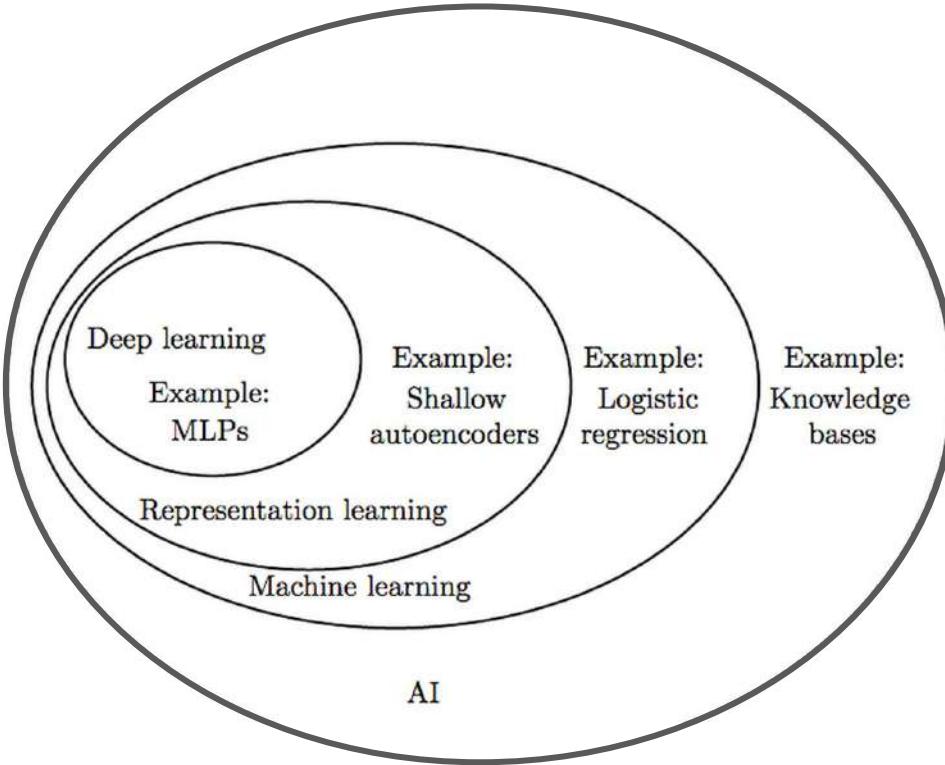
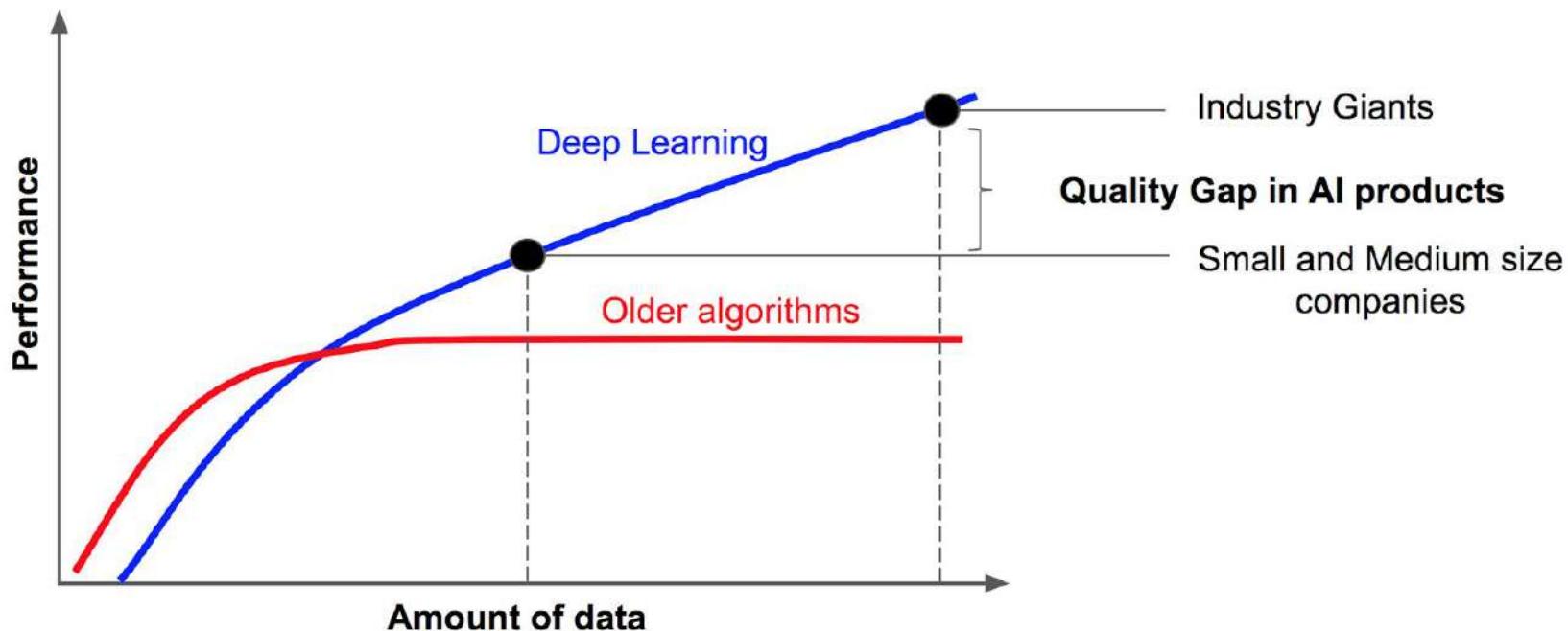


ОСНОВЫ НЕЙРОННЫХ СЕТЕЙ



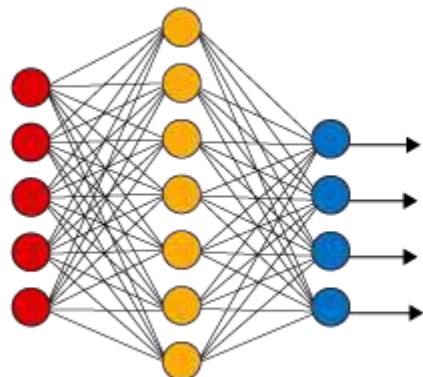
<http://www.deeplearningbook.org/>

Количество данных

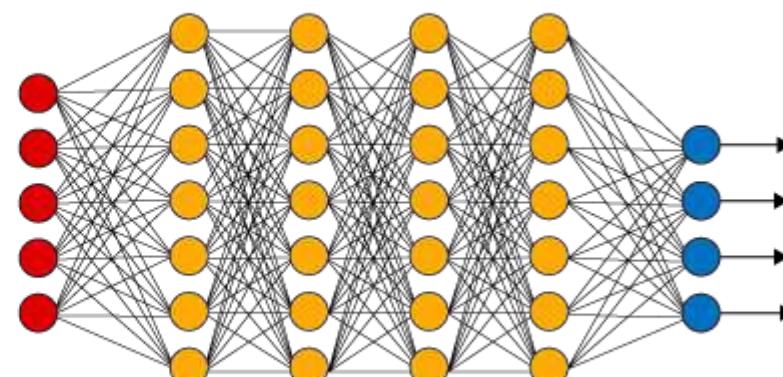


Нейронные сети

Simple Neural Network



Deep Learning Neural Network



● Input Layer

● Hidden Layer

● Output Layer

Области применения

- Компьютерное зрение
- Обработка естественного языка
- Управление
- Прогнозирование и классификация

Компьютерное зрение

Classification



CAT

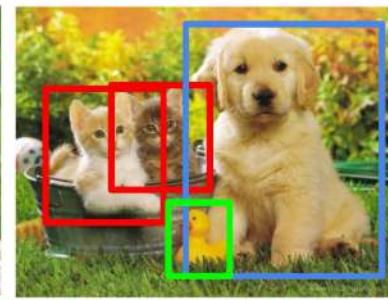
Single object

Classification + Localization



CAT

Object Detection



CAT, DOG, DUCK

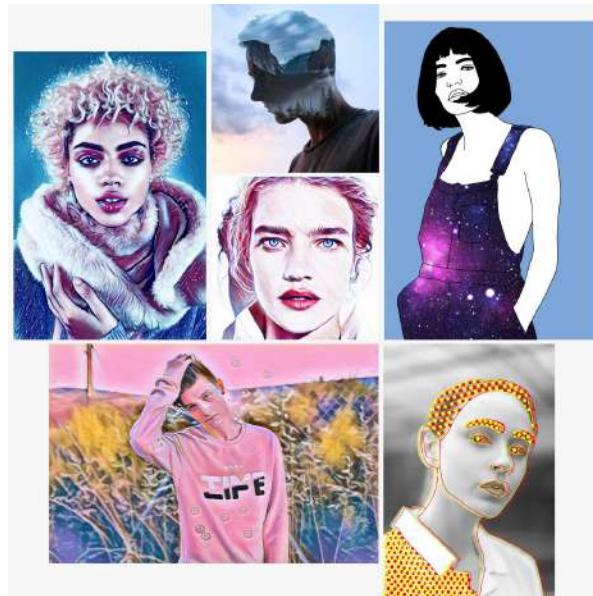
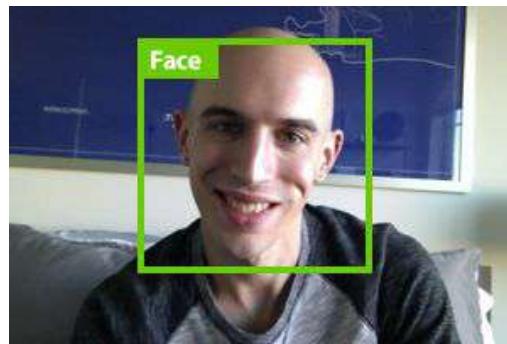
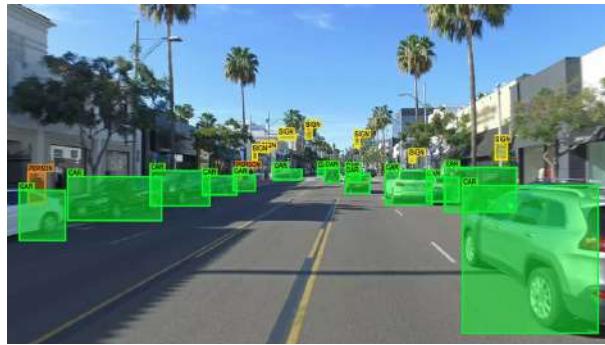
Instance Segmentation



