



УНИВЕРСИТЕТ ИТМО

# Introduction to deep learning

Natalia Khanzhina

Senior Researcher

Machine Learning Lab

[nehanzhina@corp.ifmo.ru](mailto:nehanzhina@corp.ifmo.ru)

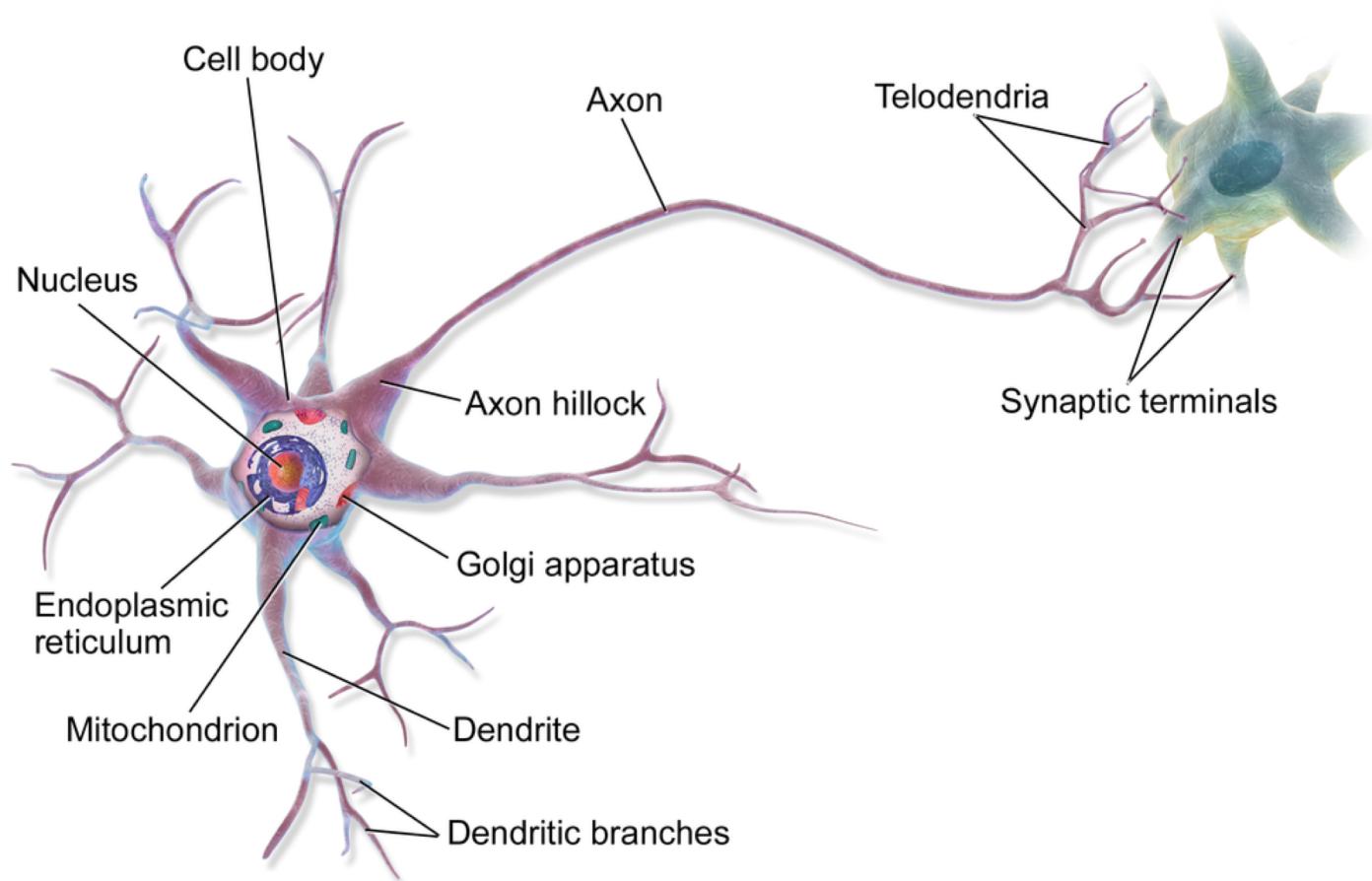
# Outline

- ✓ Neural nets
- ✓ Introduction to deep learning
- ✓ Convolutional Neural Network (CNN)
- ✓ DNN best practices
- ✓ DNN applications

# Outline

- ✓ Neural nets
- ✓ Introduction to deep learning
- ✓ Convolutional Neural Network (CNN)
- ✓ DNN best practices
- ✓ DNN applications

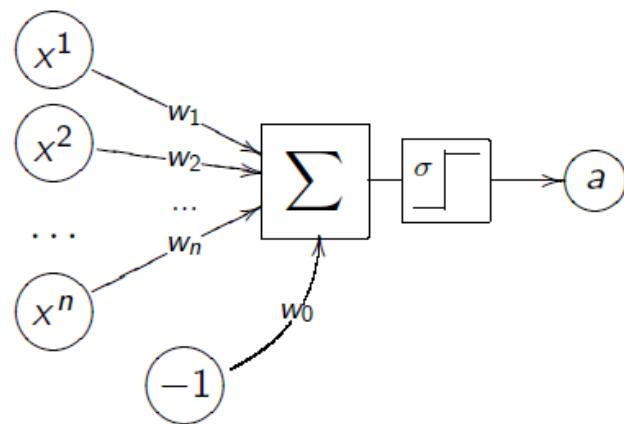
# Biological neuron



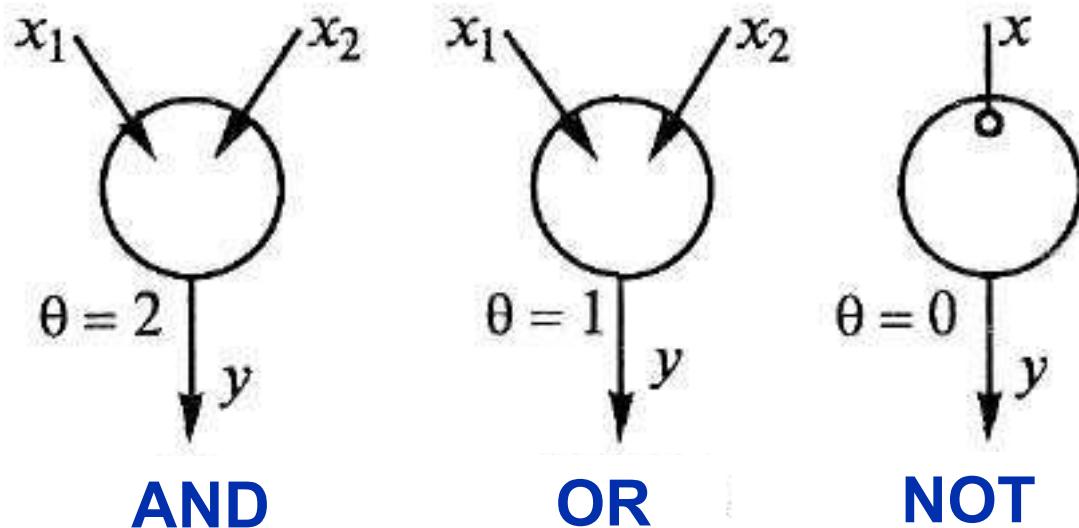
## Generalized McCulloch-Pitts neuron

$$y_w(x, T^m) = \sigma \left( \sum_{i=1}^n w_i x^i - w_0 \right),$$

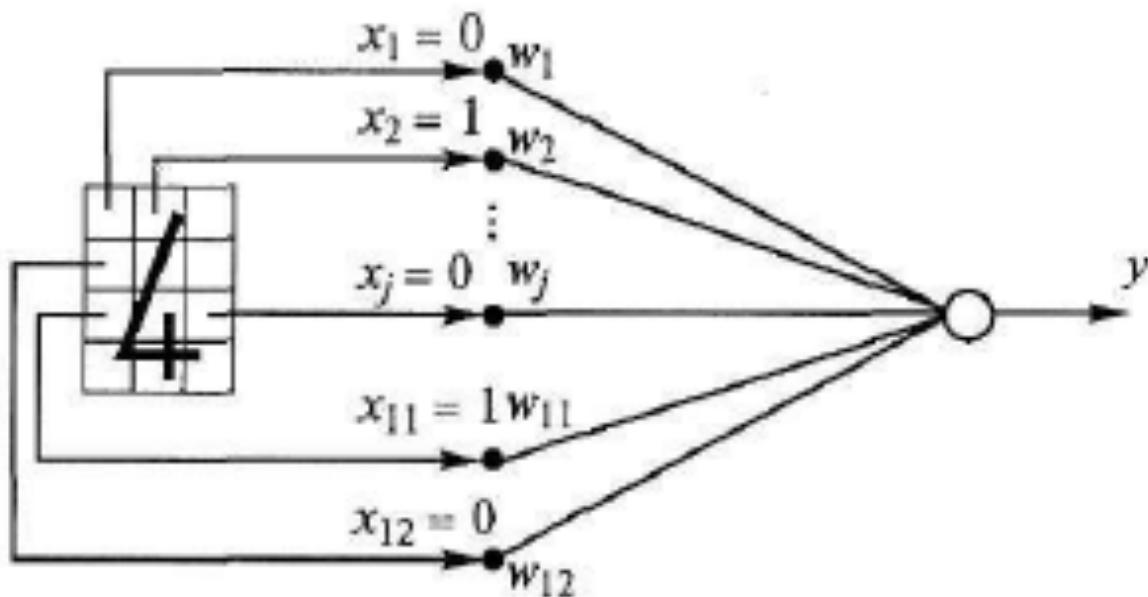
where  $\sigma(x) = 1$  if  $x > 0$  and 0 otherwise.



