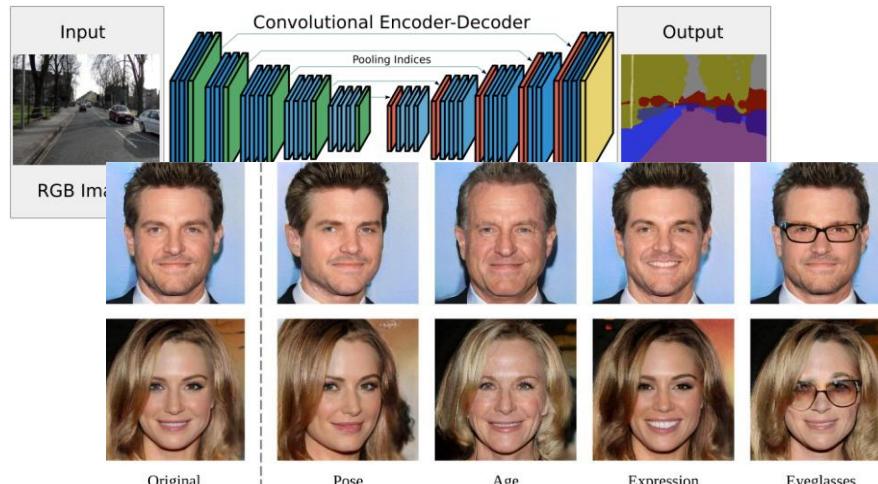
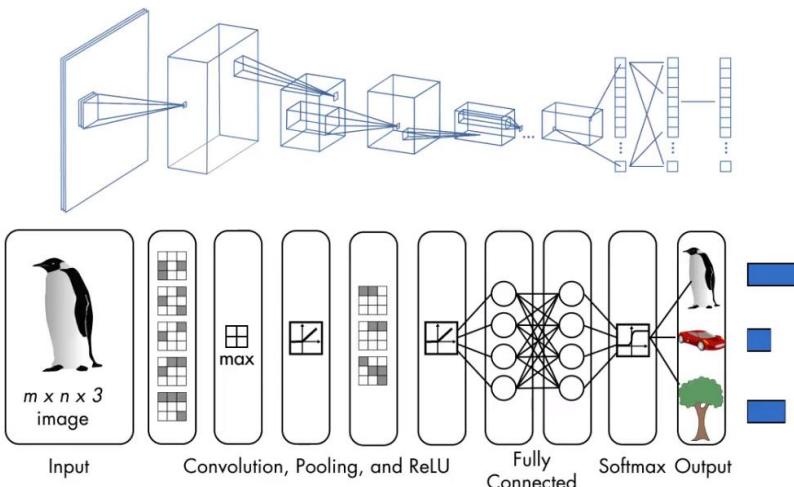


# Лекция 4. Глубокое обучение и его приложения / Deep Learning (DL) and its applications

1. Напоминание / **Contents of the previous lectures**
2. Глубокие сверточные нейронные сети / **Deep convolutional neural networks (CNNs, DCNNs)**
3. Некоторые архитектуры и приложения глубокого обучения / **DL Applications**
4. Глубокое обучение с подкреплением / **Deep reinforcement learning (DQN, DDPG)**
5. Заключение / **Conclusion**

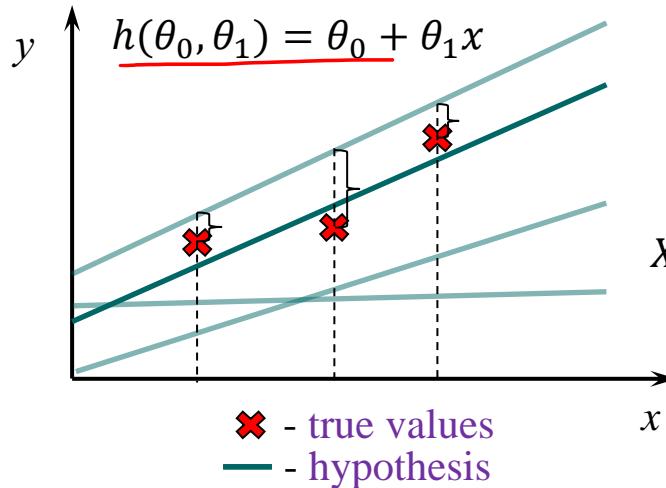


# Напоминание / Contents of the previous lectures

**Регрессионный** (от лат. regressio – обратное движение, отход) **анализ**:

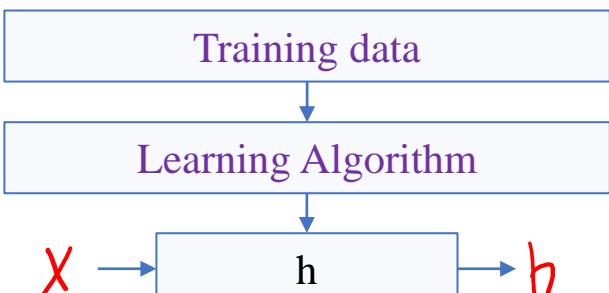
набор статистических действий для оценки связи зависимой переменной и одной или нескольких независимых переменных ([Wikipedia](#))

Основная идея линейной регрессии: поиск наилучшей функциональной зависимости множества  $Y = ((y_i))$  от множества  $X = ((x_i))$  ( $i=1 \dots m$ ) по критерию минимума некоторой функции качества:



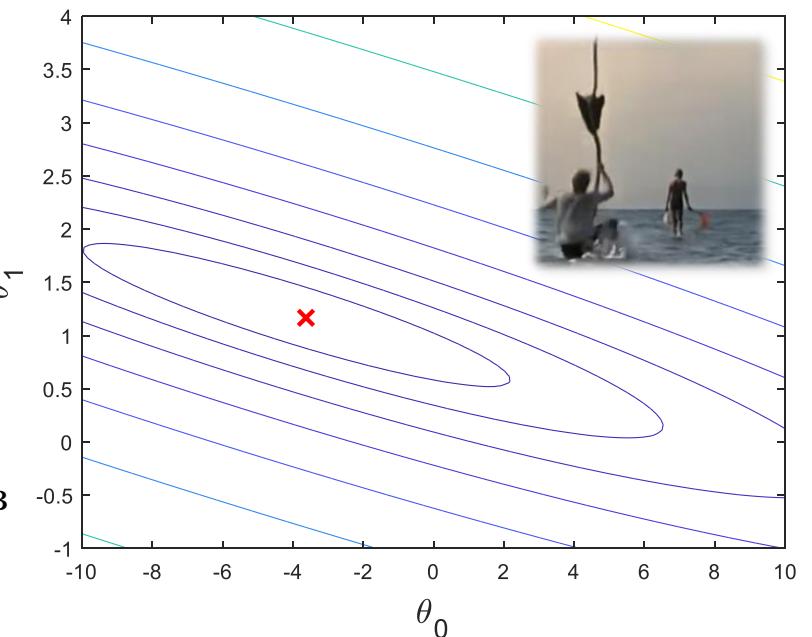
$$X = \begin{pmatrix} 1 & x^{(1)} \\ \dots & \dots \\ 1 & x^{(m)} \end{pmatrix}; Y = \begin{pmatrix} y^{(1)} \\ \dots \\ y^{(m)} \end{pmatrix}; \Theta = \begin{pmatrix} \theta_0 \\ \theta_1 \end{pmatrix}; H = X\Theta;$$

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^m (h^{(i)} - y^{(i)})^2 \Rightarrow \min.$$



Алгоритм поиска минимума функции качества.

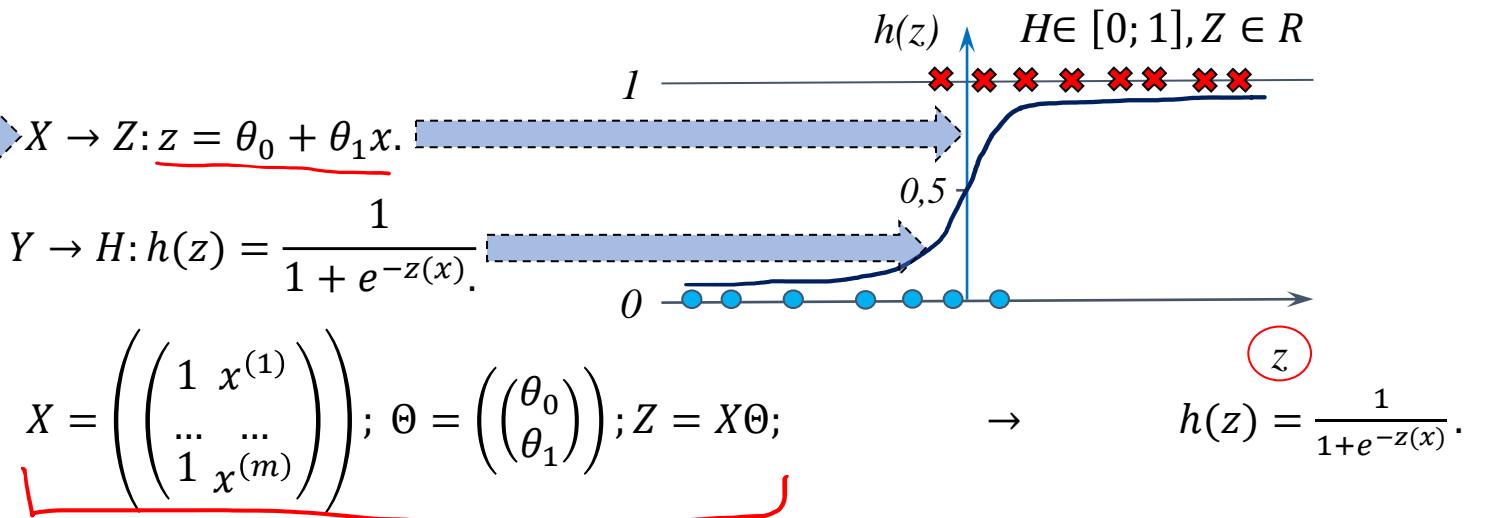
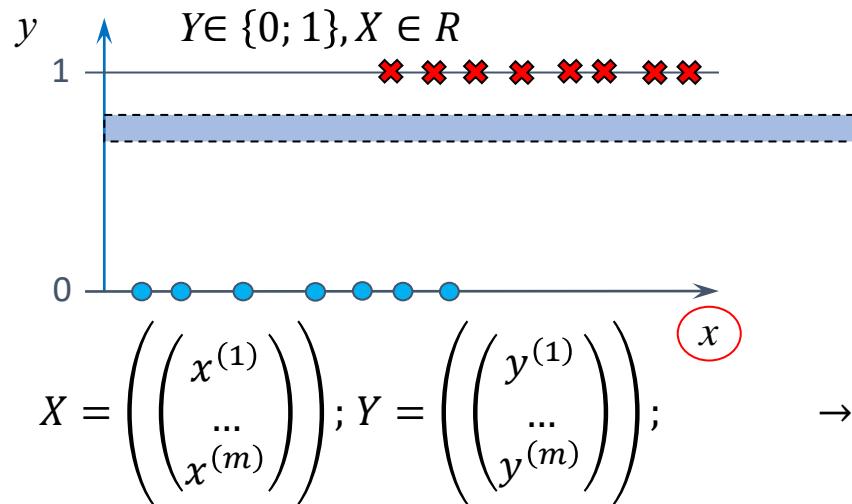
1. Задать начальное значение  $\Theta$  случайным образом.
2. Рассчитать  $H = X\Theta$  и  $\nabla J = \frac{1}{m} X^T (H - Y)$ .
3. Найти  $\Theta^H : \Theta^H = \Theta^C - \alpha \nabla J$ .
4. Повторять пункты 2-3 до выполнения одного из условий:  $J^H - J^C < \delta$ , #итер.  $> N_{max}$ .
5. Вывод результатов:  $\Theta$ .



# Напоминание / Contents of the previous lectures

Основная идея: поиск наилучшей функциональной зависимости **бинарного** множества  $Y = ((y_i))$  от множества  $X = ((x_i))$  ( $i=1 \dots m$ ) по критерию минимума некоторой функции качества:

$$J(\Theta) = -\frac{1}{m} \sum_{i=1}^m (y^{(i)} \ln(h^{(i)}) + (1 - y^{(i)}) \ln(1 - h^{(i)})) \Rightarrow \min.$$

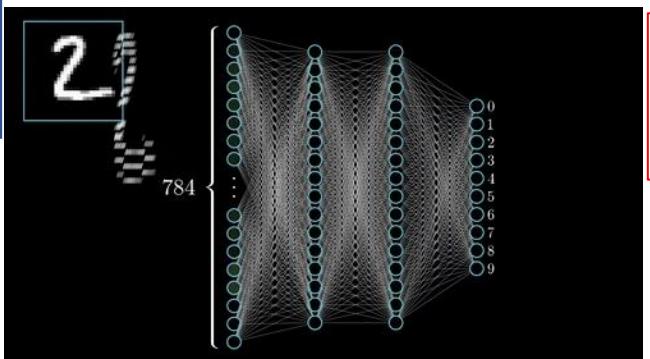


Алгоритм поиска минимума функции качества.

1. Задать начальное значение  $\Theta$  случайным образом.
2. Рассчитать  $H = X\Theta$  и  $\nabla J = \frac{1}{m} X^T (H - Y)$ .
3. Найти  $\Theta^H$ :  $\Theta^H = \Theta^C - \alpha \nabla J$ .
4. Повторять пункты 2-3 до выполнения одного из условий:  $J^H - J^C < \delta$ , #итер.  $> N_{max}$ .
5. Вывод результатов:  $\Theta$ .

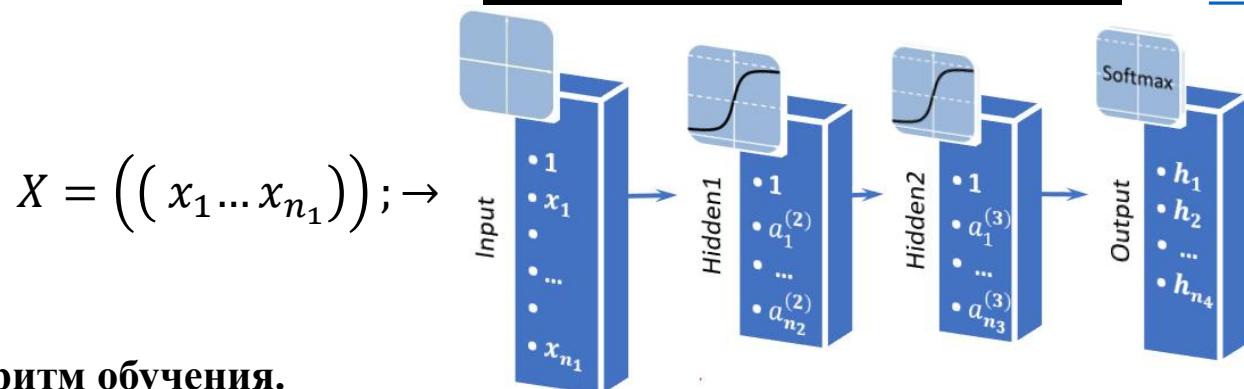
# Напоминание / Contents of the previous lectures

Поиск наилучшей функциональной зависимости множества  $Y$  от множества  $X$  по критерию минимума некоторой функции качества с помощью ИНС с количеством слоев  $l$ , количеством нейронов в  $k$ -ом слое  $n_k$  ( $n_l$  - количество классов) на основании анализа  $m$  пар значений  $Y, X$ :



$$J(\Theta^{(k)}) = -\frac{1}{m} \sum_{i=1}^m \sum_{j=1}^{n_l} \left( y_j^{(i)} \ln(h_j^{(i)}) + (1 - y_j^{(i)}) \ln(1 - h_j^{(i)}) \right) \Rightarrow \min.$$

Animations from "3Blue1Brown"



$$Y = \begin{pmatrix} 0 \\ 0 \\ 1 \\ \dots \\ 0 \end{pmatrix} \quad \Theta^{(k)} = \begin{pmatrix} \theta_{01}^{(k)} & \theta_{02}^{(k)} & \dots & \theta_{0n_{k+1}}^{(k)} \\ \theta_{11}^{(k)} & \theta_{12}^{(k)} & \dots & \theta_{1n_2}^{(k)} \\ \dots & \dots & \dots & \dots \\ \theta_{n_k 1}^{(k)} & \theta_{n_k 2}^{(k)} & \dots & \theta_{n_k n_{k+1}}^{(k)} \end{pmatrix}$$

## Алгоритм обучения.

1. Задать начальные значения компонент матрицы  $\Theta^{(k)}$  случайным образом.
2. Рассчитать вектор градиента  $\nabla J = \left[ \left[ \frac{\partial J}{\partial \theta_{ij}^{(k)}} \right] \right]$  методом обратного распространения ошибки.
3. Найти новые значения компонент  $\Theta^{(k)}$ :  $\theta_{ij}^{(k)H} = \theta_{ij}^{(k)C} - \alpha \frac{\partial J}{\partial \theta_{ij}^{(k)}}$ .
4. Повторять пп. 2-3 до достижения минимума  $J$ :  $J^H - J^C < \delta$  или #итерации  $> N_{max}$ .
5. Вывод результатов:  $\Theta^{(k)}$ .