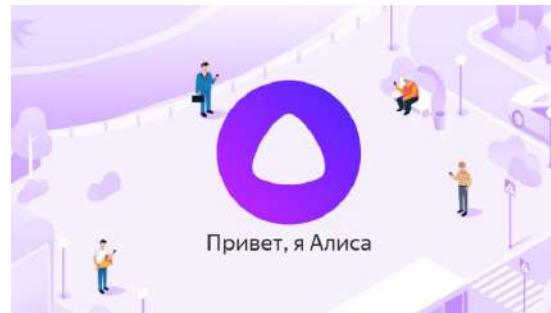
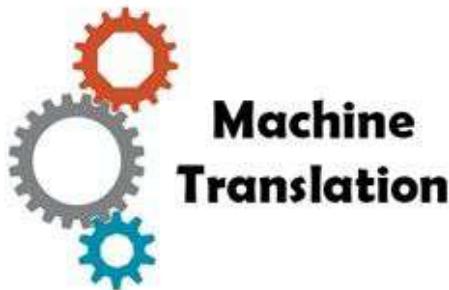
CAT, DOG, DUCK

Single object Multiple objects

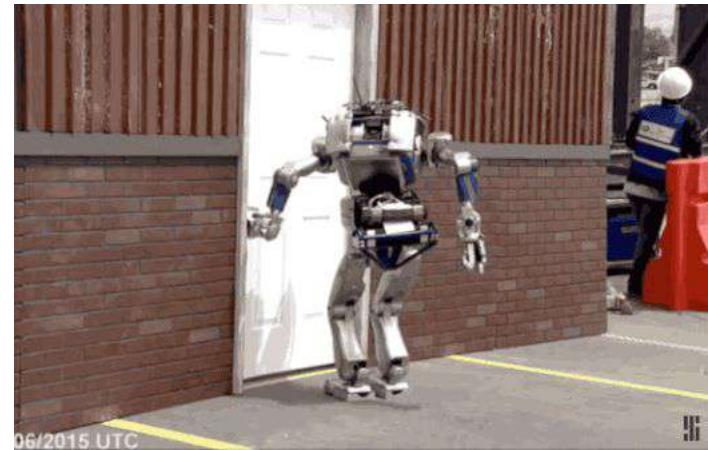
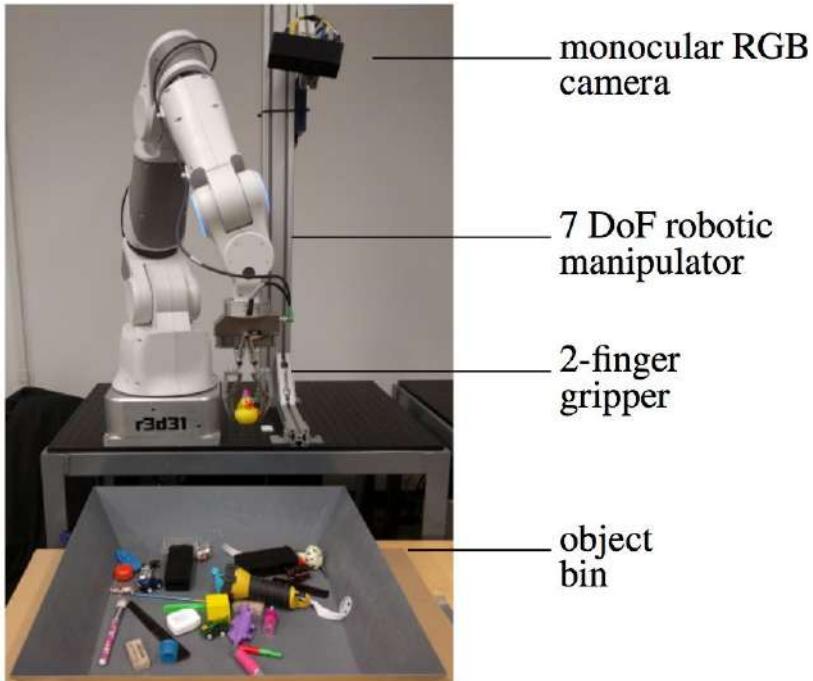
Компьютерное зрение



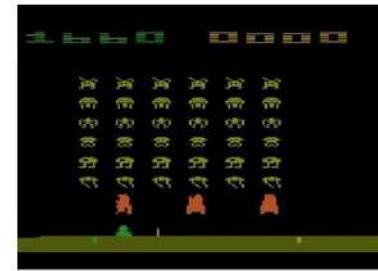
Обработка естественного языка и речи



Управление: робототехника



Управление: игры

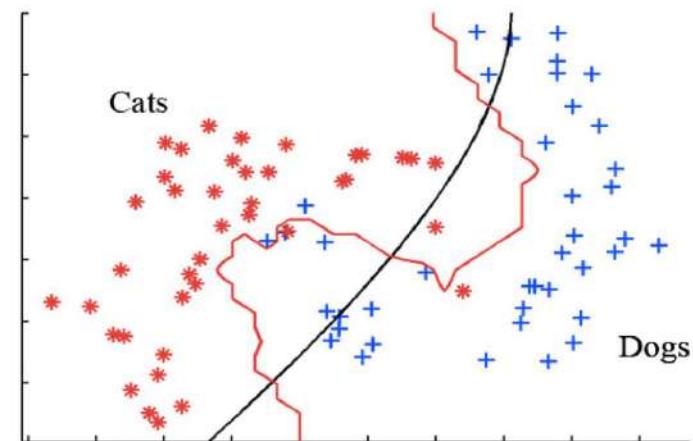
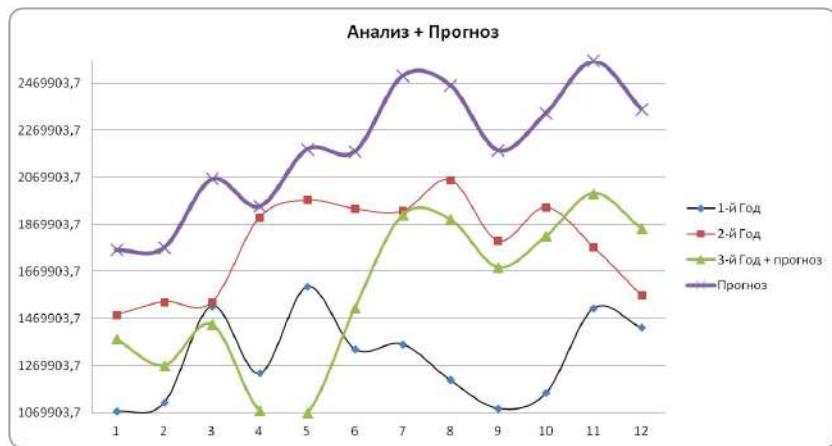


Breakout and Space Invaders, 2 of the 49 Atari games used in the paper



Задачи на структурированных данных

- Все те задачи, которые решают с помощью не нейросетевых ML-алгоритмов



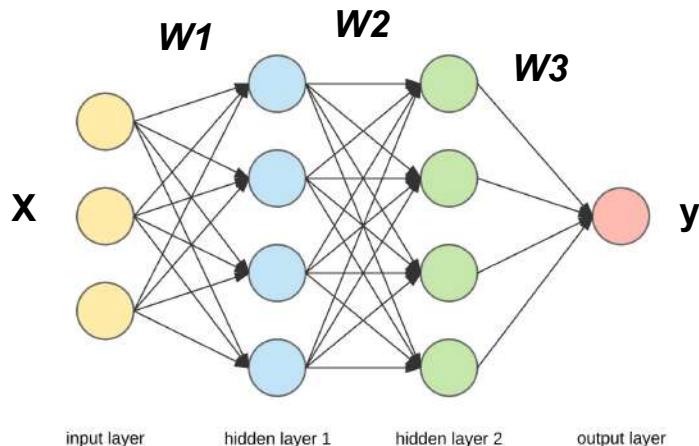
Минус: интерпретируемость



Модель нейрона

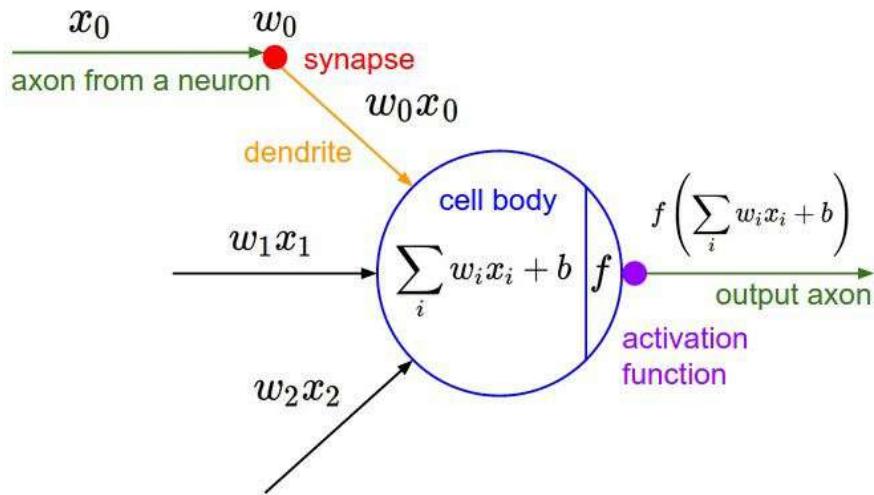
Полносвязная нейросеть

- Вход: числовая матрица X
- Внутри: матрицы параметров (веса нейросети)
- Выход: вектор ответов y
 - метки классов (классификация)
 - вещественные числа (регрессия)

