Computer vision and machine learning for perception and decision making in driverless cars

Danijel Skočaj

Visual Cognitive Systems Lab
Faculty of Computer and Information Science
University of Ljubljana

Ljubljana, 12.06. 2018

University *of Ljubljana*
Faculty *of Computer and Information Science*

Computer vision and machine learning for perception and decision making in driverless cars

- Driverless cars as intelligent agents
  - Requirements
  - Perception-action loop
- Perception and decision making
  - Computer vision
  - Deep learning

Stone age of driverless driving

- DARPA 2004 Grand Challenge
  - Goal: autonomous drive of a car through Mohave dessert
  - Main reward: 1.000.000 USD was not awarded!

- In 2004, most of the algorithms did not work well in uncontrolled environment!
- Increase the robustness, adaptability, and intelligence!

Driverless cars today

2017

Waymo

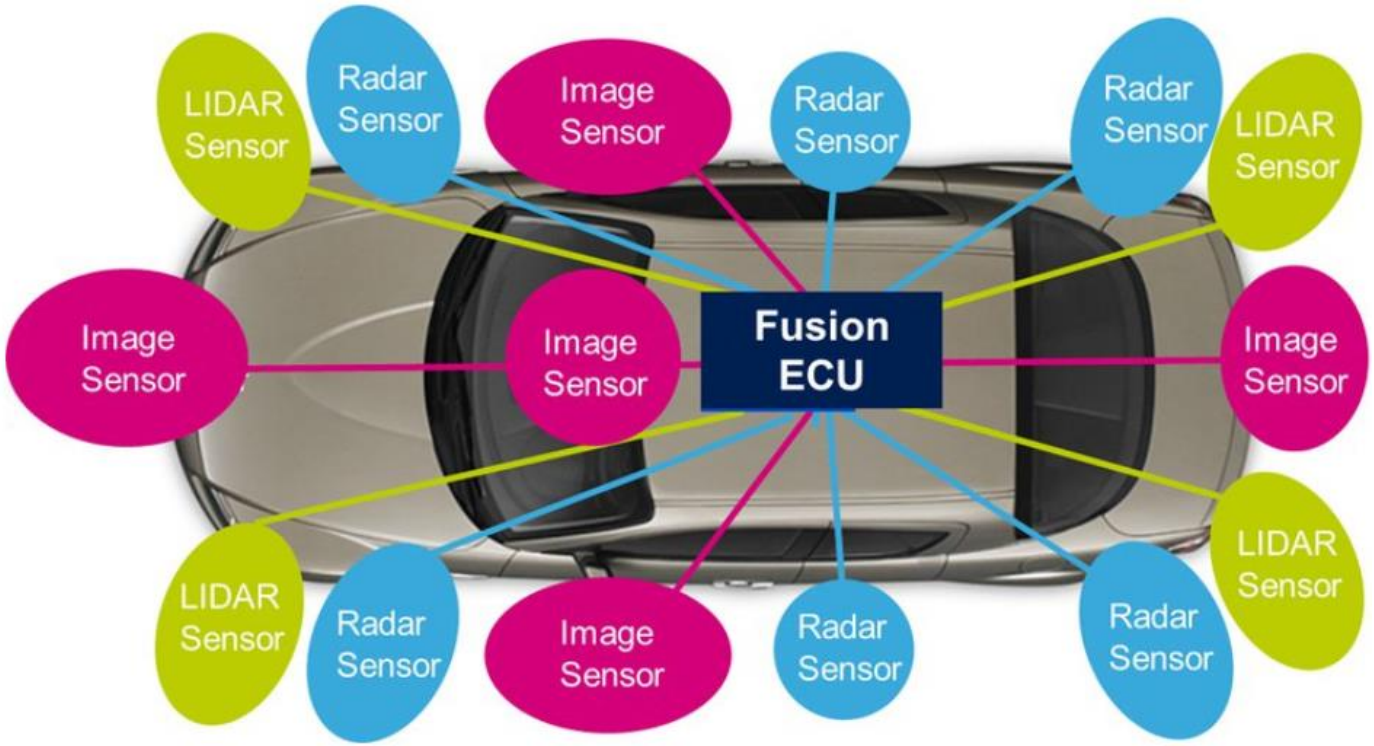
Enabling technologies

- Improved sensors!
- Improved perception!
- Improved decision making!

- Computer vision
- Machine learning
- Decision making

- Perception-action cycle

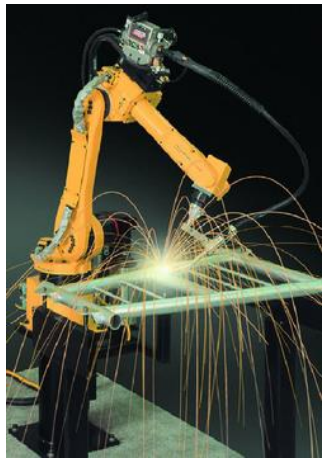
- Driverless cars
  =
  Intelligent robots



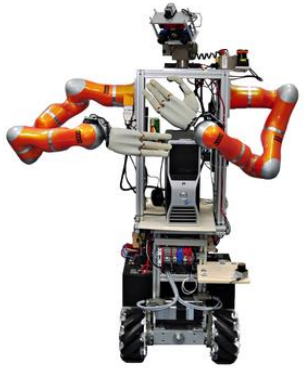
Sensors online

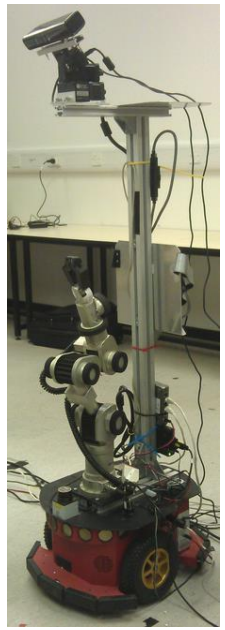
Cognitive robot systems

cognitive robots



industrial robots





SF

human

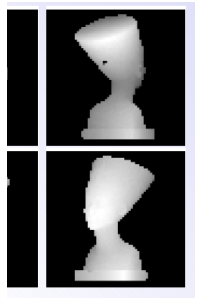
- perception
- action
- attention
- goals
- planning
- reasoning
- communication
- learning

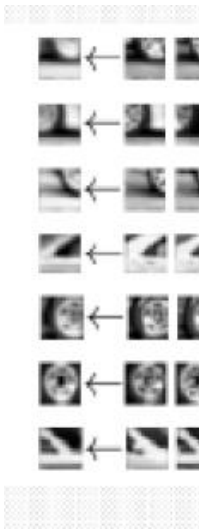
Perception

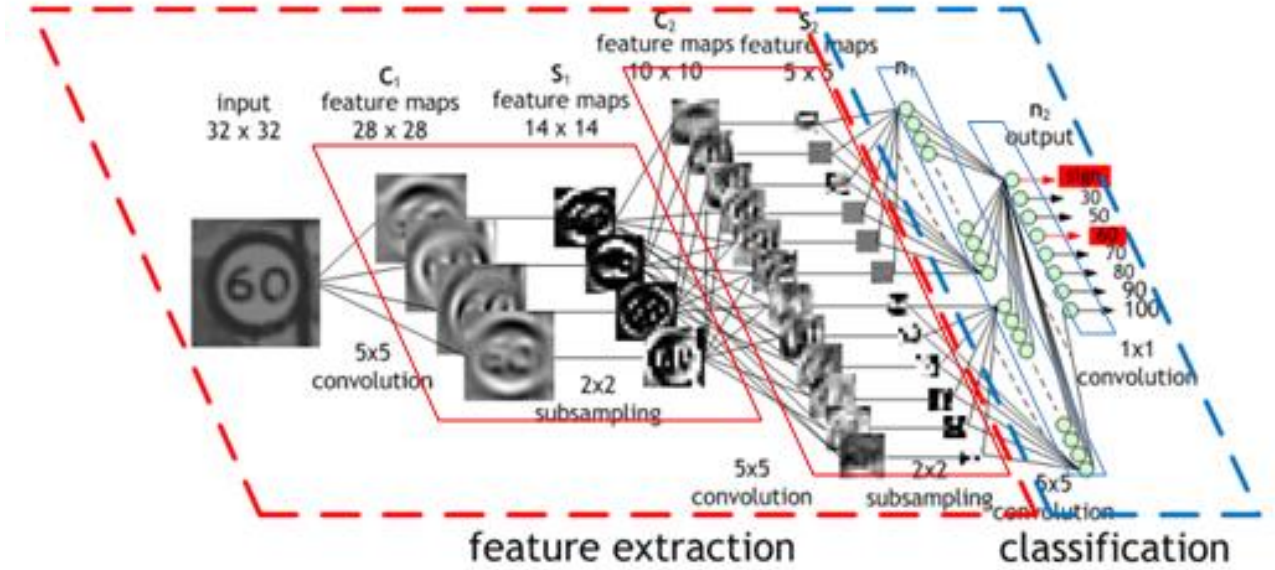
- Perception
  - Visual information (image, video; RGB, BW, IR,…)
  - Sound (speech, music, noise, …)
  - Haptic information (haptic sensors, collision detectors, ect.)
  - Range/depth/space information (range images, 3D models, 3D maps, …)
  - Many different modalities – very multimodal system
- Attention
  - Selective attention
  - Handling complexity of input signals

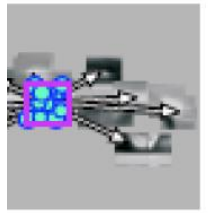
Representation of visual information

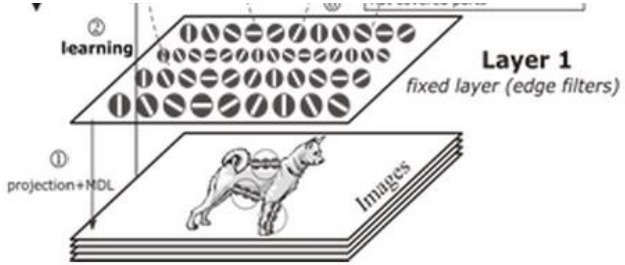
University of Ljubljana
Faculty of Computer and Information Science