$X \rightarrow H$  # слой ИНС  
 $\Theta_{i,j}^{(k)}$  # нейр.  $k+1$  слоя  
# нейрона  $k^{\text{го}}$  слоя

# Neural Networks

©2016 Fjodor van Veen - [asimovinstitute.org](http://asimovinstitute.org)

- Backfed Input Cell
- Input Cell
- △ Noisy Input Cell
- Hidden Cell
- Probabilistic Hidden Cell
- △ Spiking Hidden Cell
- Output Cell
- Match Input Output Cell
- Recurrent Cell
- Memory Cell
- △ Different Memory Cell
- Kernel
- Convolution or Pool

Perceptron (P)



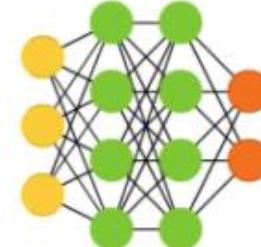
Feed Forward (FF)



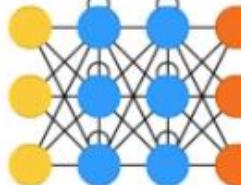
Radial Basis Network (RBF)



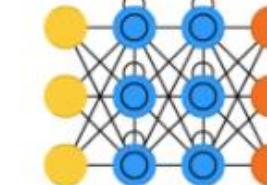
Deep Feed Forward (DFF)



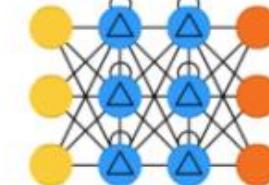
Recurrent Neural Network (RNN)



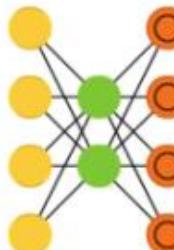
Long / Short Term Memory (LSTM)



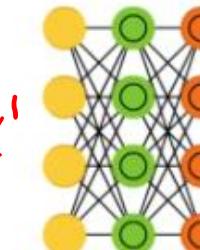
Gated Recurrent Unit (GRU)



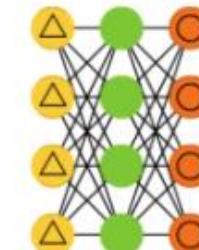
Auto Encoder (AE)



Variational AE (VAE)



Denoising AE (DAE)



Sparse AE (SAE)



Markov Chain (MC)



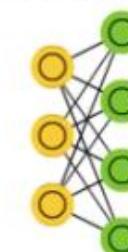
Hopfield Network (HN)



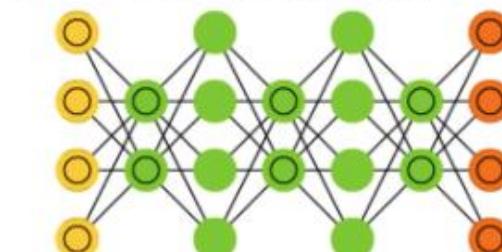
Boltzmann Machine (BM)



Restricted BM (RBM)



Deep Belief Network (DBN)



# Лекция 4. Глубокое обучение и его приложения / Deep Learning (DL) and its applications

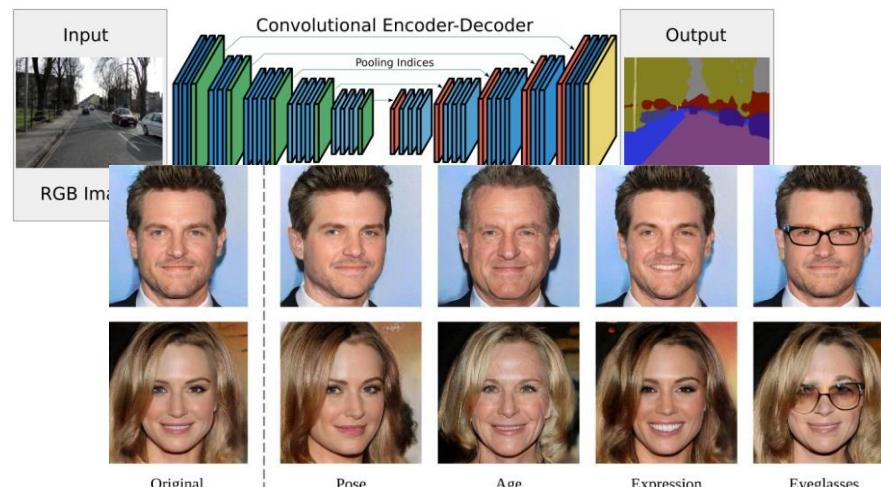
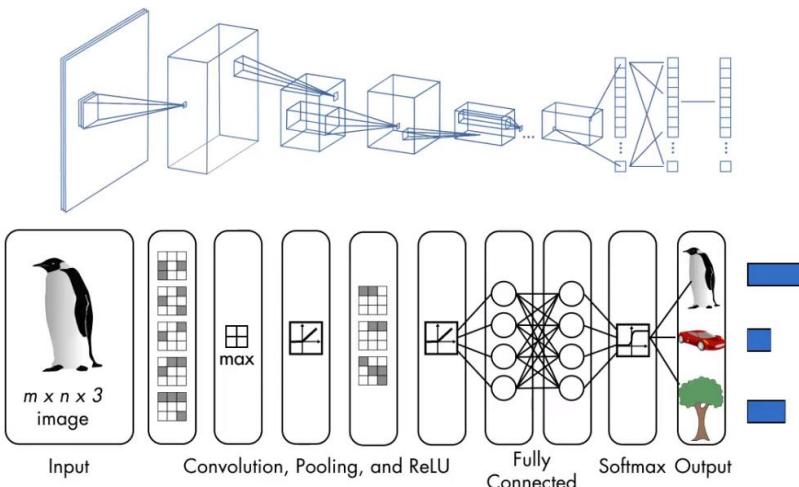
1. Напоминание / Contents of the previous lectures

2. Глубокие сверточные нейронные сети / Deep convolutional neural networks (CNNs, DCNNs)

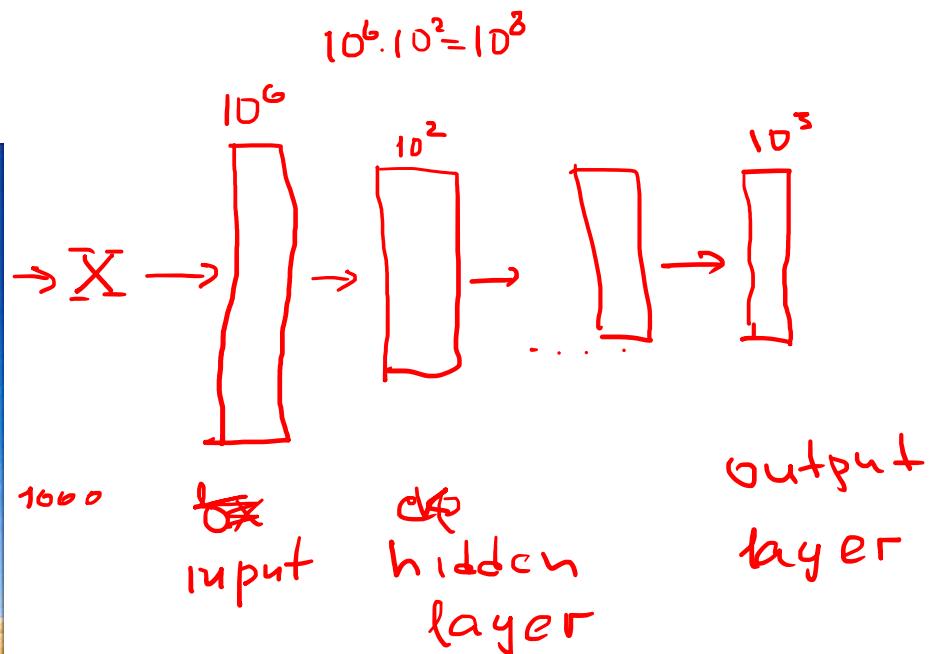
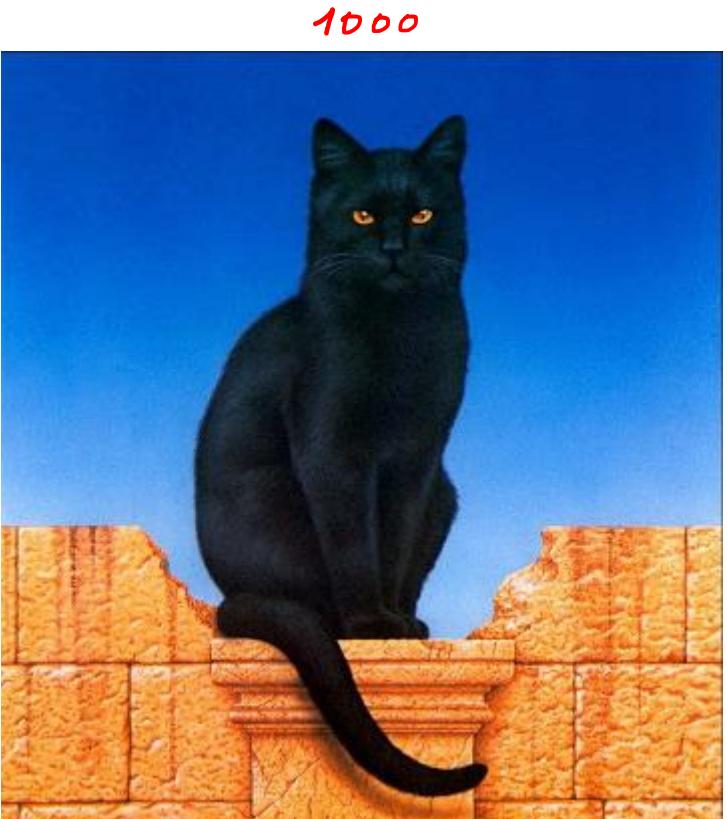
3. Некоторые архитектуры и приложения глубокого обучения / DL Applications

4. Глубокое обучение с подкреплением / Deep reinforcement learning (DQN, DDPG)

5. Заключение / Conclusion



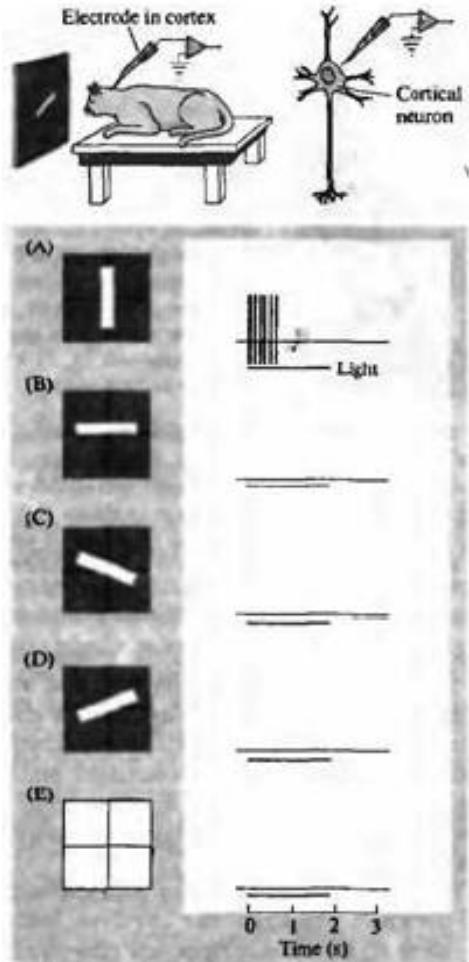
# Проблема применения полносвязных сетей в распознавании изображений / Houston, we have a problem



# Немного о кошачьей физиологии / Cat physiology

«Мы как раз вставляли слайд на стекле в виде тёмного пятна в разъём офтальмоскопа, когда внезапно, через аудиомонитор, клетка зарядила как пулемёт. Спустя некоторое время, после небольшой паники, мы выяснили, что же случилось. Конечно, сигнал не имел никакого отношения к тёмному пятну. Во время того, как мы вставляли слайд на стекле, его край отбрасывал на сетчатку слабую, но чёткую тень, в виде прямой тёмной линии на светлом фоне. Это было именно то, чего хотела клетка, и, более того, она хотела, чтобы эта линия имела строго определённую ориентацию».

/Д. Хьюбел Фрагмент нобелевской речи, 1981 г./

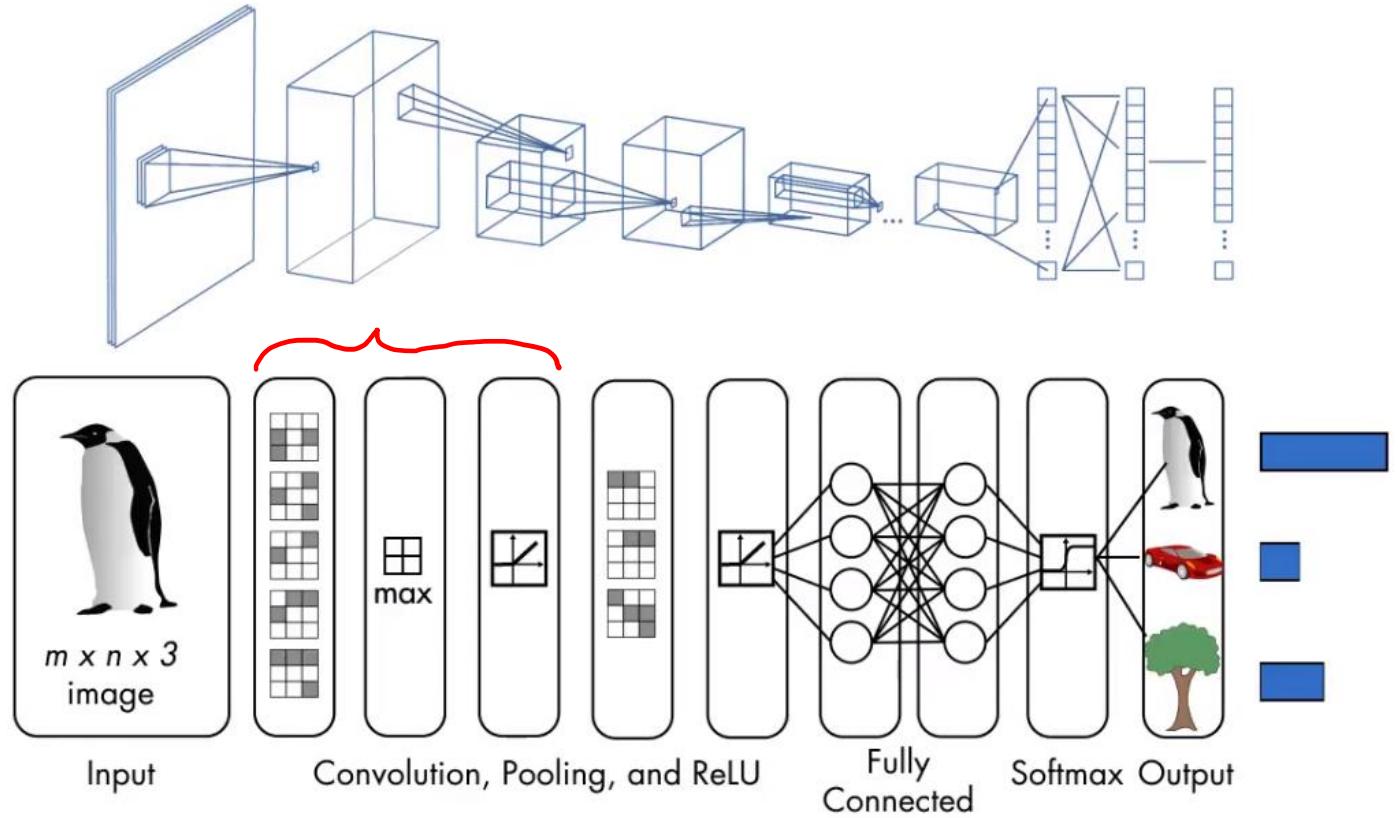
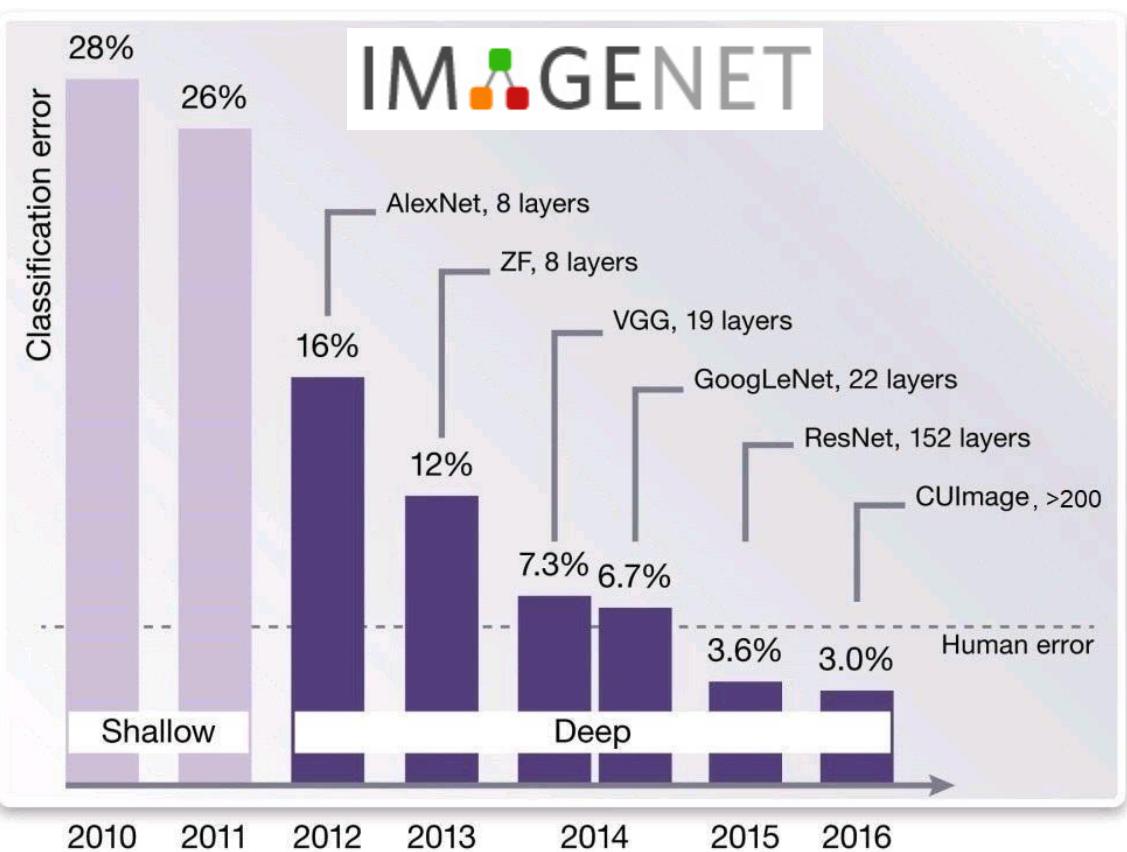