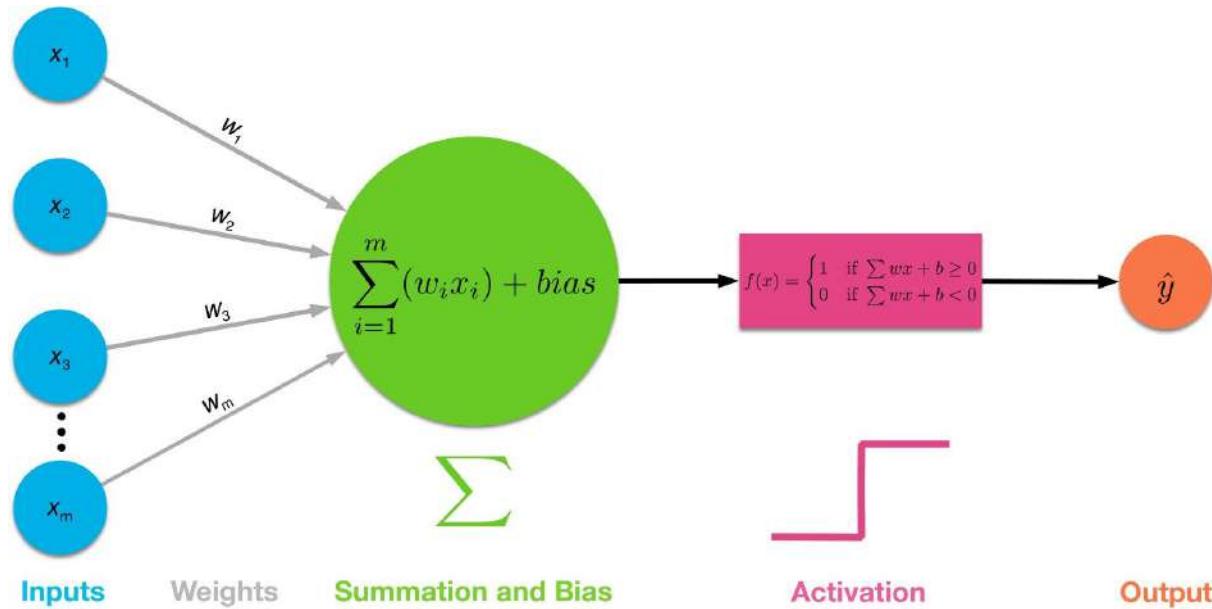


Искусственный нейрон

- Вход: вектор чисел \mathbf{x} размера $(k, 1)$
- Веса нейрона:
 - вектор чисел \mathbf{w} размера $(k, 1)$
 - скаляр b (свободный член)
- Функция активации: f
- Выход: метка класса y (0 или 1)



Перцептрон Розенблата

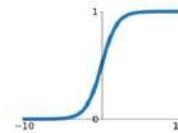


Нейрон

- Модель та же, что и у перцептрана
- Другие функции активации

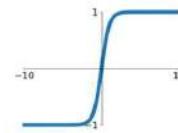
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



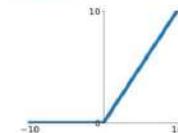
tanh

$$\tanh(x)$$



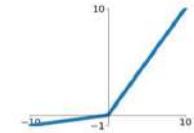
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

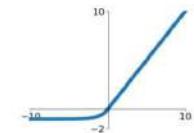


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



<https://arxiv.org/pdf/1710.05941.pdf>

Нейрон: обучение

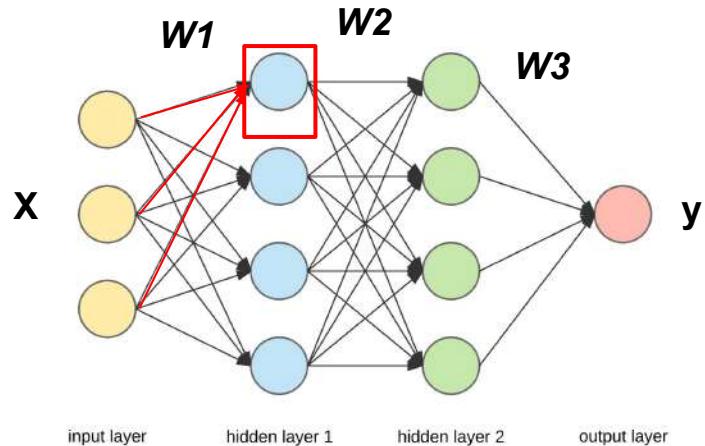
- Задан Loss: функция потерь
- Задача обучения с учителем: есть ответы на каждом из объектов
- Оптимизируем градиентным спуском

$$w^{j+1} = w^j - \alpha \frac{\partial Loss}{\partial w}(w^j)$$

Один нейрон

- Вход:
Матрица объекты-признаки \mathbf{X} размера (n, k)
- Внутри:
Вектор весов \mathbf{w} размера $(k, 1)$
- Выход:
Вектор чисел \mathbf{a} размера $(n, 1)$

$$f(\mathbf{X} @ \mathbf{w}) = \mathbf{a}$$



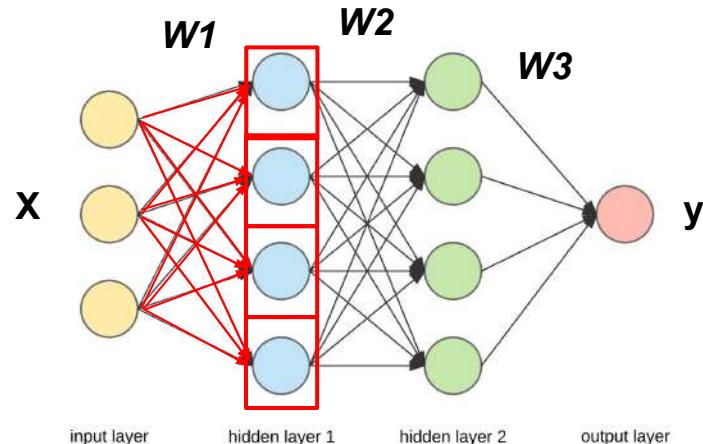
Первый слой

- Вход:
Матрица объекты-признаки \mathbf{X} размера (n, k)
- Внутри:
Вектора весов $w[1]..w[L1]$ размера $(k, 1)$
- Выход:
Вектор чисел $a[i]$ размера $(n, 1)$ от i -го нейрона

$$f(\mathbf{X} @ w[1]) = a[1]$$

⋮

$$f(\mathbf{X} @ w[L1]) = a[L1]$$

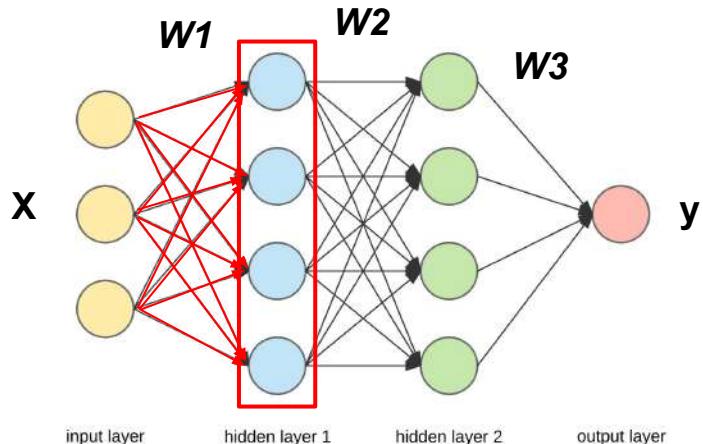


$L1$ нейронов

Первый слой

- Вход:
Матрица объекты-признаки \mathbf{X} размера (n, k)
- Внутри:
Матрица весов $\mathbf{W}[1]$ размера (k, L_1)
- Выход:
Матрица чисел $\mathbf{A}[1]$ размера (n, L_1)

$$f(\mathbf{X} @ \mathbf{W}[1]) = \mathbf{A}[1]$$

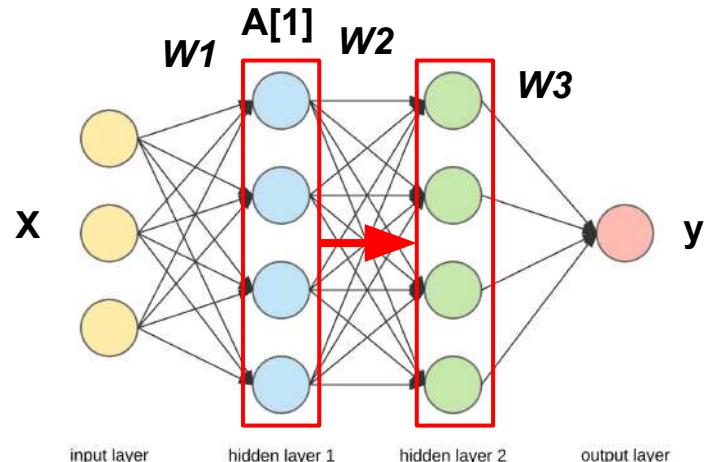


L_1 нейронов

Второй слой

- Вход:
Матрица **A[1]** с первого слоя размера (n , L1)
- Внутри:
Матрица весов **W[2]** размера (L1, L2)
- Выход:
Матрица чисел **A[2]** размера (n , L2)

$$f(A[1] @ W[2]) = A[2]$$

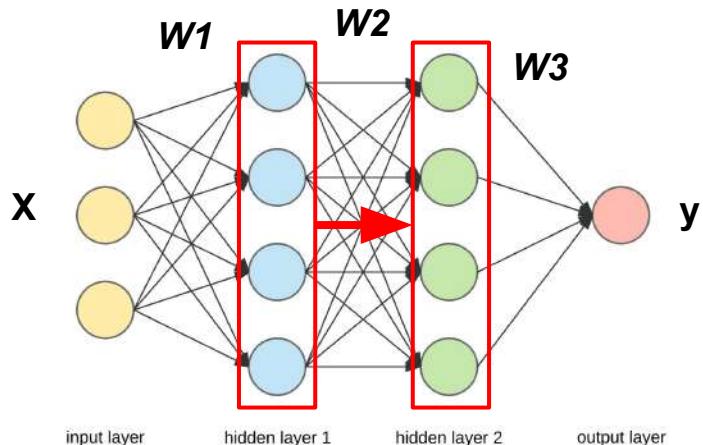


L1 L2
нейронов нейронов

Слой номер i

- Вход:
Матрица $A[i-1]$ с предыдущего слоя размера
($n, L[i-1]$)
- Внутри:
Матрица весов $W[i]$ размера ($L[i-1], L[i]$)
- Выход:
Матрица чисел $A[i]$ размера ($n, L[i]$)

