function
  - This operation can be used to detect edges in an image

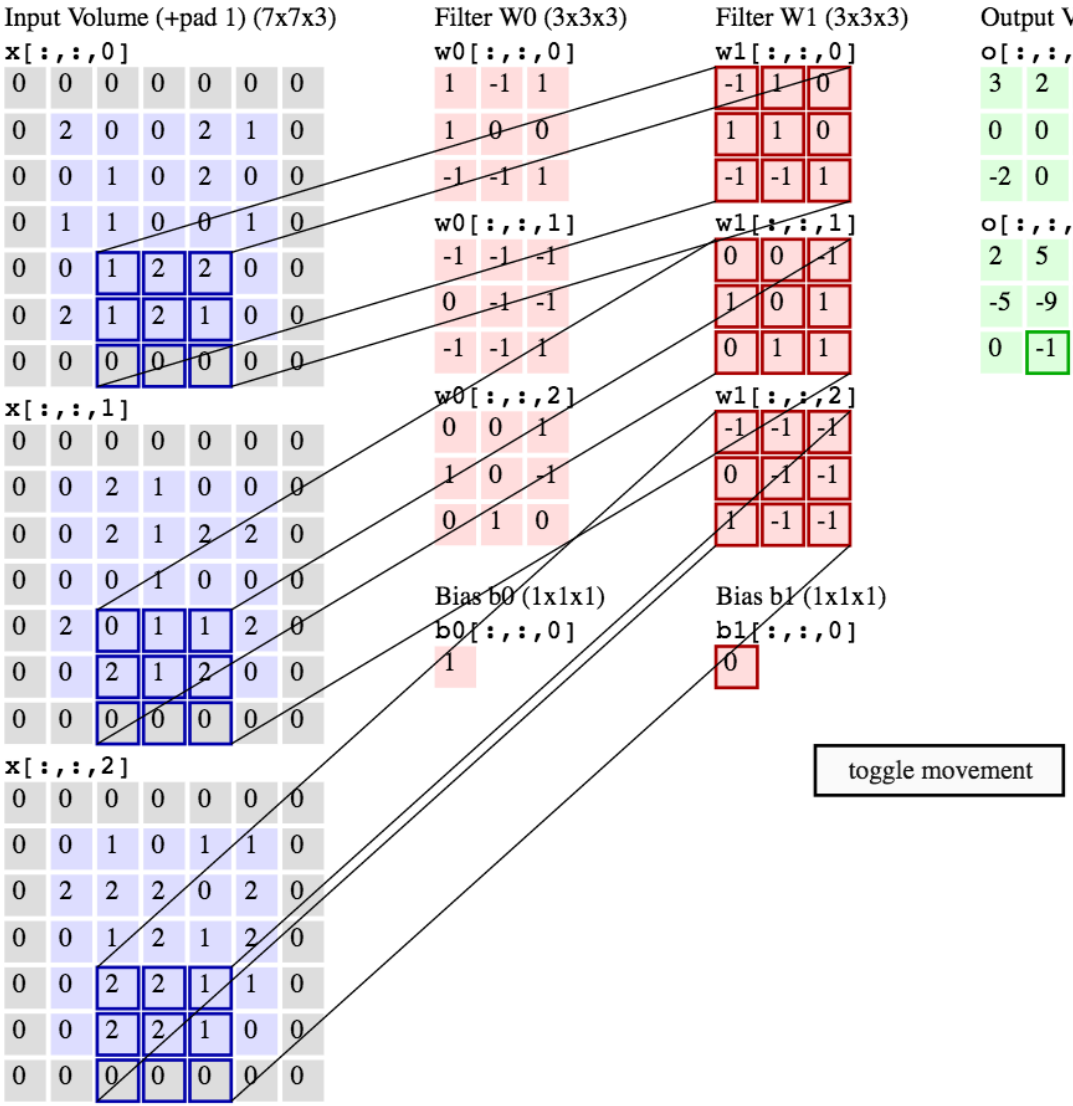
-1	0	1
-2	0	2
-1	0	1

Horizontal

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

Vertical

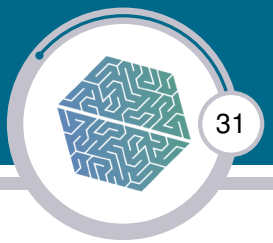
Convolutional Layers: Animation



source:

cs231n.github.io/assets/conv-demo/index.html

Convolutional Layers: Mathematics



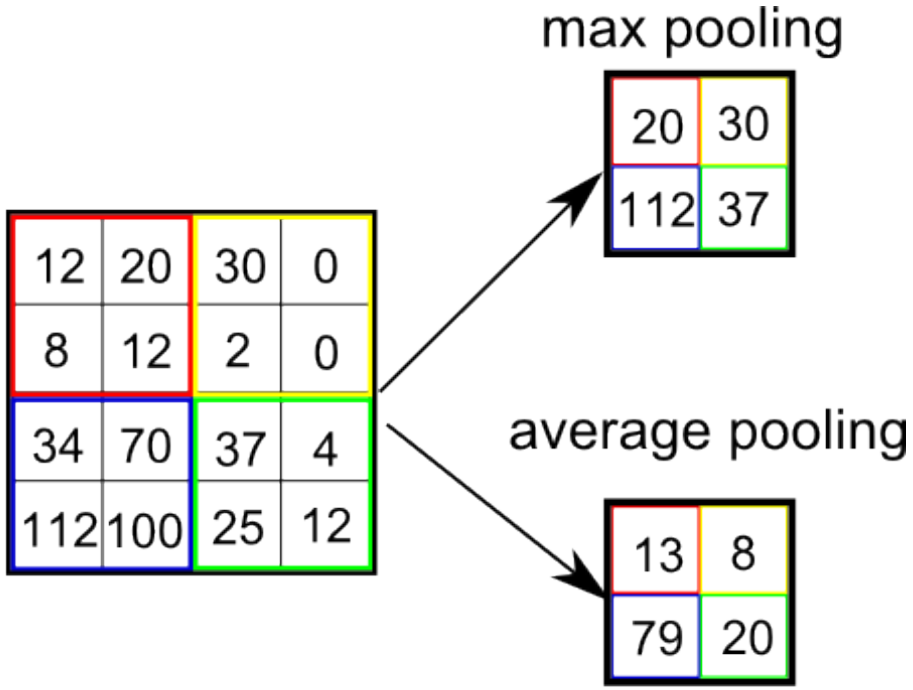
- Convolution in 2D (5x5)

$$F_{n,m}(\boldsymbol{x}; \boldsymbol{w}) = \sigma\left(b + \sum_{k=-2}^{2}\sum_{l=-2}^{2} w_{k,l} \cdot x_{n+k,m+l}\right)$$