Ответы клетки зрительной коры кошки на предъявление полосок света [Дж.Г. Николлс и др. От нейрона к мозгу, 2003]

[Как видят кошки](#)

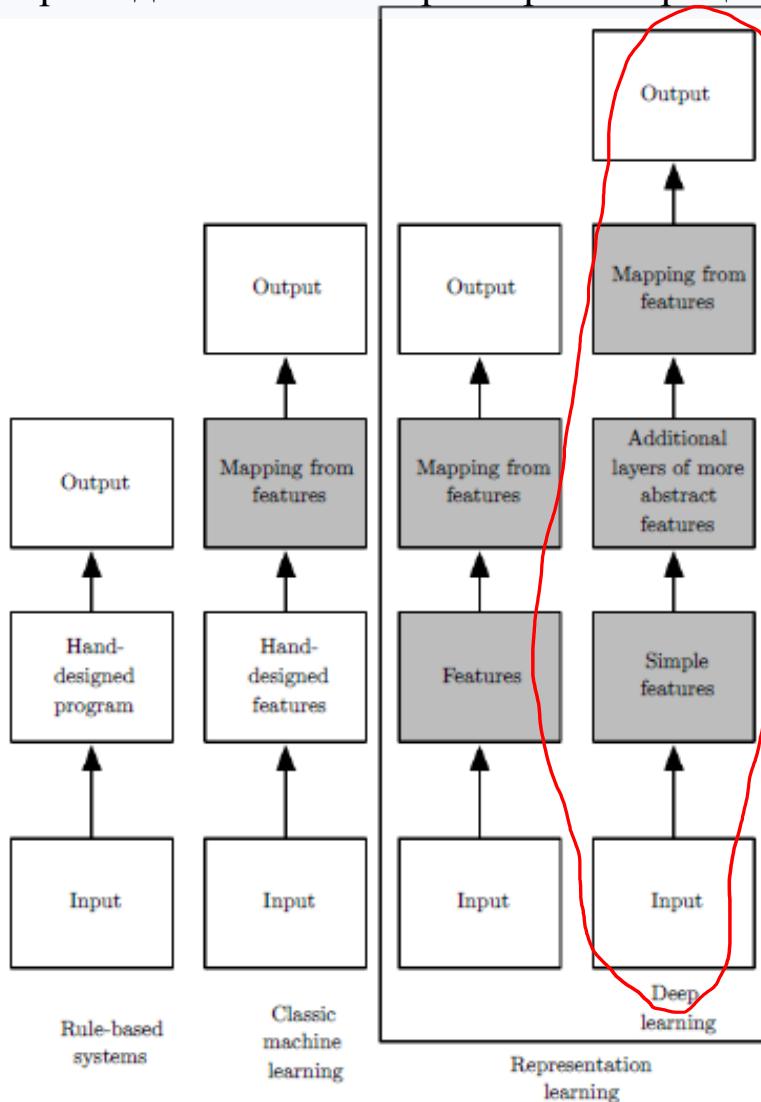
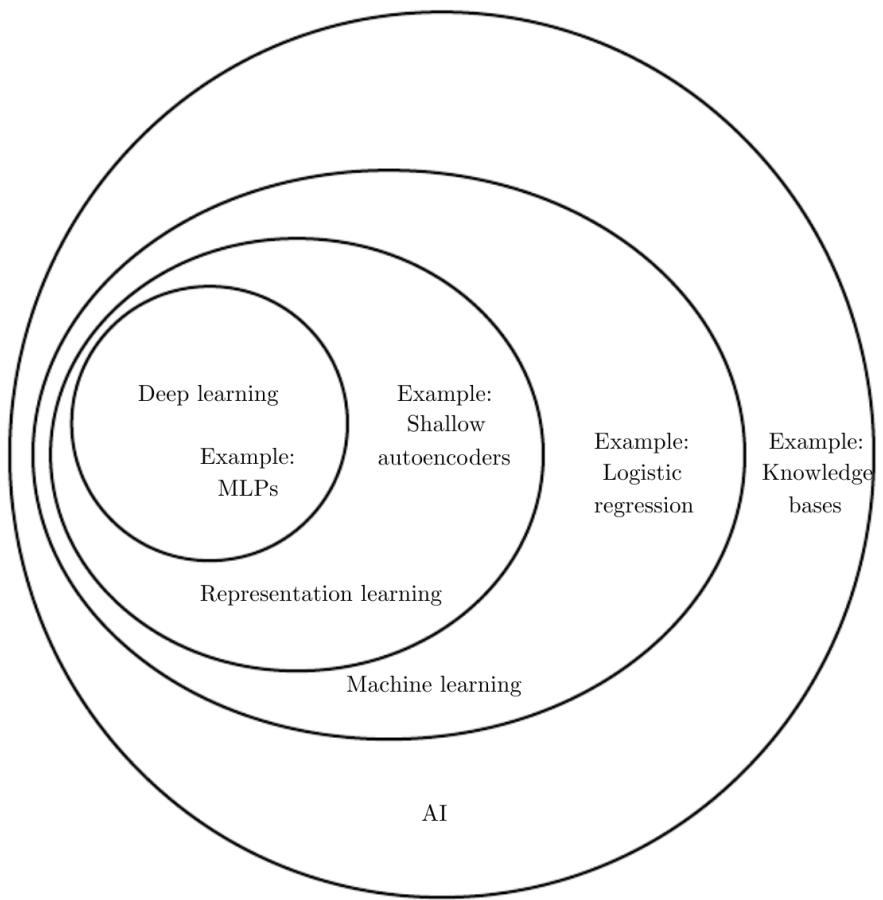
# Сверточные нейронные сети / Convolutional NN

Сверточная нейронная сеть (CHC) / Convolutional neural network (CNN) использует особенности зрительной коры, в которой простые клетки активируются на простые признаки (например, линии), а сложные на комбинацию активаций простых. CHC связана с математической операцией свертки для понижения размеров матриц. CHC обычно являются глубокими.

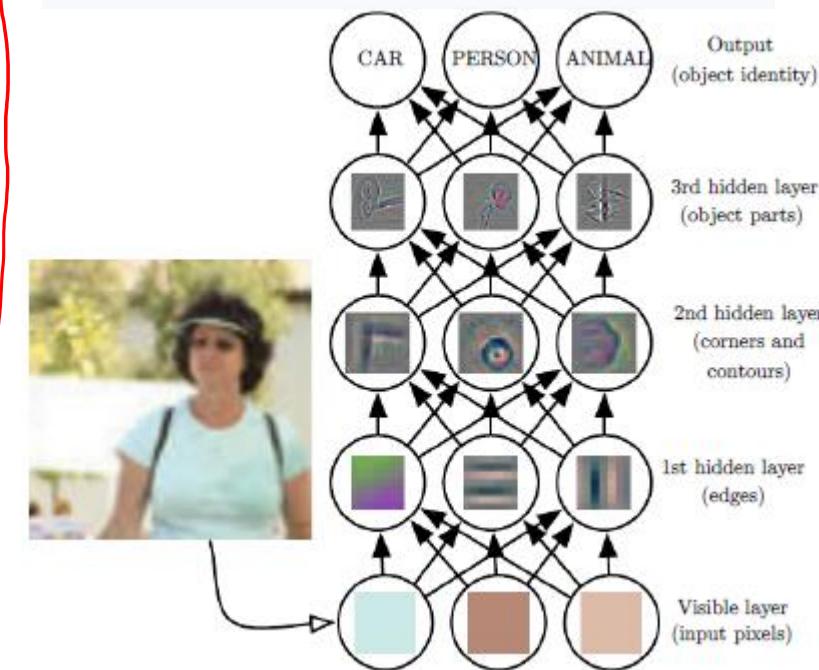


# Сверточные нейронные сети / Convolutional NN

Сверточная нейронная сеть (СНС) / Convolutional neural network (CNN) использует особенности зрительной коры, в которой простые клетки активируются на простые признаки (например, линии), а сложные на комбинацию активаций простых. СНС связана с математической операцией свертки для понижения размеров матриц. СНС обычно являются глубокими.



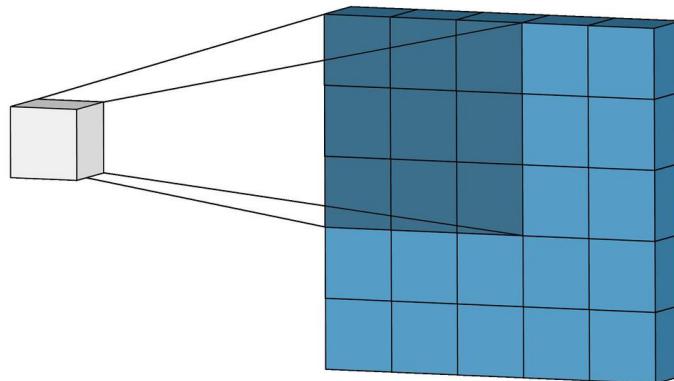
**Признаки / Features:**  
свойства объекта исследования,  
выделяемые при обучении



Deep Learning

# Прямые вычисления в СНС / Forward propagation in CNN

Свертка / Convolution операция использует особенности зрительной коры, в которой простые клетки активируются на простые признаки (например, линии), а сложные на комбинацию активаций простых. СНС связана с математической операцией свертки для понижения размеров матриц. СНС обычно являются глубокими.



3 <sub>0</sub>	3 <sub>1</sub>	2 <sub>2</sub>	1	0
0 <sub>2</sub>	0 <sub>2</sub>	1 <sub>0</sub>	3	1
3 <sub>0</sub>	1 <sub>1</sub>	2 <sub>2</sub>	2	3
2	0	0	2	2
2	0	0	0	1

12.0	12.0	17.0
10.0	17.0	19.0
9.0	6.0	14.0

input image

$$X = \left( \begin{pmatrix} 3 & 3 & 2 & 1 & 0 \\ 0 & 0 & 1 & 3 & 1 \\ 3 & 1 & 2 & 2 & 3 \\ 2 & 0 & 0 & 2 & 2 \\ 2 & 0 & 0 & 0 & 1 \end{pmatrix} \right) \rightarrow A^{(1)} =$$

$$\left( \begin{pmatrix} 3 & 3 & 2 & 0 & 0 & 1 & 3 & 1 & 2 \\ 3 & 2 & 1 & 0 & 1 & 3 & 1 & 2 & 2 \\ 2 & 1 & 0 & 1 & 3 & 1 & 2 & 2 & 3 \\ 0 & 0 & 1 & 3 & 1 & 2 & 2 & 0 & 0 \\ 0 & 1 & 3 & 1 & 2 & 2 & 0 & 0 & 2 \\ 0 & 1 & 3 & 1 & 2 & 2 & 0 & 0 & 2 \\ 1 & 3 & 1 & 2 & 2 & 3 & 0 & 2 & 2 \\ 3 & 1 & 2 & 2 & 0 & 0 & 2 & 0 & 0 \\ 1 & 2 & 2 & 0 & 0 & 2 & 0 & 0 & 0 \\ 2 & 2 & 3 & 0 & 2 & 2 & 0 & 0 & 1 \end{pmatrix} \right);$$

[9, 9]

$X^* \cdot \Theta^{(1)} = x_{ij}^* \cdot \Theta_{ij}^{(1)} - i=1 \dots 3$

kernel

$$\Theta^{(1)} = \begin{pmatrix} 0 & 1 & 2 \\ 2 & 2 & 0 \\ 0 & 1 & 2 \end{pmatrix} \rightarrow \Theta^{(1)} =$$

$$\begin{pmatrix} 0 \\ 1 \\ 2 \\ 2 \\ 2 \\ 0 \\ 0 \\ 1 \\ 2 \end{pmatrix}$$

[9, 1]

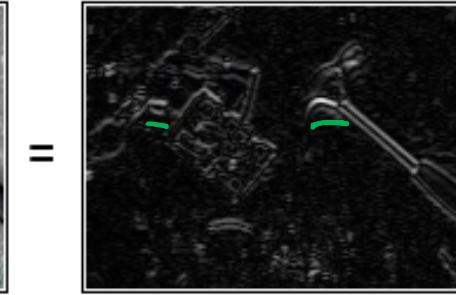
$$\begin{pmatrix} 12 \\ 12 \\ 17 \\ 10 \\ 17 \\ 19 \\ 9 \\ 6 \\ 14 \end{pmatrix}$$

$$\rightarrow Z^{(2)} = \begin{pmatrix} 12 & 12 & 17 \\ 10 & 17 & 19 \\ 9 & 6 & 14 \end{pmatrix}.$$

# Прямые вычисления в СНС / Forward propagation in CNN

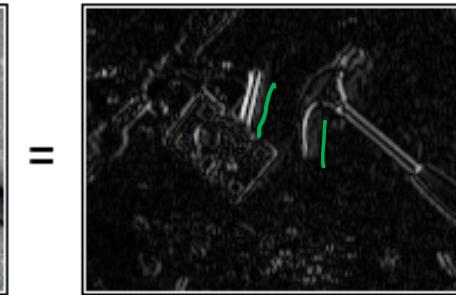
Свертка / Convolution операция использует особенности зрительной коры, в которой простые клетки активируются на простые признаки (например, линии), а сложные на комбинацию активаций простых. СНС связана с математической операцией свертки для понижения размеров матриц. СНС обычно являются глубокими.

$$\Theta^{(1)} = \begin{pmatrix} (-1 & -2 & -1) \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix} \rightarrow$$

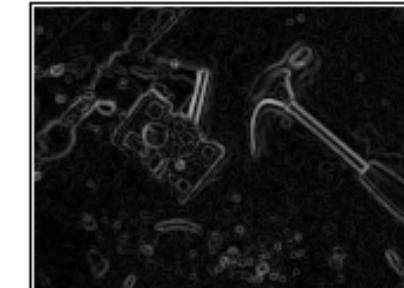
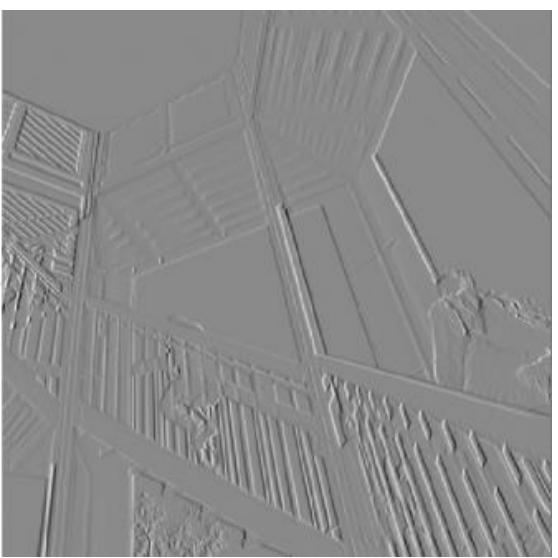


+

$$\Theta^{(1)} = \begin{pmatrix} (-1 & 0 & 1) \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix} \rightarrow$$



↓

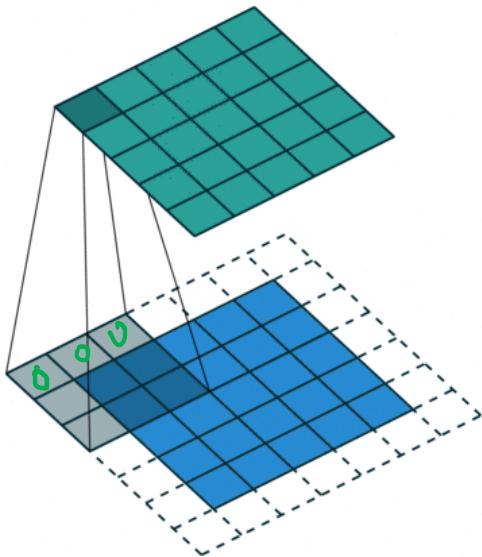


# Прямые вычисления в СНС / Forward propagation in CNN

Свертка / Convolution операция использует особенности зрительной коры, в которой простые клетки активируются на простые признаки (например, линии), а сложные на комбинацию активаций простых. СНС связана с математической операцией свертки для понижения размеров матриц. СНС обычно являются глубокими.