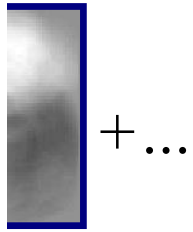






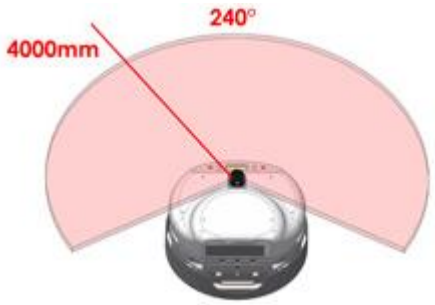


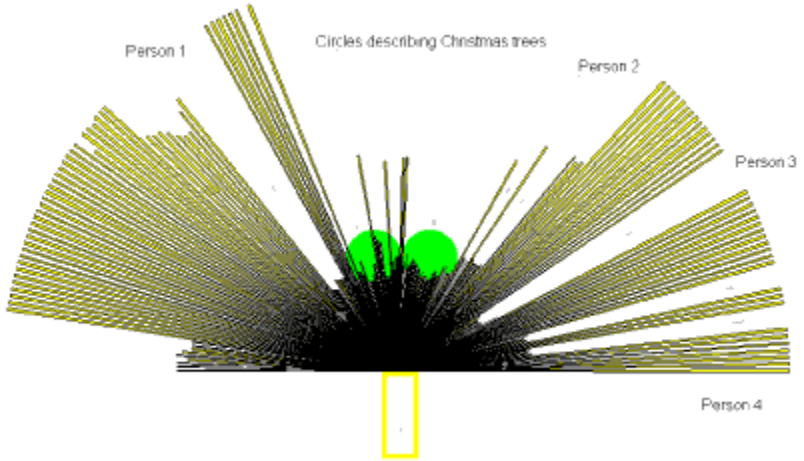




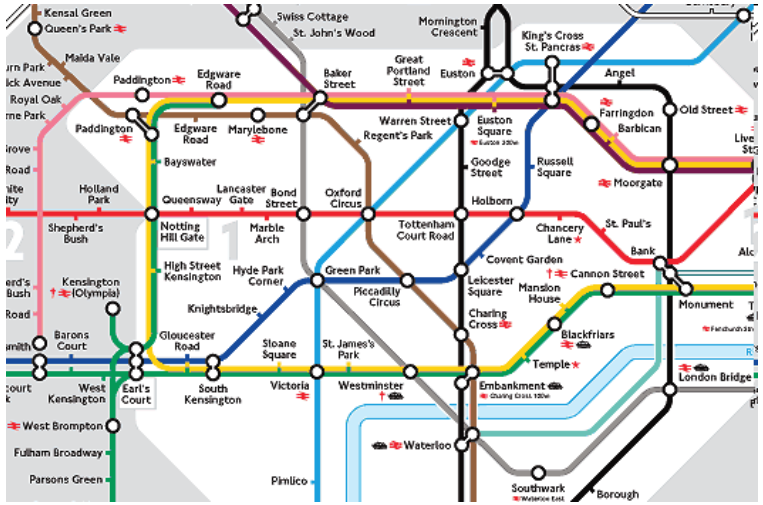
Representation of space

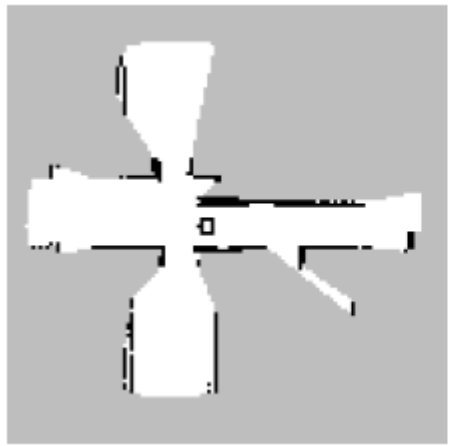
- Metric information

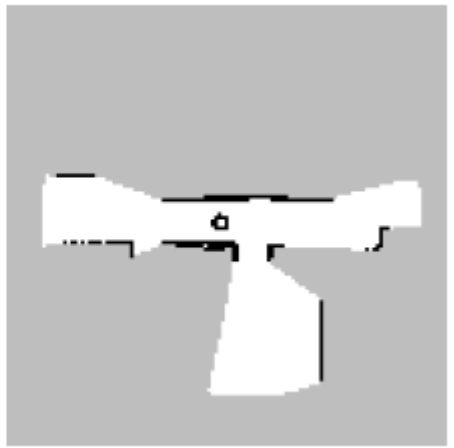




- Topological map



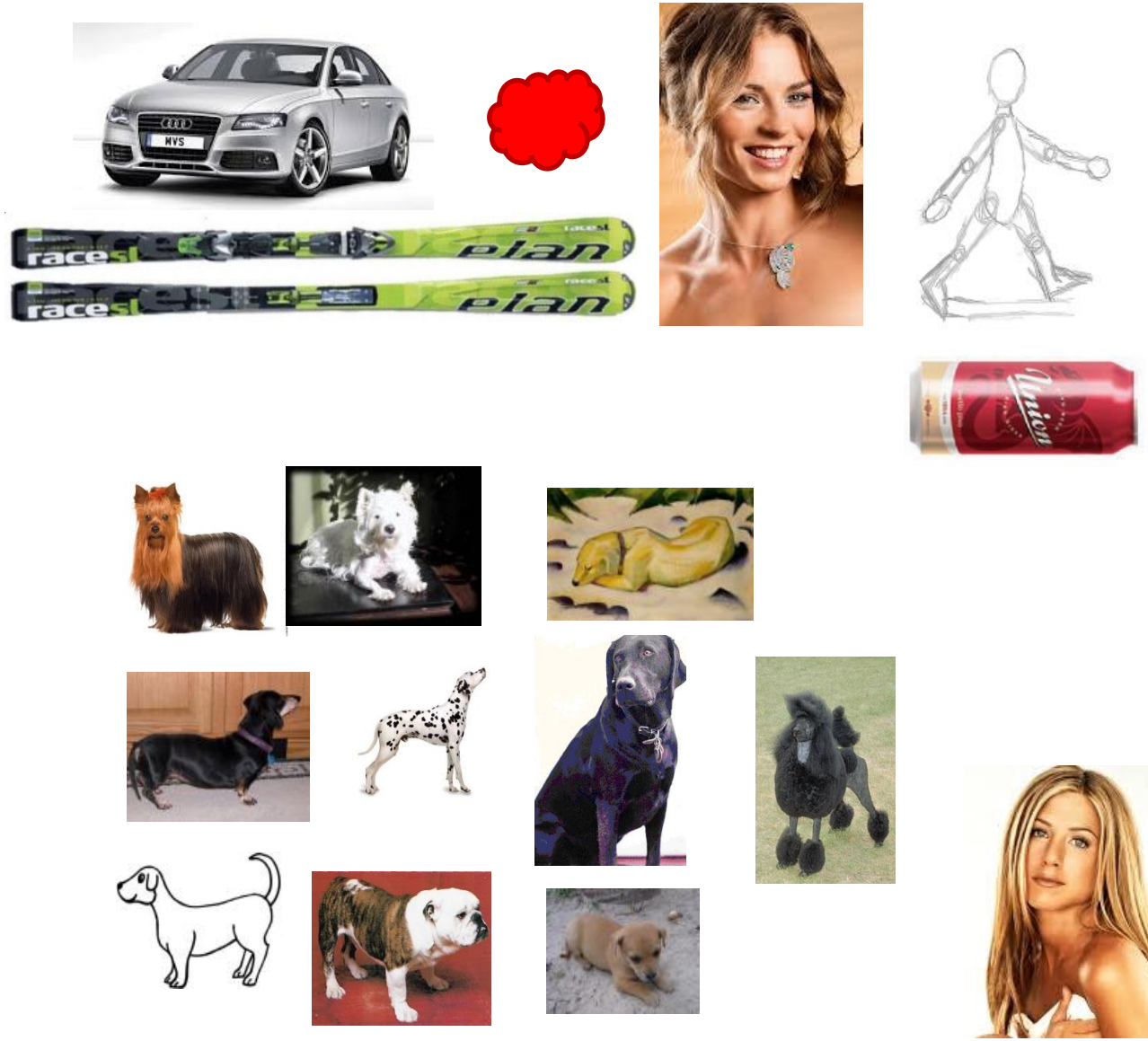




- Hierarchical representation

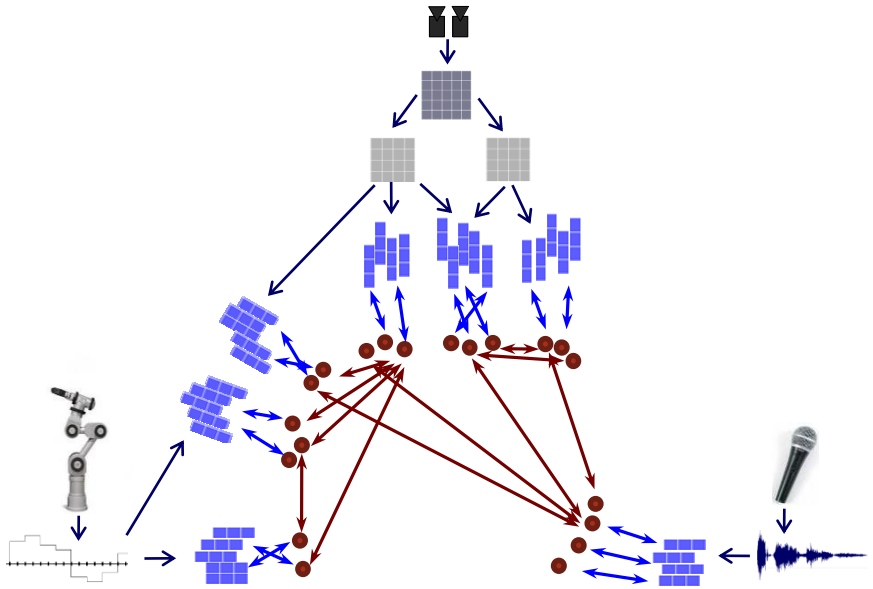
Recognition

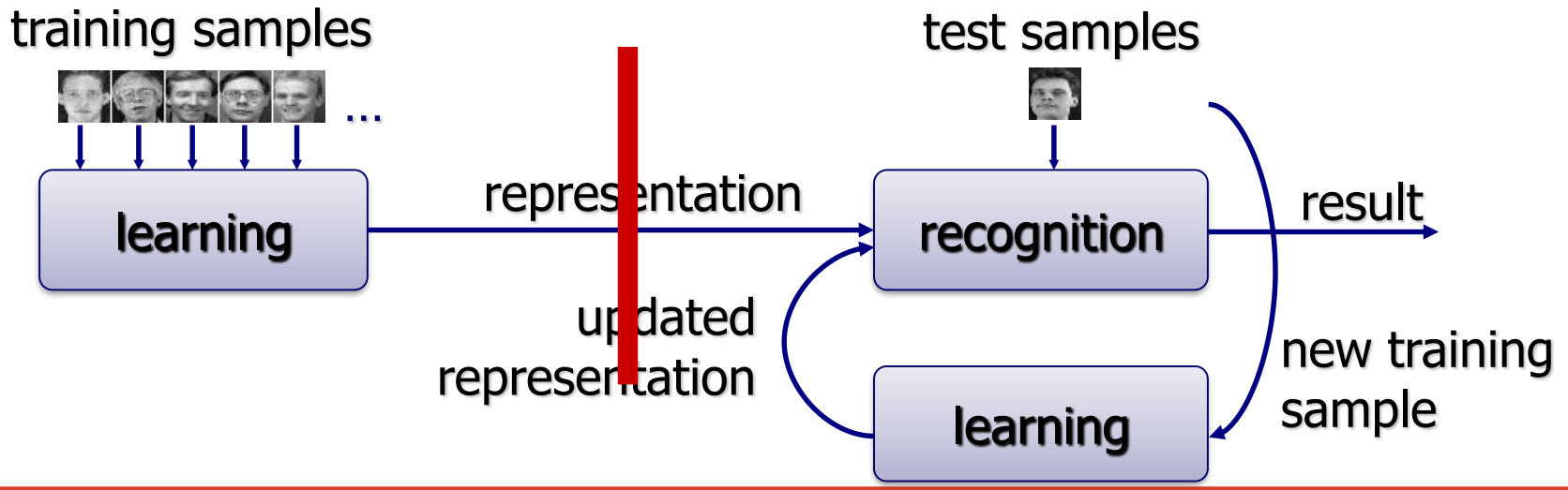
- Recognition of
  - objects
  - properties
  - faces
  - rooms
  - affordances
  - actions
  - speech
  - relations
  - intentions,…
- Categorisation
- Multimodal recognition



Learning

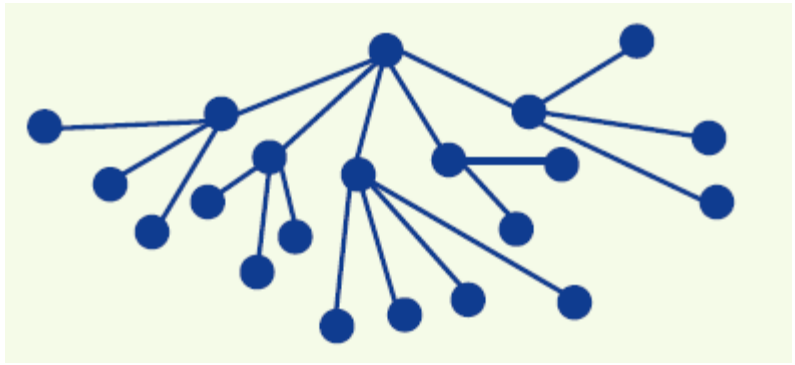
- Buildnig representations
- Continuous learning
- Different learning modes
- Multimodal learning
- Forgetting, unlearning
- Robustness
- Nature:nurture

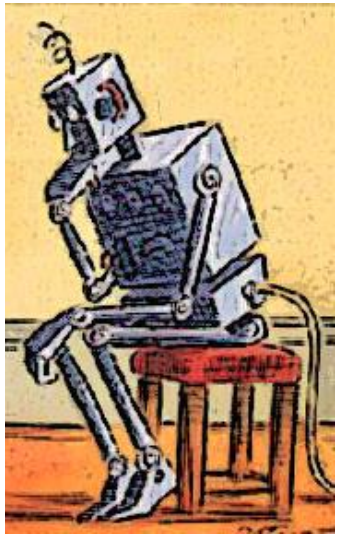




Reasoning, planning, decision making

- In unpredictable environment
- With incomplete information
- With robot limitations
- In dynamic environment
- Considering different modalities
- In real time





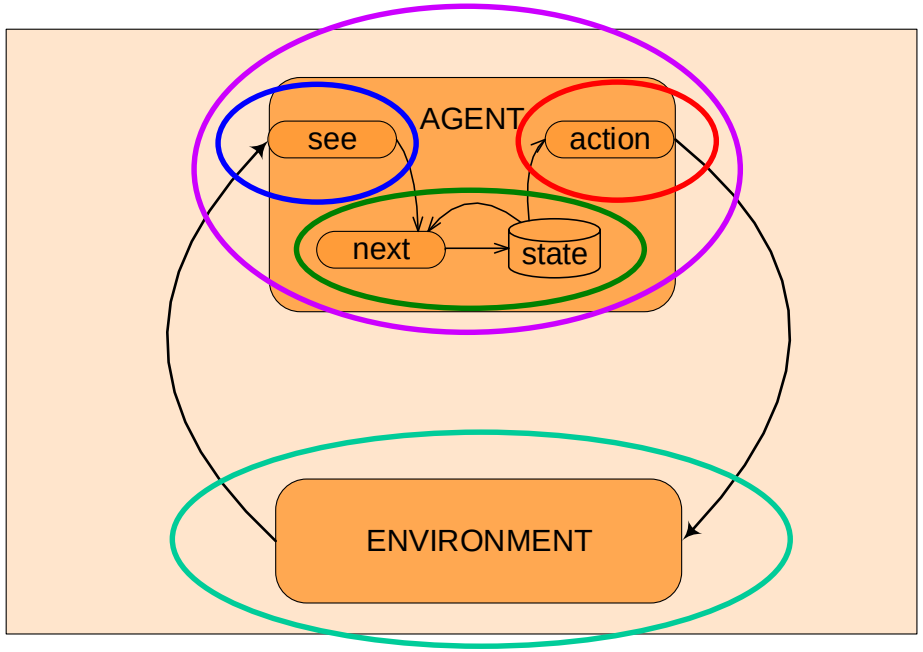
An example of a cognitive system

- Autonomous car
- City drive
- Competencies
  - Perception (image, 3D, collision)
  - Planning
  - Reasoning
  - Learning
  - Navigation
  - Obstacle avoidance
  - Action
  - Flexibility
  - Robustness
  - Efficiency
  - …

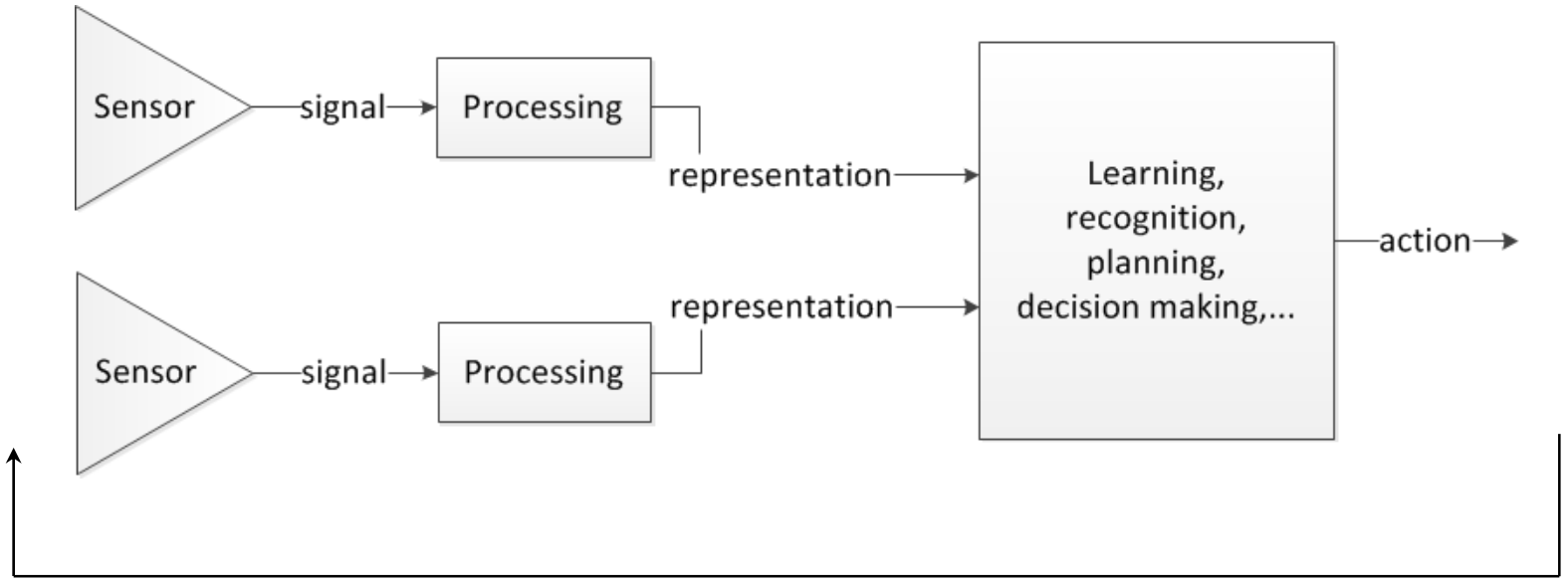
Google self-driving car

Intelligent agents

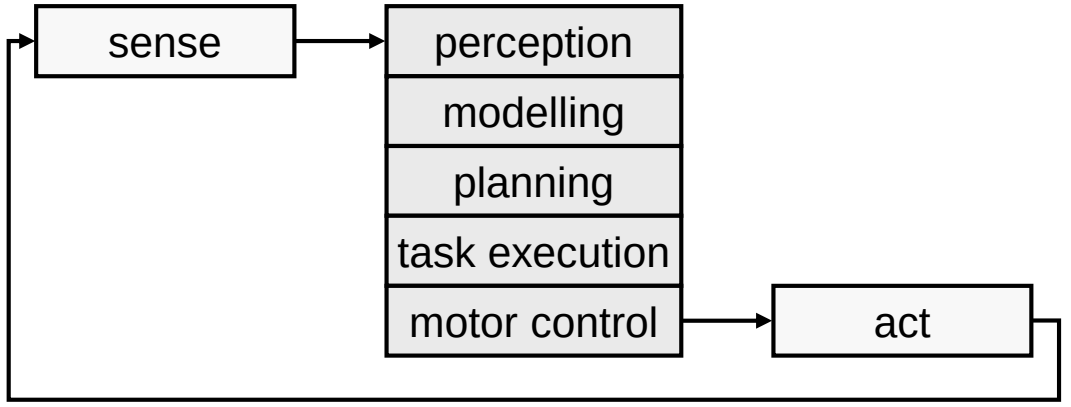
- Perception
- Action
- Reasoning, planning, decision making
- Autonomy
- Environment