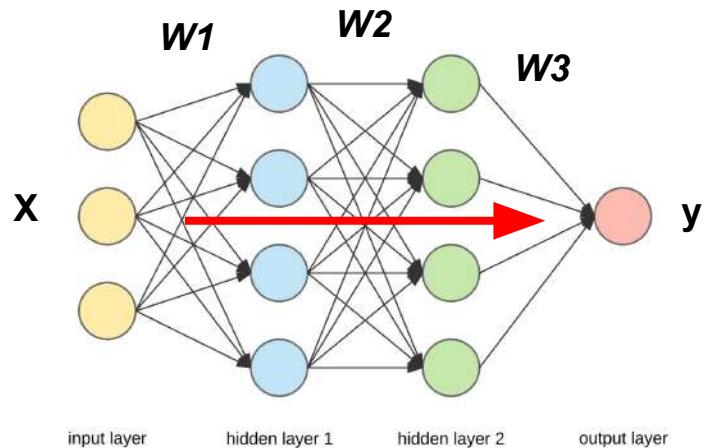
$$f(A[i-1] @ W[i]) = A[i]$$



$L[i-1]$ $L[i]$
нейронов нейронов

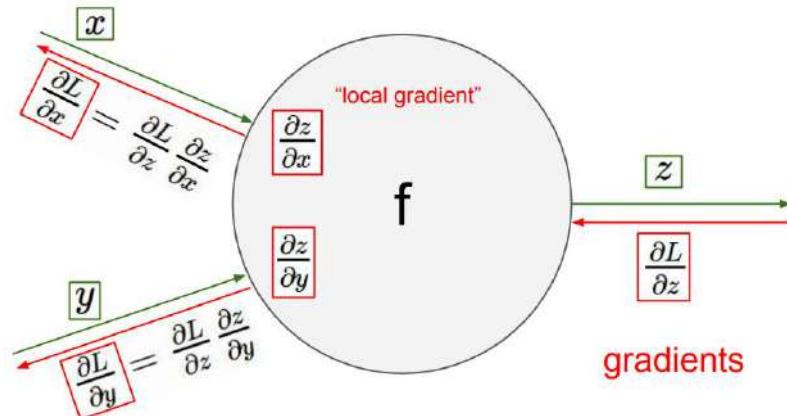
Forward pass

- Процесс прохождения данных через нейросеть
- = умножение матриц слоёв друг на друга
- Называется forward pass



Backward pass

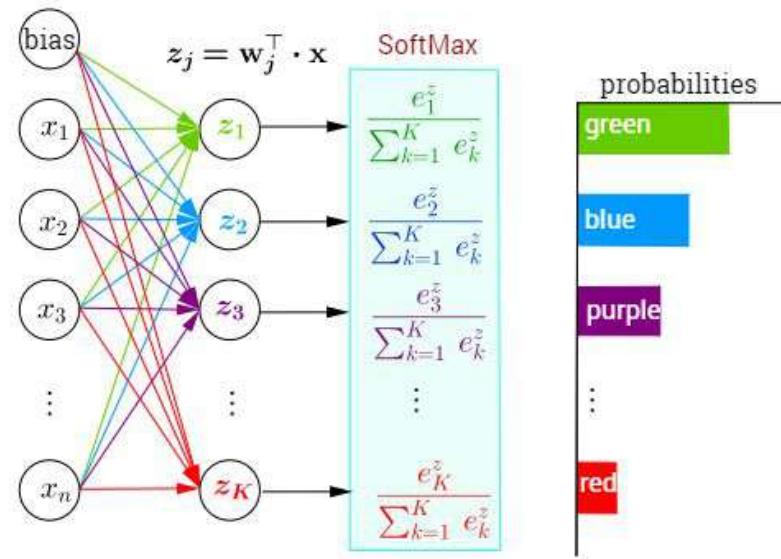
- Процесс распространения ошибок обратно для обновления весов сети
- Градиентный спуск для одной нейрона
= backpropagation для одного нейрона
- Подробнее [здесь](#)



L -- функция потерь

Многоклассовая классификация

- В случае **K** классов: **K** нейронов на выходном слое, поскольку хотим конечный размер матрицы A[out] равным (n, K)
- Выходной слой не имеет активаций
- Стока матрицы A[out] с номером **i** – это K чисел, так называемы score'ы классов



Многоклассовая классификация

- Применяем к выходной матрице Softmax, чтобы score'ы перешли в вероятности классов
- Теперь число с номером **j** в **i**-ой строке -- это вероятность принадлежности объекта **i** к классу **j** (сумма в каждой строке равна единице)

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{k=1}^K e^{z_k}}$$

$z[i]$ -- i-ое значение в строке матрицы $A[\text{out}]$ (и так для всех строк по отдельности)

Многоклассовая классификация

- Применяем к выходной матрице Softmax, чтобы score'ы перешли в вероятности классов
- Теперь число с номером j в i -ой строке -- это вероятность принадлежности объекта i к классу j (сумма в каждой строке равна единице)



Пример преобразования строки score'ов одного объекта
(здесь строка вытянута в столбец)

Многоклассовая классификация

- Применяем к выходной матрице Softmax
- Чтобы score'ы перешли в вероятности классов
- Теперь число с номером **j** в **i**-ой строке -- это вероятность принадлежности объекта **i** к классу **j**
- Результат подаётся в CrossEntropy лосс, где **p** -- матрица предсказаний, **t** -- истинные метки

$$\alpha_{ij} = \text{softmax}_j(e_{ij}) = \frac{\exp(e_{ij})}{\sum_{k \in \mathcal{N}_i} \exp(e_{ik})}.$$

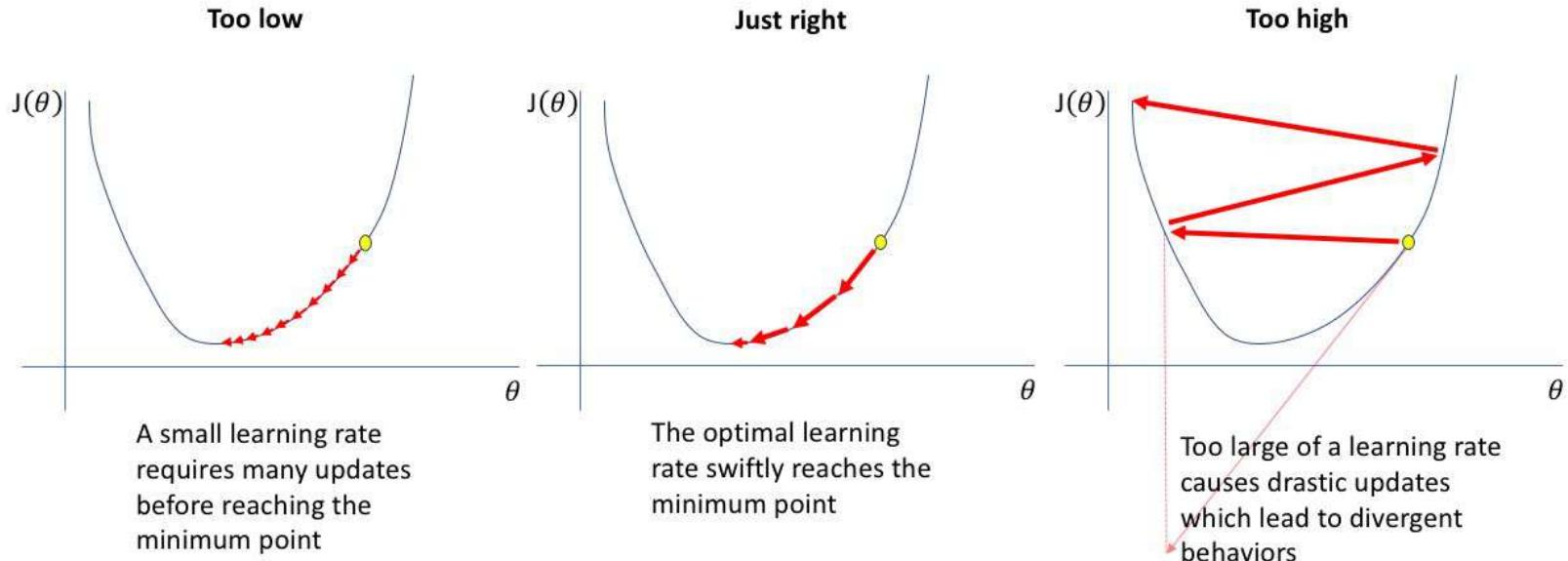
$$\text{cross-entropy} = -\frac{1}{N} \sum_{i=1}^N \sum_{j=1}^k t_{i,j} \log(p_{i,j})$$

Эффективное обучение нейросетей

Learning rate

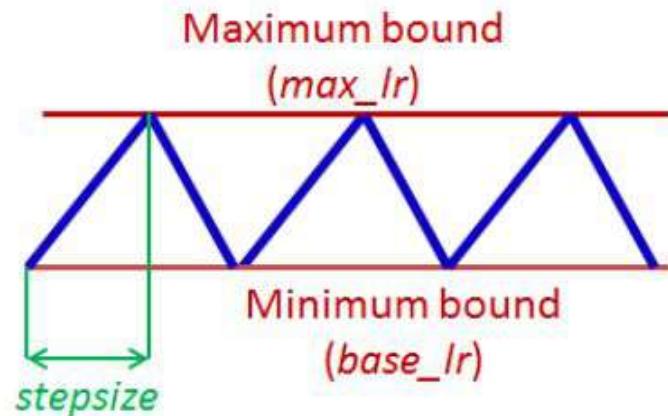
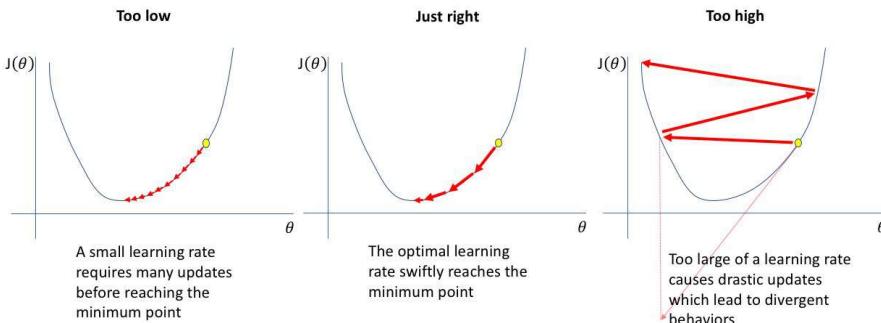
$$w^{j+1} = w^j - \boxed{\alpha} \frac{\partial Loss}{\partial w}(w^j)$$

Learning rate schedule



<https://arxiv.org/abs/1506.01186>

Cyclic learning rate



<https://arxiv.org/abs/1506.01186>

Initialization

- Нулями
- Не нулями, но одинаковыми числами
- Случайными числами
- Не init / Xavier init
- Новая статья: <https://arxiv.org/pdf/1704.08863.pdf>

Weight decay

$$J(\theta) = \frac{1}{m} \sum_{i=1}^m [-y^{(i)} \log(h_\theta(x^{(i)})) - (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)}))] + \frac{\lambda}{2m} \sum_{j=1}^n \theta_j^2.$$

L1 Regularization

$$\text{Cost} = \sum_{i=0}^N (y_i - \sum_{j=0}^M x_{ij} W_j)^2 + \lambda \sum_{j=0}^M |W_j|$$