Операцию свертки обычно комбинируют с операциями дополнения, шагания и группирования.

Дополнение / Padding



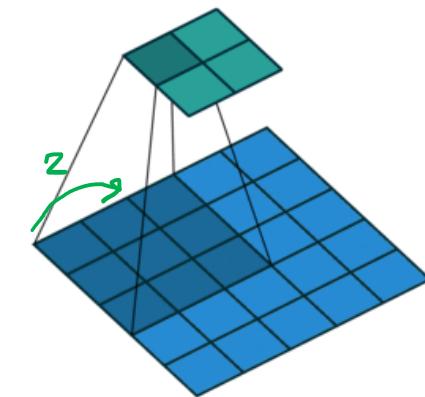
Группирование / Pooling

2	2	7	3
9	4	6	1
8	5	2	4
3	1	2	6

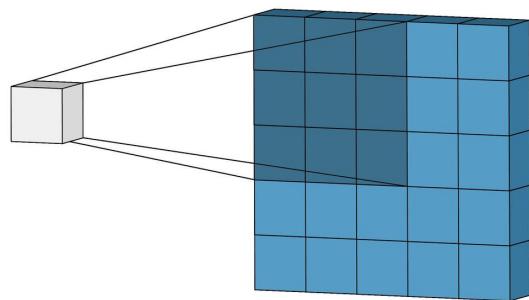
Max Pool  
Filter - (2 x 2)  
Stride - (2, 2)

9	7
8	6

Шагание / Striding



# Прямые вычисления в СНС / Forward propagation in CNN



**Упражнение.** Найти результат свертки изображения цифры «4» размером [3 3]. Kernel  $\Theta^{(1)} = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 0 \end{pmatrix}$ , padding (1), stride [1 1].

(11)

0	0	0	0	0
0	0.5	1	0.5	0
0	0.5	0.5	0.5	0
0	1	1	0.5	0
0	0	0	0	0

$$X = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0.5 & 1 & 0.5 & 0 \\ 0 & 0.5 & 0.5 & 0.5 & 0 \\ 0 & 1 & 1 & 0.5 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$A^{(1)} = \begin{pmatrix} 0 & 0 & 0 & \frac{1}{2} & 1 & 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 0 & \frac{1}{2} & 1 & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 0 & 1 & \frac{1}{2} & 0 & \frac{1}{2} & 0 \\ 0 & \frac{1}{2} & 1 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 1 \\ 0 & \frac{1}{2} & \frac{1}{2} & 0 & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & 0 \\ 0 & \frac{1}{2} & 0 & \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 \\ 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$\Theta^{(1)} = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 0 \end{pmatrix} \rightarrow \Theta^{(1)} =$$

$$\begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 2 \\ 1 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

$$\begin{pmatrix} 2.5 \\ 3.5 \\ 2.5 \\ 3 \\ 4 \\ 2.5 \\ 3.5 \\ 4 \\ 2.5 \end{pmatrix} \rightarrow$$

2.5	3.0	3.5
3.5	4.0	4.0
2.5	2.5	2.5

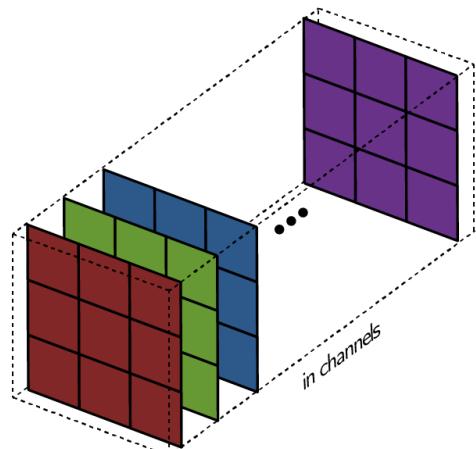
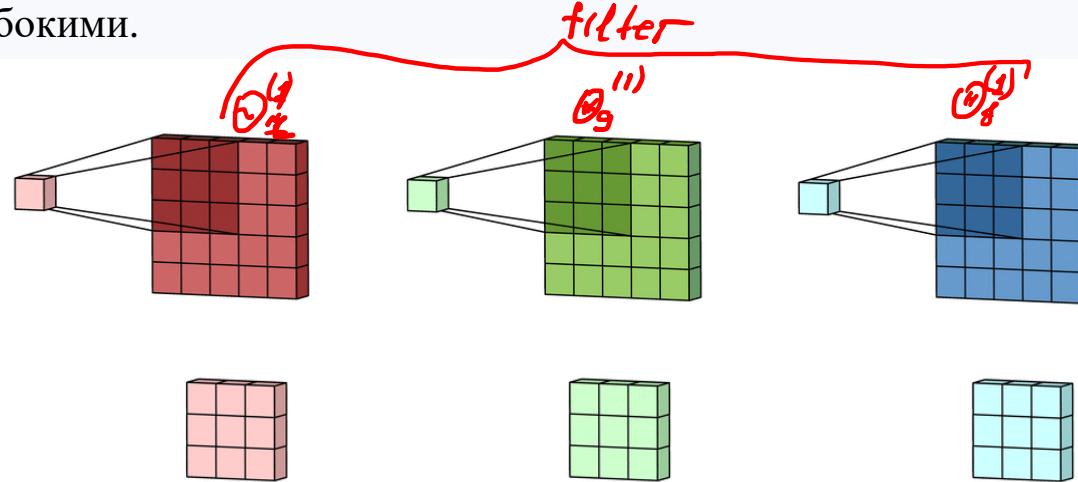
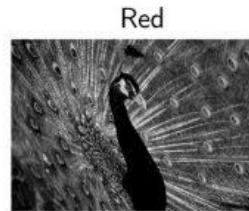
$$Z^{(2)} = \begin{pmatrix} 2.5 & 3.0 & 3.5 \\ 3.5 & 4.0 & 4.0 \\ 2.5 & 2.5 & 2.5 \end{pmatrix}.$$

$$A^{(2)} = \text{activation}(Z^{(2)})$$

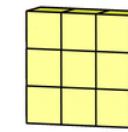
$$A^{(2)} = Z^{(2)} - \text{feature map}$$

# Прямые вычисления в СНС / Forward propagation in CNN

Свертка / Convolution операция использует особенности зрительной коры, в которой простые клетки активируются на простые признаки (например, линии), а сложные на комбинацию активаций простых. СНС связана с математической операцией свертки для понижения размеров матриц. СНС обычно являются глубокими.



$$h = \Theta_0 + \Theta_1 x$$



bias

## Пример свертки.

Дано: изобр. [5 5 3], свертка 2 фильтрами [3 3 3], дополнение (1), шаг [2 2].

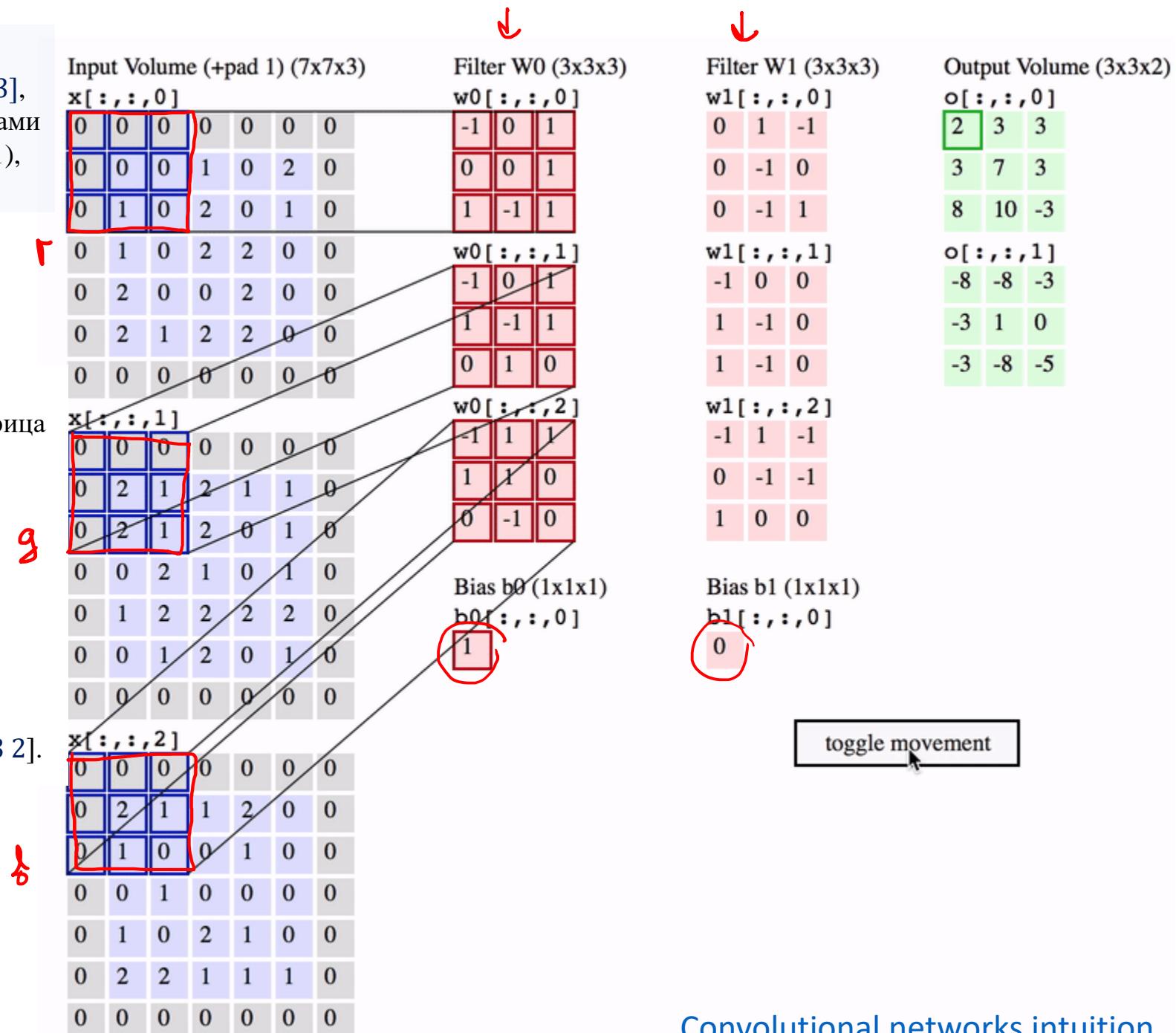
Результатом свертки изображения  $[q q]$  ядром/kernel  $[k k]$ , с дополнением/padding (p) и шагом/stride [s s] является матрица размером  $[r r]$ :  $r = \frac{q-k+2p}{s} + 1 = \frac{5-3+2}{2} + 1 = 3$ .

Количество операций свертки:  $r^2 = 9$ .

$X, [5 5 3] \rightarrow X, [7 7 3] \rightarrow A^{(1)}, [9 2]$ ;

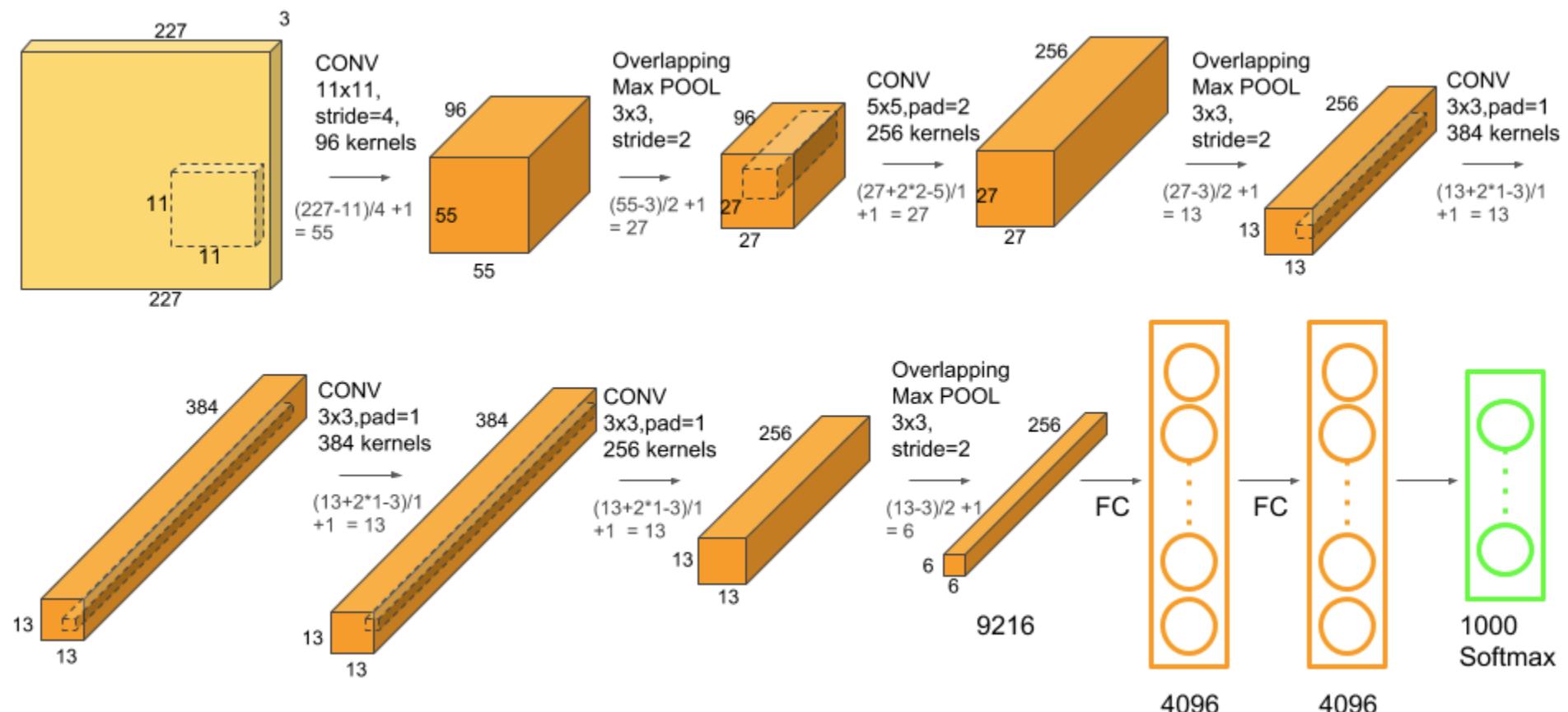
$\Theta^{(1)}, [3 3 3 2] \rightarrow \Theta^{(1)}, [27 2]$ ;

$Z^{(2)} = A^{(1)}\Theta^{(1)}, [9 2] \rightarrow Z^{(2)} = Z^{(2)} + \Theta_0^{(1)}, [9 2] \rightarrow Z^{(2)}, [3 3 2]$ .



# Сверточные нейронные сети / Convolutional NN

AlexNet (designed by Alex Krizhevsky) – глубокая сверточная нейронная сеть для распознавания 1000 классов, признанная лучшей в 2012 г. в конкурсе [ImageNet Large Scale Visual Recognition Challenge](#).

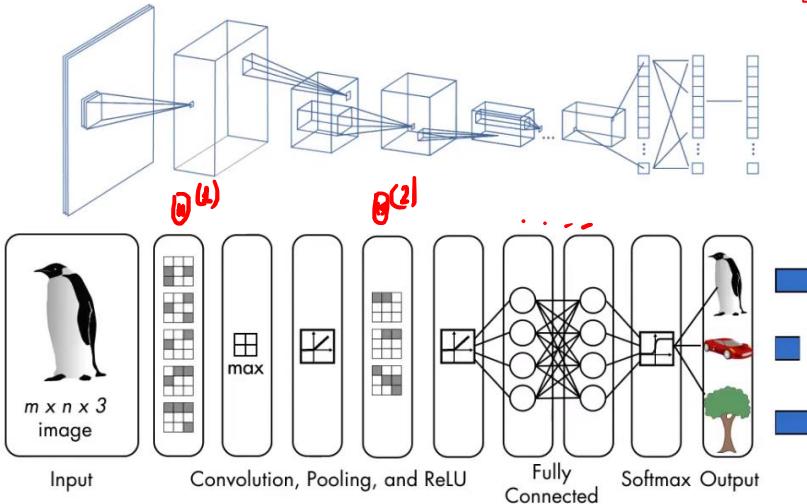


AlexNet architecture

# Обучение СНС / CNN Training algorithm

Поиск наилучшей функциональной зависимости множества  $Y$  от множества  $X$  по критерию минимума некоторой функции качества с помощью **СНС**, с количеством слоев  $l$ , количеством нейронов в  $k$ -ом слое  $n_k$  ( $n_l$  - количество классов), параметризованной весами  $\Theta^{(k)}$ , на основании анализа  $m$  пар значений  $Y, X$ :

$$J(\Theta^{(k)}) = -\frac{1}{m} \sum_{i=1}^m \sum_{j=1}^{n_l} \left( y_j^{(i)} \ln(h_j^{(i)}) + (1 - y_j^{(i)}) \ln(1 - h_j^{(i)}) \right) \Rightarrow \min.$$



## Алгоритм обучения.

1. Задать начальные значения компонент матрицы  $\Theta^{(k)}$  случайным образом.
2. Рассчитать вектор градиента  $\nabla J = \left[ \left[ \partial J / \partial \theta_{ij}^{(k)} \right] \right]$  методом обратного распространения ошибки.
3. Найти новые значения компонент  $\Theta^{(k)} : \theta_{ij}^{(k)H} = \theta_{ij}^{(k)C} - \alpha \frac{\partial J}{\partial \theta_{ij}^{(k)}}.$
4. Повторять пп. 2-3 до достижения минимума  $J : J^H - J^C < \delta$  или #итерации  $> N_{max}$ .
5. Вывод результатов:  $\Theta^{(k)}$ .