Perception action cycle



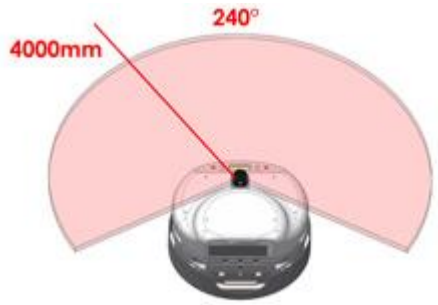
- Significant abstraction of the real world

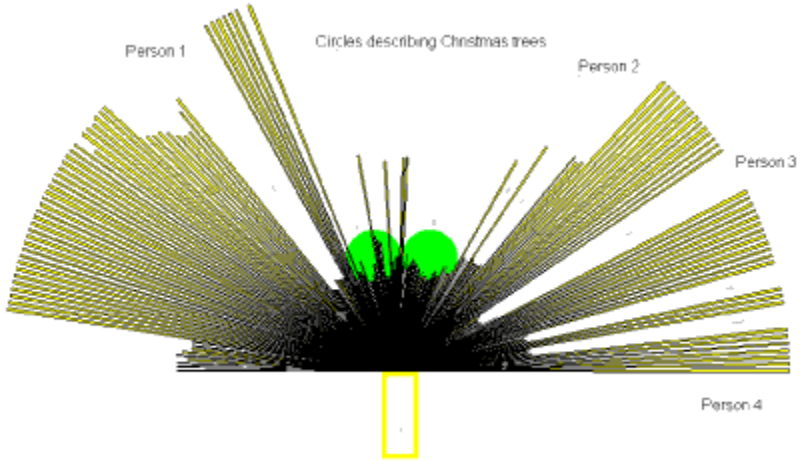


Simulation of robot perception and control

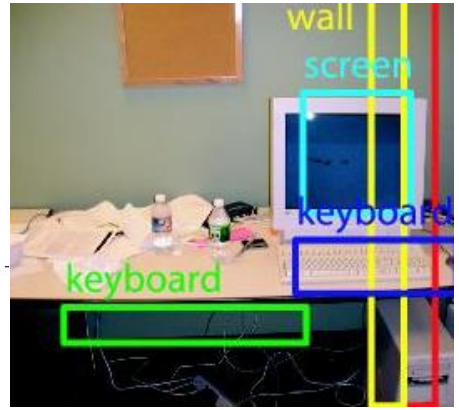
Sensors

- Range sensors





- Object recognition

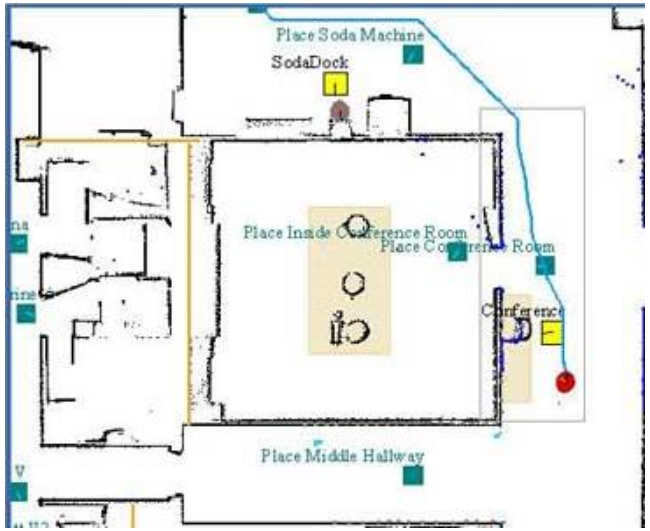


- Bumper – collision detector

- Odometer

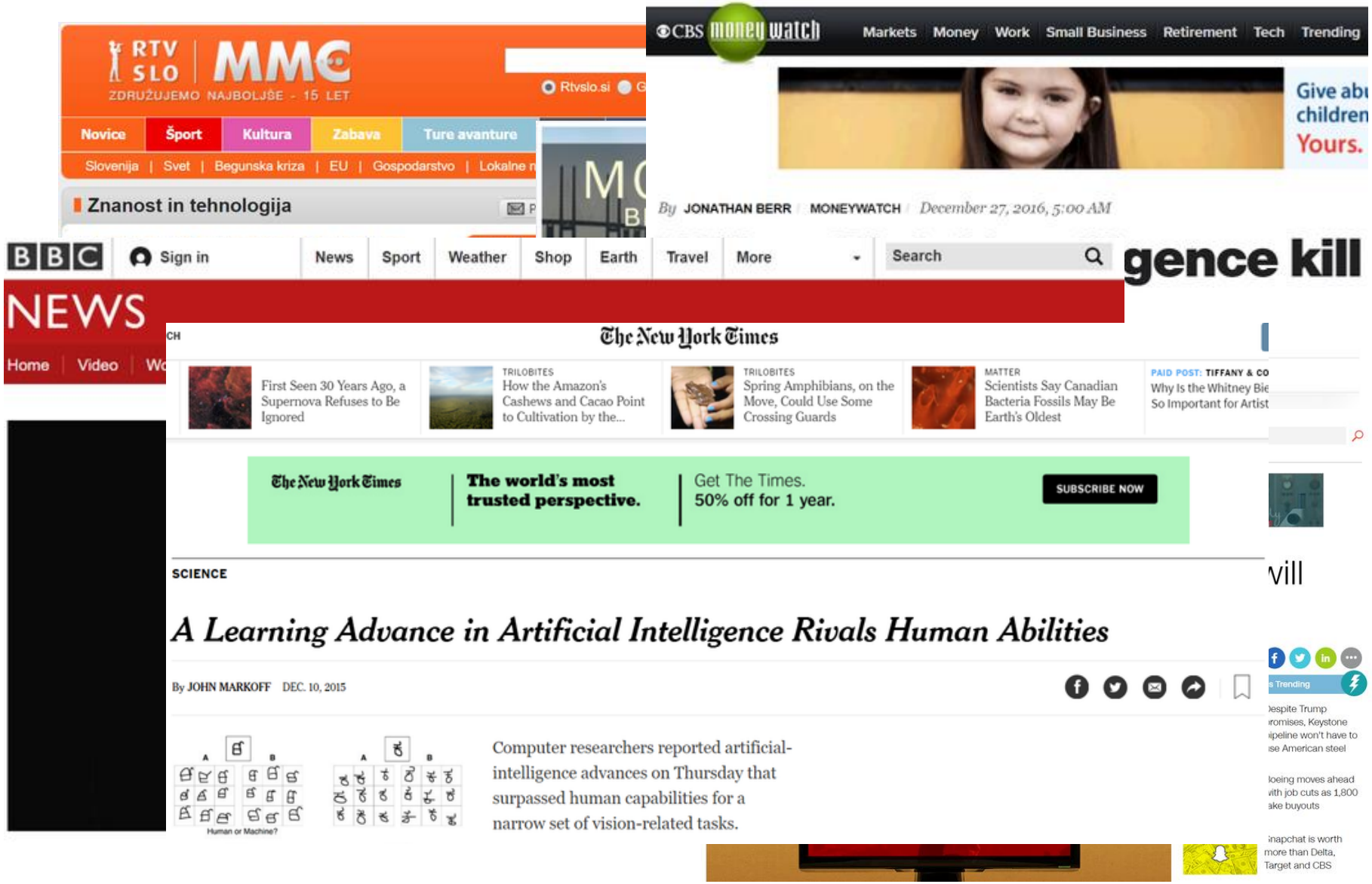
Planning and control

- Planning



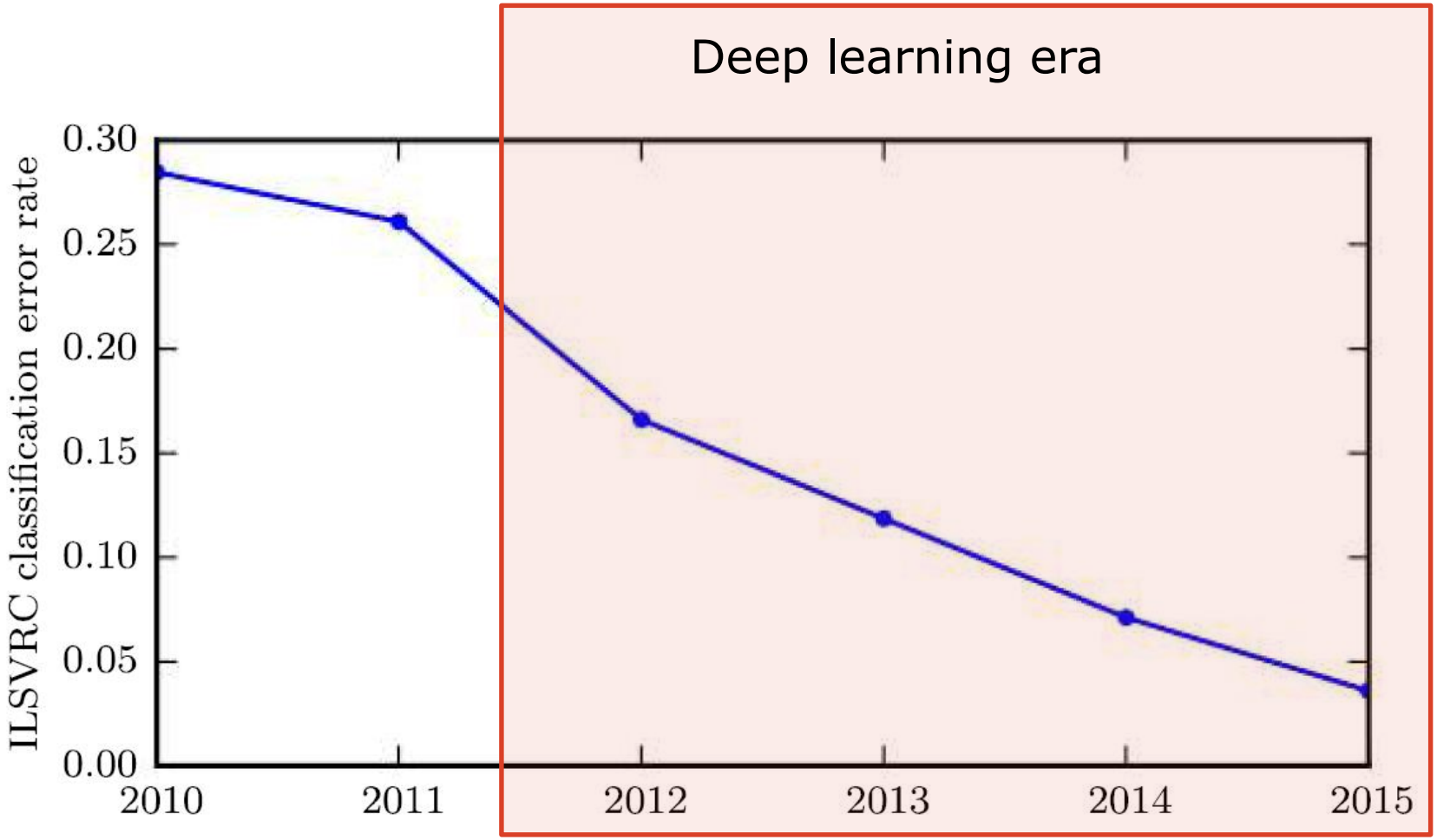
- Control

Deep learning



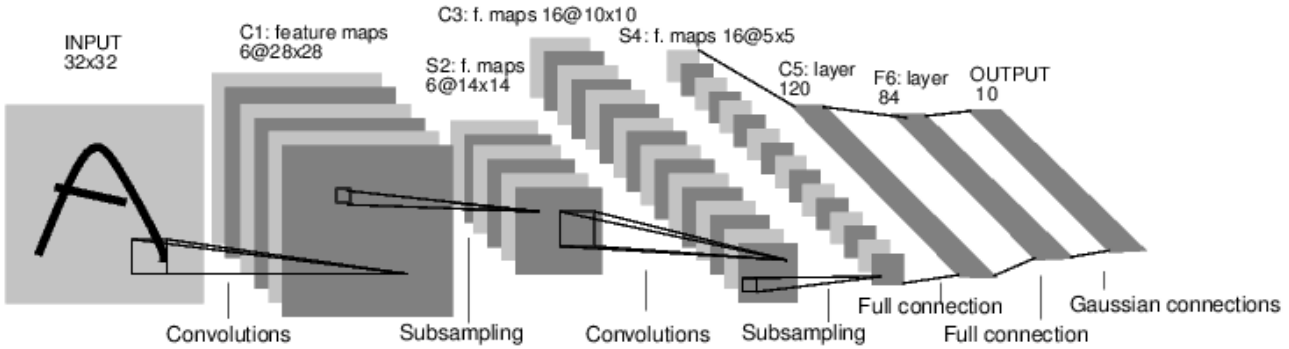
Excellent results

- ILSVRC results

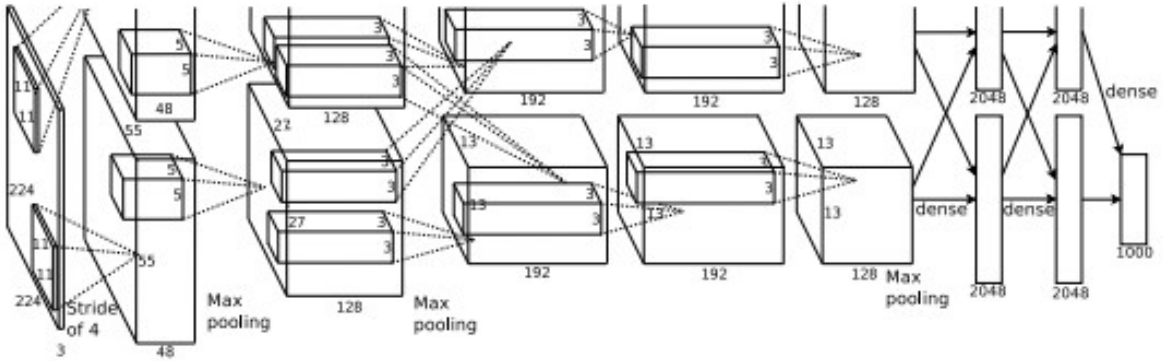


Modern deep learning

- More data!