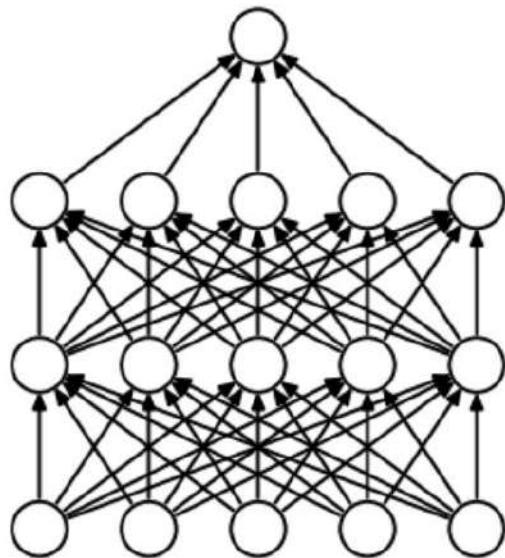
L2 Regularization

$$\text{Cost} = \sum_{i=0}^N (y_i - \sum_{j=0}^M x_{ij} W_j)^2 + \lambda \sum_{j=0}^M W_j^2$$

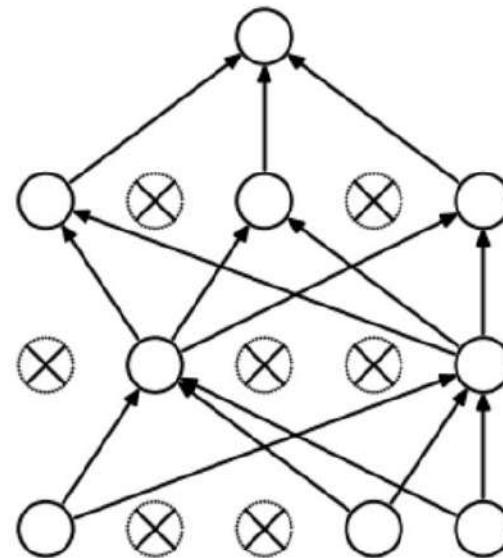
Loss function

Regularization Term

Dropout



(a) Standard Neural Net



(b) After applying dropout.

Обучение на больших выборках

Gradient Descent

$$Loss(\hat{y}, y) = \frac{1}{2n} \sum_{i=1}^n (\hat{y}_i - y_i)^2 = \frac{1}{2n} \sum_{i=1}^n (\sigma(w \cdot X_i) - y_i)^2$$

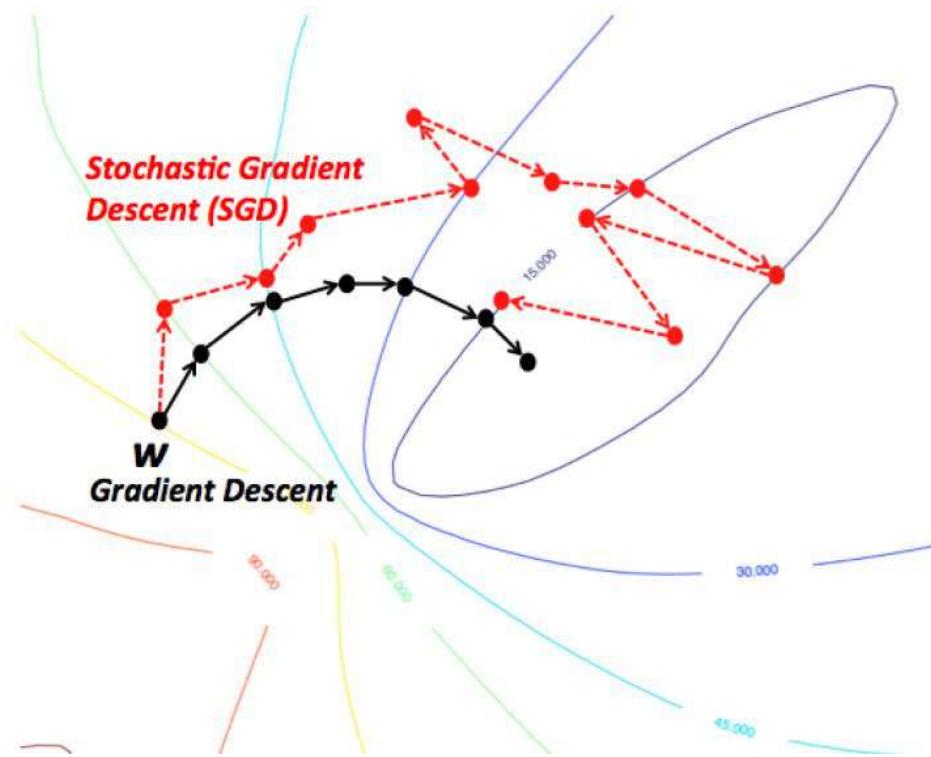
$$\frac{\partial Loss}{\partial w} = \frac{1}{n} X^T (\sigma(w \cdot X) - y) \sigma(w \cdot X) (1 - \sigma(w \cdot X))$$

Stochastic Gradient Descent

$$Loss(\hat{y}, y) = (\hat{y}_i - y_i)^2 = (\sigma(w \cdot X_i) - y_i)^2$$

$$\frac{\partial Loss}{\partial w} = X_i^T (\sigma(w \cdot X_i) - y) \sigma(w \cdot X_i) (1 - \sigma(w \cdot X_i))$$

Stochastic Gradient Descent



Batch

	X	y
x1	features	label
x2	features	label
x3	features	label
x4	features	label
x5	features	label
x6	features	label
x7	features	label

Batch of data

Batch

	X	y
x1	features	label
x2	features	label
x3	features	label
x4	features	label
x5	features	label
x6	features	label
x7	features	label

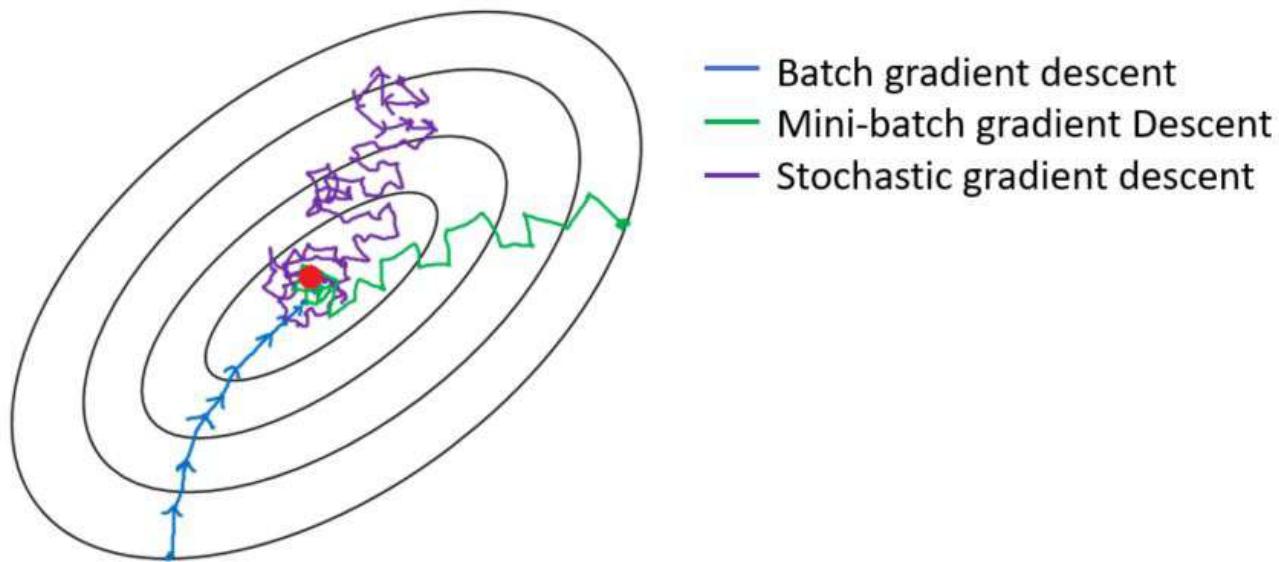
Random batch
of data

Mini-batch Gradient Descent

$$Loss(\hat{y}, y) = \frac{1}{2 \cdot \text{batch_size}} \sum_{i=1}^{\text{batch_size}} (\hat{y}_i - y_i)^2 = \frac{1}{2 \cdot \text{batch_size}} \sum_{i=1}^{\text{batch_size}} (\sigma(w \cdot X_i) - y_i)^2$$

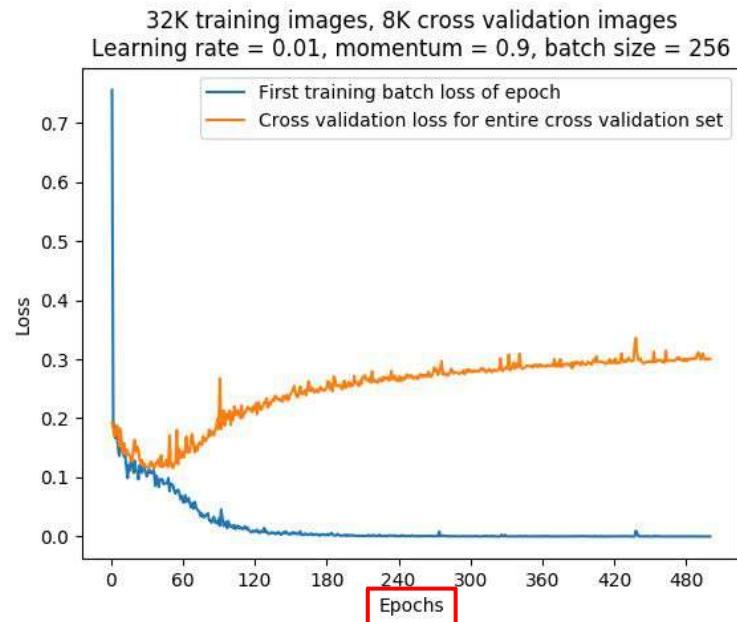
$$\frac{\partial Loss}{\partial w} = \frac{1}{\text{batch_size}} X_{batch}^T (\sigma(w \cdot X_{batch}) - y) \sigma(w \cdot X_{batch})(1 - \sigma(w \cdot X_{batch}))$$

Визуализация на линиях уровня



Итерация и эпоха

- **Итерация:** оптимизация по одному батчу данных
- **Эпоха:** оптимизация по всей выборке, т.е. когда мы прошлись по всем батчам
- Например, если в выборке **N** элементов, а в батче **B** элементов, то **в одной эпохе будет $N // B$ итераций** (но можно сделать и больше)