  - (n, m): center of receptive field
  - $\boldsymbol{x}$: image (2D pixel field)
  - $\boldsymbol{w}$: weights = arranged as a 2D mask
  - related to convolution in mathematics

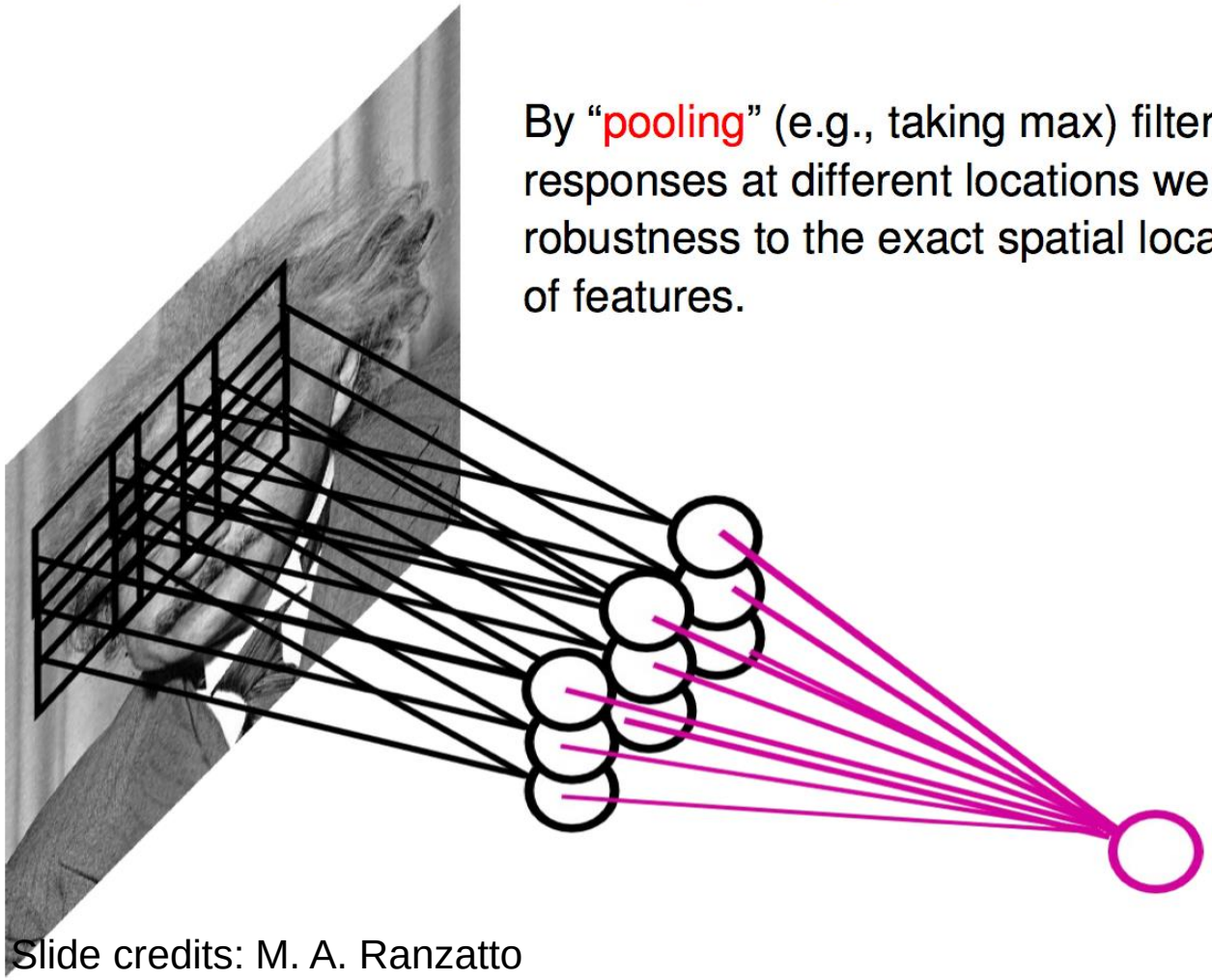
Pooling

- Reduce size of convolutional layers by down-sampling
- Take average over window (e.g. 2x2)
- Common practice: max pooling = take maximum in window



Pooling

Pooling Layer



By "pooling" (e.g., taking max) filter responses at different locations we gain robustness to the exact spatial location of features.