# Generalized McCulloch-Pitts neuron



# Rosenblatt's perceptron



## Rosenblatt's rule and Hebb's rule

**Rosenblatt's rule** for  $\{1; 0\}$  classification case for weight learning is for each object  $x_{(k)}$  change weight vector:

$$w^{[k+1]} := w^{[k]} - \eta(a_w(x_{(k)}) - y_{(k)}).$$

**Hebb's rule** for  $\{1; -1\}$  classification case for weight learning is for each object  $x_{(k)}$  change weight vector:

If  $\langle w^{[k]} x_{(k)} \rangle y_{(k)} < 0$  then  $w^{[k+1]} := w^{[k]} + \eta x_{(k)} y_{(k)}.$

# Perceptron Convergence Theorem

If  $X_m = \{(x_1, y_1), \dots, (x_m, y_m)\}$  describes a linearly separable dichotomy, then the fixed-increment perceptron algorithm terminates after a finite number of weight updates.

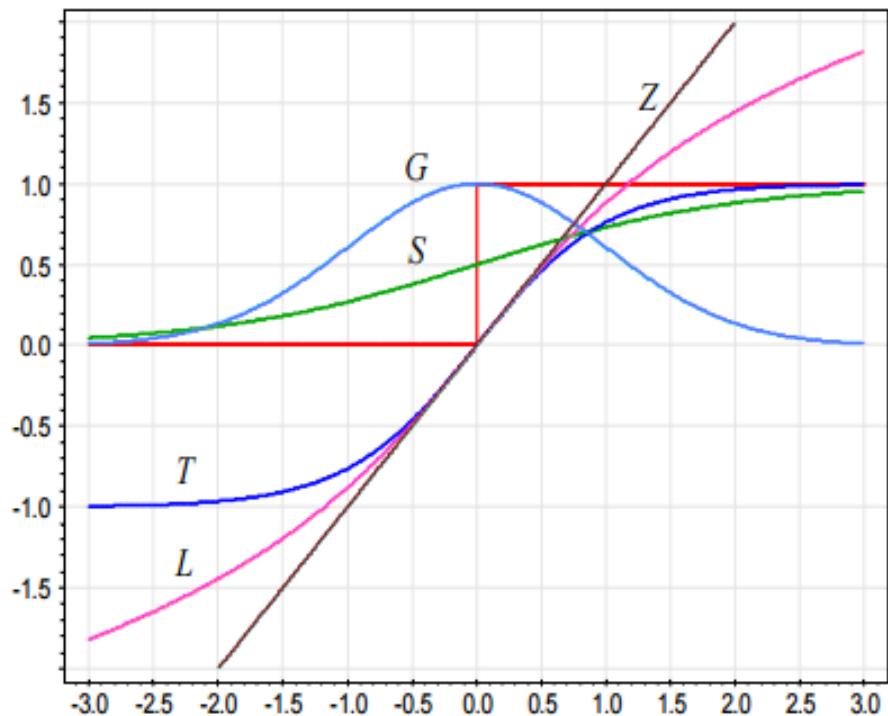
## Delta-rule

Let  $L(a_w, x) = (\langle w, x \rangle - 1)^2$ .

**Delta-rule** for weight learning is for each object  $x_{(k)}$  change weight vector:

$$w^{[k+1]} := w^{[k]} - \eta(\langle w, x_{(k)} \rangle - y_{(k)}).$$

# Activation functions



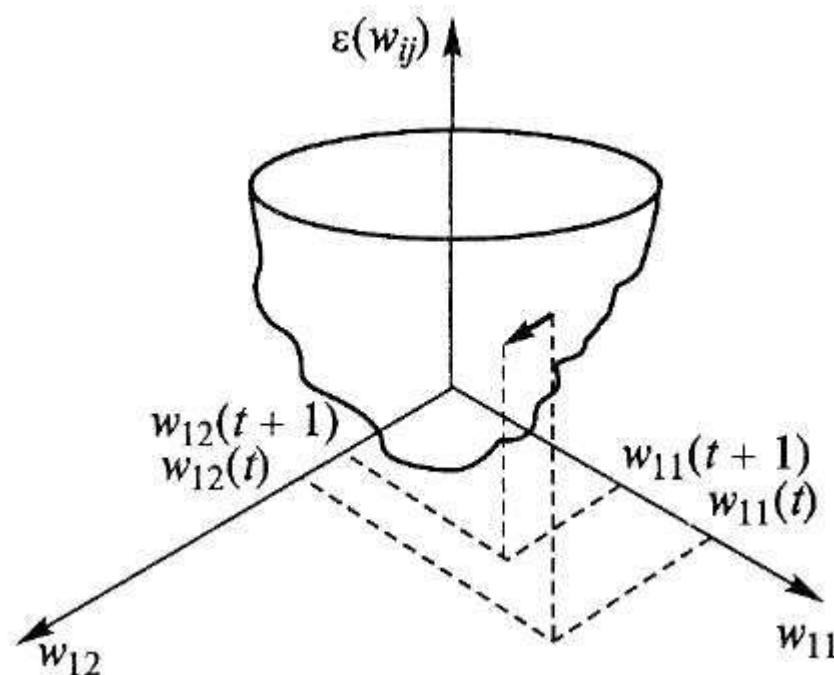
$$\begin{array}{ll} \sigma(z) = (1 + e^{-z})^{-1} & S \\ \tanh(z) = 2\sigma(2z) - 1 & T \\ \ln(z + \sqrt{z^2 + 1}) & L \\ \exp(-z^2/2) & G \\ z & Z \end{array}$$

# Gradient descent

Widrow, Hoff, 1960

$$\varepsilon = \frac{1}{2} \sum_{i=1}^I (d_i - y_i)^2$$

$$\varepsilon = \varepsilon(w_{ij})$$



## Gradient descent

Changing  $\Delta w_{ij}$  in the direction opposite to the gradient of the hypersurface :

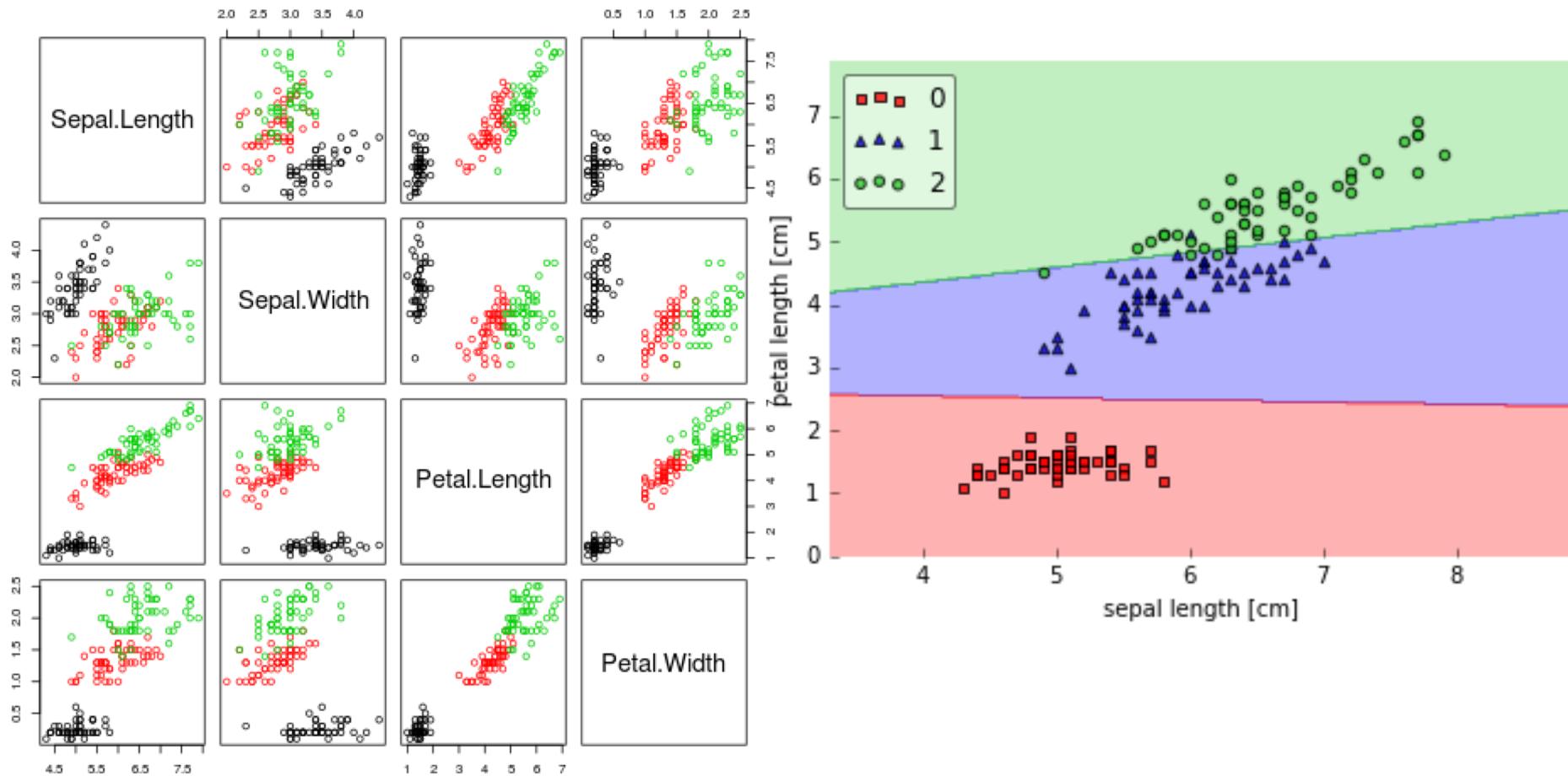
$$\Delta w_{ij} = -\eta \frac{\partial \varepsilon}{\partial w_{ij}}, \quad \frac{\partial \varepsilon}{\partial w_{ij}} = \frac{\partial \varepsilon}{\partial y_i} \frac{\partial y_i}{\partial w_{ij}}$$

$$\frac{\partial y_i}{\partial w_{ij}} = \frac{\partial f_\sigma(S_i)}{\partial S_i} \frac{\partial S_i}{\partial w_{ij}} = f'_\sigma(S_i) x_j$$

$$\frac{\partial \varepsilon}{\partial y_i} = -(d_i - y_i)$$

$$\boxed{\Delta w_{ij} = -\eta(-(d_i - y_i)f'_\sigma(S_i)x_j) = \eta(d_i - y_i)f'_\sigma(S_i)x_j}$$