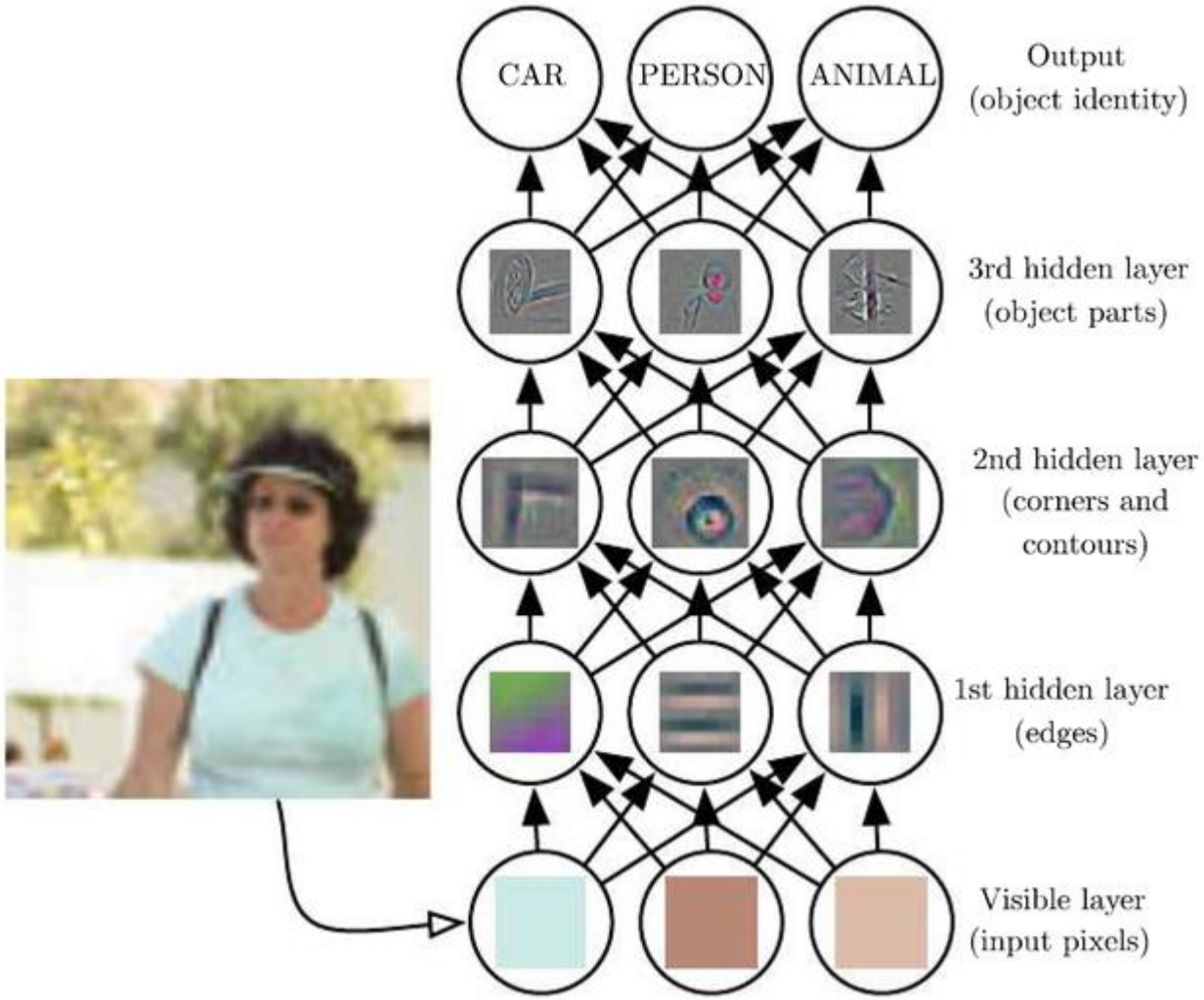


- More computational power!



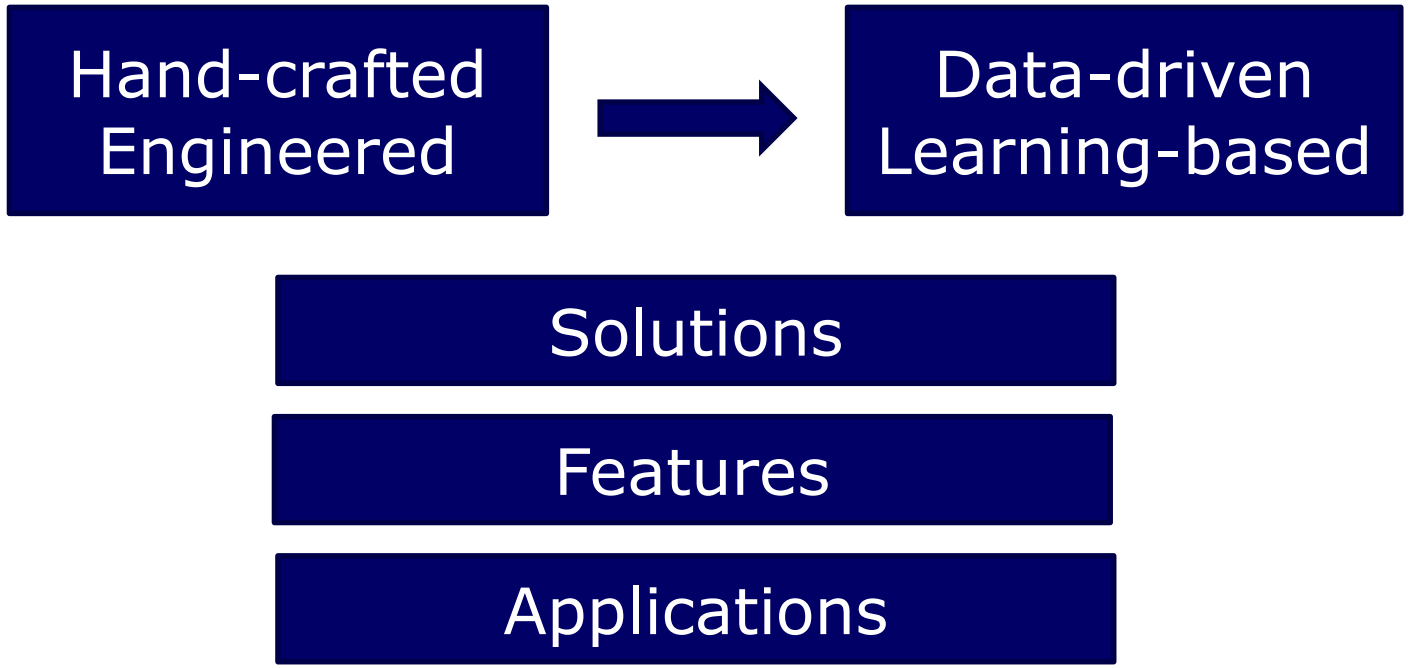
- Improved learning details!

The main concept



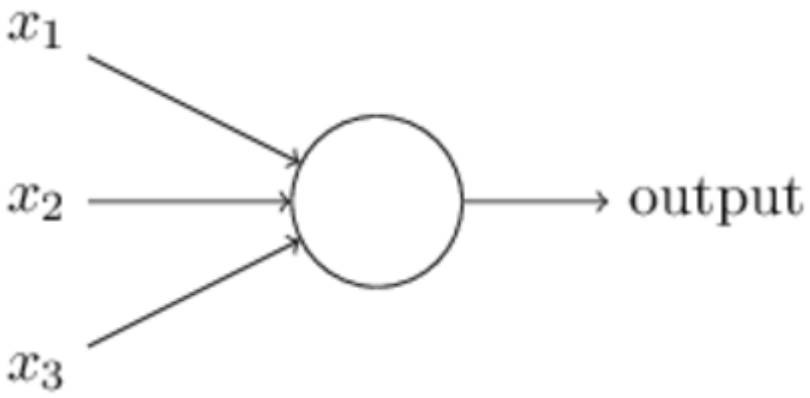
Zelier and Fergus, 2014

Shift in paradigm



Perceptron

- Rosenblatt, 1957
- Binary inputs and output
- Weights
- Threshold
- Bias
- Very simple!

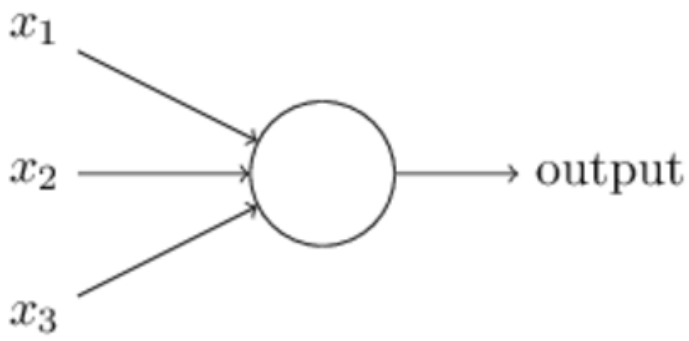


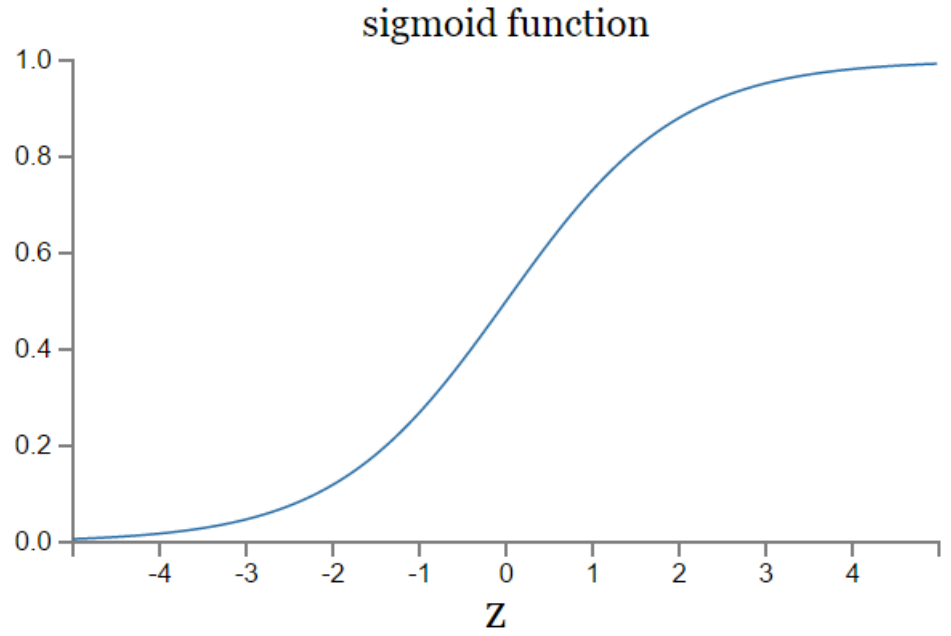
$$\text{output} = \begin{cases} 0 & \text{if } \sum_j w_j x_j \leq \text{ threshold} \\ 1 & \text{if } \sum_j w_j x_j > \text{ threshold} \end{cases}$$

$$\text{output} = \begin{cases} 0 & \text{if } w \cdot x + b \leq 0 \\ 1 & \text{if } w \cdot x + b > 0 \end{cases}$$

Sigmoid neurons

- Real inputs and outputs from interval [0,1]