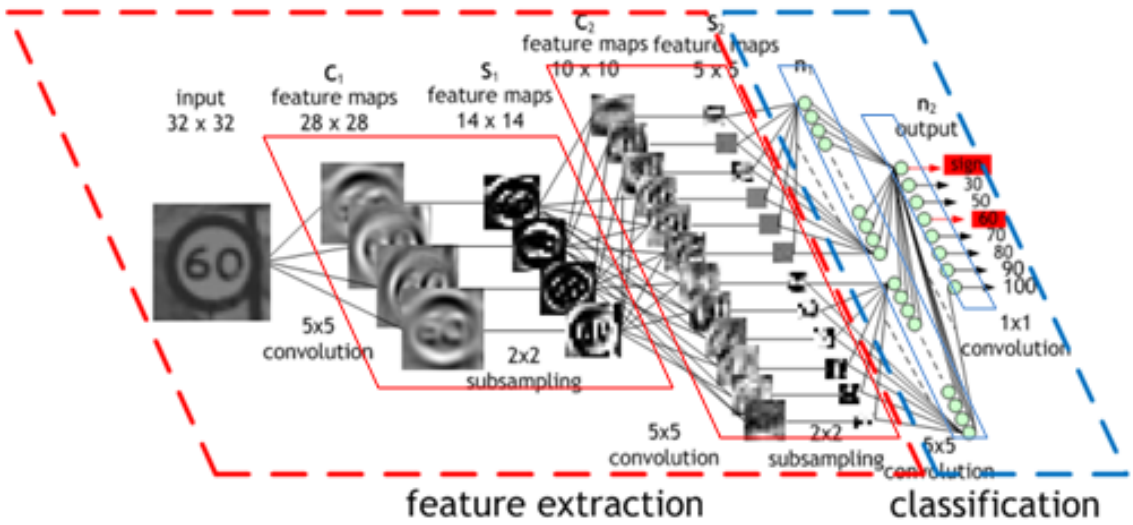
68

Ranzato

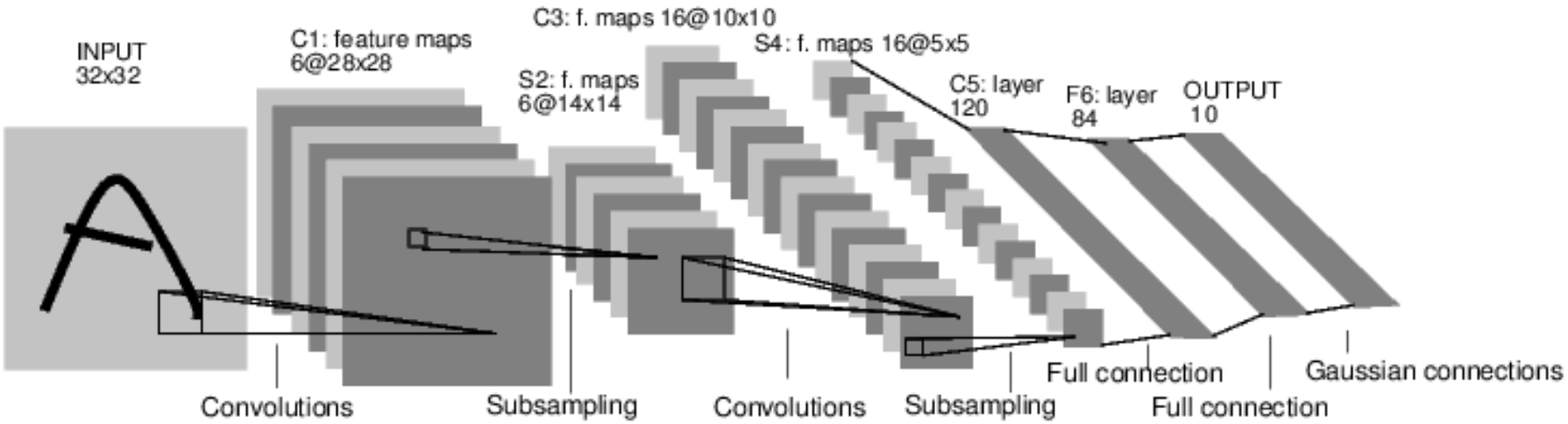
Slide credits: M. A. Ranzatto

Fully-connected layer

- High-level reasoning
- Connects all neurons in the previous layer to every single neuron it has

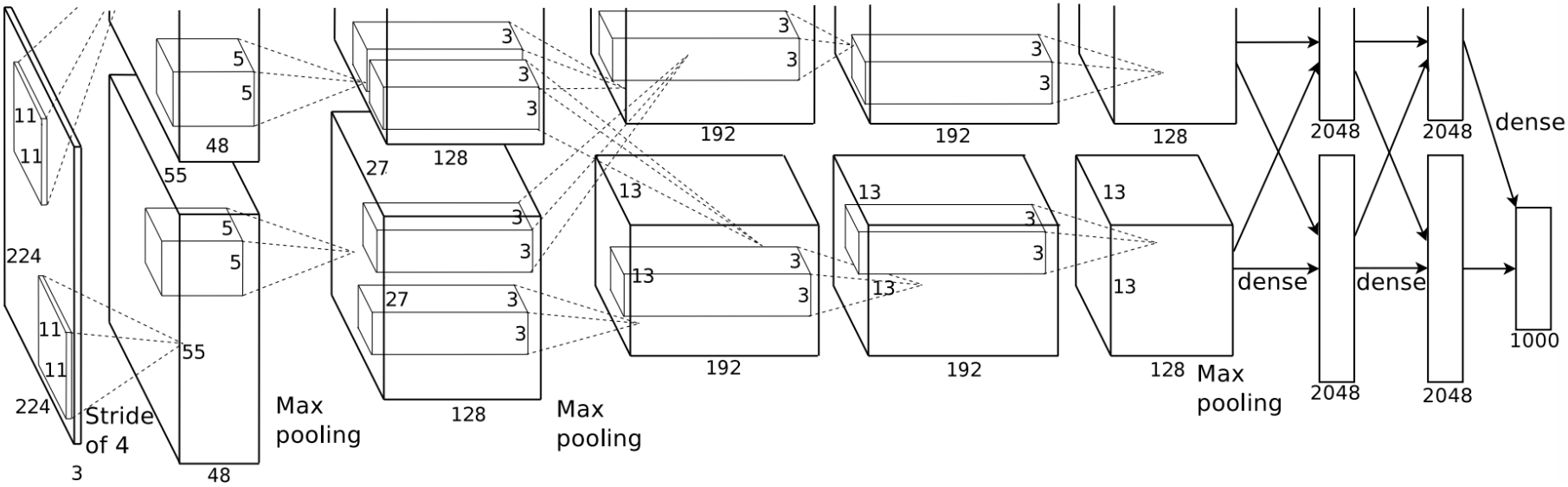


LeNet5



- Architecture LeNet5
  - layers C1/S2: 6 channels, cutting at border, 2x subsampling (4704 neurons)
  - layers C3/S4: 16 channels, cutting at border, 2x subsampling (1600 neurons)
  - layers F5/F6: fully-connected
  - output: Gaussian noise model (squared loss)