## Want to know more?

[Convolutional neural networks](#): видео с визуализацией, на английском языке

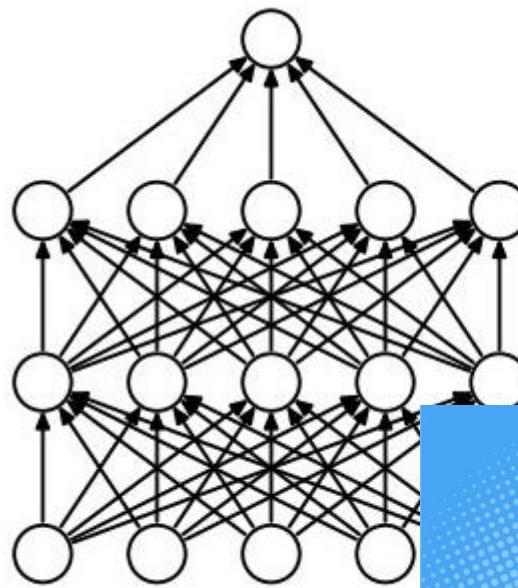
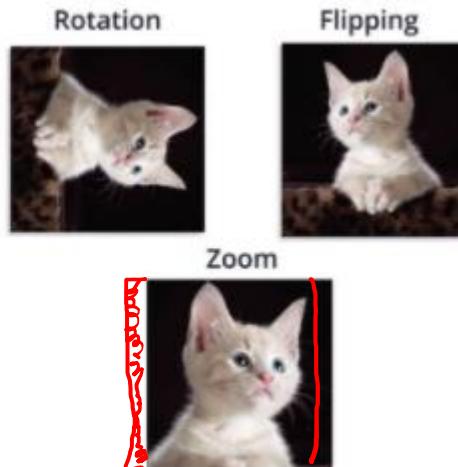
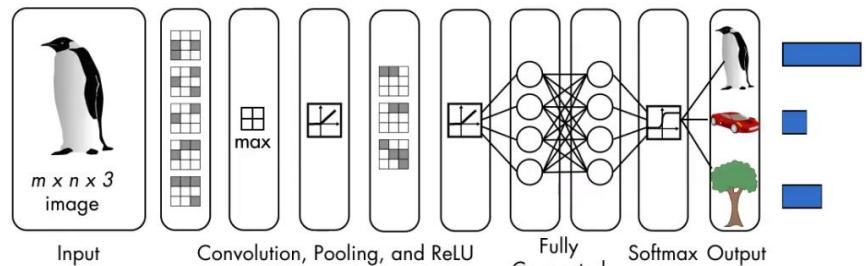
[Convolutions Over Volumes](#): видео фрагмент специализации «Deep Learning» на платформе «Coursera» от проф. Andrew Ng

# Настройка обучения СНС / CNN Training settings

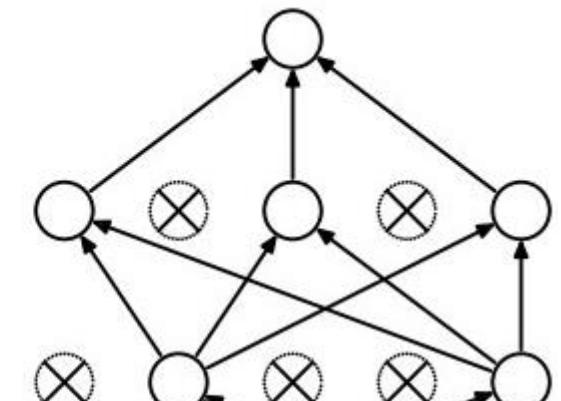
1. Скорость обучения  $\alpha$  / Learning rate
2. Погрешность  $\delta$  и количество итераций  $N_{\max}$  / Error and # of iterations
3. Количество данных для градиентного метода / Batch gradient descent (GD) – Mini-Batch GD. – Stochastic GD
4. Регуляризация / Regularization
5. Аугментация / Augmentation
6. Дропаут / Dropout
7. Трансферное обучение / Transfer learning



[www.element14.com](http://www.element14.com)



<https://habr.com>

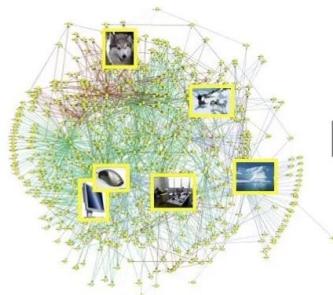


Want to know more?

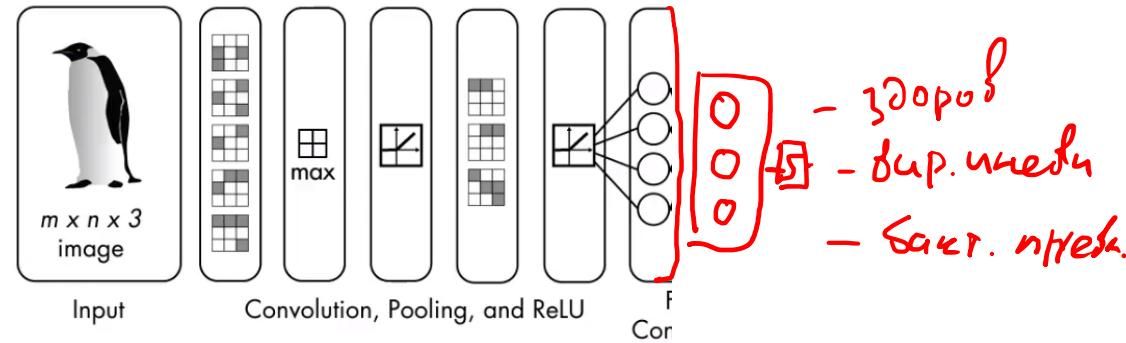
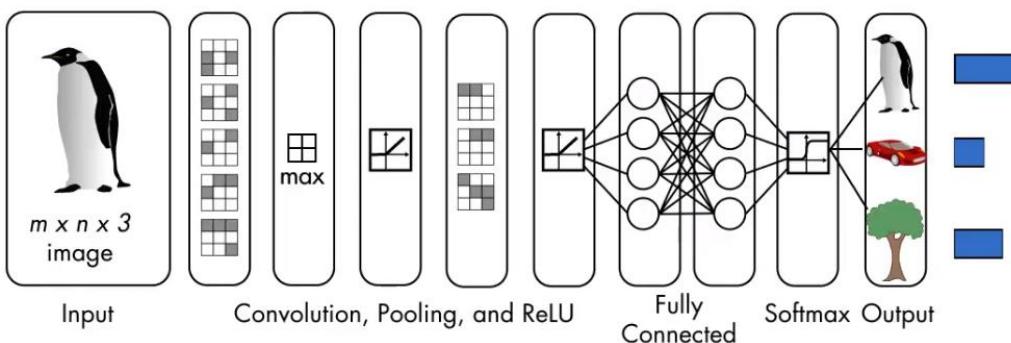
Нейросети на практике (настройка сетей): лекция Семена Козлова (Simon)

# Настройка обучения СНС / CNN Training settings

1. Скорость обучения  $\alpha$  / Learning rate
2. Погрешность  $\delta$  и количество итераций  $N_{\max}$  / Error and # of iterations
3. Количество данных для градиентного метода / Batch gradient descent (GD) – Mini-Batch GD. – Stochastic GD
4. Регуляризация / Regularization
5. Аугментация / Augmentation
6. Дропаут / Dropout
7. Трансферное обучение / Transfer learning



IMGENET

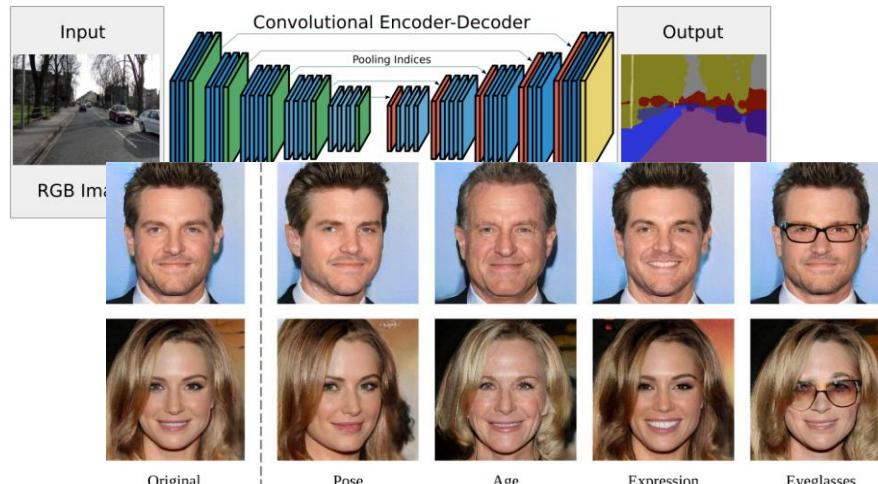
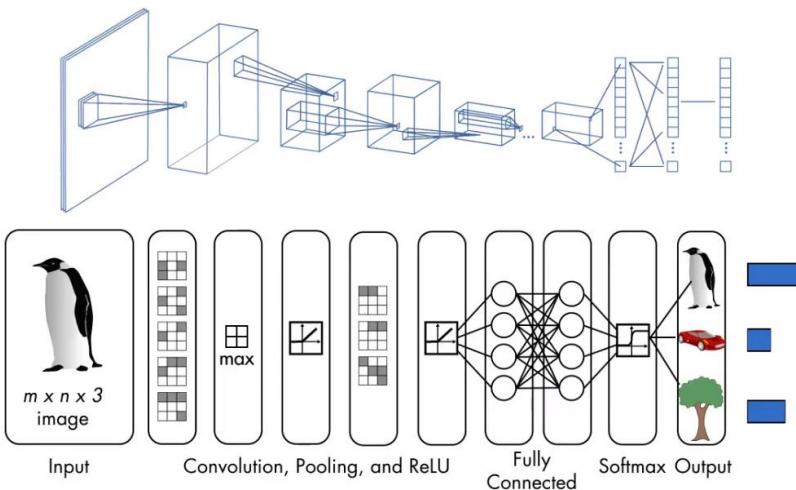


Want to know more?

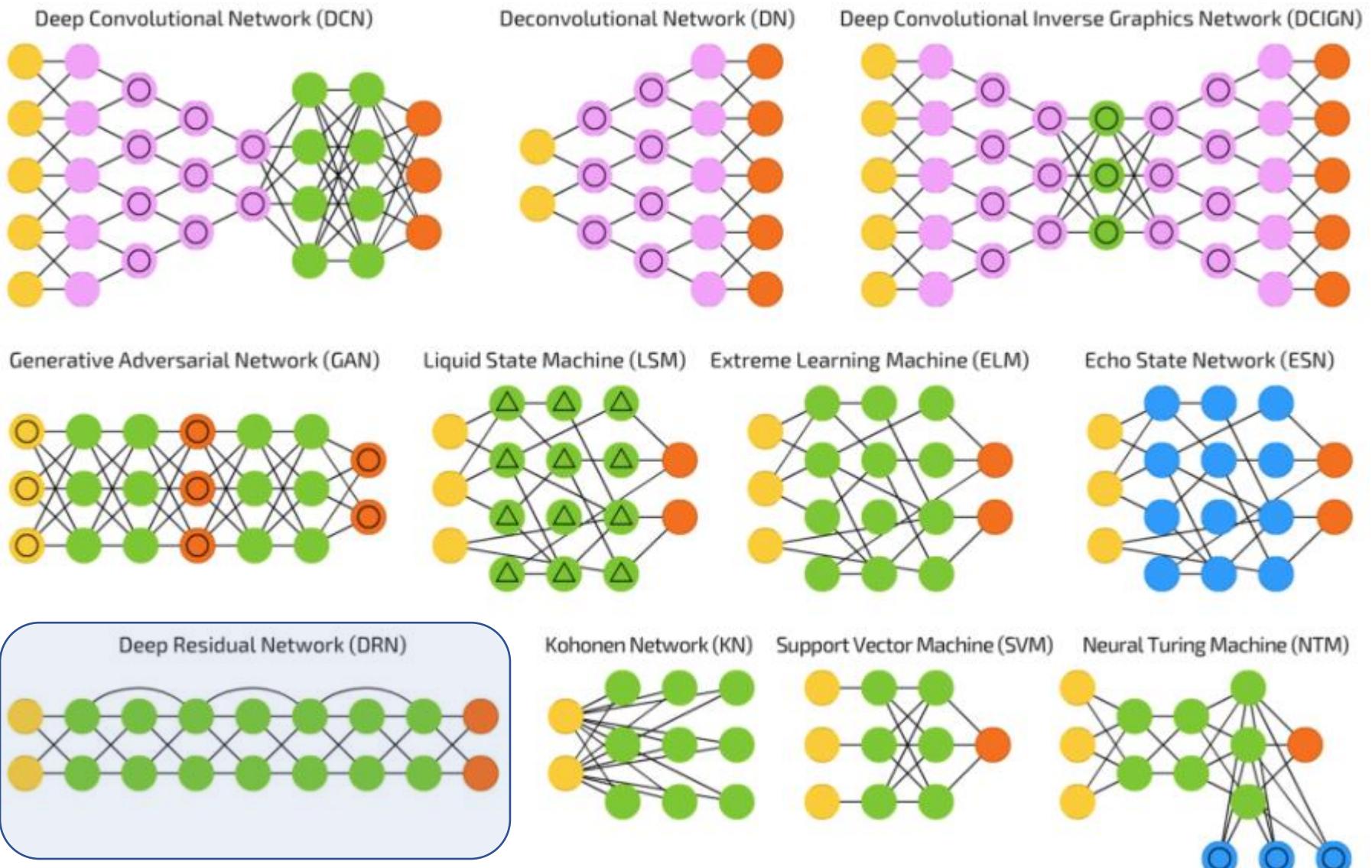
[Нейросети на практике \(настройка сетей\)](#): лекция Семена Козлова (Simon)

# Лекция 4. Глубокое обучение и его приложения / Deep Learning (DL) and its applications

1. Напоминание / Contents of the previous lectures
2. Глубокие сверточные нейронные сети / Deep convolutional neural networks (CNNs, DCNNs)
3. Некоторые архитектуры и приложения глубокого обучения / DL Applications
4. Глубокое обучение с подкреплением / Deep reinforcement learning (DQN, DDPG)
5. Заключение / Conclusion

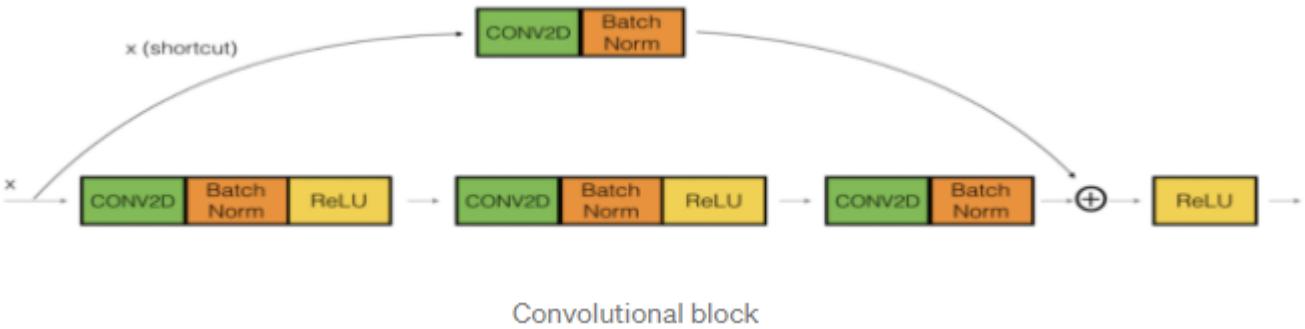
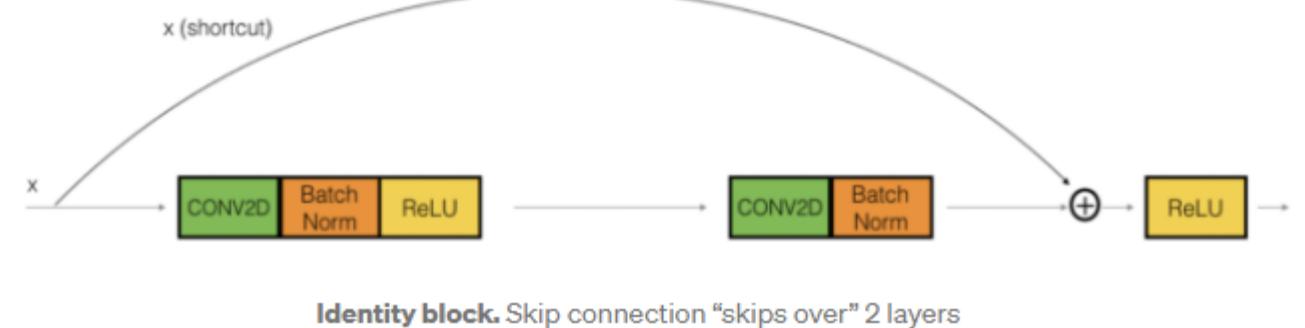
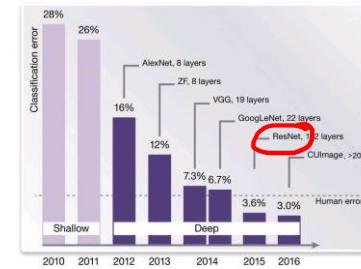