Алгоритмы оптимизации

Gradient Descent

$$Loss(\hat{y}, y) = \frac{1}{2n} \sum_{i=1}^n (\hat{y}_i - y_i)^2 = \frac{1}{2n} \sum_{i=1}^n (\sigma(w \cdot X_i) - y_i)^2$$

$$\frac{\partial Loss}{\partial w} = \frac{1}{n} X^T (\sigma(w \cdot X) - y) \sigma(w \cdot X) (1 - \sigma(w \cdot X))$$

Mini-batch Gradient Descent

$$Loss(\hat{y}, y) = \frac{1}{2 \cdot \text{batch_size}} \sum_{i=1}^{\text{batch_size}} (\hat{y}_i - y_i)^2 = \frac{1}{2 \cdot \text{batch_size}} \sum_{i=1}^{\text{batch_size}} (\sigma(w \cdot X_i) - y_i)^2$$

$$\frac{\partial Loss}{\partial w} = \frac{1}{\text{batch_size}} X_{batch}^T (\sigma(w \cdot X_{batch}) - y) \sigma(w \cdot X_{batch})(1 - \sigma(w \cdot X_{batch}))$$

Momentum

Обозначим:

$$\begin{aligned} J &= Loss \\ \theta &= w \end{aligned}$$

Тогда: $\frac{\partial Loss}{\partial w} = \nabla_{\theta} J$

$$v_t = \gamma v_{t-1} + \eta \nabla_{\theta} J(\theta - \gamma v_{t-1})$$

$$\theta = \theta - v_t$$

<https://habr.com/post/318970/>

Adagrad (adaptive gradient)

$$g_t \equiv \nabla_{\theta} J(\theta_t)$$

$$G_t = G_t + g_t^2$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{G_t + \epsilon}} g_t$$

<http://jmlr.org/papers/volume12/duchi11a/duchi11a.pdf>

<https://arxiv.org/abs/1002.4908>

RMSProp (root mean square propagation)

$$g_t \equiv \nabla_{\theta} J(\theta_t)$$

$$E[g^2]_t = \gamma E[g^2]_{t-1} + (1 - \gamma) g_t^2$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} g_t$$

$$RMS[g]_t = \sqrt{E[g^2]_t + \epsilon}$$

https://www.youtube.com/watch?v=76lj_cKBvmq

Adam (adaptive moments estimation)

$$g_t \equiv \nabla_{\theta} J(\theta_t)$$

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \quad v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$$

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}, \quad \hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t + \epsilon}} \hat{m}_t$$

<https://arxiv.org/abs/1412.6980>

Типы нейросетей

A mostly complete chart of
Neural Networks

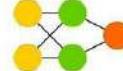
©2016 Fjodor van Veen - asimovinstitute.org

- 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

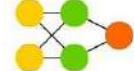
Perceptron (P)



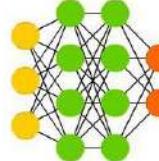
Feed Forward (FF)



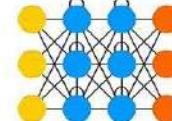
Radial Basis Network (RBF)



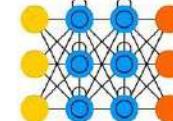
Deep Feed Forward (DFF)



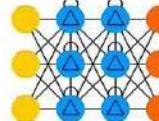
Recurrent Neural Network (RNN)



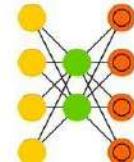
Long / Short Term Memory (LSTM)



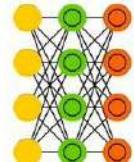
Gated Recurrent Unit (GRU)



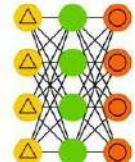
Auto Encoder (AE)



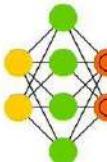
Variational AE (VAE)



Denoising AE (DAE)



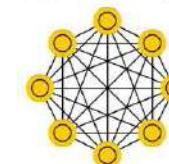
Sparse AE (SAE)



Markov Chain (MC)



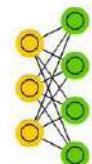
Hopfield Network (HN)



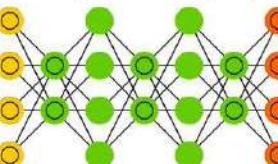
Boltzmann Machine (BM)

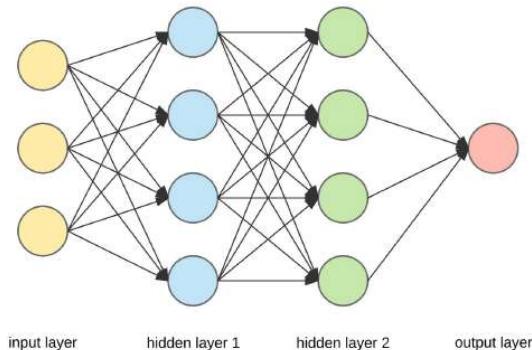


Restricted BM (RBM)

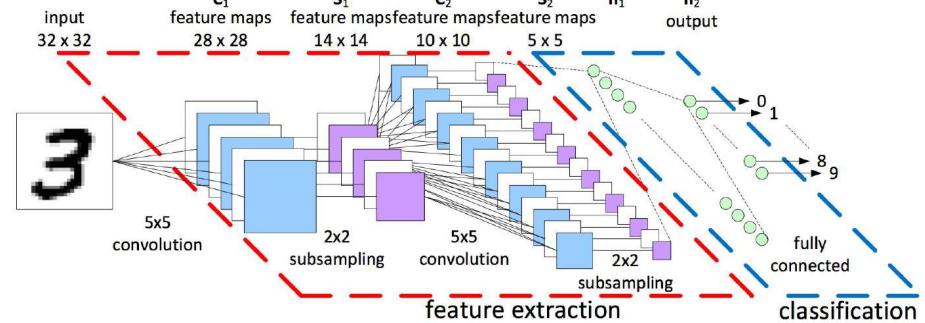


Deep Belief Network (DBN)