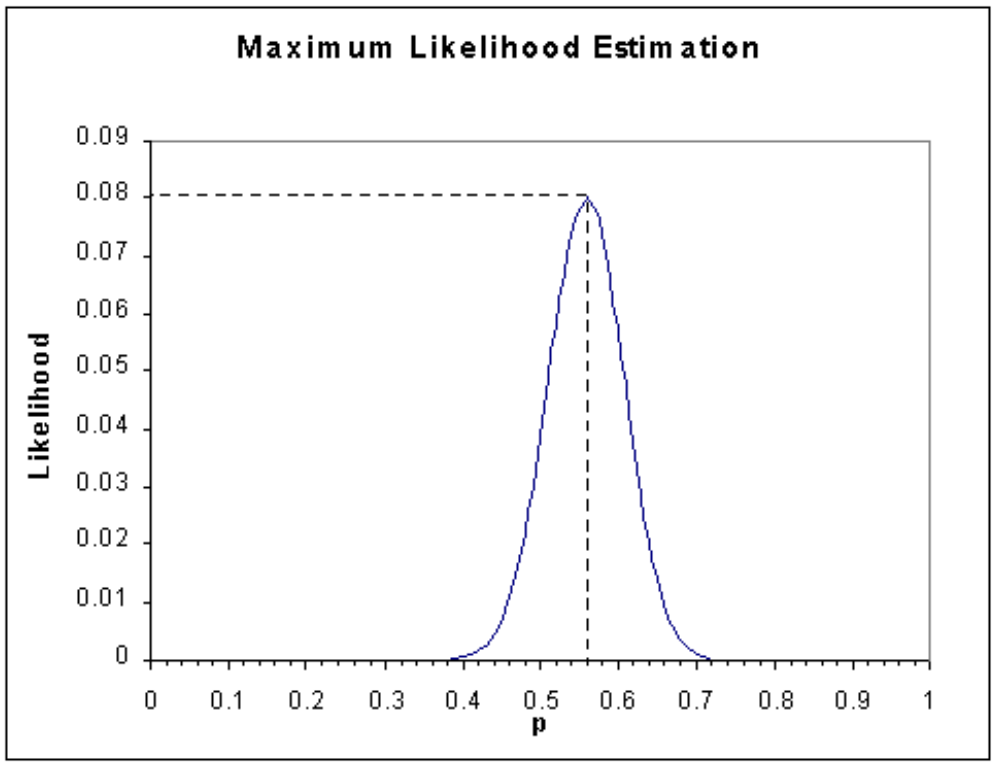
AlexNet



- AlexNet
  - Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton, 2012 ImageNet Classification with Deep Convolutional NN
  - 60 million parameters and 500,000 neurons
  - 5 convolutional layers, some followed by max-pooling
  - 2 globally connected layers with a final 1000-way softmax

Optimization

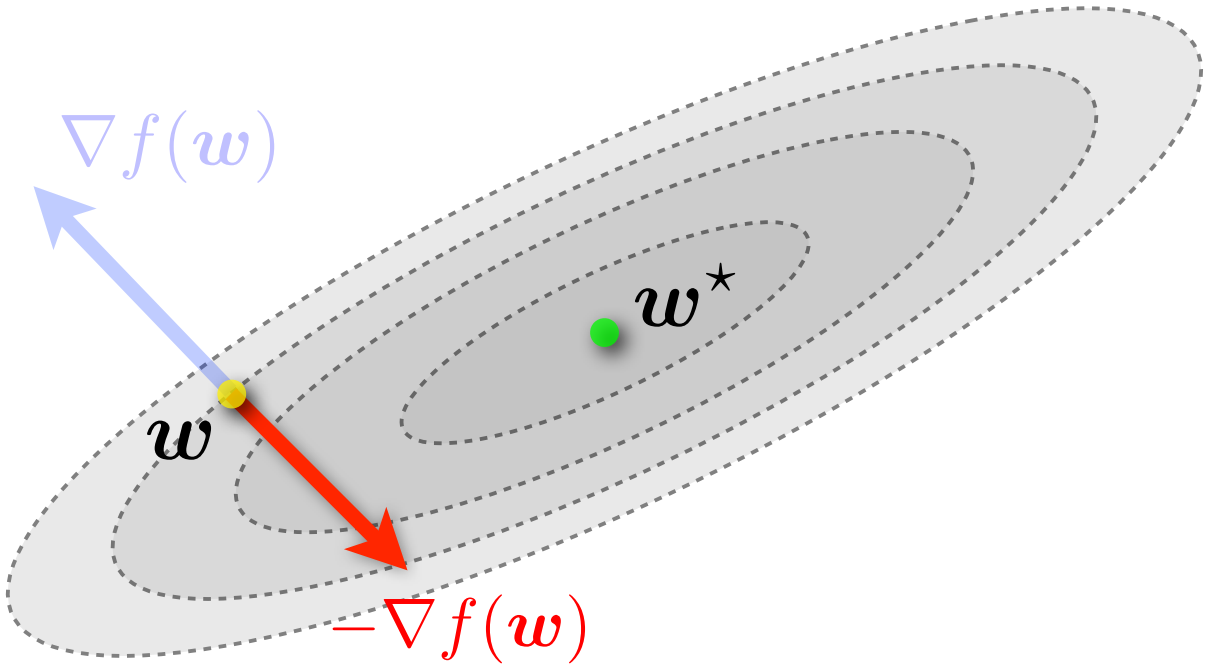
- The typical loss function $L(\boldsymbol{x}; \boldsymbol{w})$ used to train CNNs is **maximum likelihood**
  - Find the parameters $\boldsymbol{w}$ that are most likely to have generated the data $\boldsymbol{x}$