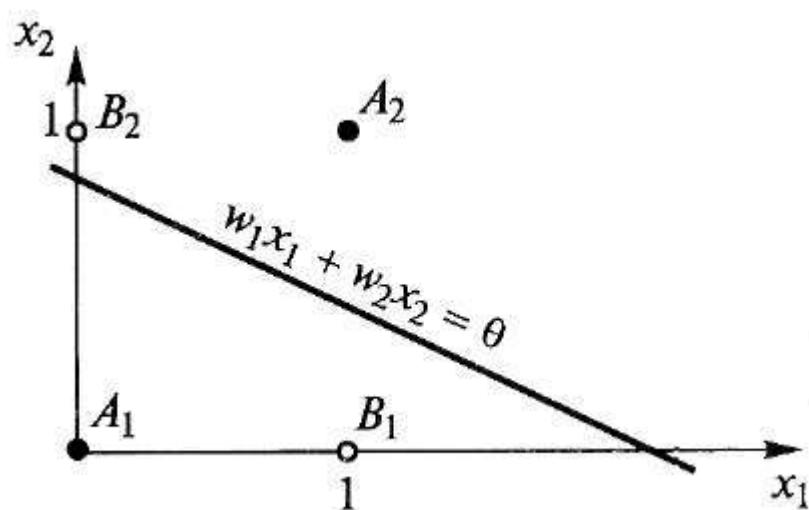
# Linear decision boundary



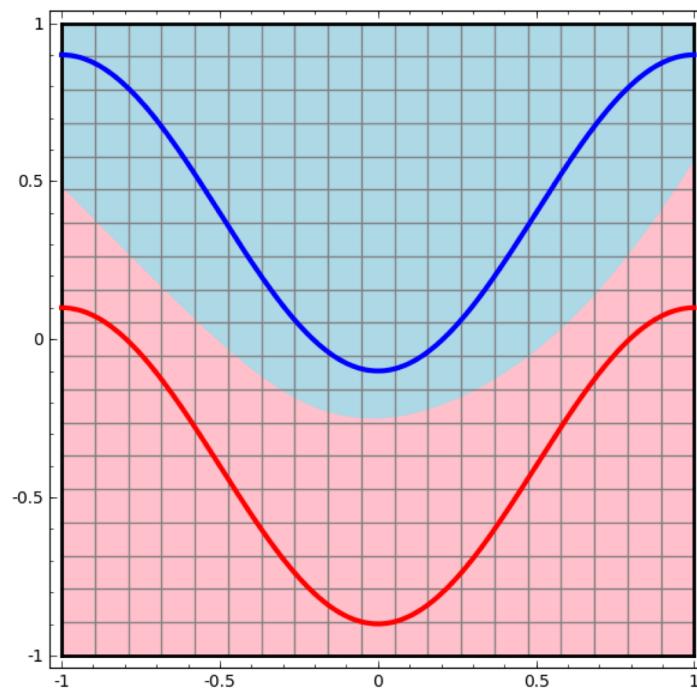
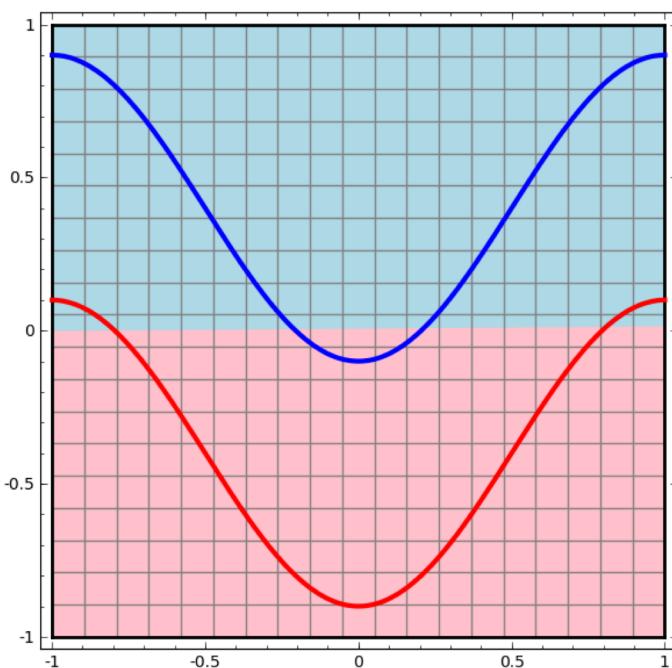
# Linear Separability and the XOR Problem

Minsky, Papert, 1969

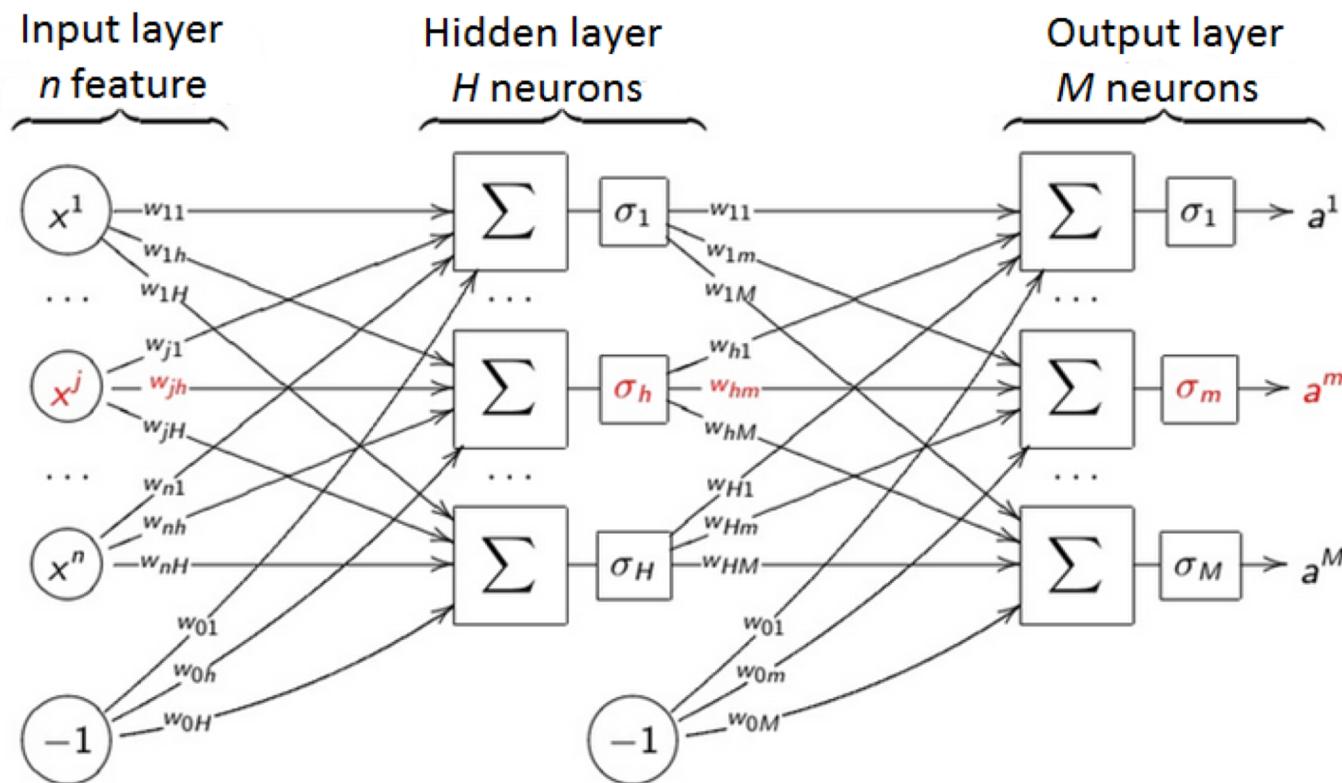
✓ «XOR»