- Backfed Input Cell
- Input Cell
- △ Noisy Input Cell
- Hidden Cell
- Probabilistic Hidden Cell
- △ Spiking Hidden Cell
- Output Cell
- Match Input Output Cell
- Recurrent Cell
- Memory Cell
- △ Different Memory Cell
- Kernel
- Convolution or Pool

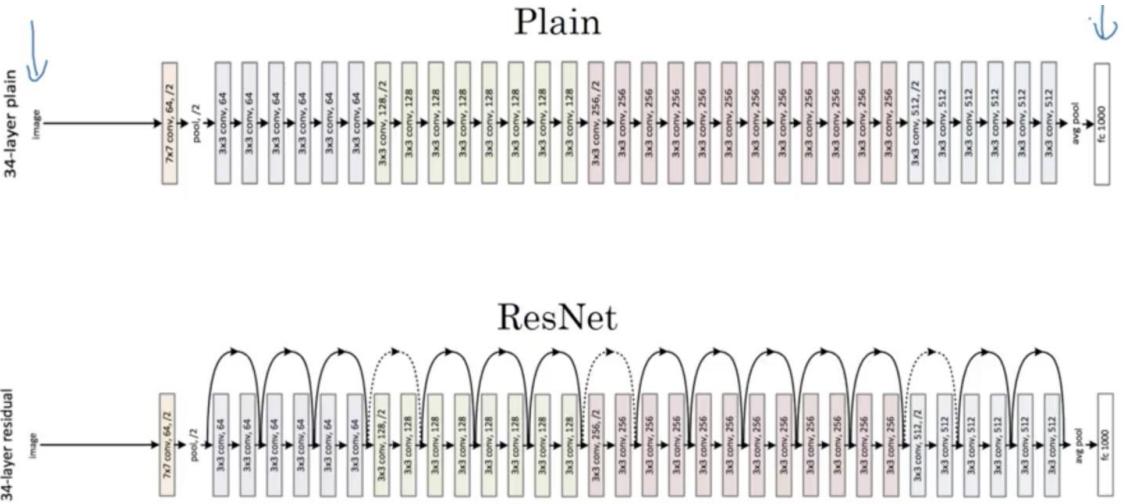


[A mostly complete chart of NN \(2016\)](#)

# Residual neural network (ResNet)

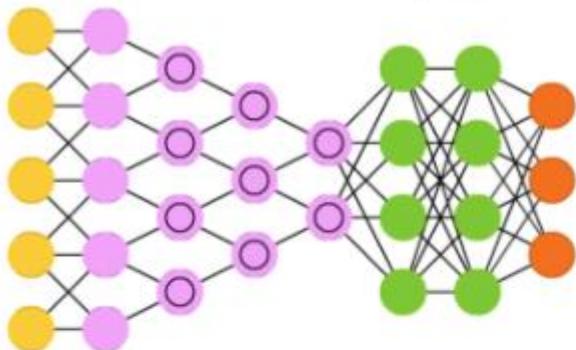


[medium.com](https://medium.com)

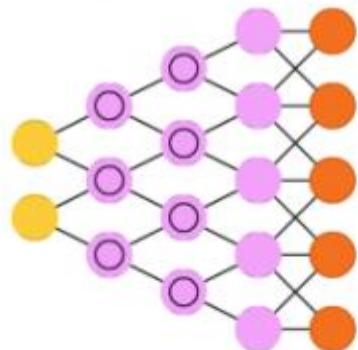


- Backfed Input Cell
- Input Cell
- △ Noisy Input Cell
- Hidden Cell
- Probabilistic Hidden Cell
- △ Spiking Hidden Cell
- Output Cell
- Match Input Output Cell
- Recurrent Cell
- Memory Cell
- △ Different Memory Cell
- Kernel
- Convolution or Pool

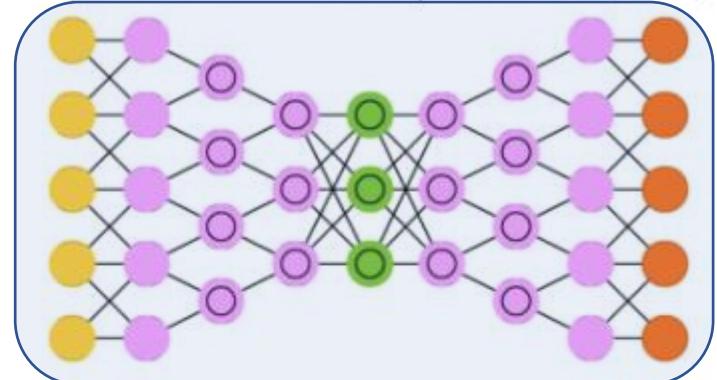
Deep Convolutional Network (DCN)



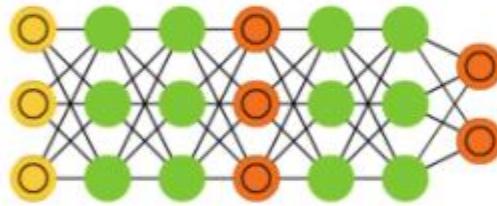
Deconvolutional Network (DN)



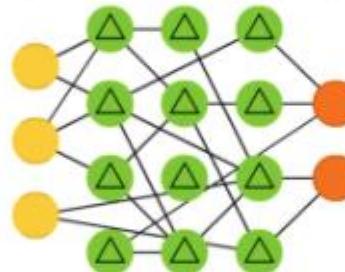
Deep Convolutional Inverse Graphics Network (DCIGN)



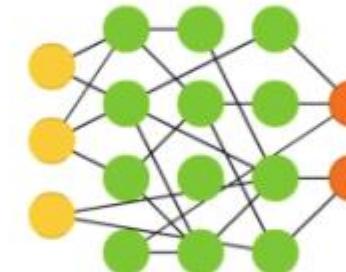
Generative Adversarial Network (GAN)



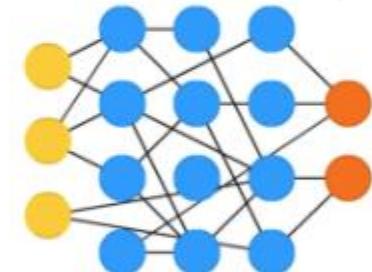
Liquid State Machine (LSM)



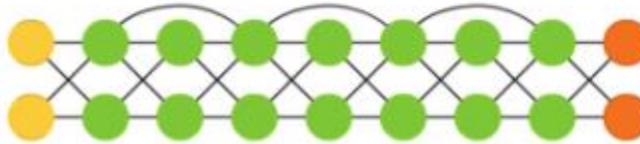
Extreme Learning Machine (ELM)



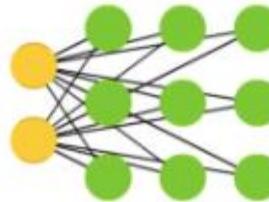
Echo State Network (ESN)



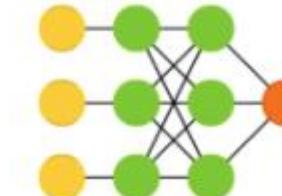
Deep Residual Network (DRN)



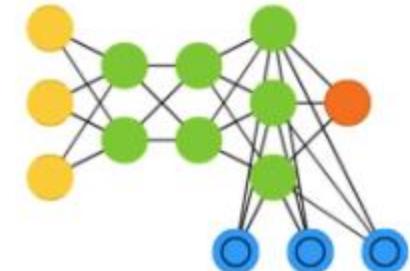
Kohonen Network (KN)



Support Vector Machine (SVM)

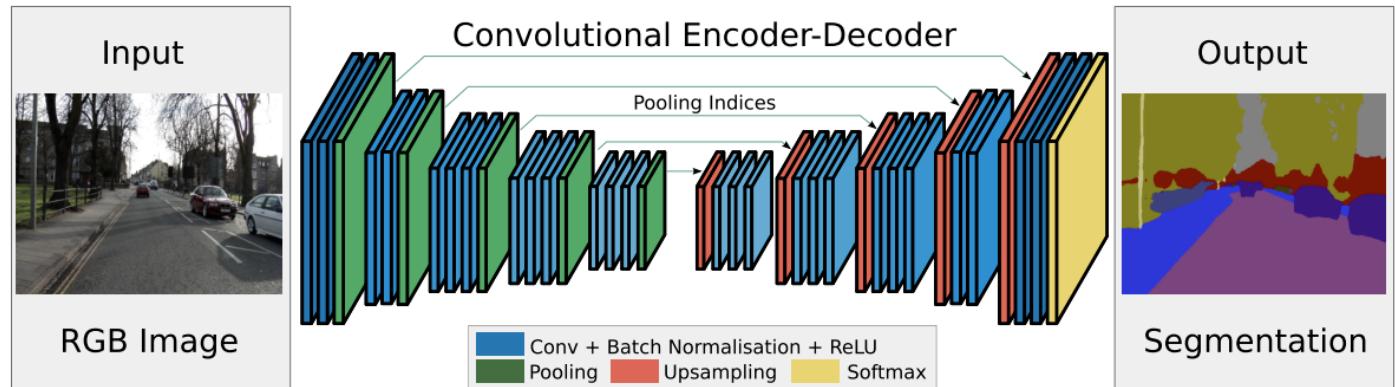
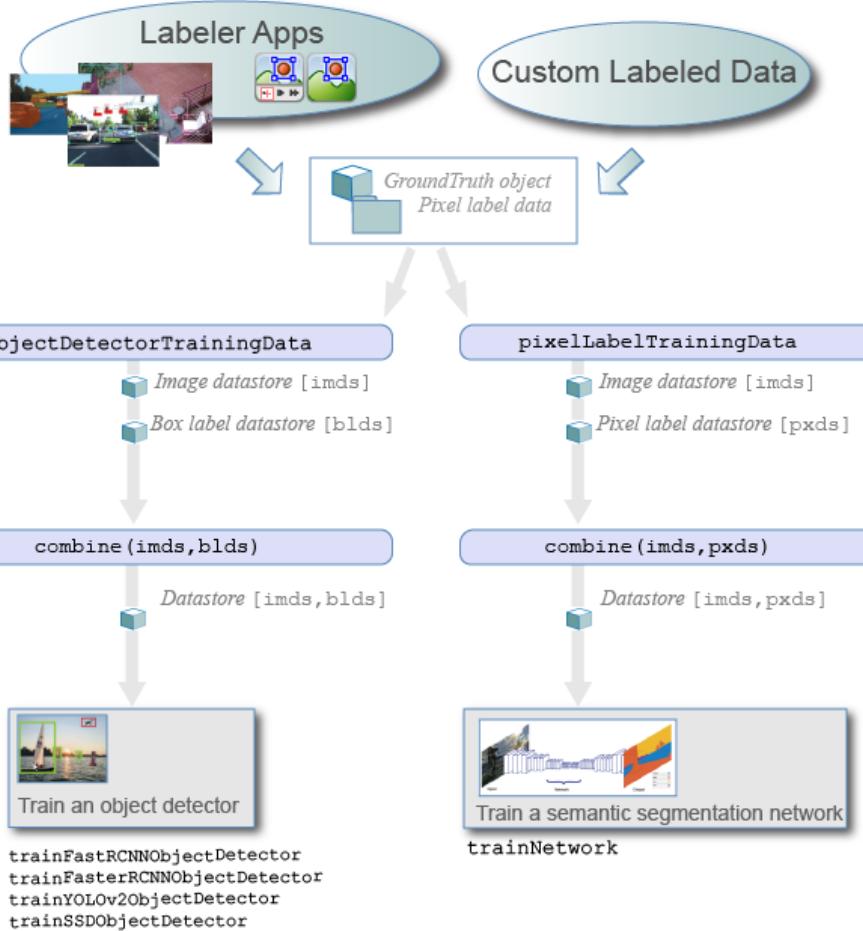


Neural Turing Machine (NTM)

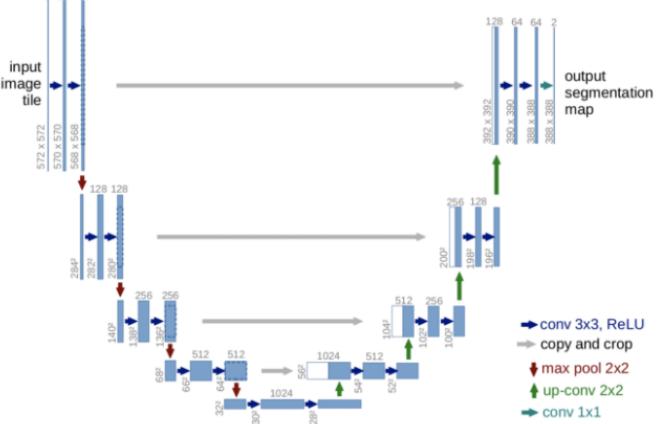


[A mostly complete chart of NN \(2016\)](#)

# Сегментация изображений / Segmentation



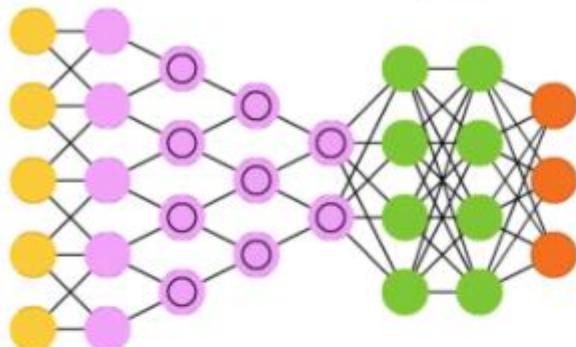
## Review on SegNet



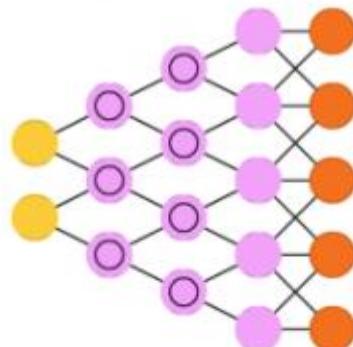
## UNet

- Backfed Input Cell
- Input Cell
- △ Noisy Input Cell
- Hidden Cell
- Probabilistic Hidden Cell
- △ Spiking Hidden Cell
- Output Cell
- Match Input Output Cell
- Recurrent Cell
- Memory Cell
- △ Different Memory Cell
- Kernel
- Convolution or Pool

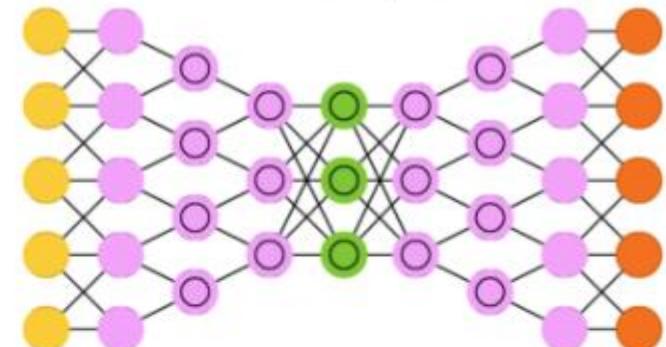
Deep Convolutional Network (DCN)



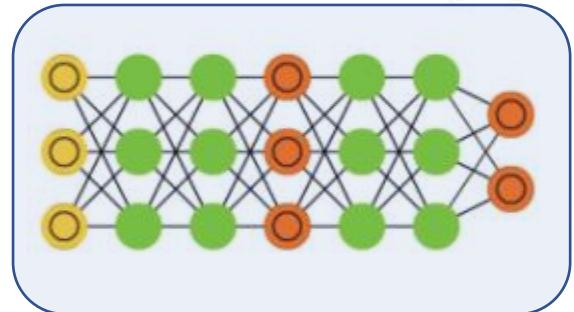
Deconvolutional Network (DN)



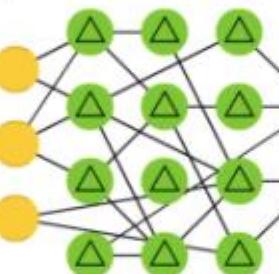
Deep Convolutional Inverse Graphics Network (DCIGN)



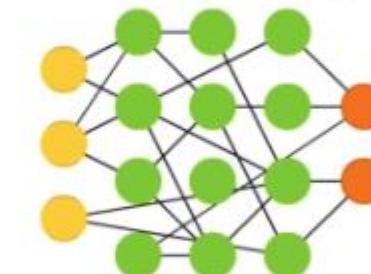
Generative Adversarial Network (GAN)



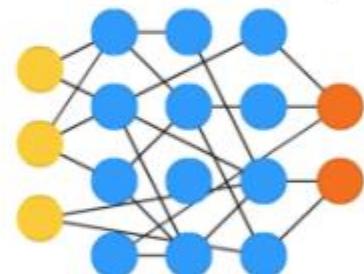
Liquid State Machine (LSM)



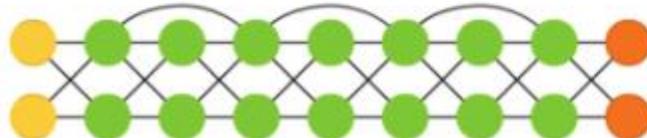
Extreme Learning Machine (ELM)



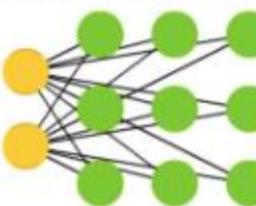
Echo State Network (ESN)



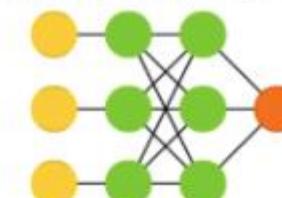
Deep Residual Network (DRN)



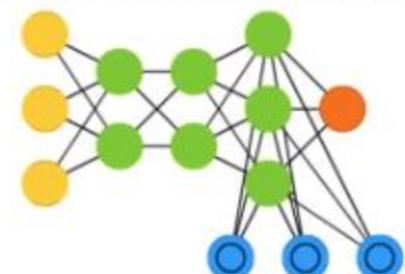
Kohonen Network (KN)



Support Vector Machine (SVM)



Neural Turing Machine (NTM)



[A mostly complete chart of NN \(2016\)](#)

# Генеративно-состязательная сеть / GAN

Генеративно-состязательная сеть / Generative adversarial network основана на алгоритме [машинного обучения без учителя](#), построенный на комбинации из двух [нейронных сетей](#), одна из которых (сеть G) генерирует образцы (см. [Генеративная модель\[en\]](#)), а другая (сеть D) старается отличить правильные («подлинные») образцы от неправильных (см. [Дискриминативная модель\[en\]](#)). Так как сети G и D имеют противоположные цели — создать образцы и отбраковать образцы — между ними возникает [антагонистическая игра](#) [wikipedia.org].