Optimization

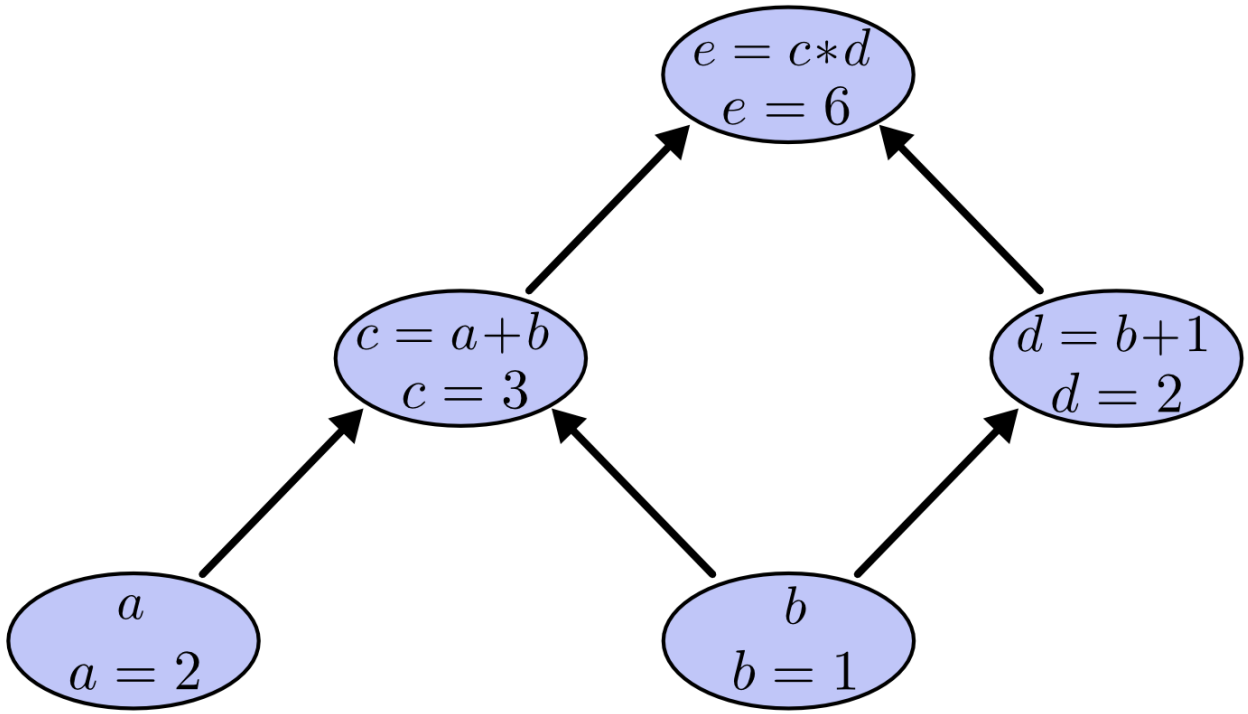
- Minimize $L(\boldsymbol{x}; \boldsymbol{w})$ using stochastic gradient descent, i.e.

$$\boldsymbol{w} \leftarrow \boldsymbol{w} - \eta \nabla_{\boldsymbol{w}} L(\boldsymbol{x}_i; \boldsymbol{w})$$



Computational graph

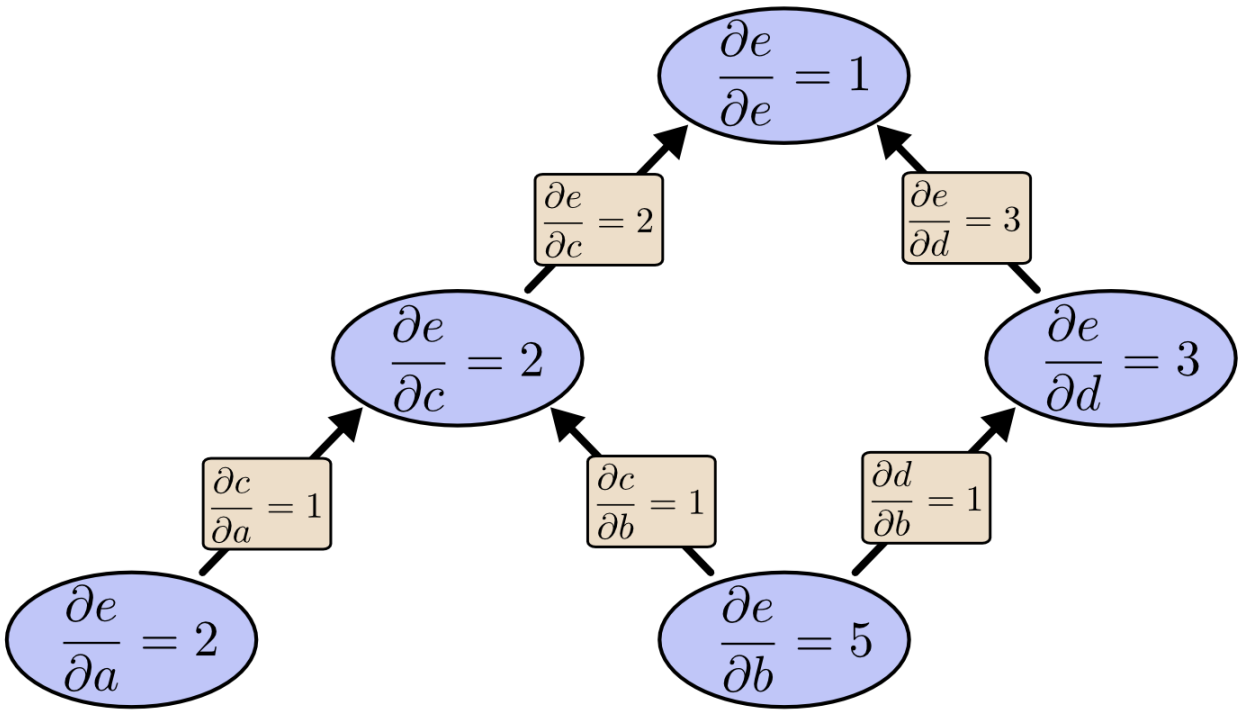
- Consider the function $f(a, b) = \underbrace{(a + b)}_{=c} \times \underbrace{(b + 1)}_{=d}$



source: `http://colah.github.io/posts/2015-08-Backprop/`

Computational graph

- Compute $\frac{\partial e}{\partial a}$?
- Chain rule: $\frac{\partial e}{\partial a} = \frac{\partial e}{\partial c} \cdot \frac{\partial c}{\partial a} = (b+1) \cdot 1$