# Linear Separability

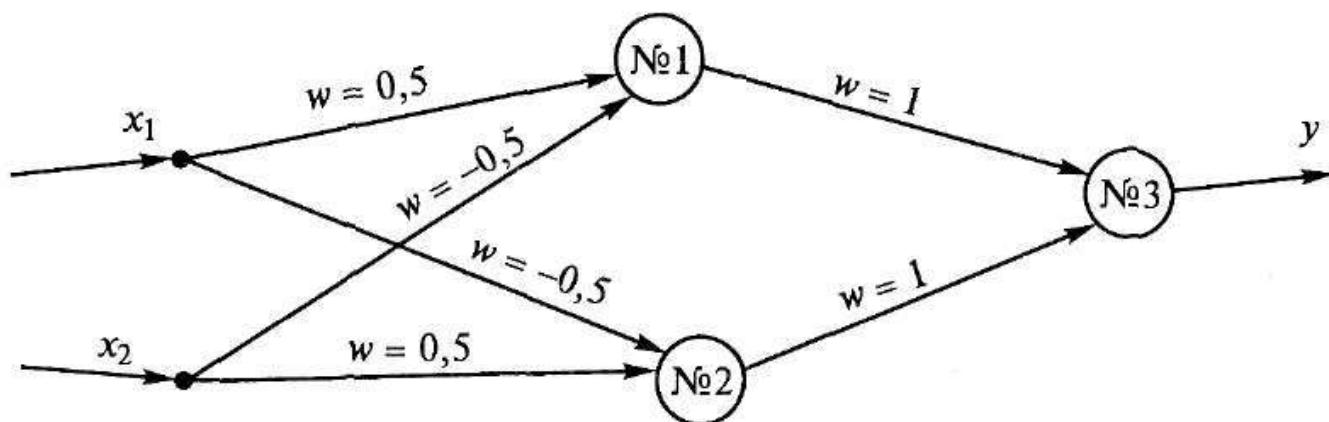


# Multilayer perceptron (MLP)





## MLP. The modeling of XOR

N<sub>1</sub>:

$$S_1 = 0,5 \times x_1 + (-0,5) \times x_2;$$

$$\begin{aligned}y_1 &= 1, & S_1 &\geq \theta; \\y_1 &= 0, & S_1 &< \theta.\end{aligned}$$

N<sub>2</sub>:

$$S_2 = (-0,5) \times x_1 + 0,5 \times x_2;$$

$$\begin{aligned}y_2 &= 1, & S_2 &\geq \theta; \\y_2 &= 0, & S_2 &< \theta.\end{aligned}$$

N<sub>3</sub>:

$$S_3 = 1 \times y_1 + 1 \times y_2;$$

$$\begin{aligned}y_3 &= 1, & S_3 &\geq \theta; \\y_3 &= 0, & S_3 &< \theta.\end{aligned}$$

# Back propagation

LeCun, Werbos, Galushkin, 1974

✓ For output layer:

$$w_{ij}^{(k)}(t+1) = w_{ij}^{(k)}(t) + \Delta w_{ij}^{(k)}(t+1),$$

$$\Delta w_{ij}^{(k)}(t+1) = \eta \delta_i^{(k)} y_j^{(k-1)}, \quad \delta_i^{(K)} = (d_i - y_i) f'_\sigma(S_i).$$

✓ For hidden layers:

$$(d_j - y_j) = \sum_{i=1}^I \delta_i w_{ij}$$

$$\delta_i^{(K)} = f'_\sigma(S_i) \sum_{l=1}^{H_{k+1}} \delta_l^{(k+1)} w_{li}^{(k+1)}.$$

# Flexibility MLP

- ✓ 13<sup>th</sup> Hilbert problem
- ✓ Solved by Kolmogorov, in case of neural nets – by Hecht-Nielsen

# Problems with MLP

Main problems:

- ✓ Too much connections
- ✓ Lose complex structure of objects
- ✓ Tend to overfitting

# Outline

- ✓ Neural nets
- ✓ **Introduction to deep learning**
- ✓ Convolutional Neural Network (CNN)
- ✓ DNN best practices
- ✓ DNN applications

# Deep architecture

**Definition:** Deep architectures are composed of multiple levels of non-linear operations, such as neural nets with many hidden layers

- ✓ Most machine learning algorithms have shallow (1-3 layers) architecture (SVM, PCA, kNN, LogisticRegression, etc.)

**Goal:** Deep learning methods aim at:

- ✓ Learning feature hierarchies, **no more feature engineering!**
- ✓ Where features from higher levels of the hierarchy are formed by lower level features.

# Why to go deep?

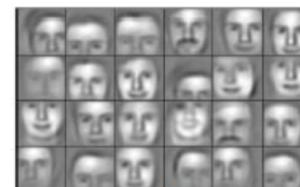
- ✓ Some functions cannot be efficiently represented by shallow architectures
- ✓ Functions that can be compactly represented by a depth  $k$  architecture might require an exponential number of computational elements to be represented by a depth  $k - 1$  architecture
- ✓ Allow non-local generalization and comprehensibility
- ✓ State of the art results in many fields

# Levels of abstraction

## Hierarchical Learning:

- ✓ Natural progression from low level to high level structure as seen in natural complexity
- ✓ Easier to monitor what is being learnt
- ✓ A good lower level representation can be used for many tasks

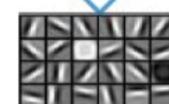
Feature representation



3rd layer  
“Objects”



2nd layer  
“Object parts”



1st layer  
“Edges”



Pixels

# Discrete convolution

$$(x \times k)_{ij} = \sum_{pq} x_{i+p,j+q} k_{r-p,r-q}$$

$$\begin{array}{c}
 \begin{matrix} & 1 & 0 & + & 0.5 & \times & 80 & + & 0.25 & \times & 20 & + & 0 & \times & 40 \\ \downarrow & & & & & & & & & & & & & & & \\ \begin{matrix} 0 & 1 & 80 & 0.5 & 40 \\ 1 & 0 & 0.5 & 40 \\ 0.25 & 40 & 0 & 0 \\ 0 & 0 & 40 \\ x & & & \end{matrix} & * & \begin{matrix} 0 & 0.25 \\ 0.5 & 1 \end{matrix} & = & \begin{matrix} 45 \end{matrix} \end{matrix} \\
 k \\
 1 \times 80 + 0.5 \times 40 + 0.25 \times 40 + 0 \times 0
 \end{array}$$

$$\begin{matrix} x \\ \begin{matrix} 0 & 80 & 40 & 0.5 \\ 20 & 1 & 0.25 & 0 \\ 0 & 0 & 40 \\ x \end{matrix} \end{matrix} * \begin{matrix} 1 & 0.5 & 0.25 & 0 \\ 0.5 & 1 \end{matrix} = \begin{matrix} 45 & 110 \end{matrix}$$

# Discrete convolution

$$(x \times k)_{ij} = \sum_{pq} x_{i+p, j+q} k_{r-p, r-q}$$

The diagram shows the computation of a single output unit in a convolution layer. An input feature map  $x$  of size 3x3 is multiplied by a kernel  $k$  of size 2x2. The result is a 2x2 output unit. The calculation for the top-left output unit is shown:  $1 \times 20 + 0.5 \times 40 + 0.25 \times 0 + 0 \times 0 = 40$ . The input values are labeled with their respective weights from the kernel.

0	80	40
20	40	0.5
0.25	0	40

$x$

$* \quad k$

$=$

45	110
40	

The diagram shows the computation of a single output unit in a convolution layer. An input feature map  $x$  of size 3x3 is multiplied by a kernel  $k$  of size 2x2. The result is a 2x2 output unit. The calculation for the bottom-right output unit is shown:  $1 \times 40 + 0.5 \times 0 + 0.25 \times 0 + 0 \times 40 = 40$ . The input values are labeled with their respective weights from the kernel.

0	80	40
20	40	1
0	0.25	40

$x$

$* \quad k$

$=$

45	110
40	40



# What can kernels do?



\*

0	0	0	0	0
0	1	1	1	0
0	1	1	1	0
0	1	1	1	0
0	0	0	0	0

=



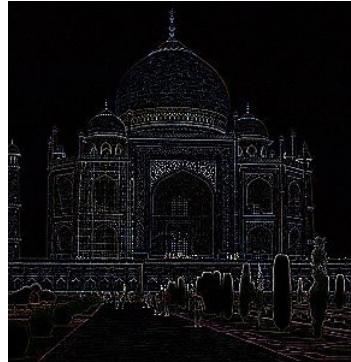
blur



\*

0	1	0		
1	-4	1		
0	1	0		

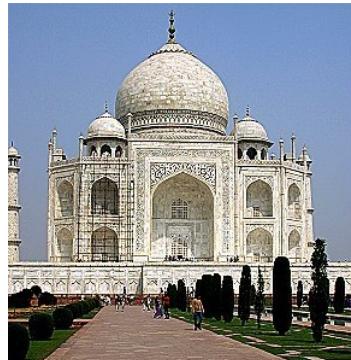
=

edge  
detection

\*

0	0	0	0	0
0	0	-1	0	0
0	-1	5	-1	0
0	0	-1	0	0
0	0	0	0	0

=



sharpen

# Outline

- ✓ Neural nets
- ✓ Introduction to deep learning
- ✓ **Convolutional Neural Network (CNN)**
- ✓ DNN best practices
- ✓ DNN applications

# Convolutional Neural Network (CNN)

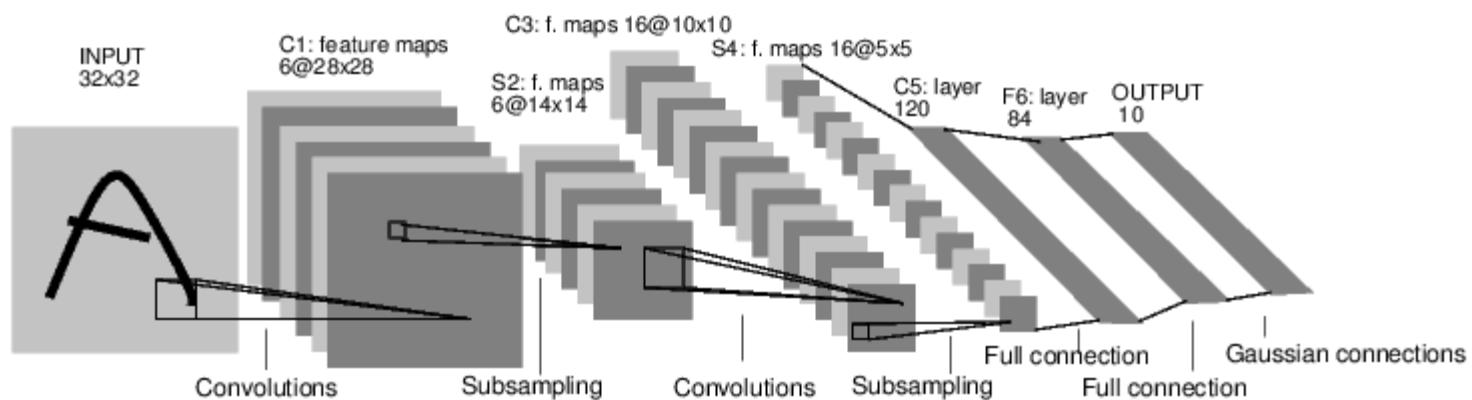
## Core concepts:

- ✓ Local perception – each neuron sees a small part of the object. Use kernels (filters) to capture 1D or 2D structure of objects. For instance, capture all pixel's neighbors for an image
- ✓ Weight sharing – use small and the same sets of kernels for all objects, this leads to less adjusting parameters than MLP
- ✓ Subsampling/pooling – use dimensionality reduction in order, for images, to provide invariance to scale

# Convolutional Neural Network

- ✓ Convolutional layers
- ✓ Subsampling (pooling) layers
- ✓ Fully connected layers

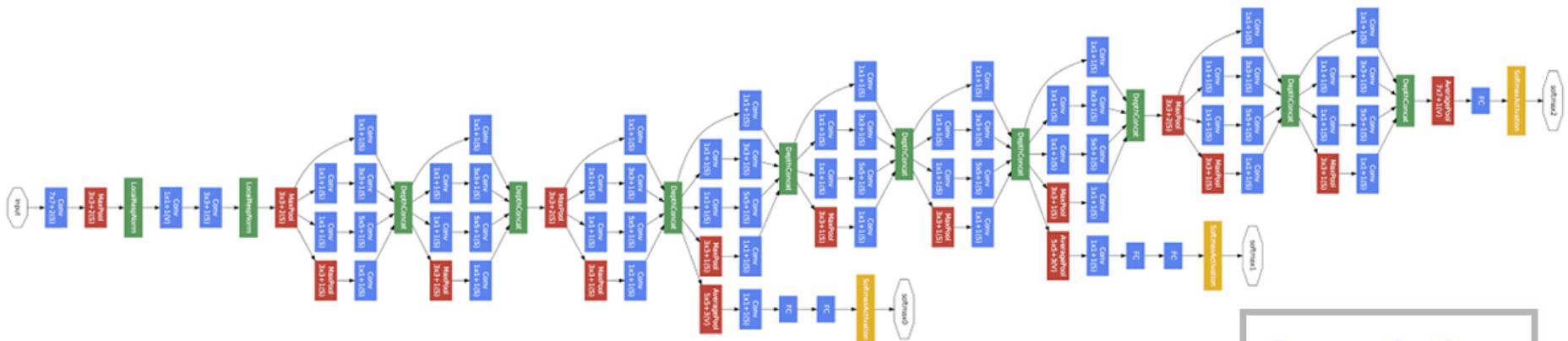
# LeNet



LeCun, Yann, et al. "Gradient-based learning applied to document recognition."  
*Proceedings of the IEEE* 86.11 (1998): 2278-2324.



# GoogLeNet

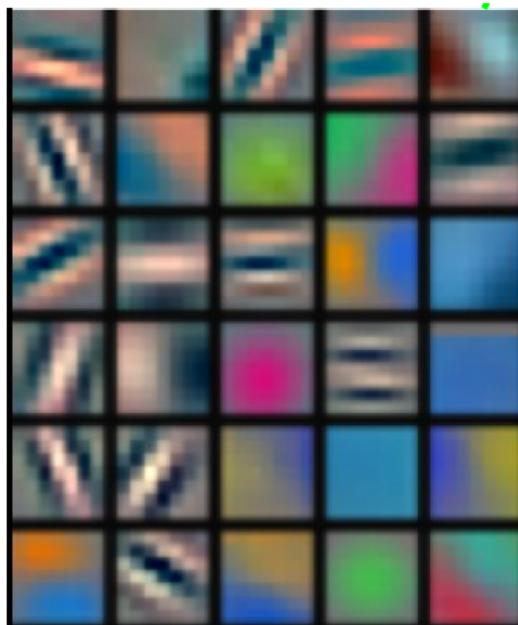


Convolution  
Pooling  
Softmax  
Other

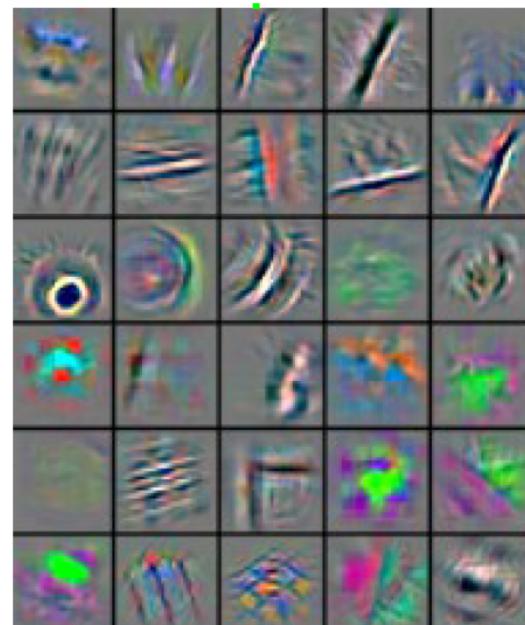
Szegedy, Christian, et al. "Going deeper with convolutions." *arXiv preprint arXiv:1409.4842* (2014).

# What do trained kernels look like?

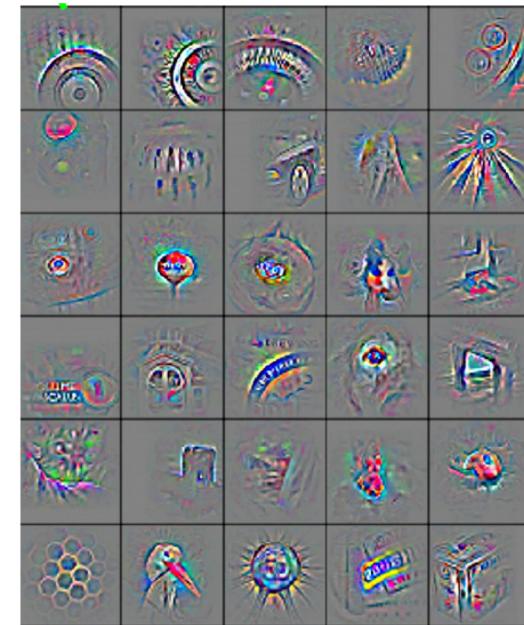
low feature



medium feature



high feature



- ✓ Each kernel composes a local patch of lower-level features into high level representation

## Pros and cons of deep learning

- ✓ Universal, flexible and powerful tool
- ✓ Suitable for any task of AI
- ✓ Shows **the best results** in complex tasks
  
- ✓ It is difficult, long and **expensive to adjust**
- ✓ For good results, you need **a lot of data**

# Outline

- ✓ Neural nets
- ✓ Introduction to deep learning
- ✓ Convolutional Neural Network (CNN)
- ✓ **DNN best practices**
- ✓ DNN applications

# Data augmentation

- ✓ Image shifting, rotation
- ✓ Horizontal/vertical flips + cropping
- ✓ Changing RGB intensities

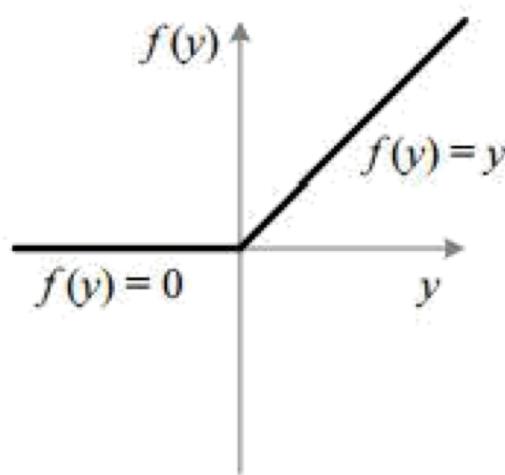
# Dropout

Dropout: set the output of each hidden neuron to zero w.p. 0.5

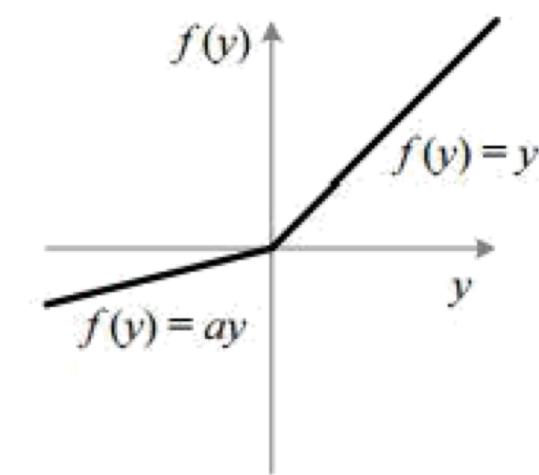
- ✓ The neurons do not contribute to the forward pass and in backpropagation
- ✓ Allows to try different architectures
- ✓ It is forced to learn more robust features
- ✓ Allows to avoid overfitting

# ReLU, PReLU

ReLU



PReLU



- ✓ Sigmoid, hyperbolic tangent activation functions have a problem with vanishing of gradients and tend to overfitting