- Activation function: sgimoid function

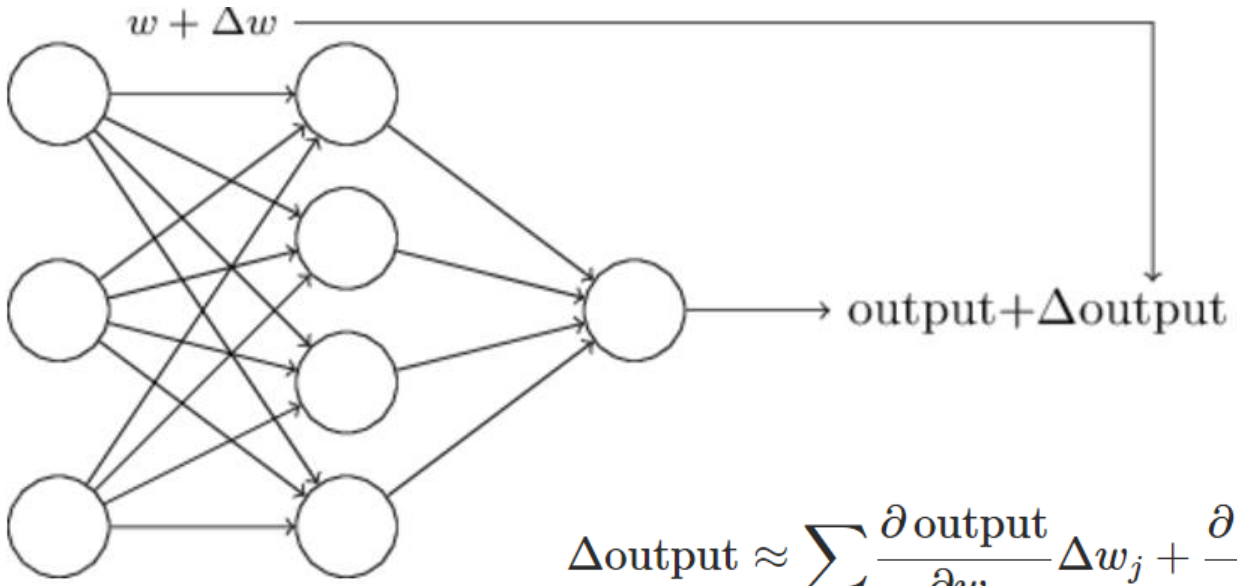
- *output* = $\dfrac{1}{1+\exp(-\sum_j w_j x_j - b)}$

$$\sigma(z) \equiv \frac{1}{1+e^{-z}}$$

$$\sigma(w \cdot x + b)$$

Sigmoid neurons

- Small changes in weights and biases causes small change in output

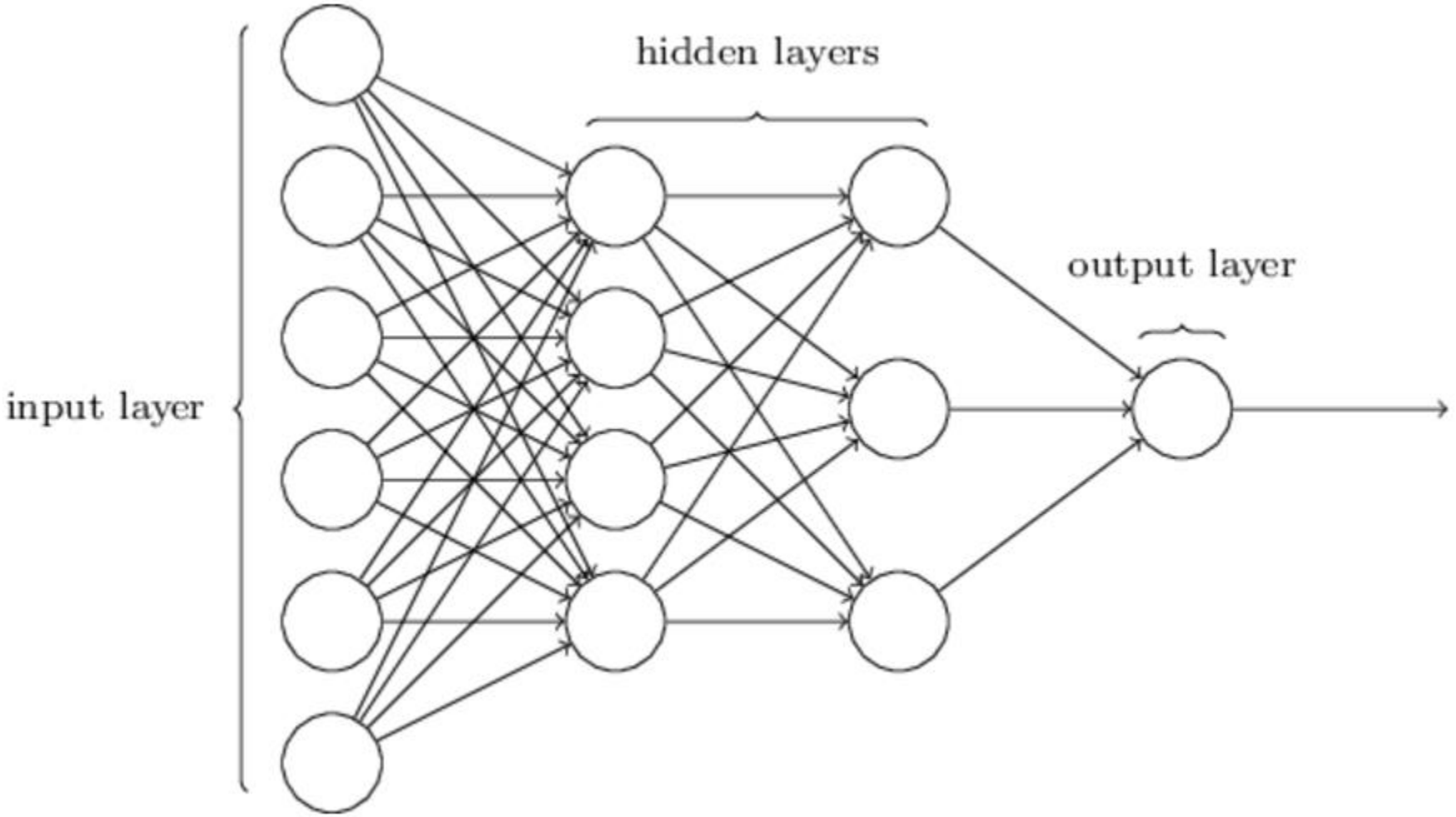


$$\Delta\text{output} \approx \sum_j \frac{\partial\,\text{output}}{\partial w_j}\Delta w_j + \frac{\partial\,\text{output}}{\partial b}\Delta b$$

- Enables learning!

Feedfoward neural networks

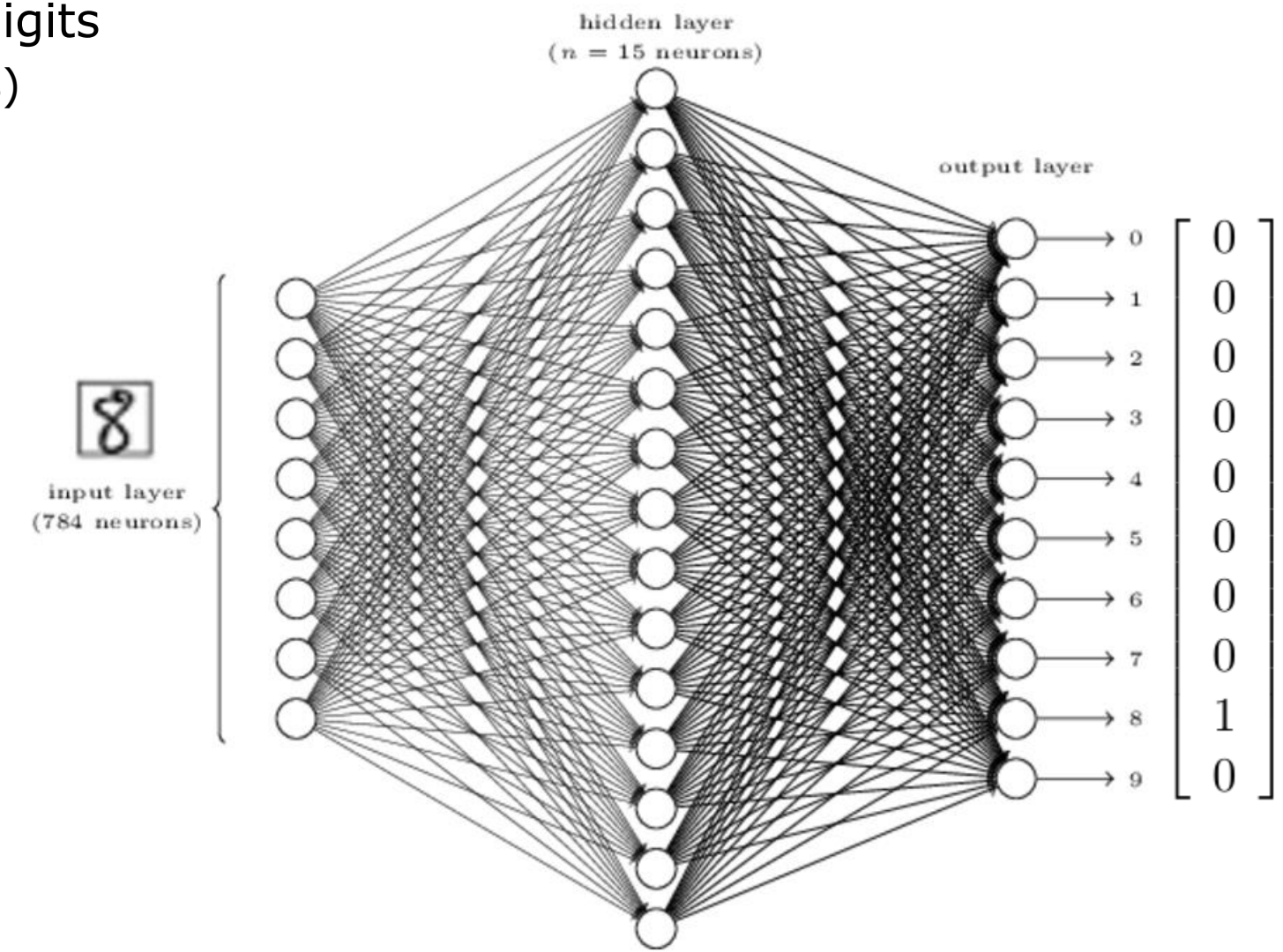
- Network architecture:



Example: recognizing digits

University of Ljubljana
Faculty of Computer and Information Science

- MNIST database of handwritten digits
  - 28x28 pixes (=784 input neurons)
  - 10 digits
  - 50.000 training images
  - 10.000 validation images
  - 10.000 test images



http://neuralnetworksanddeeplearning.com

Loss function

- Given:

$$y\left(\text{  }\right) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

for all training images

- Loss function: $C(w,b) \equiv \frac{1}{2n} \sum_x \|y(x) - a\|^2$
  - (mean sqare error – quadratic loss function)

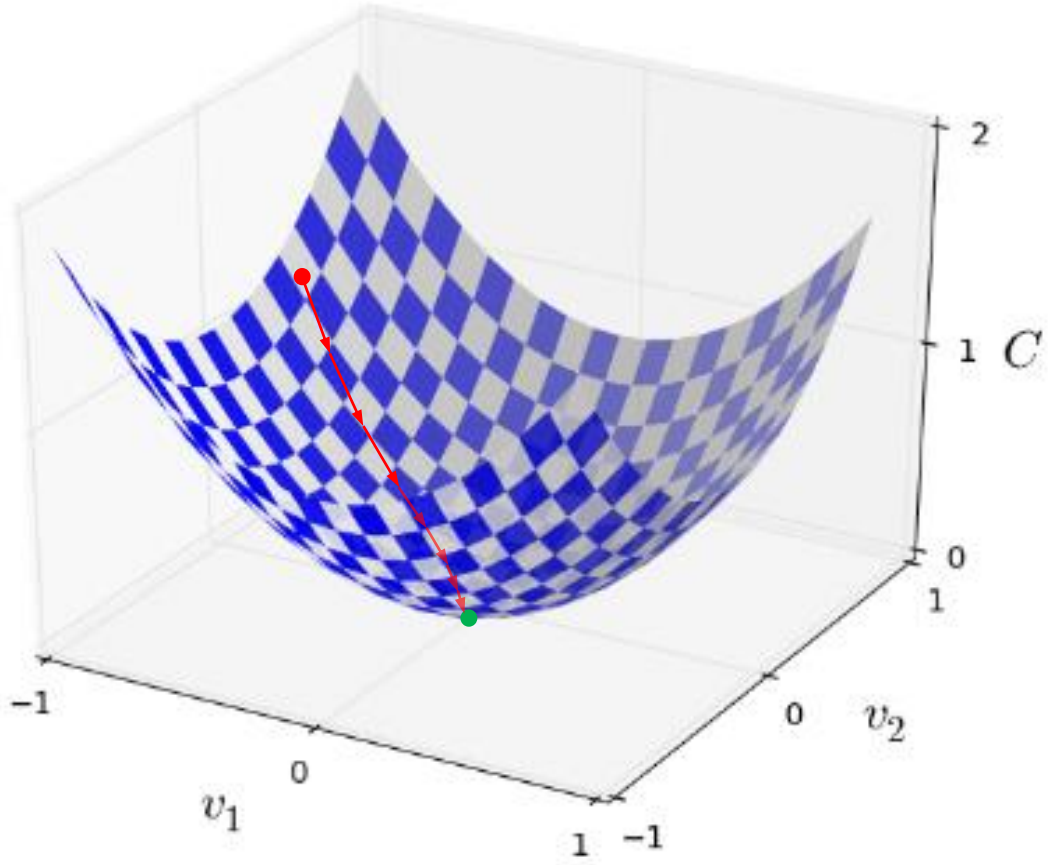
- Find weigths w and biases b that for given input x produce output a that minimizes Loss function C

Gradient descend

- Find minimum of $C(v_1, v_2)$
- Change of C: $\Delta C \approx \frac{\partial C}{\partial v_1}\Delta v_1 + \frac{\partial C}{\partial v_2}\Delta v_2 = \nabla C \cdot \Delta v = -\eta \|\nabla C\|^2$
- Gradient of C: $\nabla C \equiv \left(\frac{\partial C}{\partial v_1}, \frac{\partial C}{\partial v_2}\right)^T$
- *Change v in the opposite direction of the gradient:* $\Delta v = -\eta \nabla C$

Learning rate

- Algorithm:
  - Initialize v
  - Until stopping criterium riched
    - Apply udate rule $v \to v' = v - \eta \nabla C.$



Gradient descend in neural networks

- Loss function $C(w, b)$

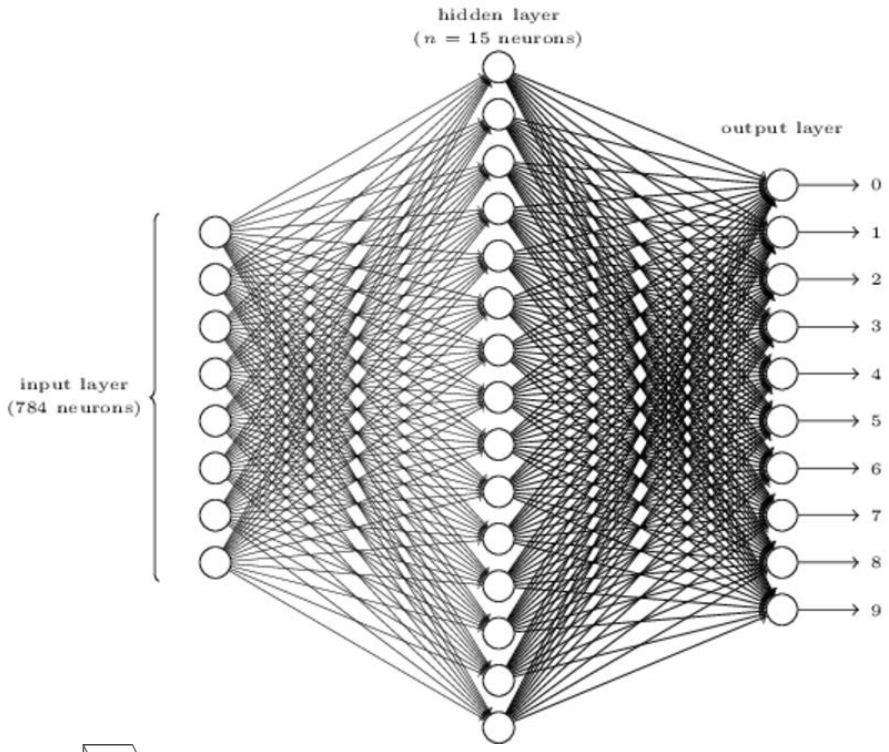
- Update rules:

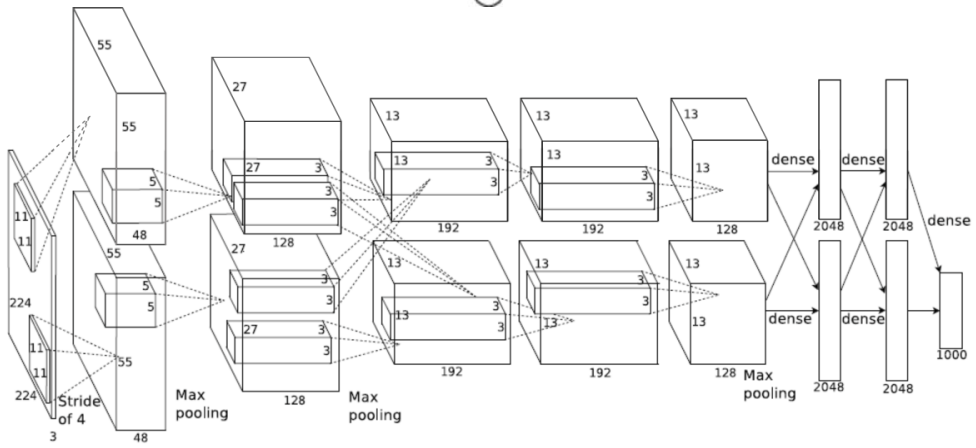
$$w_k \to w'_k = w_k - \eta \frac{\partial C}{\partial w_k}$$

$$b_l \to b'_l = b_l - \eta \frac{\partial C}{\partial b_l}$$

- Consider all training samples
- Very many parameters
  => computationaly very expensive

- Use Stochastic gradient descend instead
  - Compute gradient only for a subset of m training samples





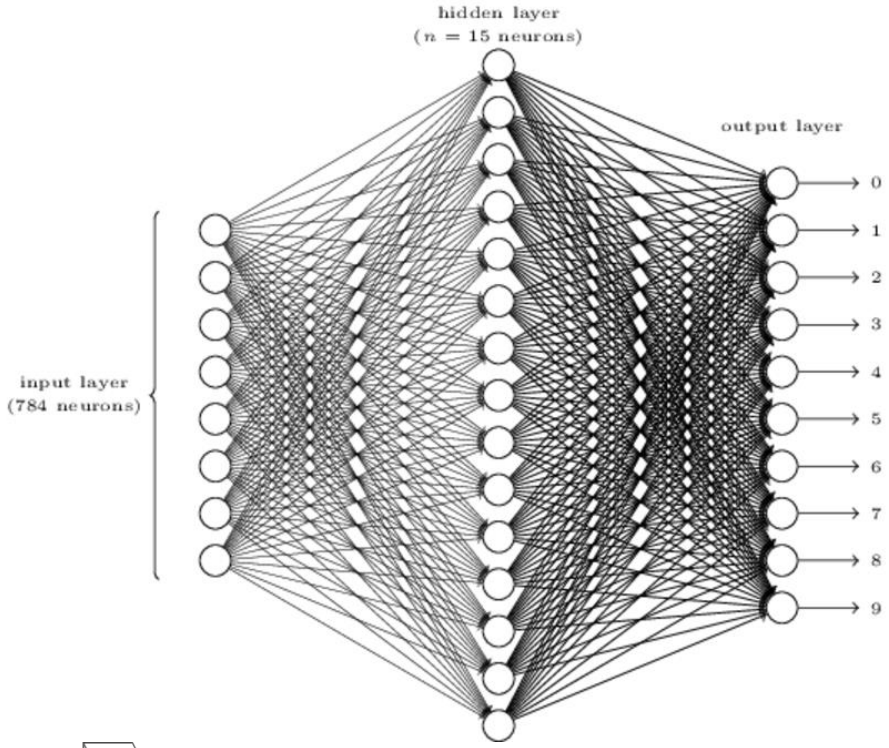
Backpropagation

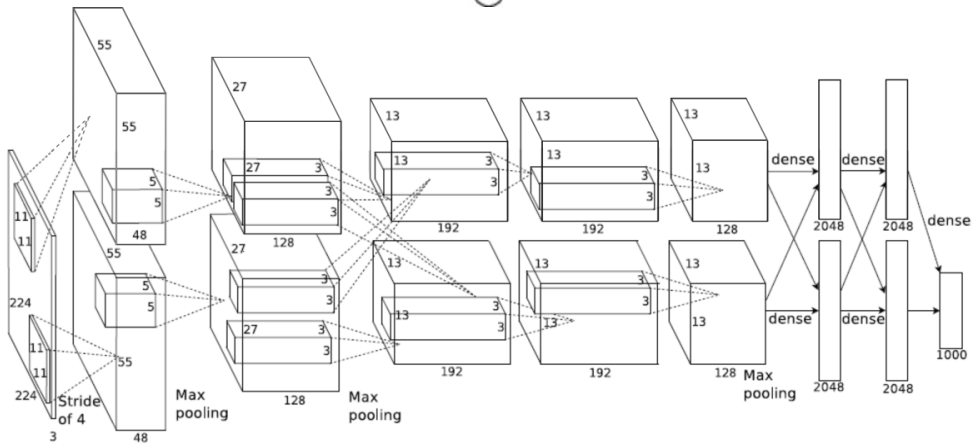
- All we need is gradient of loss function ∇C
  - Rate of change of C wrt. to change in any weigt
  - Rate of change of C wrt. to change in any biase

$$\frac{\partial C}{\partial b_j^l} \qquad \frac{\partial C}{\partial w_{jk}^l}$$

- How to compute gradient?
  - Numericaly
    - Simple, approximate, extremely slow ☹
  - Analyticaly for entire C
    - Fast, exact, nontractable ☹
  - Chain individual parts of netwok
    - Fast, exact, doable ☺

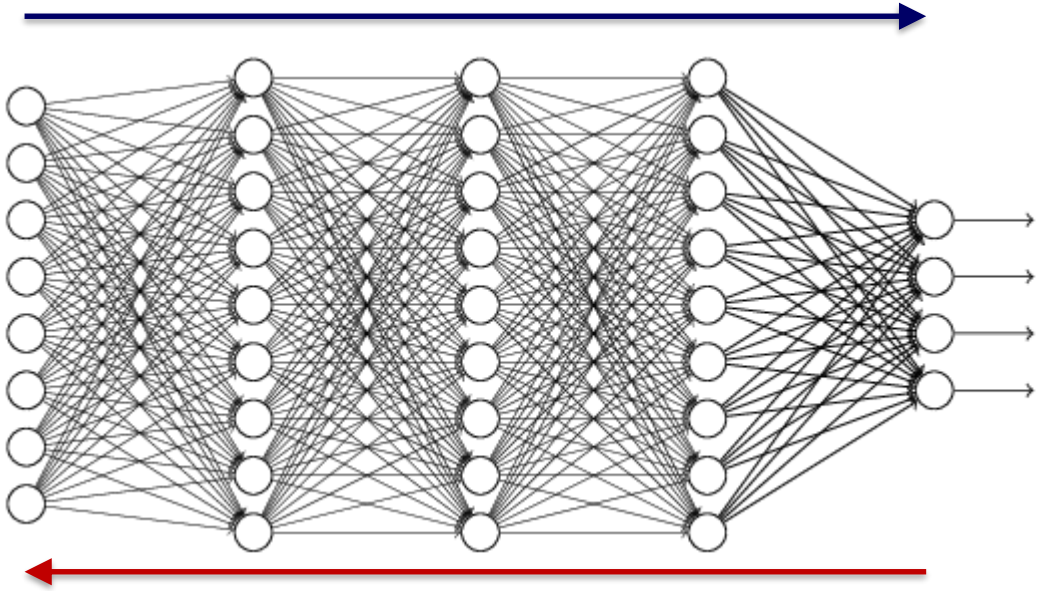
Backpropagation!





Main principle

- We need the gradient of the Loss function ∇C $\frac{\partial C}{\partial b^l_j}$ $\frac{\partial C}{\partial w^l_{jk}}$
- Two phases:
  - Forward pass; propagation: the input sample is propagated through the network and the error at the final layer is obtained



  - Backward pass; weight update: the error is backpropagated to the individual levels, the contribution of the individual neuron to the error is calculated and the weights are updated accordingly

Backpropagation and SGD

For a number of **epochs**

Until all training images are used

Select a **mini-batch** of m training samples

For each training sample x in the mini-batch

Input: set the corresponding activation $a^{x,1}$

Feedforward: for each $l = 2, 3, \ldots, L$

compute $z^{x,l} = w^l a^{x,l-1} + b^l$ and $a^{x,l} = \sigma(z^{x,l})$

Output error: compute $\delta^{x,L} = \nabla_a C_x \odot \sigma'(z^{x,L})$

Backpropagation: for each $l = L-1, L-2, \ldots, 2$

compute $\delta^{x,l} = ((w^{l+1})^T \delta^{x,l+1}) \odot \sigma'(z^{x,l})$

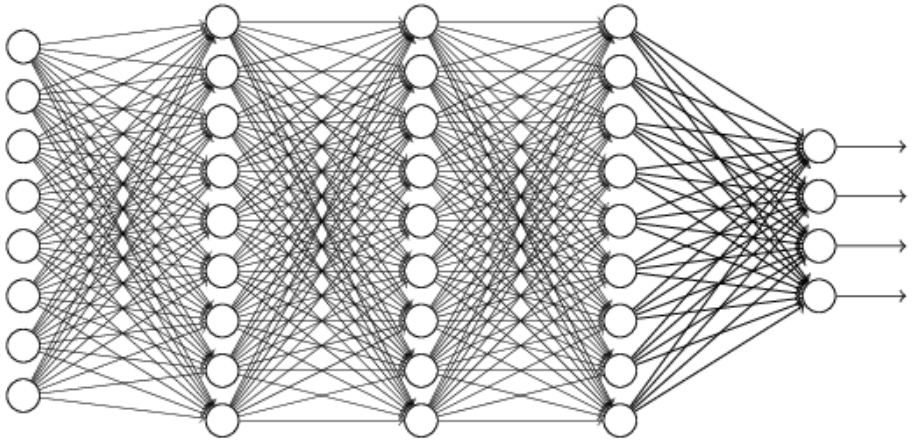
Gradient descend: for each $l = L, L-1, \ldots, 2$ and x update:

$$w^l \to w^l - \frac{\eta}{m} \sum_x \delta^{x,l} (a^{x,l-1})^T$$

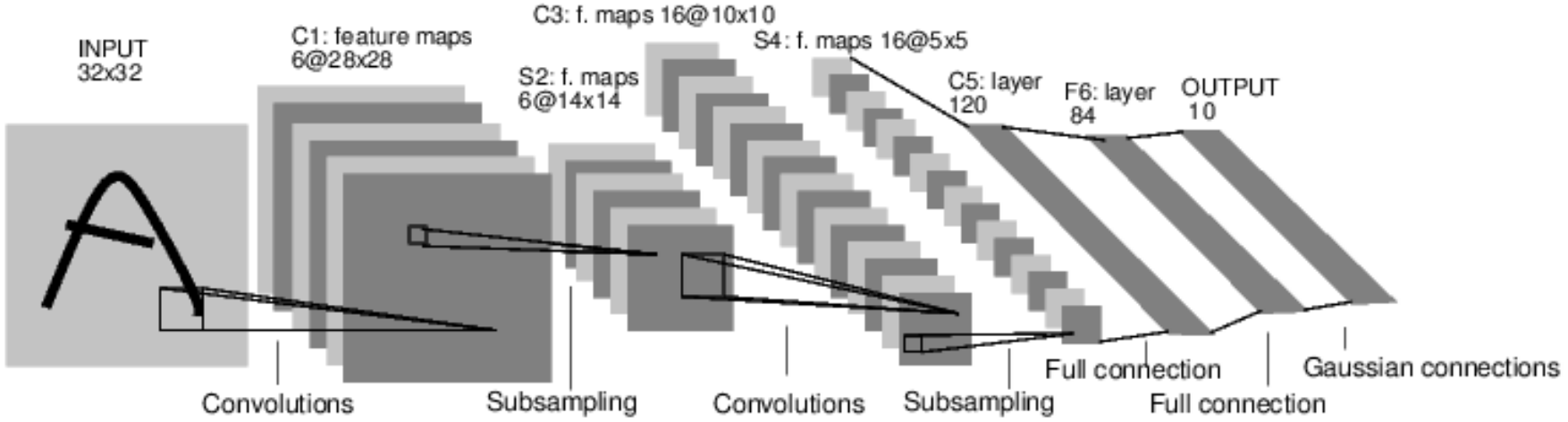
$$b^l \to b^l - \frac{\eta}{m} \sum_x \delta^{x,l}$$