Optimization

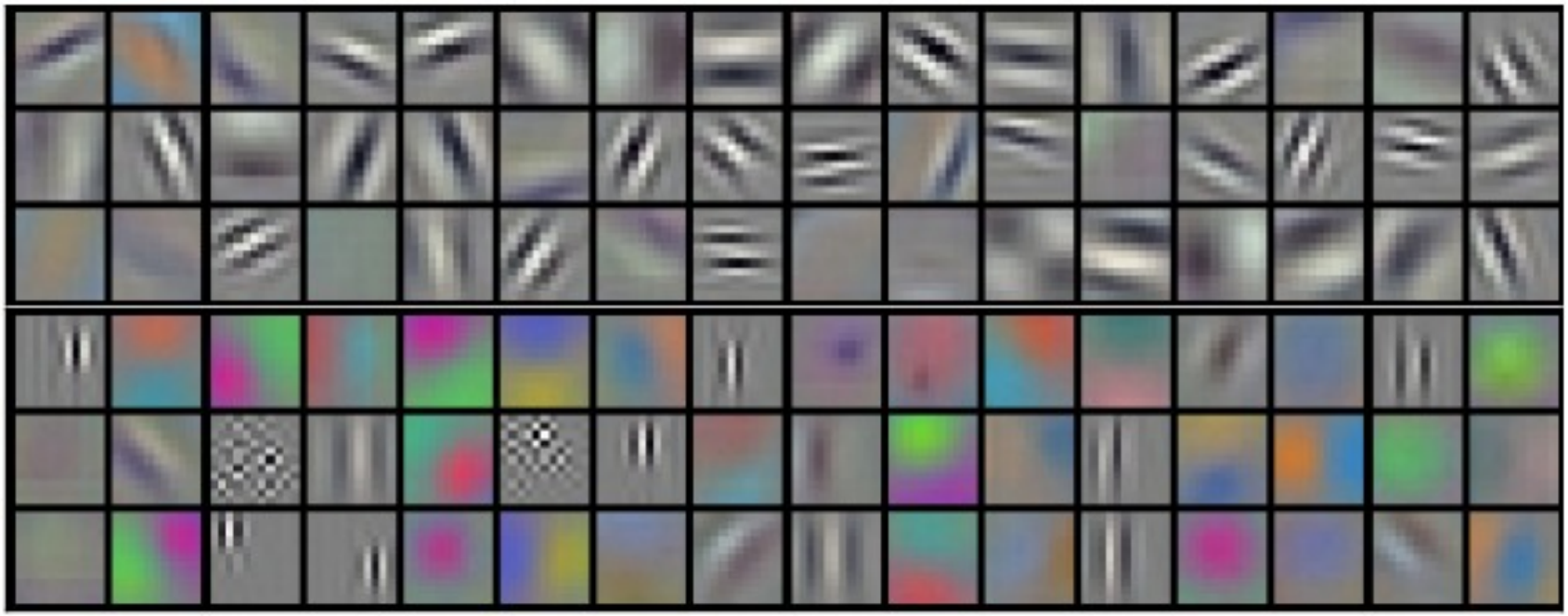
Algorithm 1 SGD to train deep neural networks

1: Given an initial $\boldsymbol{w}_0$
2: **for** $t = 1$ to T **do**
3: Compute $\nabla_{\boldsymbol{w}_t} L(\boldsymbol{x}_i; \boldsymbol{w}_t)$ using back-propagation
4: $\boldsymbol{w}_{t+1} \leftarrow \boldsymbol{w}_t - \eta \nabla_{\boldsymbol{w}_t} L(\boldsymbol{x}_i; \boldsymbol{w}_t)$
5: **end for**

- Practical considerations:
  - Pick initial parameters $\boldsymbol{w}_0$ (see Xavier's initialization)
  - Need to tune the step-size η
  - Fix a maximum number of iterations T (or use another stopping criterion)

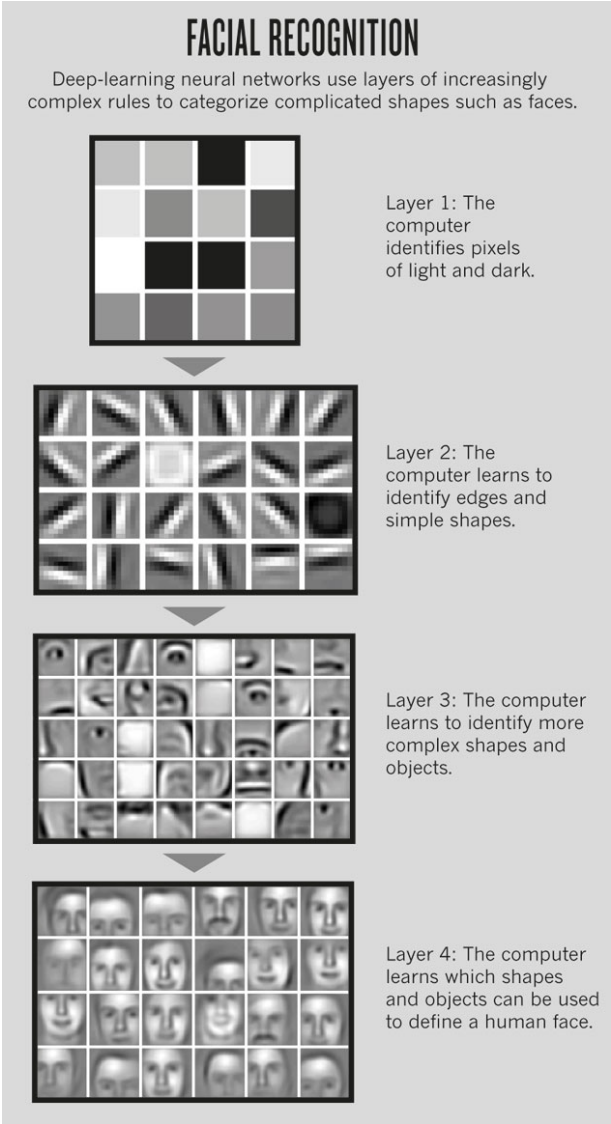
Learning Local Image Features

- Example: filters learned at first layer
- cf. Krizhevsky et al.: 96 filters of size 11x11x3