# Loss functions

✓ Softmax

✓  $L = \frac{1}{N} \sum_i -\log \frac{e^{f_{y_i}}}{\sum_j e^{f_j}}$

✓ Where  $j \in [1, K]$ ,  $N$  – number of samples

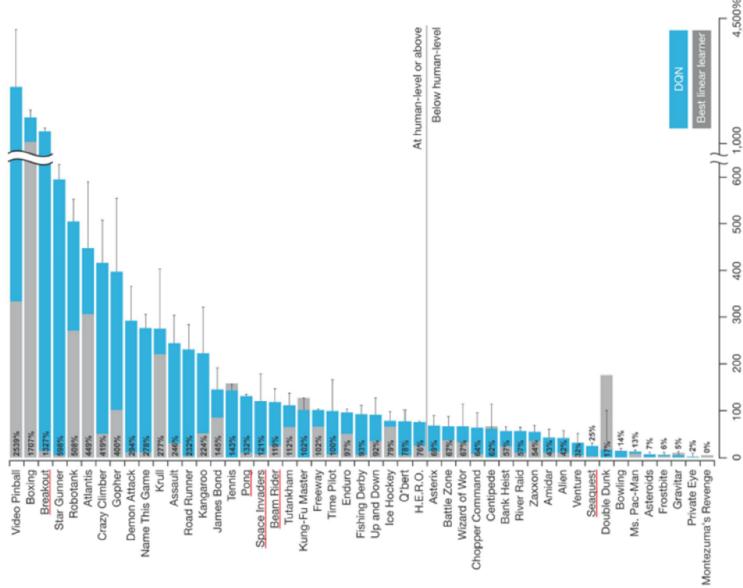
# Outline

- ✓ Neural nets
- ✓ Introduction to deep learning
- ✓ Convolutional Neural Network (CNN)
- ✓ DNN best practices
- ✓ DNN applications**

# Deep learning applications

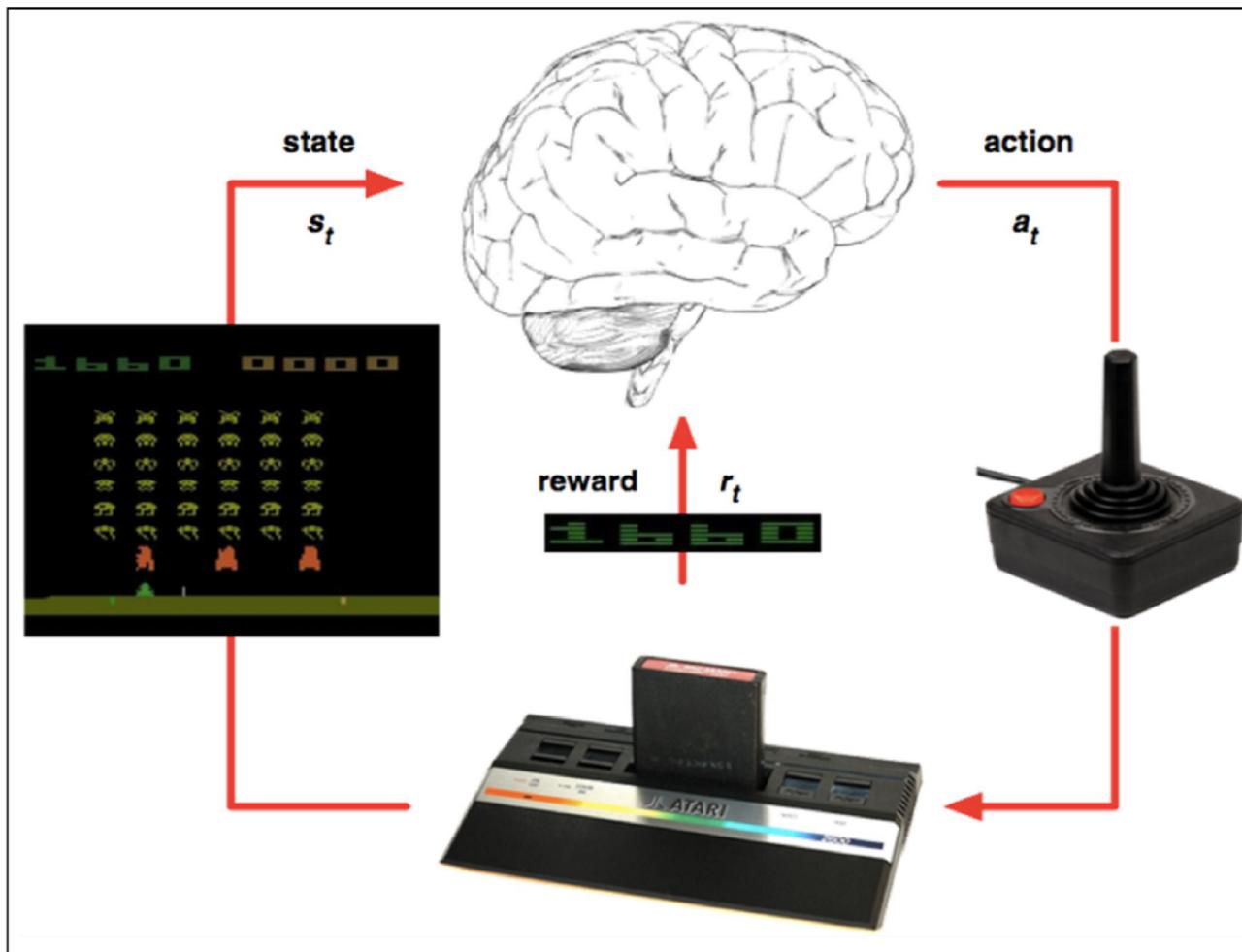
- ✓ Image Recognition
- ✓ Image Search
- ✓ Visual Question Answering
- ✓ NLP
- ✓ Speech Recognition
- ✓ ...