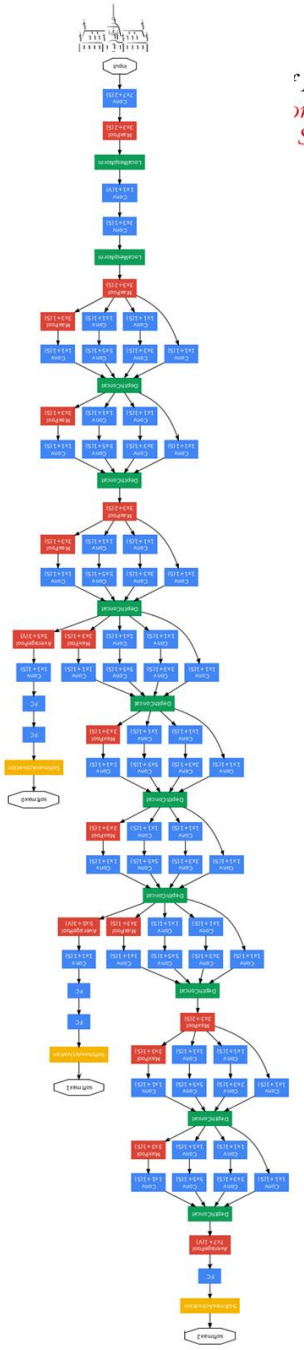
Convolutional neural networks

- From feedforward fully-connected neural networks



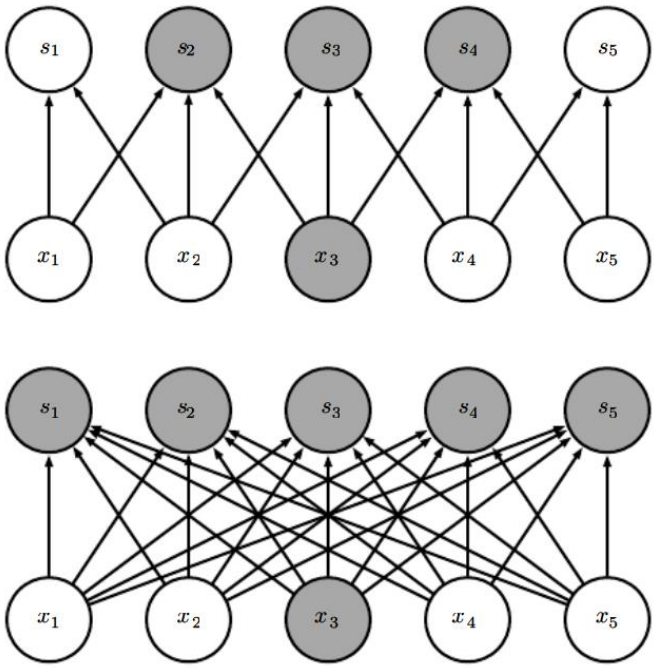
- To convolutional neural networks



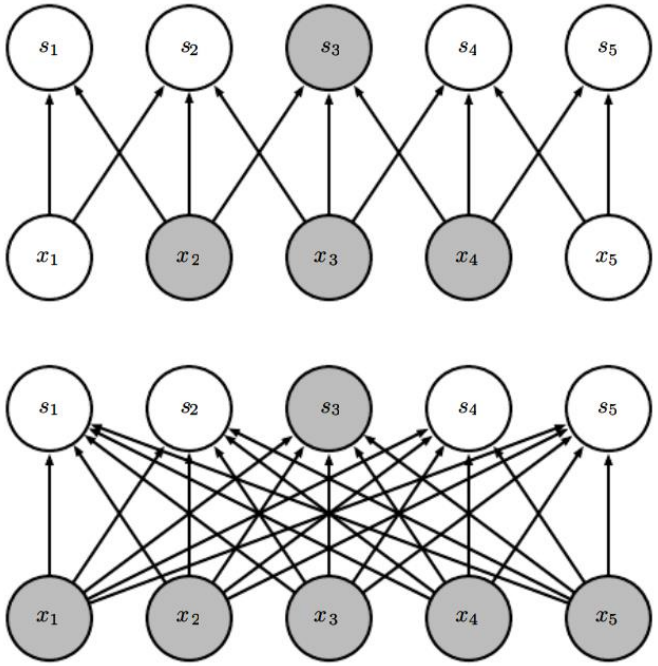


Sparse connectivity

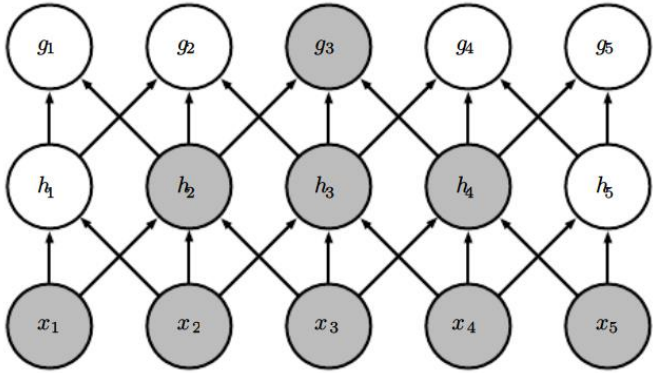
- Local connectivity – neurons are only locally connected (**receptive field**)
  - Reduces memory requirements
  - Improves statistical efficiency
  - Requires fewer operations



from below



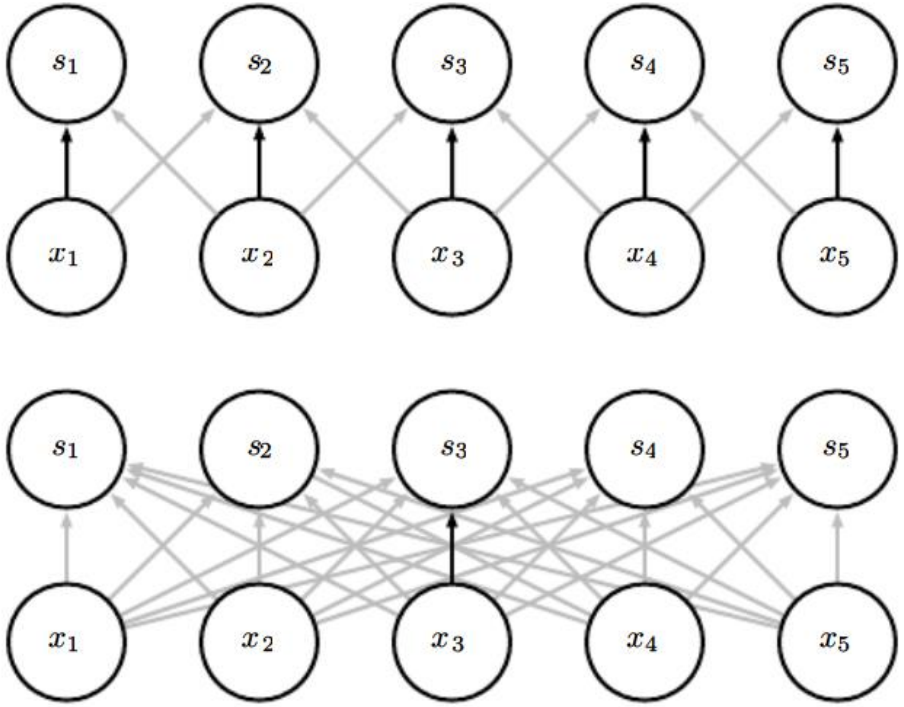
from above



The receptive field of the units in the deeper layers is large
=> Indirect connections!

Parameter sharing

- **Neurons share weights!**
  - Tied weights
- Every element of the kernel is used at every position of the input
- All the neurons at the same level detect the same feature (everywhere in the input)
- Greatly reduces the number of parameters!



- **Equivariance to translation**
  - Shift, convolution = convolution, shift
  - Object moves => representation moves

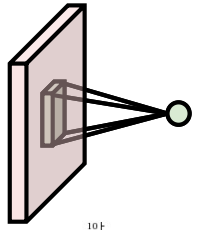
- Fully connected network with an infinitively strong prior over its weights
  - Tied weights
  - Weights are zero outside the kernel region

  => learns only local interactions and is equivariant to translations

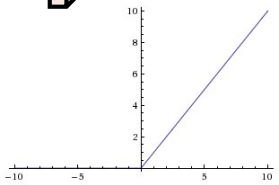
CNN layers

- Layers used to build ConvNets:
  - INPUT:
    raw pixel values

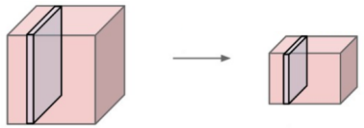
  - CONV:
    convolutional layer



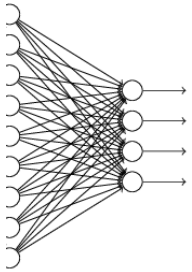
  - ReLU:
    introducing nonlinearity

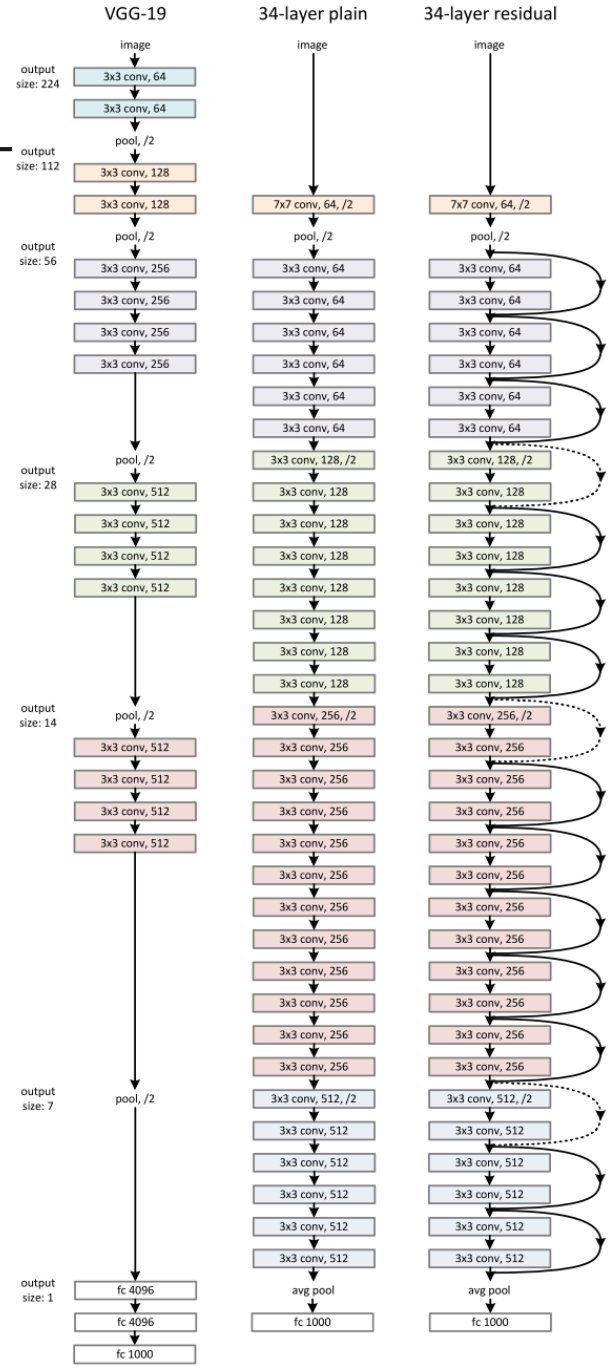


  - POOL:
    downsampling



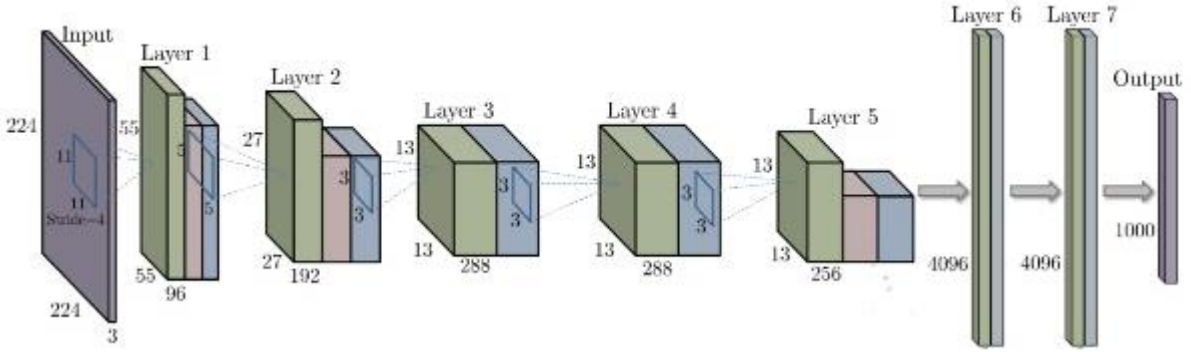
  - FC:
    for computing class scores



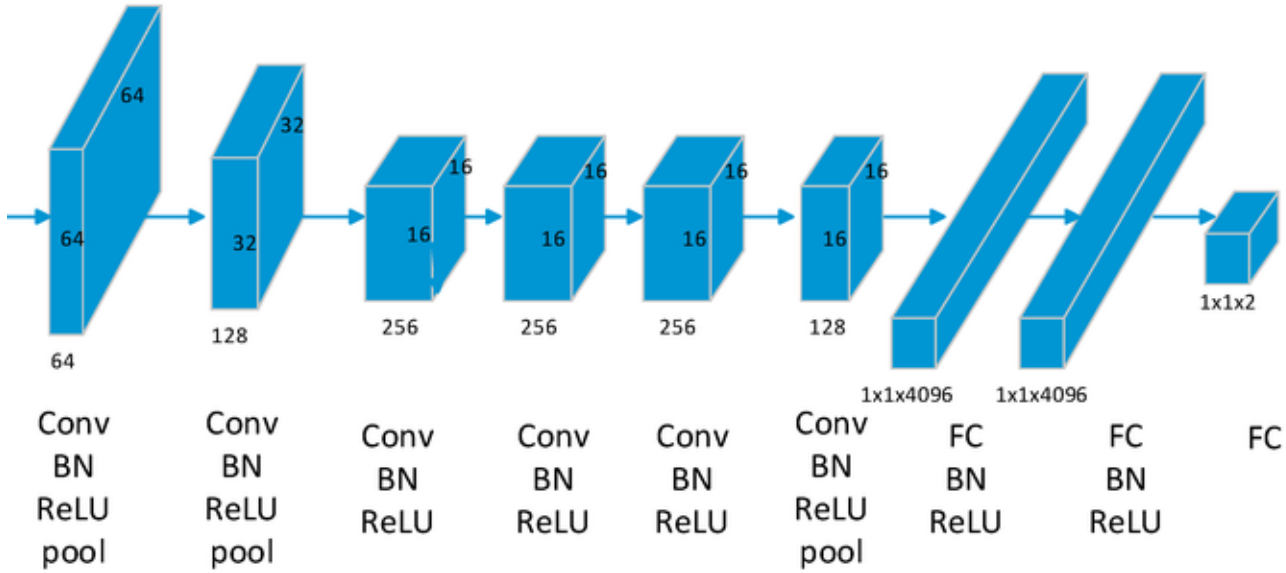


CNN architecture

- Stack the layers in an appropriate order



Babenko et. al.



Hu et. al.