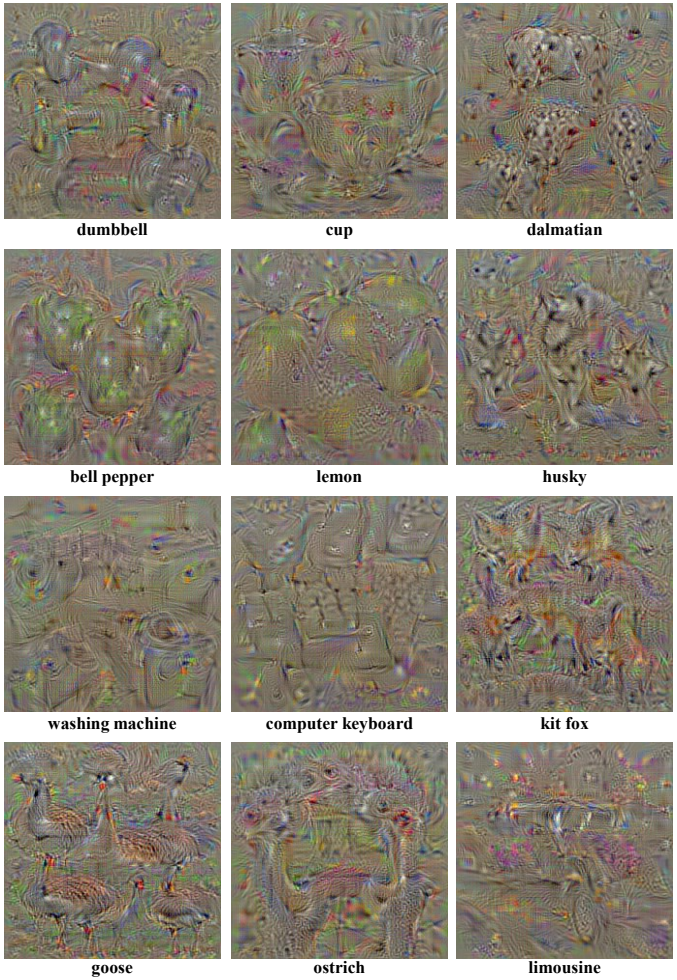


Learning Higher Level Features

- (c) Andrew Ng, trained on face images



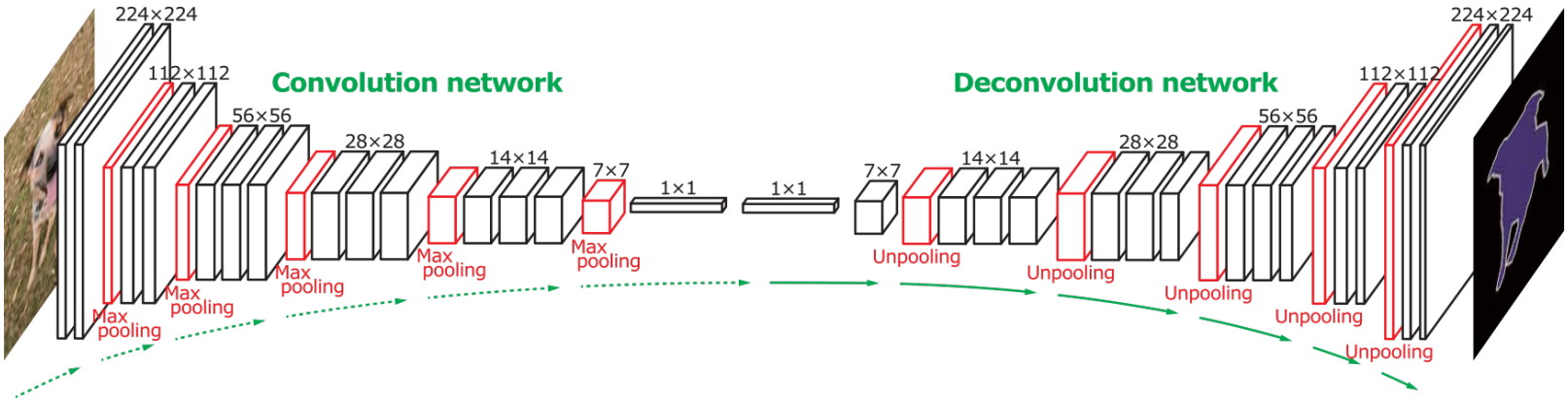
Inside CNNs



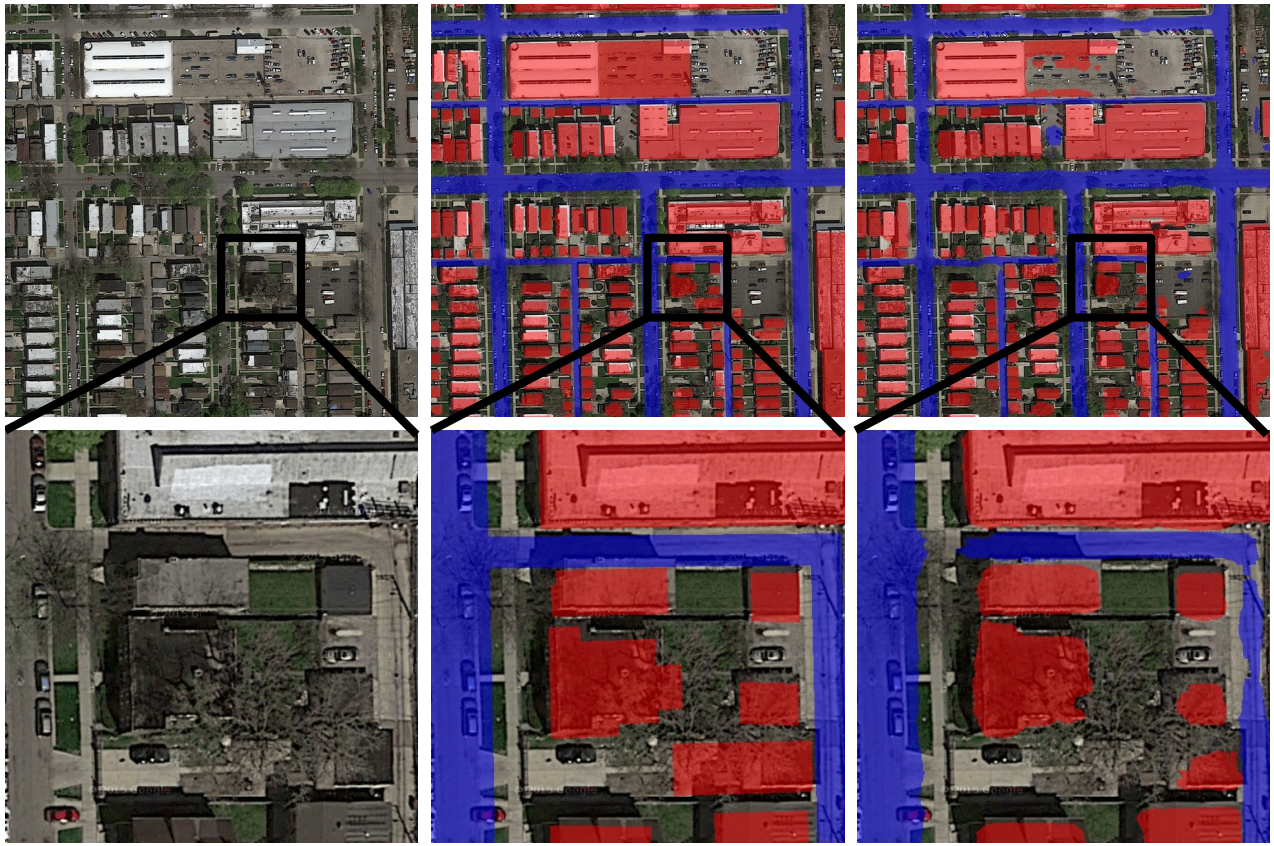
- Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps, Simonyan et al, 2015