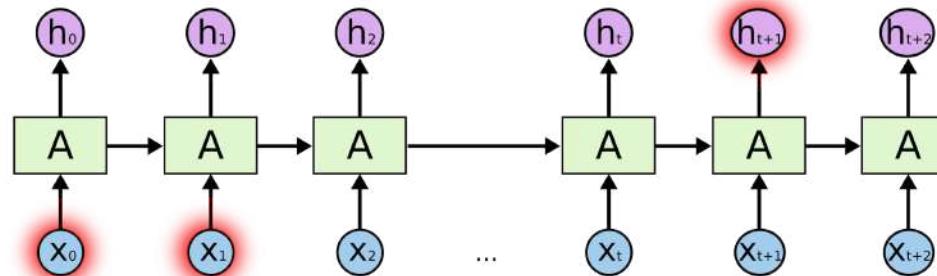




Dense =
 Fully Connected =
 Multi Layered Perceptron



Convolutional Neural Networks

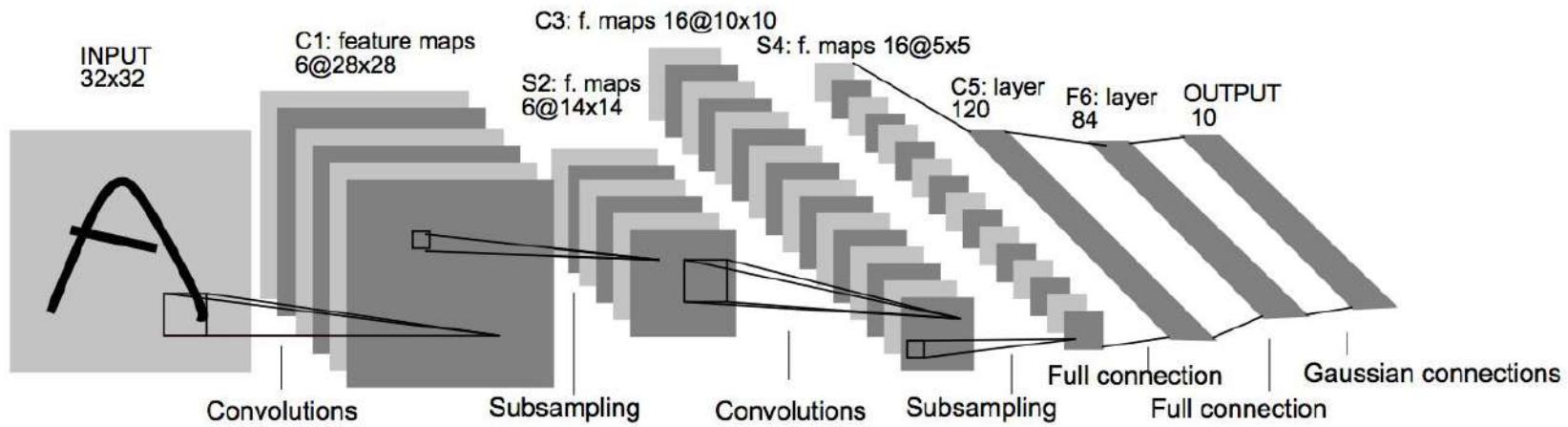


Recurrent Neural Networks

Архитектуры нейросетей

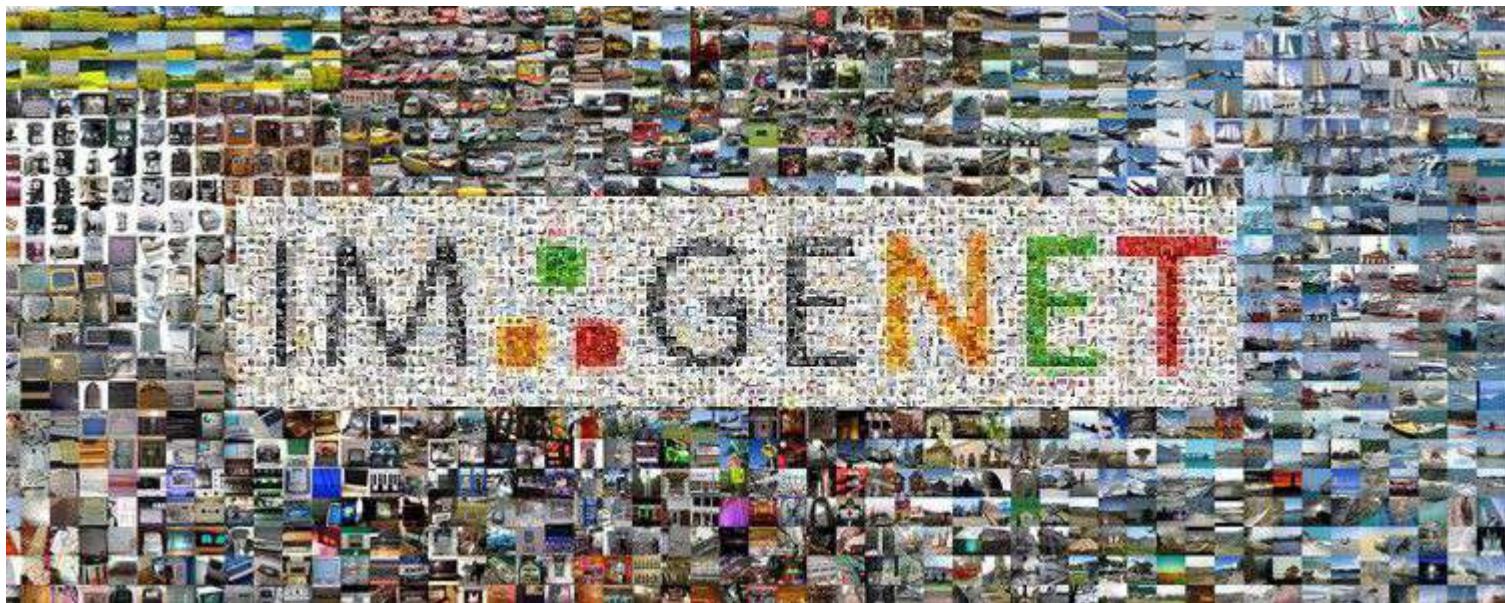
Свёрточные нейросети

LeNet



<http://yann.lecun.com/exdb/publis/pdf/lecun-98.pdf>

ImageNet



<http://www.image-net.org>

ILSVRC

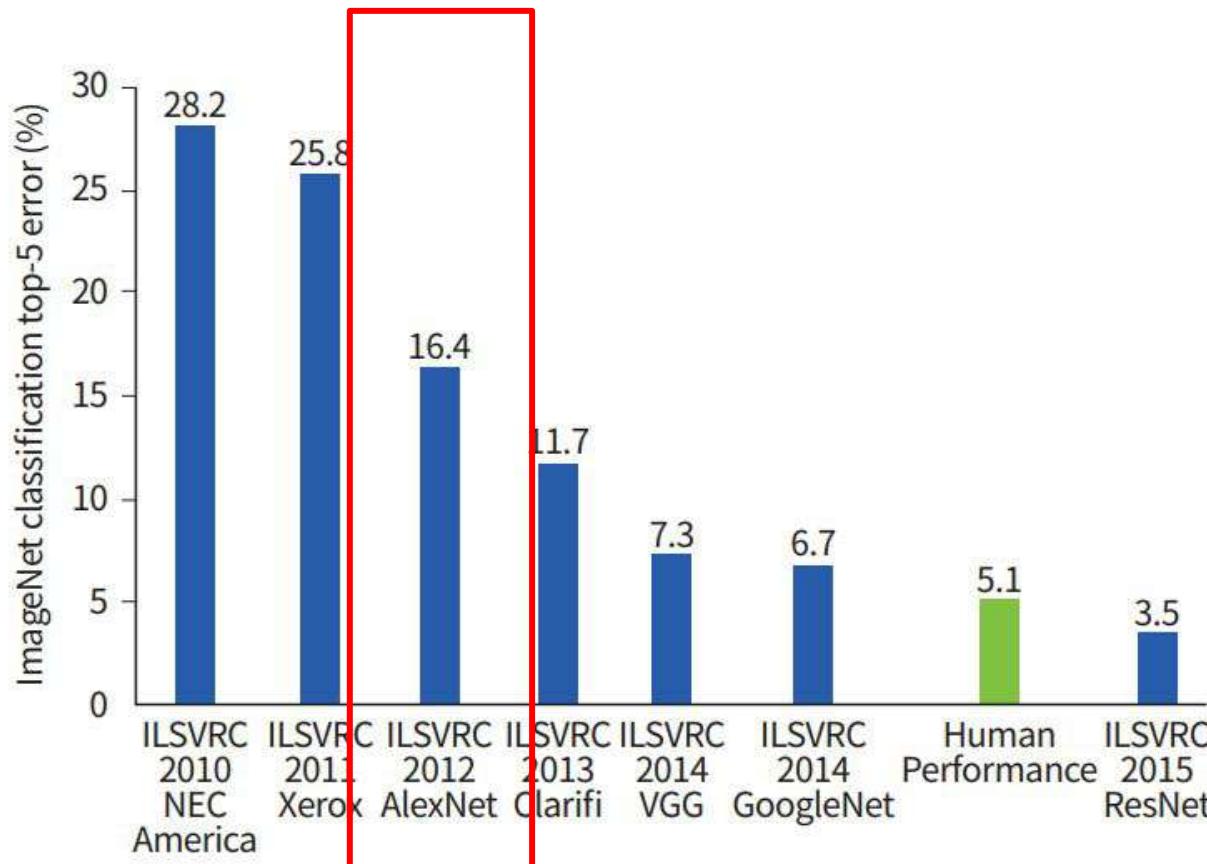


ImageNet Large Scale Visual Recognition Challenges



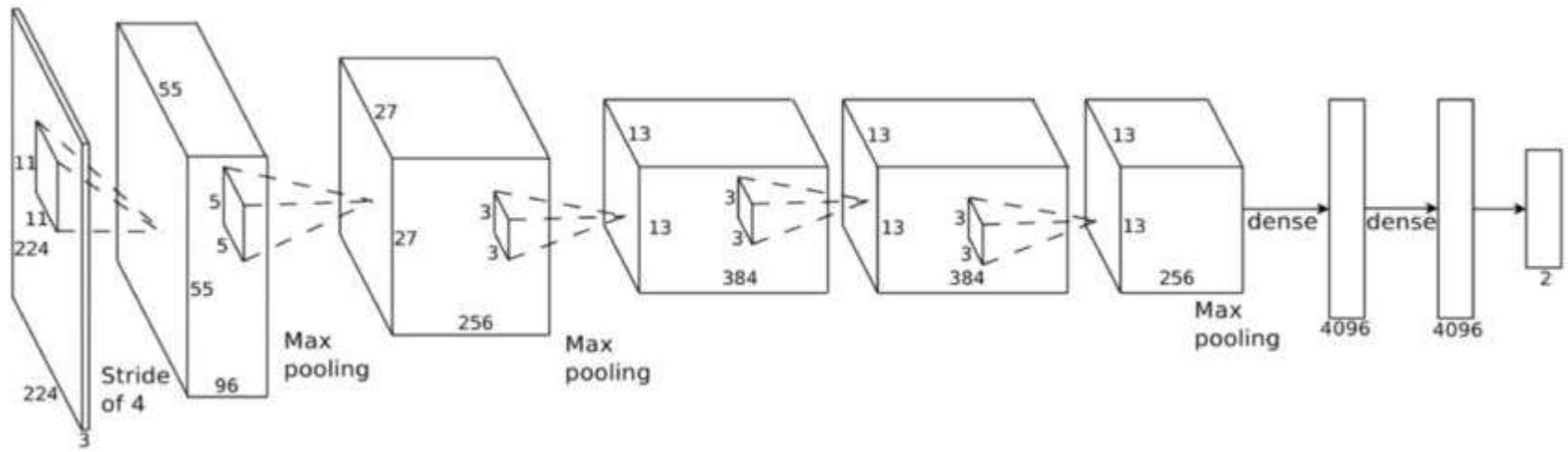
<http://www.image-net.org/challenges/LSVRC/>

ILSVRC



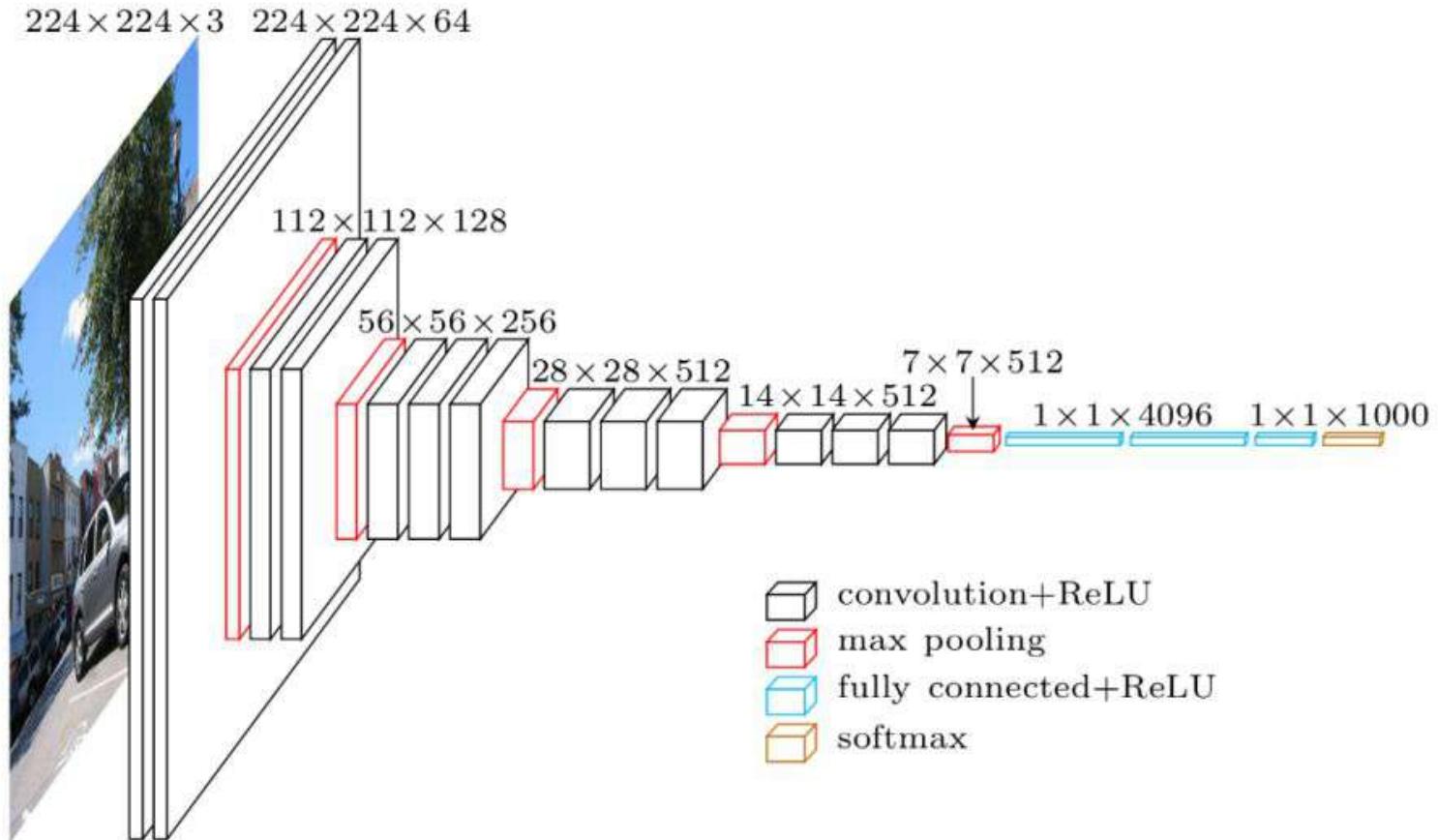
<http://www.image-net.org/challenges/LSVRC/>