Detection of traffic signs

University of Ljubljana
Faculty of Computer and Information Science

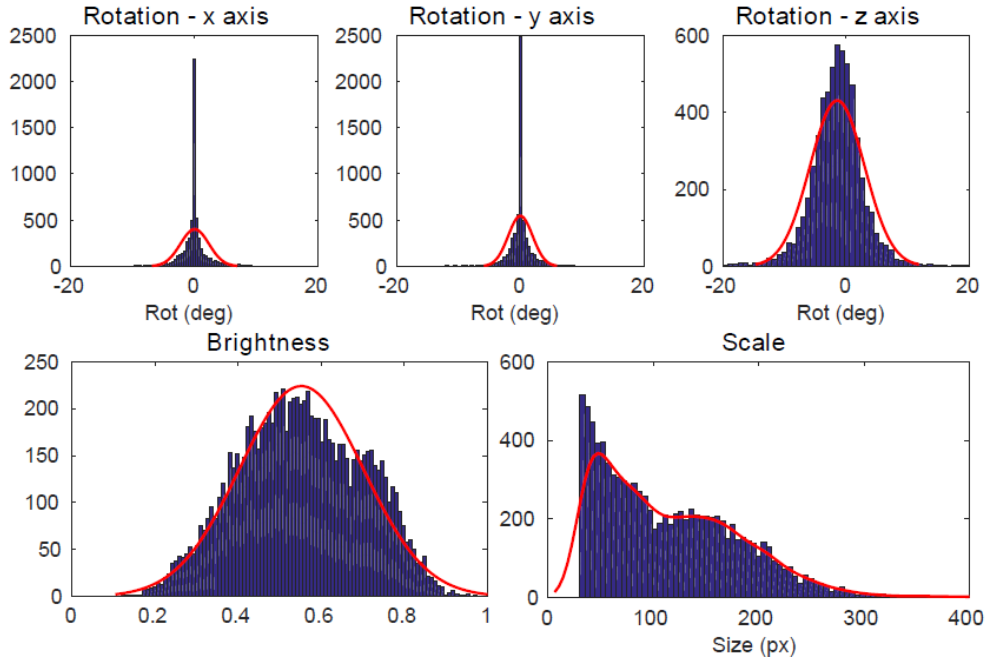
- DFG database
- 200 categories

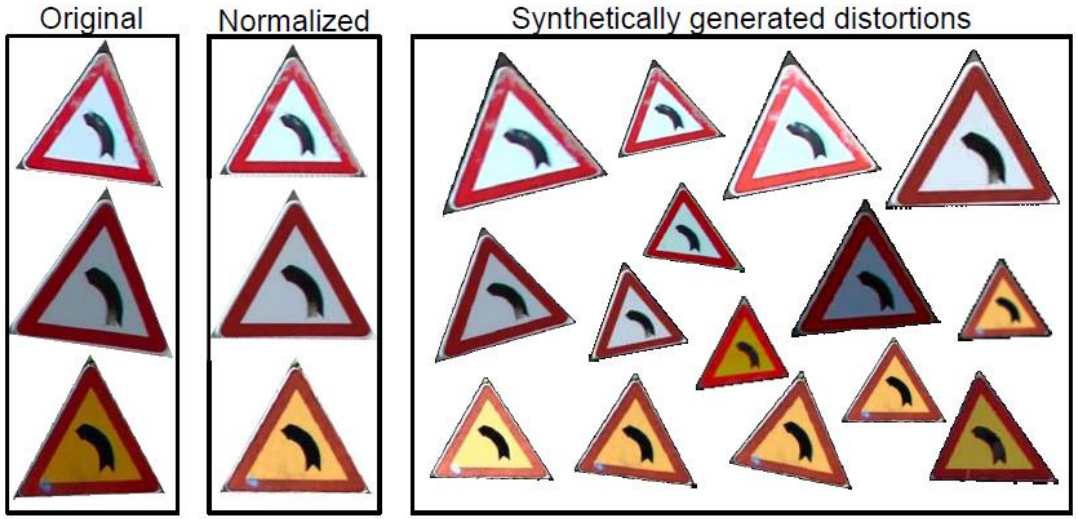
Tabernik, Skočaj, Deep Learning for for Large-Scale Traffic Sign Detection and Recognition, submitted

ViCOS visual cognitive systems lab

Detection of traffic signs

- Data augmentation





- Mask R-CNN +
  - Online hard-example mining
  - Distribution of selected training samples
  - Sample weighting
  - Adjusting region pass-through during detection

Experimental results

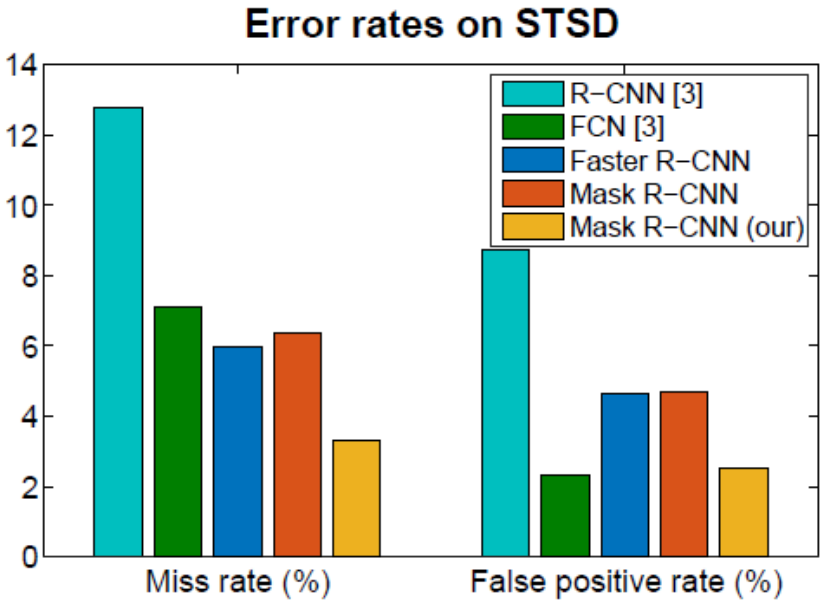
University of Ljubljana
Faculty of Computer and Information Science

- Swedish traffic sign database

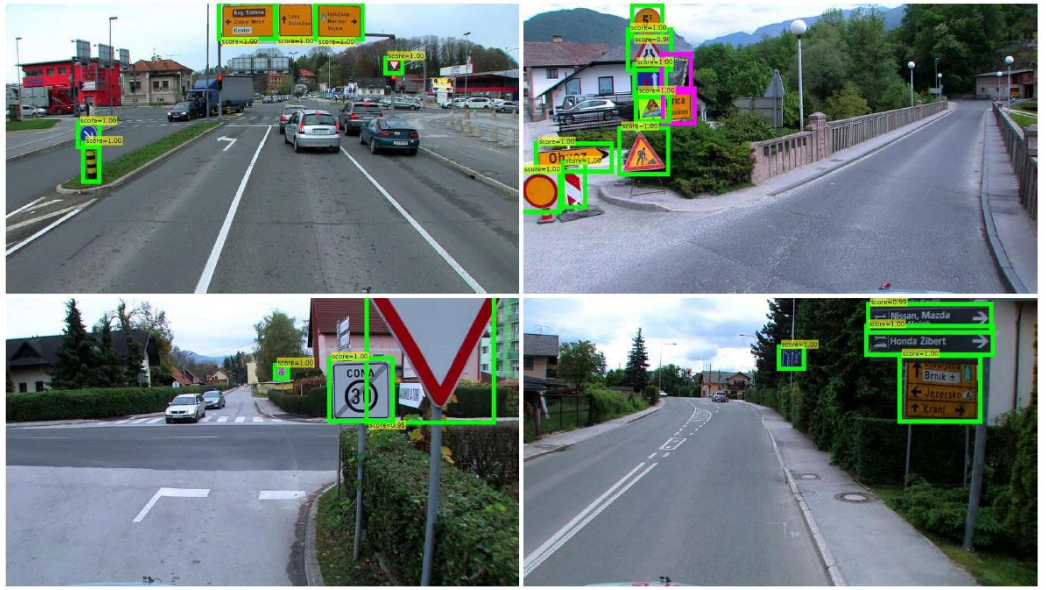
Average	R-CNN [6]	FCN [6]	Faster R-CNN	Mask R-CNN (ResNet-50)	
				No adapt.	Adapt. (ours)
Precision	91.2	**97.7**	95.4	95.3	97.5
Recall	87.2	92.9	94.0	93.6	**96.7**
F-measure	88.8	95.0	94.6	93.8	**97.0**
mAP^{50}	/	/	94.3	94.9	**95.2**

DFG traffic sign database

	Faster R-CNN	Mask R-CNN (ResNet-50)		
		No adapt.	With adapt.	With adapt. and data augment.
mAP^{50}	92.4	93.0	95.2	**95.5**
$mAP^{50:95}$	80.4	82.3	82.0	**84.4**
Max recall	93.8	94.6	**96.5**	**96.5**

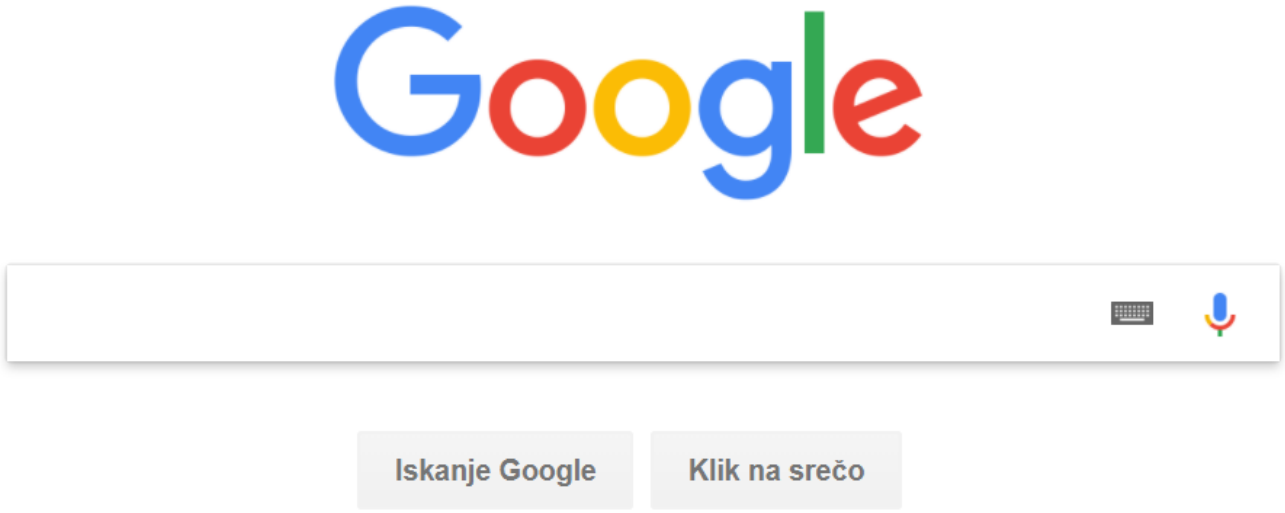


ViCOS visual cognitive systems lab



Traffic sign detection

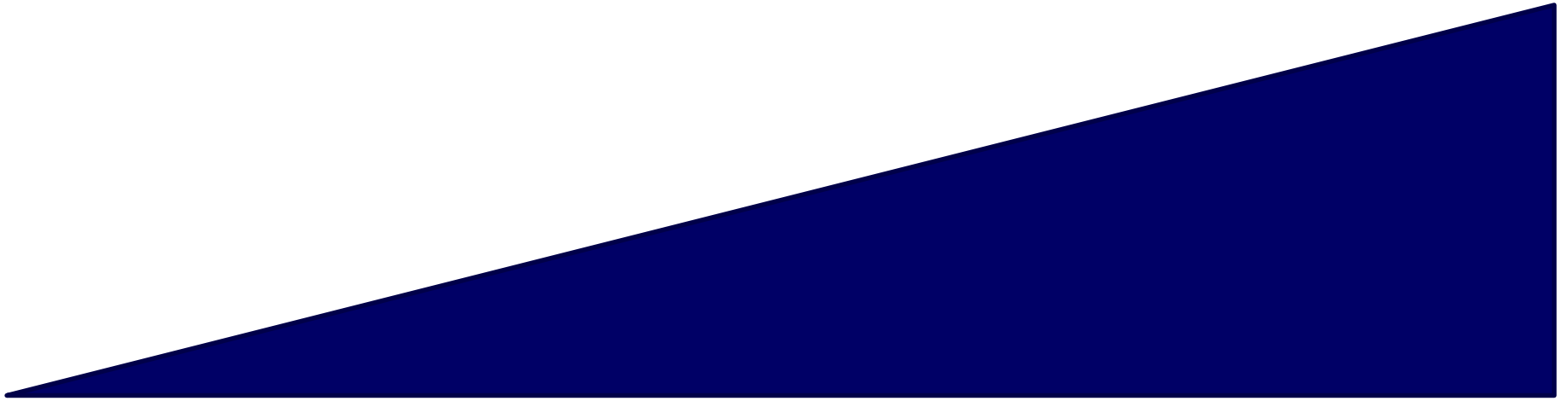
Knowing everything



- Data, big data!
- Machine learning!
- Make sense of huge amount of data!

Digitalisation of decision making

- Different problem complexities



Simple,
well defined problems

Rule-based
decision making

Programming

Complex, vaguely
defined problems

Data-driven
decision making

Machine learning

Digital decision making

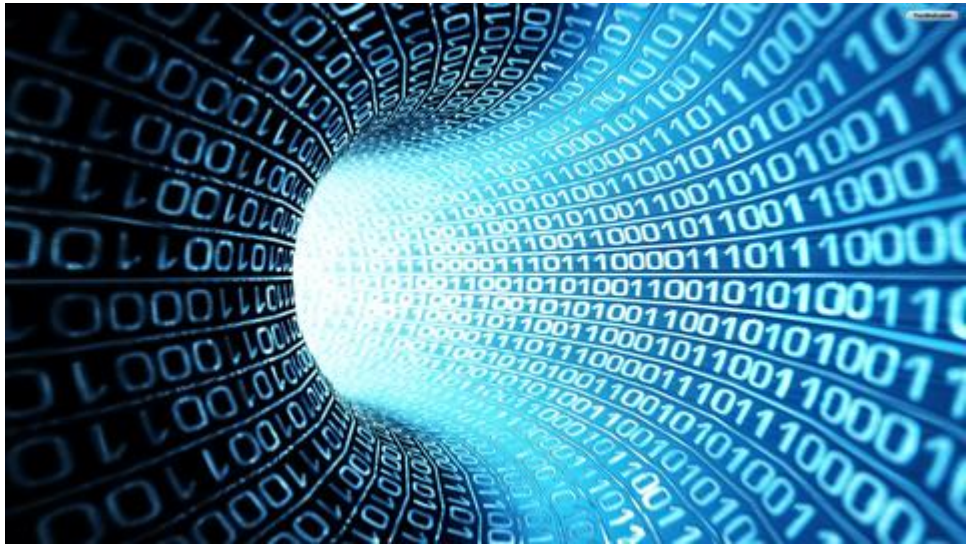
University of Ljubljana
Faculty of Computer and Information Science

- Decision making based on history
- Decision making modelling based on previous decisions
- Explicit capturing + digitalisation of all important attributes

- Bias in favor of past decisions
- Size and representativeness of the training set

- Updating the model through time
- Combining the training model with new rules

- Explainabilty of decisions



Conclusion

T-60

T-30

T

T+30