<https://thispersondoesnotexist.com/>

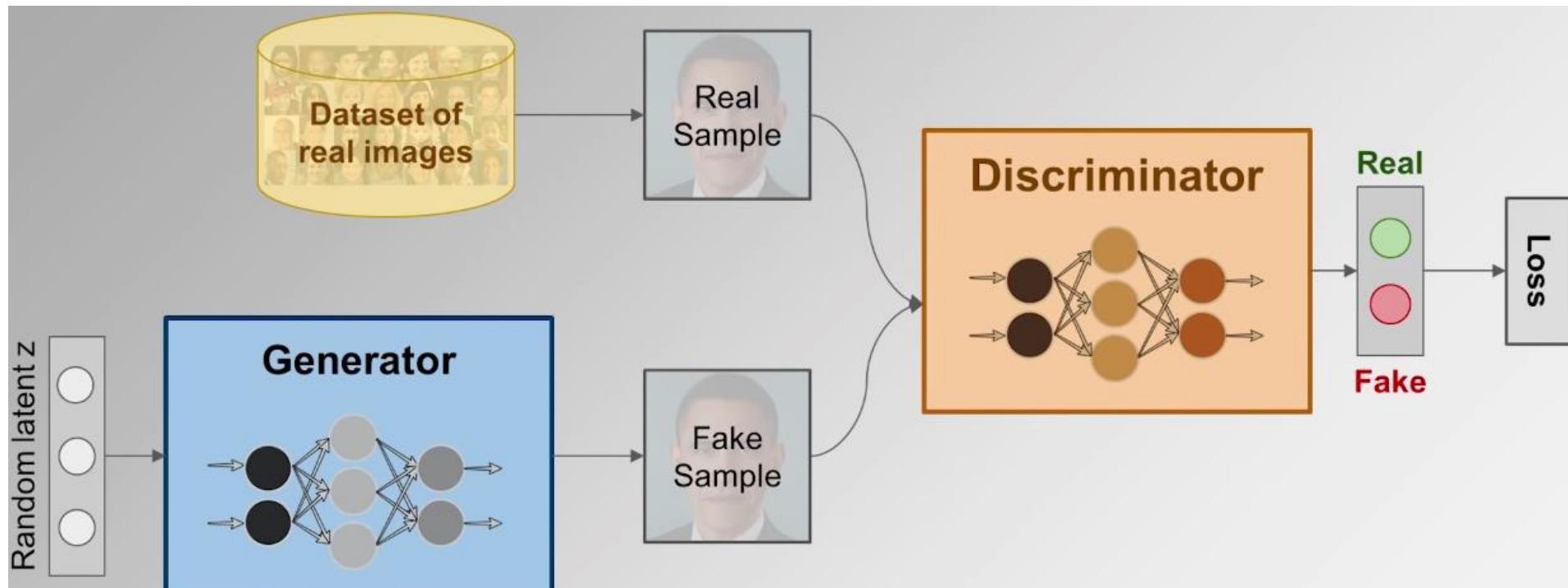


Земля в иллюминаторе

# Генеративно-состязательная сеть / GAN

Генеративно-состязательная сеть / Generative adversarial network основана на алгоритме [машинного обучения без учителя](#), построенный на комбинации из двух нейронных сетей, одна из которых (сеть G) генерирует образцы (см. [Генеративная модель\[en\]](#)), а другая (сеть D) старается отличить правильные («подлинные») образцы от неправильных (см. [Дискриминативная модель\[en\]](#)).

$$\min_G \max_d J(G, d) = E[\ln(d(X))] + E[(\ln(1 - d(G(z)))].$$

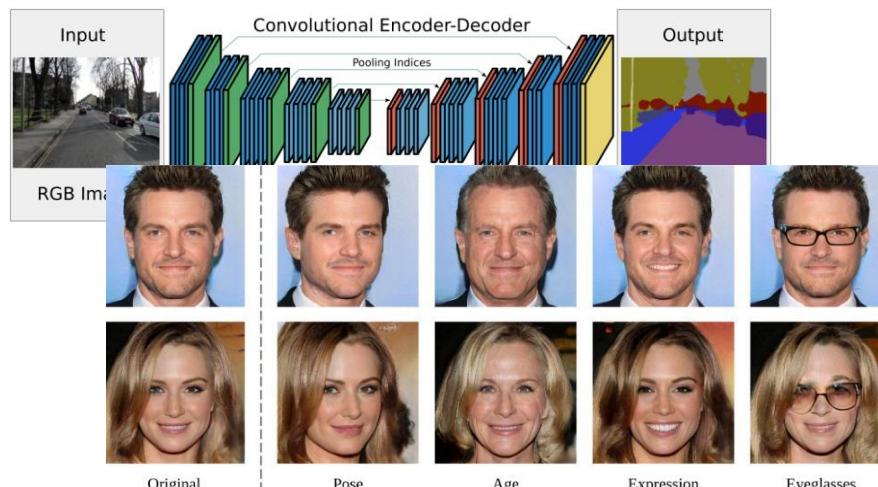
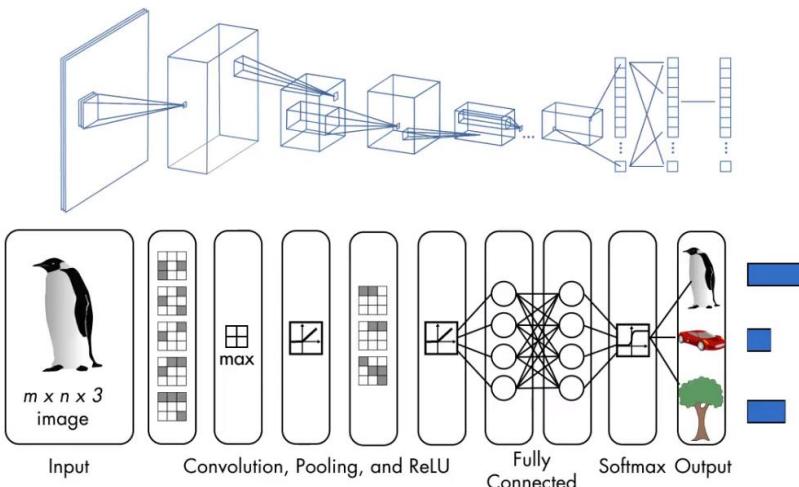


Want to know more?

[Face editing with Generative Adversarial Networks](#) : видео с визуализацией, на английском языке

# Лекция 4. Глубокое обучение и его приложения / Deep Learning (DL) and its applications

1. Напоминание / Contents of the previous lectures
2. Глубокие сверточные нейронные сети / Deep convolutional neural networks (CNNs, DCNNs)
3. Некоторые архитектуры и приложения глубокого обучения / DL Applications
4. Глубокое обучение с подкреплением / Deep reinforcement learning (DQN, DDPG)
5. Заключение / Conclusion



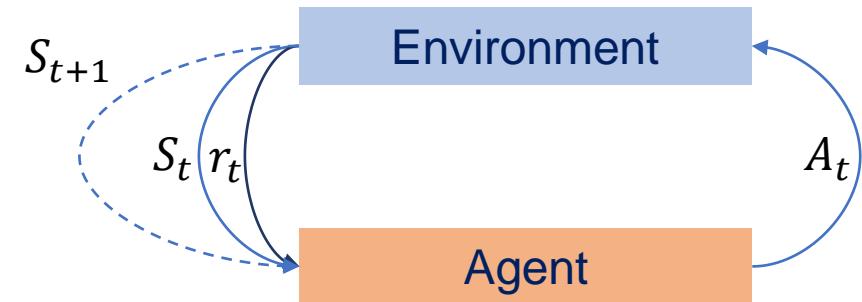
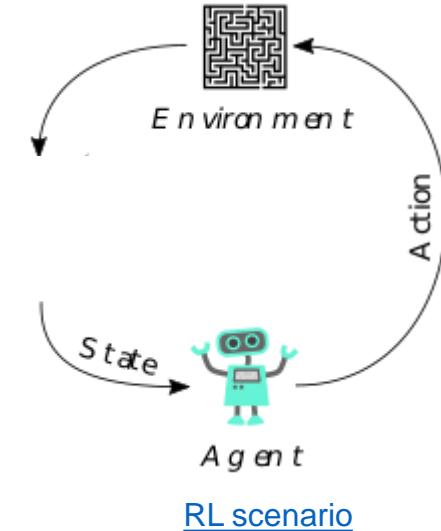
# Напоминание / Contents of the previous lecture

Control systems based on Reinforcement Learning (RL)

**Control or decision process:** at each time step, the controller (agent) receives feedback from the system (environment) in the form of a state signal, and takes an action in response. We supposed that current state completely characterizes the state of the system ([Markov decision process](#)).

**The main problem** is that the correct actions are unknown sometimes.

**The main idea** is to learn agent after the event, giving him higher **reward** for the better **actions**.



**Характеристики модели RL:**

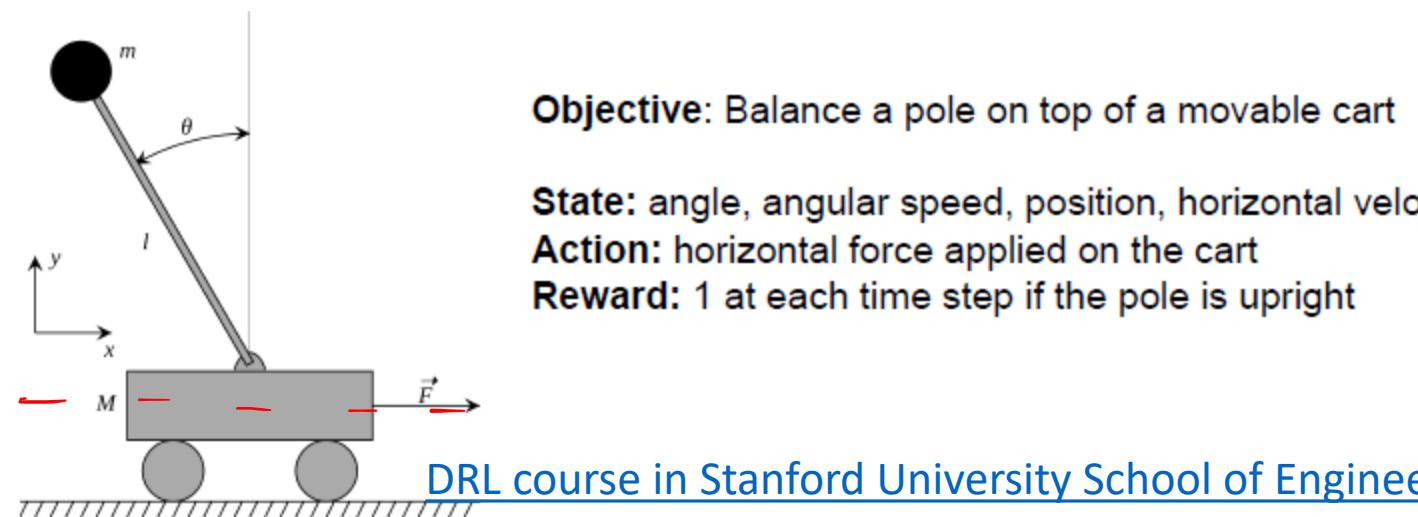
$S_t (s_t)$  – состояние среды (**state**) в момент времени  $t$ ;

$A_t (a_t)$  – действие агента (**action**) в момент времени  $t$ ;

$r_t$  – награда агента (**reward**) в момент времени  $t$ ;

$g_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \dots$  - будущая награда (**return**);

$\gamma$  - дисконт (**discount**).



# Напоминание / Contents of the previous lecture

**Основная идея:** на основании результатов исследования окружающей среды (**environment**) обучить модель (**critic**), которая при данных  $S_t, A_t$  предсказывает будущую награду  $g_t$  для любого возможного действия  $A_{t+1}$ . Тогда агенту (**agent**) при данных  $S_t, A_t$  из множества возможных дальнейших действий  $A_{t+1}$ , следует предпринимать то, которое приведет к наибольшей будущей награде  $g_t$ .

**Формализация.** Наилучшее действие  $q_t^*$  в момент времени  $t$  описывается уравнением Беллмана:

$$q_t^*(S_t, A_t) = r_t + \gamma \max_A [q_{t+1}(S_{t+1}, A_{t+1})]. \quad (3)$$

Процесс накопления опыта состоит в проигрывании **эпизодов**. В процессе обучения необходимо достичь минимизации ошибки между обучаемой функцией  $Q(S, A)$  и оптимальной:

$$Q(S, A) - Q^*(S, A) \Rightarrow \min.$$

Тогда в результате обучения агент будет способен производить действия с максимальной наградой.

## Алгоритм обучения [[MATLAB Documentation](#)]/ Training algorithm.

Инициализировать  $Q(S, A)$  случайными значениями или нулями. Задать гиперпараметры: вероятность  $\epsilon$ , скорость обучения  $\alpha$ .

Для каждого эпизода обучения:

1. Получить данные о начальном состоянии  $S_t$  ( $t = 1$ )

2. Повторять для каждого шага  $t$  до достижения **terminal state**:

2.1 Для текущего состояния  $S_t$  выбрать случайное действие  $A_t$  с вероятностью  $\epsilon$  (может уменьшаться в процессе обучения), иначе действие, для которого значение наибольшее:  $A_t = \max_A [Q_t(S_t, A_t)]$

2.2 Выполнить действие  $A_t$ , получить награду  $r_t$  и данные о новом состоянии  $S_{t+1}$ .

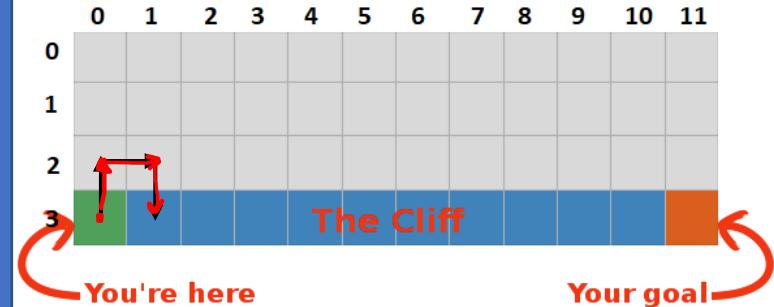
2.3 Если новое состояние  $S_{t+1}$  терминальное, то установить значение целевой функции  $y_t = r_t$ . Иначе  $y_t = r_t + \gamma \max_A [q_{t+1}(S_{t+1}, A_{t+1})]$

2.4 Обновить компоненту матрицы  $Q(S, A)$ :

$$q(S_t, A_t) = q(S_t, A_t) + \alpha [y_t - q(S_t, A_t)].$$

UP												
0	U: -6.76	U: -6.70	U: -6.42	U: -6.14	U: -5.82	U: -5.51	U: -5.12	U: -4.58	U: -4.01	U: -3.57	U: -2.65	U: -2.26
	D: -6.73	D: -6.74	D: -6.53	D: -6.09	D: -5.77	D: -5.37	D: -4.97	D: -4.49	D: -4.02	D: -3.37	D: -2.69	D: -1.90
1	R: -6.75	R: -6.60	R: -6.34	R: -6.06	R: -5.74	R: -5.36	R: -4.94	R: -4.49	R: -3.94	R: -3.36	R: -2.65	R: -2.07
	L: -6.71	L: -6.62	L: -6.34	L: -6.12	L: -5.78	L: -5.53	L: -5.32	L: -4.69	L: -4.28	L: -3.70	L: -3.01	L: -2.15
2	U: -6.81	U: -6.80	U: -6.51	U: -6.17	U: -5.76	U: -5.55	U: -4.73	U: -4.37	U: -3.92	U: -3.80	U: -2.84	U: -1.72
	D: -6.96	D: -6.71	D: -6.43	D: -6.08	D: -5.68	D: -5.21	D: -4.68	D: -4.09	D: -3.44	D: -2.71	D: -1.90	D: -1.00
3	R: -6.89	R: -6.70	R: -6.45	R: -6.09	R: -5.68	R: -5.21	R: -4.68	R: -4.09	R: -3.44	R: -2.71	R: -1.90	R: -1.43
	L: -6.89	L: -6.75	L: -6.67	L: -6.39	L: -5.98	L: -5.63	L: -4.92	L: -4.23	L: -3.98	L: -2.93	L: -2.34	L: -2.27
4	U: -7.06	U: -6.96	U: -6.71	U: -6.37	U: -6.09	U: -5.60	U: -5.07	U: -4.60	U: -4.07	U: -3.34	U: -2.64	U: -1.72
	D: -7.40	D: -99.95	D: -93.75	D: -96.88	D: -99.61	D: -99.22	D: -99.22	D: -96.88	D: -98.44	D: -98.44	D: 0.00	
5	R: -6.86	R: -6.51	R: -6.13	R: -5.70	R: -5.22	R: -4.69	R: -4.10	R: -3.44	R: -2.71	R: -1.90	R: -1.00	R: -1.00
	L: -7.14	L: -7.15	L: -6.86	L: -6.16	L: -6.12	L: -5.68	L: -5.18	L: -4.07	L: -3.98	L: -3.35	L: -2.63	L: -1.81
6	U: -7.18	U: 0.00	U: 0.00	U: 0.00								
	D: -7.46	D: 0.00	D: 0.00	D: 0.00								
7	R: -99.22	R: 0.00	R: 0.00	R: 0.00								
	L: -7.45	L: 0.00	L: 0.00	L: 0.00								

# Напоминание / Contents of the previous lectures



## Алгоритм обучения [MATLAB Documentation]

Инициализировать  $Q(S, A)$  случайными значениями или нулями. Задать гиперпараметры: вероятность  $\epsilon$ , скорость обучения  $\alpha$ , дисконт  $\gamma$ .