Example - Road segmentation

- Recall that pooling layers shrink the dimension
- For segmentation, we want the final output to be the same size as the input

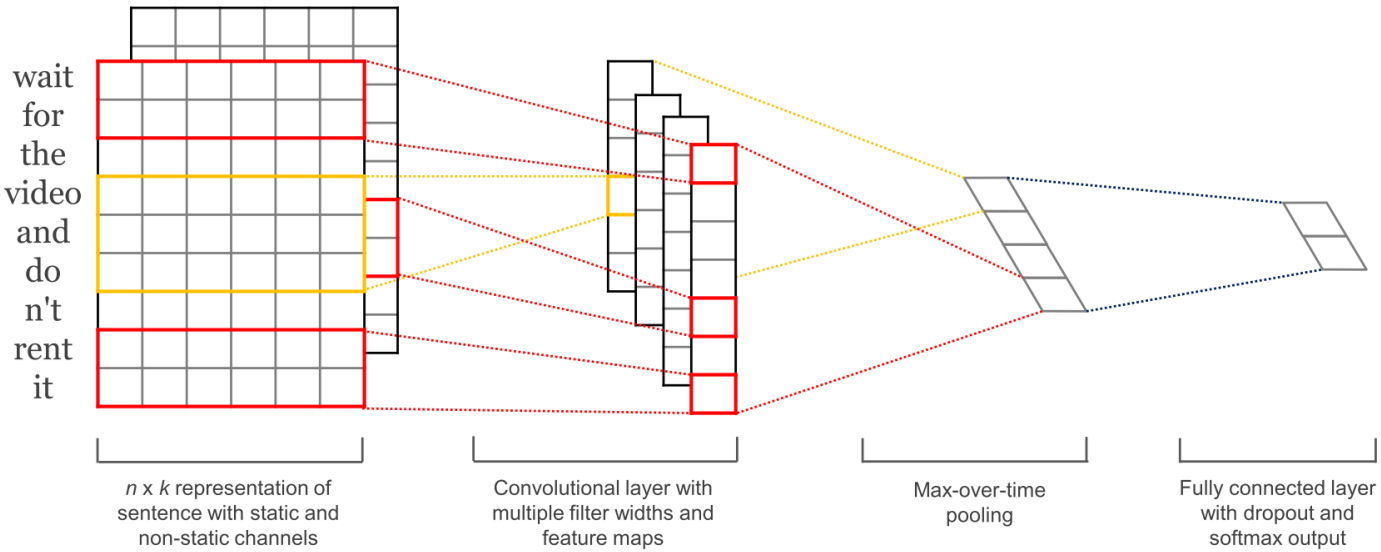


Example - Road segmentation

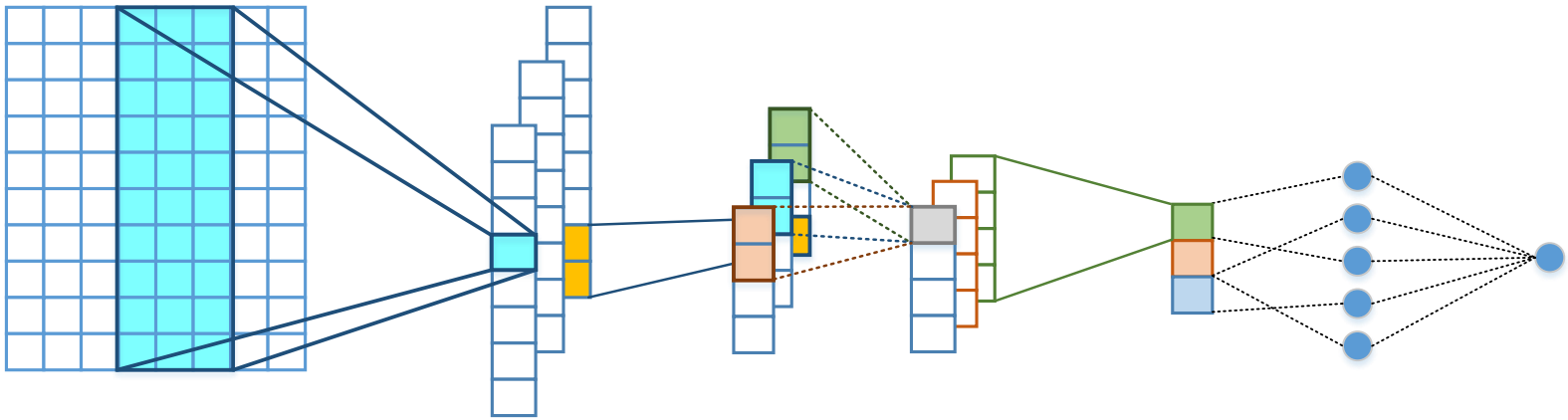


(a) aerial image (b) ground truth labels (c) FCN-8s prediction

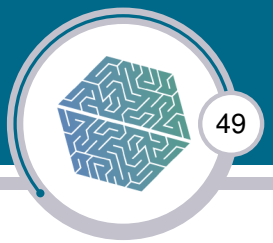
Example - Sentiment classification



Example - Sentiment classification



Available libraries



Language	Library
Python	Theano
Python	Tensorflow
Matlab	MatConvNet

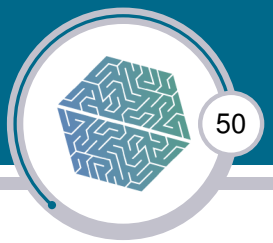
- For more, see `http://deeplearning.net/software_links/`

Thank you!

image credit: Mania Orand

Latex template credit: Lilyana Vankova

Credits



- Some slides, figures, ... were taken from Thomas Hofmann, Martin Jaggi, Yann Lecun, Andrej Karpathy, Hadi Daneshmand
- Additional figures taken from Wikipedia