# Atari

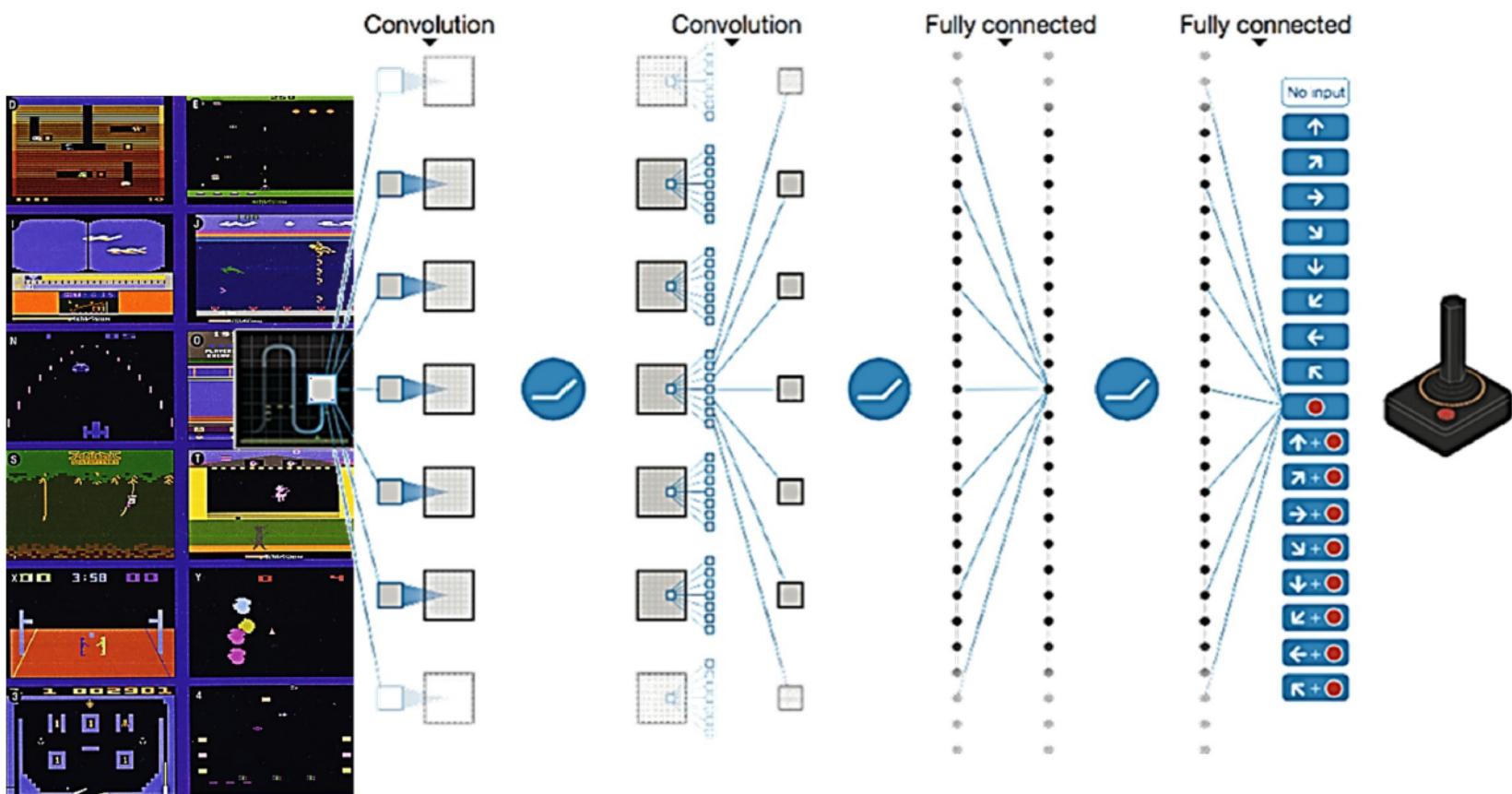


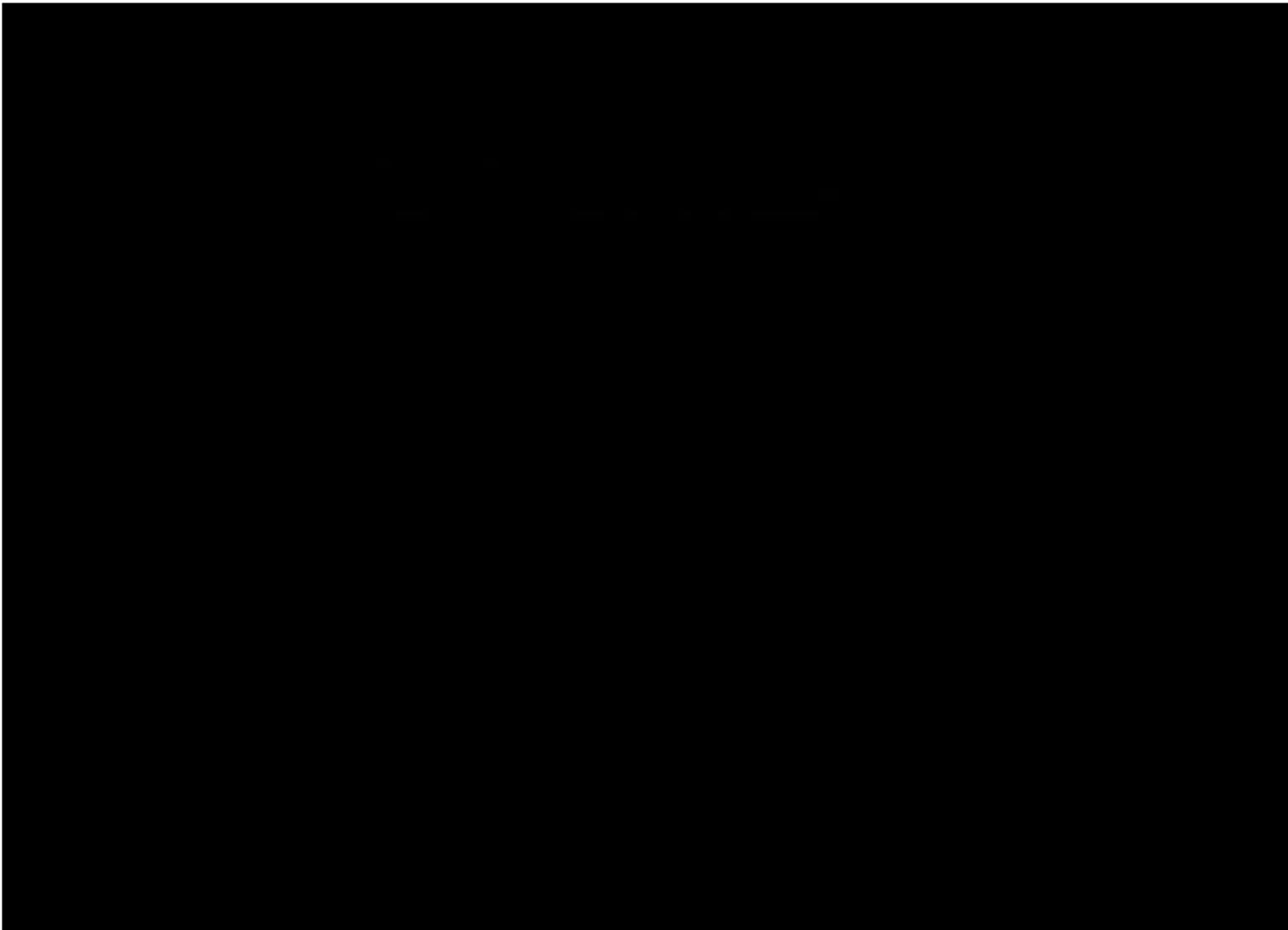
Mnih, Volodymyr, et al. "Human-level control through deep reinforcement learning." *nature* 518.7540 (2015): 529-533.