Для каждого эпизода обучения:

1. Получить данные о начальном состоянии  $S_t$  ( $t = 1$ )
2. Повторять для каждого шага  $t$  до достижения terminal state:

2.1 Для текущего состояния  $S_t$  выбрать случайное действие  $A_t$  с вероятностью  $\epsilon$  (может уменьшаться в процессе обучения), иначе действие, для которого значение наибольшее:  $A_t = \max_A [Q_t(S_t, A_t)]$

2.2 Выполнить действие  $A_t$ , получить награду  $r_t$  и данные о новом состоянии  $S_{t+1}$ .

2.3 Если новое состояние  $S_{t+1}$  терминальное, то установить значение целевой функции  $y_t = r_t$ . Иначе  $y_t = r_t + \gamma \max_A [q_{t+1}(S_{t+1}, A_{t+1})]$

2.4 Обновить компоненту матрицы  $Q(S, A)$ :

$$q(S_t, A_t) = q(S_t, A_t) + \alpha [y_t - q(S_t, A_t)].$$

	$a^{(1)}$ ↑	$a^{(2)}$ ↓	$a^{(3)}$ →	$a^{(4)}$ ←
$s^{(0,0)}$	0	0	0	0
$s^{(0,1)}$	0	0	0	0
...	0	0	0	0
$s^{(2,0)}$	0	0	-0.1	0
$s^{(2,1)}$	0	-90	0	0
$s^{(2,2)}$	0	0	0	0
...	0	0	0	0
$s^{(3,0)}$	-0.1	0	0	0
$s^{(3,1)}$	0	0	0	0
$s^{(3,2)}$	0	0	0	0
...	0	0	0	0
$s^{(3,11)}$	0	0	0	0

The  $Q$  table aims to optimal

**Пример Q-обучения.** Задача о прогулке по скале: необходимо найти кратчайший путь до цели, не упав с обрыва [<https://habr.com/ru/post/443240/>]. Награда за обычный шаг -1, за достижение цели +0, за срыв с обрыва -100. Дисконт 0.9.  $S = \llbracket s^{(i,j)} \rrbracket$ ,  $A = \llbracket a^{(k)} \rrbracket \rightarrow Q = \llbracket q^{(i,j,k)} \rrbracket$ .  $Q$ ?

11

## Реализация алгоритма.

Инициализация  $Q(S, A)$  нулями. Задание:  $\epsilon = 1, \alpha = 0.1, \gamma = 0.9$ .

Эпизод 1.  $\epsilon = 1$ , тогда все действия 1 эпизода случайные.

1.  $t = 1. \epsilon = 1$  (все действия случайные),  $s_t = s^{(3,0)}$ .
  - 2.1 Выбор сл. действия  $a_t = a^{(1)}$ .
  - 2.2 Вып. действия  $a_t = a^{(1)}$ , награда  $r_t = -1, s_{t+1} = s^{(2,0)}$ .
  - 2.3 Расчет цел. функц.  $y_t = -1 + \gamma \max_A [0,0,0] = -1$ .
  - 2.4 Обновление  $q^{(3,0,1)} = 0 + 0.1[-1 - 0] = -0.1$ .

$t = 2. \epsilon = 1$  (все действия случайные),  $s_t = s^{(2,0)}$ .

- 2.1 Выбор сл. действия  $a_t = a^{(3)}$ .
- 2.2 Вып. действия  $a_t = a^{(3)}$ , награда  $r_t = -1, s_{t+1} = s^{(2,1)}$ .
- 2.3 Расчет цел. функц.  $y_t = -1 + \gamma \max_A [0,0,0] = -1$ .
- 2.4 Обновление  $q^{(2,0,3)} = 0 + 0.1[-1 - 0] = -0.1$ .

$t = 3. \epsilon = 1$  (все действия случайные),  $s_t = s^{(2,1)}$ .

- 2.1 Выбор сл. действия  $a_t = a^{(2)}$ .
- 2.2 Вып. действия  $a_t = a^{(2)}$ , награда  $r_t = -100, s_{t+1} = s^{(3,1)}$ .
- 2.3 Расчет цел. функц.  $y_t = -100$  (терминальное состояние).
- 2.4 Обновление  $q^{(2,1,2)} = 0 + 0.1[-100 - 0] = -90$ .

# Глубокое Q-обучение / Deep Q-Learning

Для оценки функции будущей награды  $g_t$  агент включает 2 аппроксиматора: критик  $q(S, A)$ , который предсказывает награду, и целевой критик  $q'(S, A)$ , который с целью повышения стабильности оптимизации периодически обновляется (копирует) критика. Оба критика имеют одинаковую структуру и начальные параметры.

**Алгоритм обучения** [[MATLAB Documentation](#)] Инициализировать  $q(S, A)$  случайными весами  $\Theta^{(k)}$ , те же значения весов передать целевому критику  $q'(S, A)$ :  $\Theta'^{(k)} = \Theta^{(k)}$ . Задать гиперпараметры: вероятность  $\epsilon$ , скорость обучения  $\alpha$ , дисконт  $\gamma$ .

Повторять для каждого шага  $t$ :

1. Для текущего состояния  $S_t$  выбрать случайное действие  $A_t$  с вероятностью  $\epsilon$  (может уменьшаться в процессе обучения), иначе действие, для которого значение наибольшее:  
$$A_t = \max_A [q_t(S_t, A_t | \Theta^{(k)})]$$
2. Выполнить действие  $A_t$ , получить награду  $r_t$  и данные о новом состоянии  $S_{t+1}$ .
3. Записать данные  $(S_t, A_t, r_t, S_{t+1})$  в буфер для формирования датасета.
4. Выбрать из буфера случайный мини-батч из  $m$  экземпляров  $(S_t, A_t, r_t, S_{t+1})$
5. Если новое состояние  $S_{t+1}$  терминальное, то установить значение целевой функции  $y_t = r_t$ . Иначе  $y_t = r_t + \gamma \max_A [q_{t+1}(S_{t+1}, A_{t+1})]$
6. Рассчитать целевую функцию  $L(\Theta^{(k)}) = \frac{1}{m} \sum_{i=1}^m (y_i - q(S_i, A_i | \Theta^{(k)}))^2$  и вектор градиента  $\nabla L$ , обновить параметры  $\Theta^{(k)}$ :  
$$\theta_{ij}^{(k)H} = \theta_{ij}^{(k)C} - \alpha \frac{\partial L}{\partial \theta_{ij}^{(k)}}$$

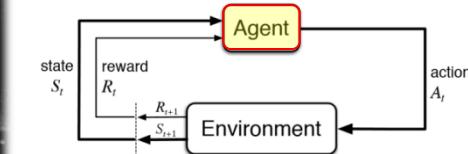


# Deep Deterministic Policy Gradient (DDPG)

The DDPG Agent includes two networks:

<- the Critic

and the Actor ->



## Deep Deterministic Policy Gradient (DDPG) Agent:

Randomly initialize critic network  $Q(s, a|\theta^Q)$  and actor  $\mu(s|\theta^\mu)$  with weights  $\theta^Q$  and  $\theta^\mu$ .

Initialize target network  $Q'$  and  $\mu'$  with weights  $\theta^{Q'} \leftarrow \theta^Q, \theta^{\mu'} \leftarrow \theta^\mu$

Initialize replay buffer  $R$

for episode = 1, M do

    Initialize a random process  $\mathcal{N}$  for action exploration

    Receive initial observation state  $s_1$

    for t = 1, T do

        Select action  $a_t = \mu(s_t|\theta^\mu) + \mathcal{N}_t$  according to the current policy and exploration noise

        Execute action  $a_t$  and observe reward  $r_t$  and observe new state  $s_{t+1}$

        Store transition  $(s_t, a_t, r_t, s_{t+1})$  in  $R$

        Sample a random minibatch of  $N$  transitions  $(s_i, a_i, r_i, s_{i+1})$  from  $R$

        Set  $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$

        Update critic by minimizing the loss:  $L = \frac{1}{N} \sum_i (y_i - Q(s_i, a_i|\theta^Q))^2$

        Update the actor policy using the sampled policy gradient:

$$\nabla_{\theta^\mu} J \approx \frac{1}{N} \sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu} \mu(s|\theta^\mu)|_{s_i}$$

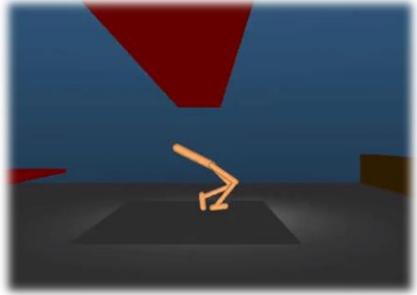
    Update the target networks:

$$\theta^{Q'} \leftarrow \tau \theta^Q + (1 - \tau) \theta^{Q'}$$

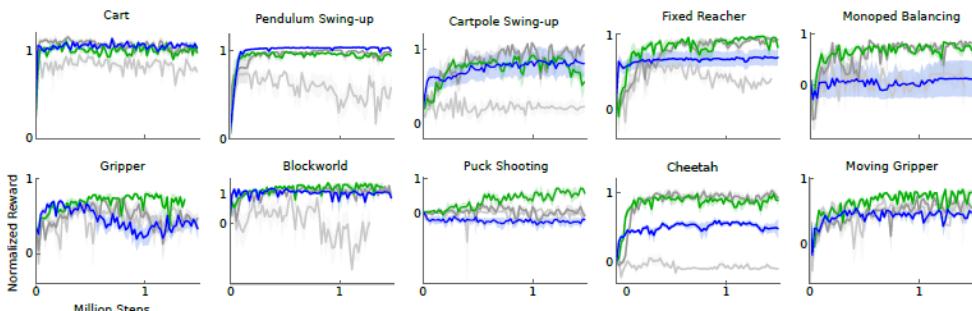
$$\theta^{\mu'} \leftarrow \tau \theta^\mu + (1 - \tau) \theta^{\mu'}$$

    end for

end for

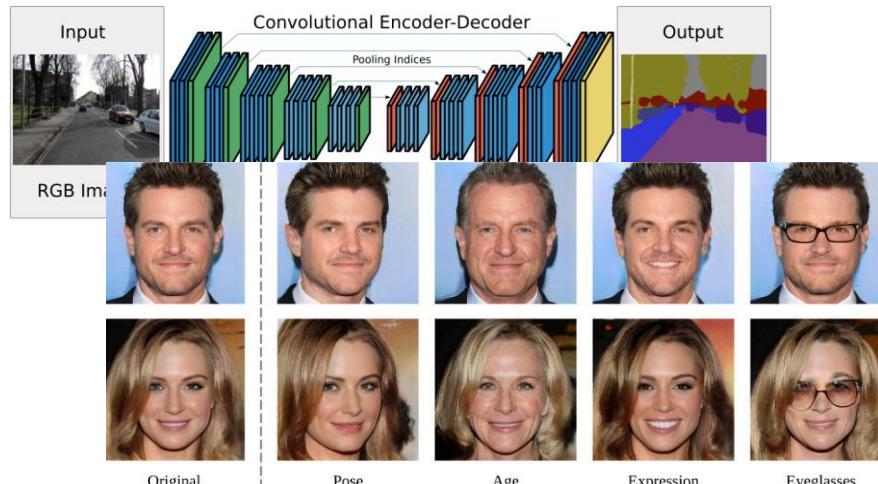
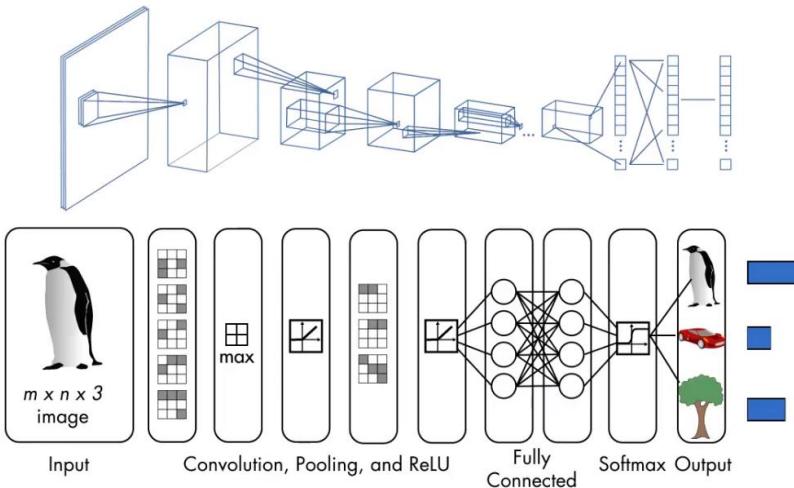


DeepMind walking



# Лекция 4. Глубокое обучение и его приложения / Deep Learning (DL) and its applications

1. Напоминание / Contents of the previous lectures
2. Глубокие сверточные нейронные сети / Deep convolutional neural networks (CNNs, DCNNs)
3. Некоторые архитектуры и приложения глубокого обучения / DL Applications
4. Глубокое обучение с подкреплением / Deep reinforcement learning (DQN, DDPG)
5. Заключение / Conclusion



# Полезные ссылки / References, Links

## Онлайн курсы, обучающие ресурсы:

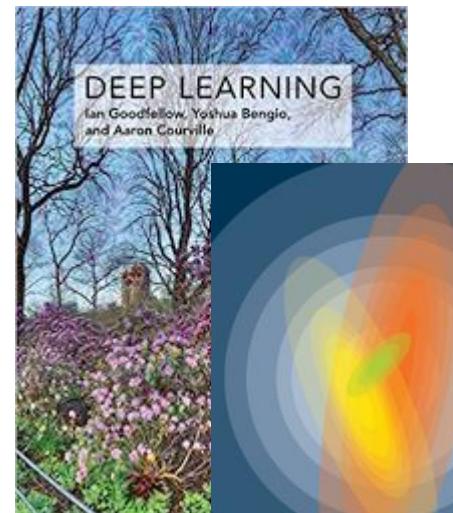
- [Neural Networks](#): серия видео с отличной визуализацией с канала «3Blue1Brown» на английском языке, плюс видео с русским дублированием на смежном канале;
- [Andrew Ng – Machine Learning](#): 11-недельный бесплатный интерактивный курс на английском языке по основам машинного обучения на платформе «Coursera»;
  - [sim0nsays – Deep Learning на пальцах](#): бесплатный видео курс лекций на русском языке, плюс на канале есть ссылка на ресурс с полным курсом и заданиями;
- [RL Course by David Silver](#): курс из 10 лекций Д.Силвера «Введение в обучение с подкреплением»
- [Stanford CS234: Reinforcement Learning](#): курс лекций Стэнфордского университета
- [Kaggle](#): курсы, базы данных для машинного обучения (дата сеты), соревнования на английском языке.

## Книги, статьи:

[Deep Learning](#) by Ian Goodfellow and Yoshua Bengio and Aaron Courville, 2016.

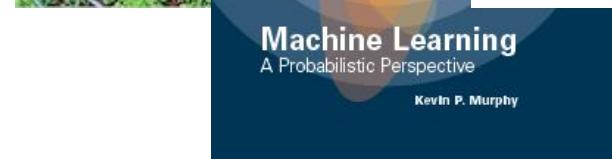
[Pattern Recognition and Machine Learning](#) by C.M. Bishop, 2006.

[Machine Learning: A Probabilistic Perspective](#) by K.P. Murphy, 2012.



Christopher M. Bishop

Pattern Recognition and  
Machine Learning



Springer

[Elsevier](#), [Springer](#): поисковые системы статей крупнейших издательств

[SJR](#), [WoS](#): поисковые системы журналов, рейтинг журналов