# Atari: how it works



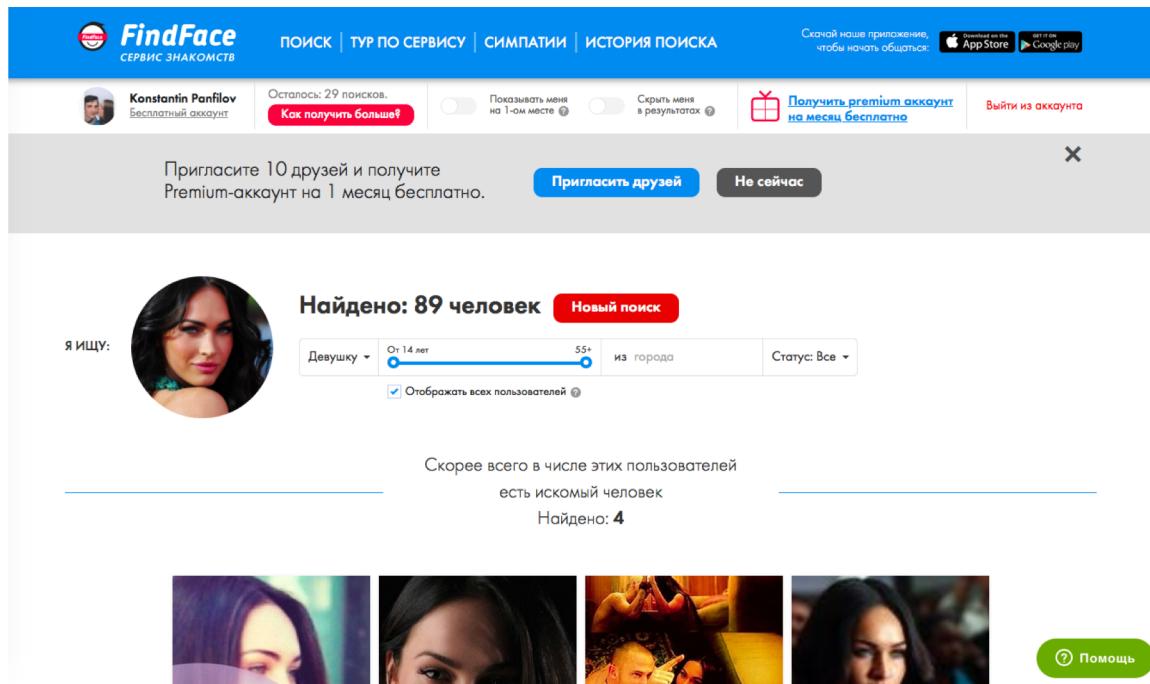
# DQN architecture





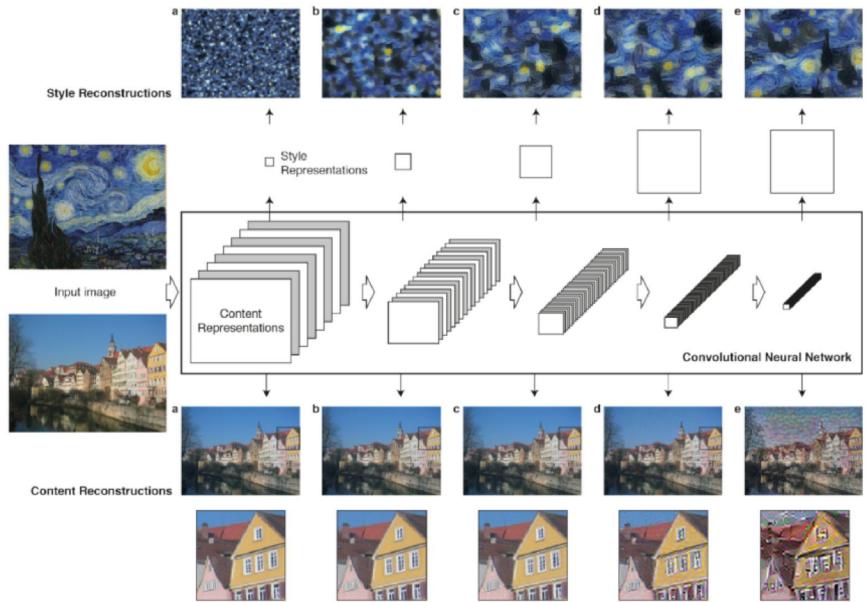
# Faces recognition

FindFace uses CNN to provide the search of faces that look similar

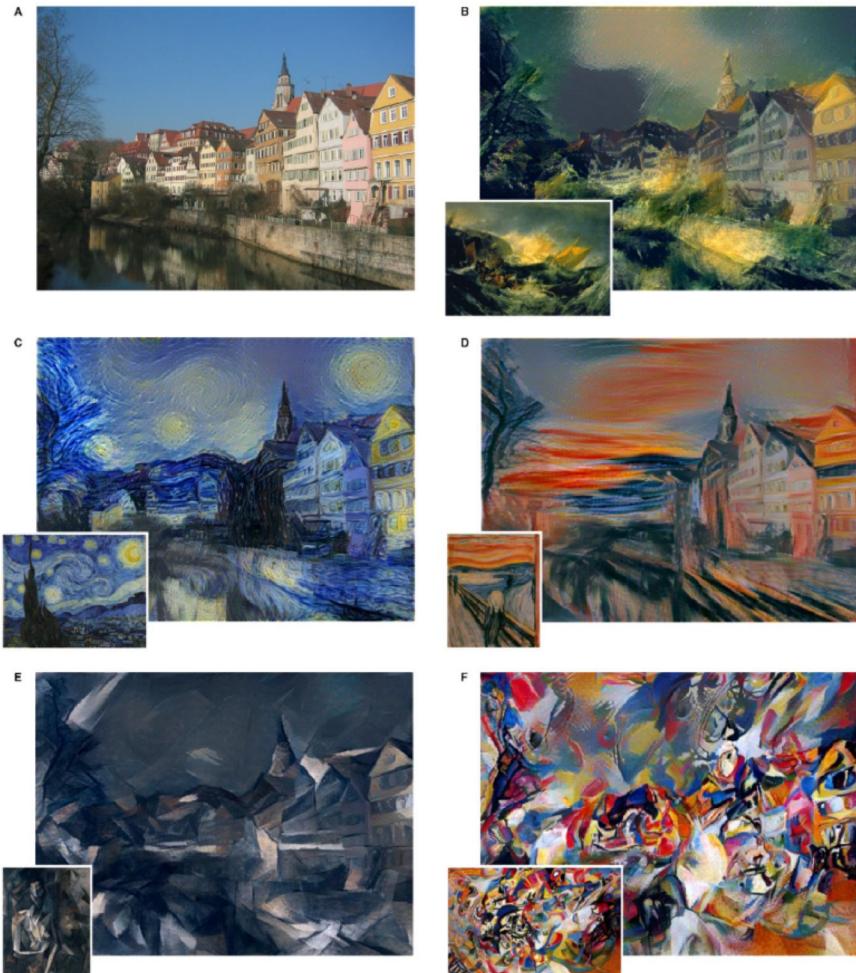


The screenshot shows the FindFace website interface. At the top, there is a blue header bar with the FindFace logo, search navigation links (ПОИСК, ТУР ПО СЕРВИСУ, СИМПАТИИ, ИСТОРИЯ ПОИСКА), download links for the app (App Store, Google Play), and account-related buttons (Получить premium аккаунт на месяц бесплатно, Выйти из аккаунта). Below the header, a promotional banner encourages users to invite 10 friends to get a free Premium account for a month. The main search results area displays a profile picture of a woman and the text "Найдено: 89 человек" (Found: 89 people). It includes filters for gender (Девушку), age (От 14 лет to 55+), location (из города), and status (Статус: Все). A checkbox for "Отображать всех пользователей" (Show all users) is checked. Below the filters, a message states that 4 users are likely to be the target person. Four small profile pictures are shown at the bottom, and a "Помощь" (Help) button is located in the bottom right corner.

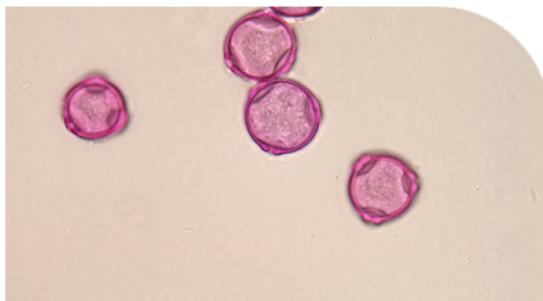
# Prisma, DeepArt



Gatys, L.A., Ecker, A.S. and Bethge, M., 2015. A neural algorithm of artistic style.  
*arXiv preprint arXiv:1508.06576.*



## Pollen recognition



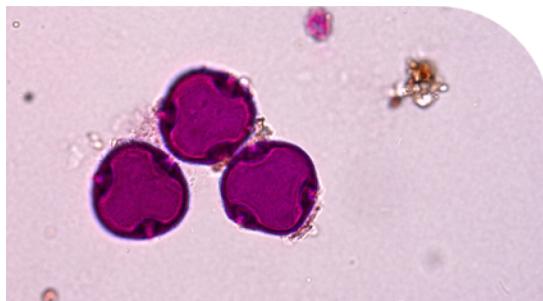
Береза повислая



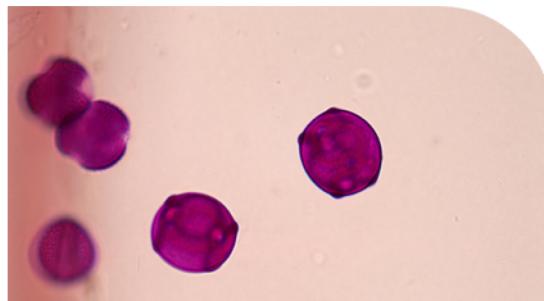
Свербига восточная



Ива белая



Липа сердцевидная



Ольха клейкая



Иван-чай узколистный

# Multiple focuses and forms

Свербига восточная



Дудник лесной



Гвоздика-травянка



Гречиха посевная



Дудник лекарственный



Клевер гибридный



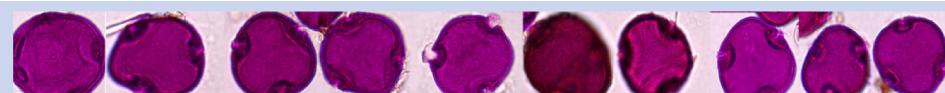
Иван-чай узколистный



Ива белая

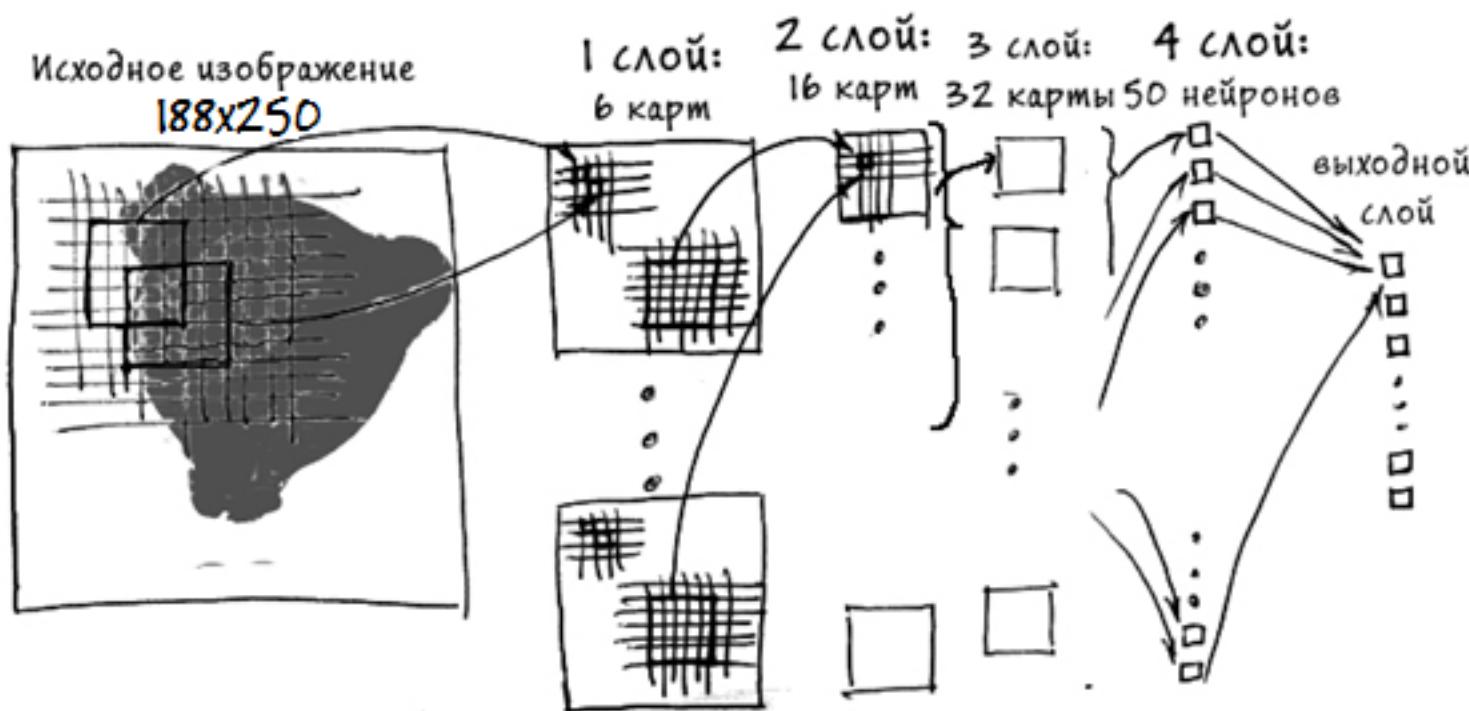


Липа сердцевидная





# CNN architecture



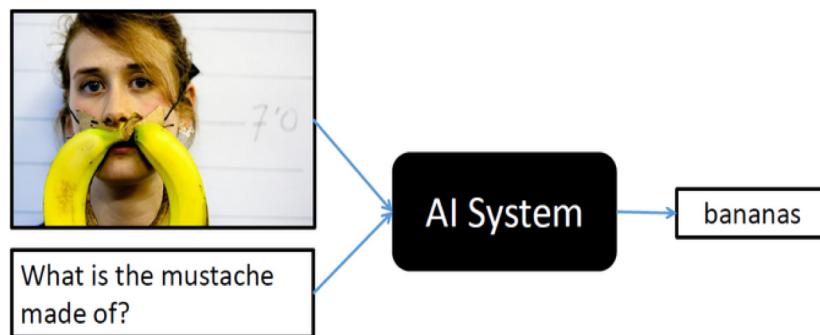
On 5 classes the accuracy = **99.7%** (RF 85.6%)

# What else?



NVIDIA Pascal

Image from visualqa.org



# Tools

- ✓ Python:  
Keras, Theano, TensorFlow, Caffe, Pylearn2, Deepnet
- ✓ C:  
Torch
- ✓ C++:  
Caffe, nnForge, CXXNET, Cuda-convent
- ✓ Matlab:  
Caffe, DeepLearningToolbox, Deepmat, Cuda CNN