AlexNet



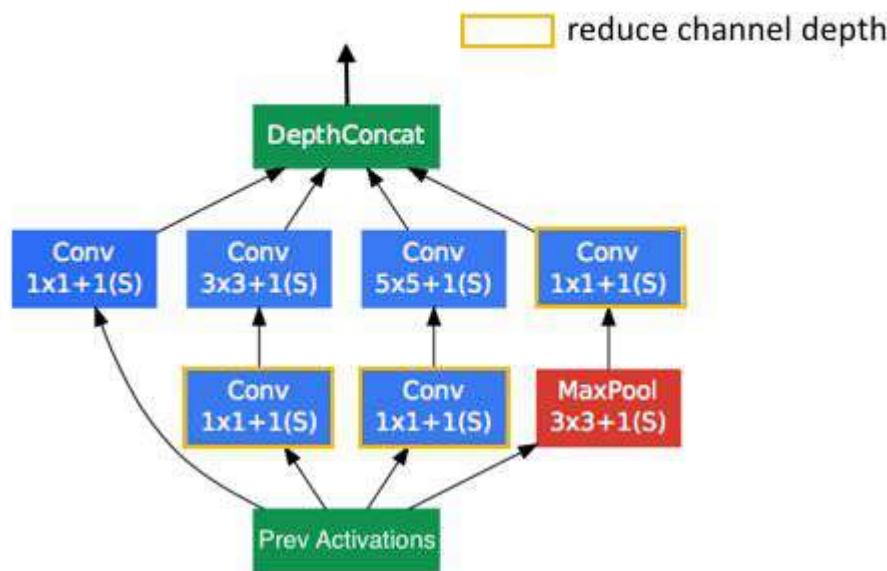
<http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf>

VGG



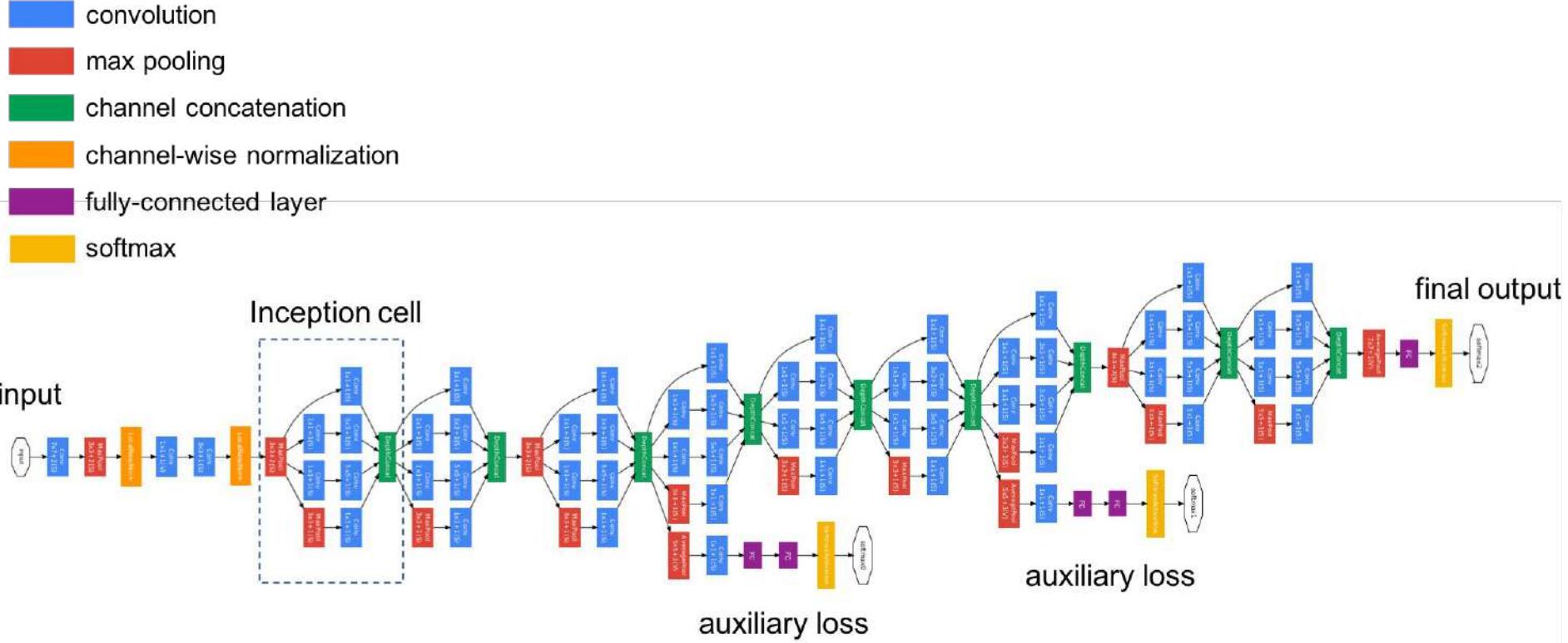
<https://arxiv.org/pdf/1409.1556.pdf>

Inception-v1 (GoogLeNet)



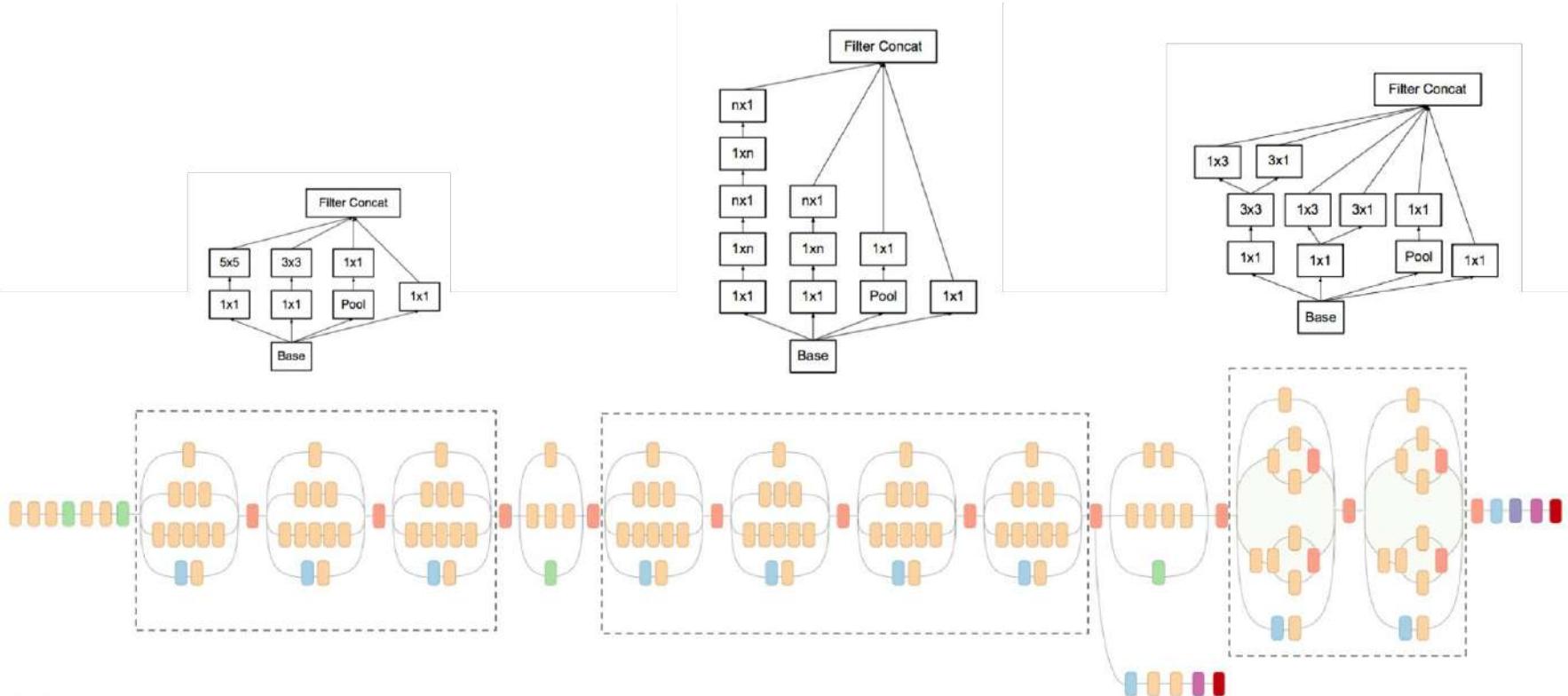
<https://arxiv.org/abs/1409.4842>

Inception-v1 (GoogLeNet)



<https://arxiv.org/abs/1409.4842>

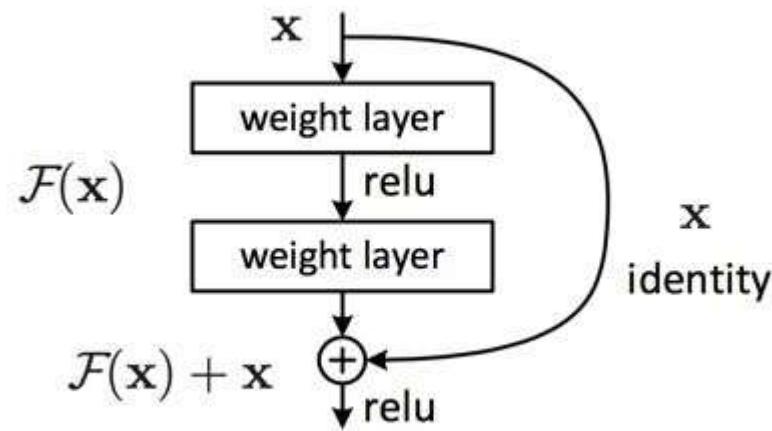
Inception-v2



- Convolution
- AvgPool
- MaxPool
- Concat
- Dropout
- Fully connected
- Softmax

<https://arxiv.org/pdf/1512.00567.pdf>

ResNet



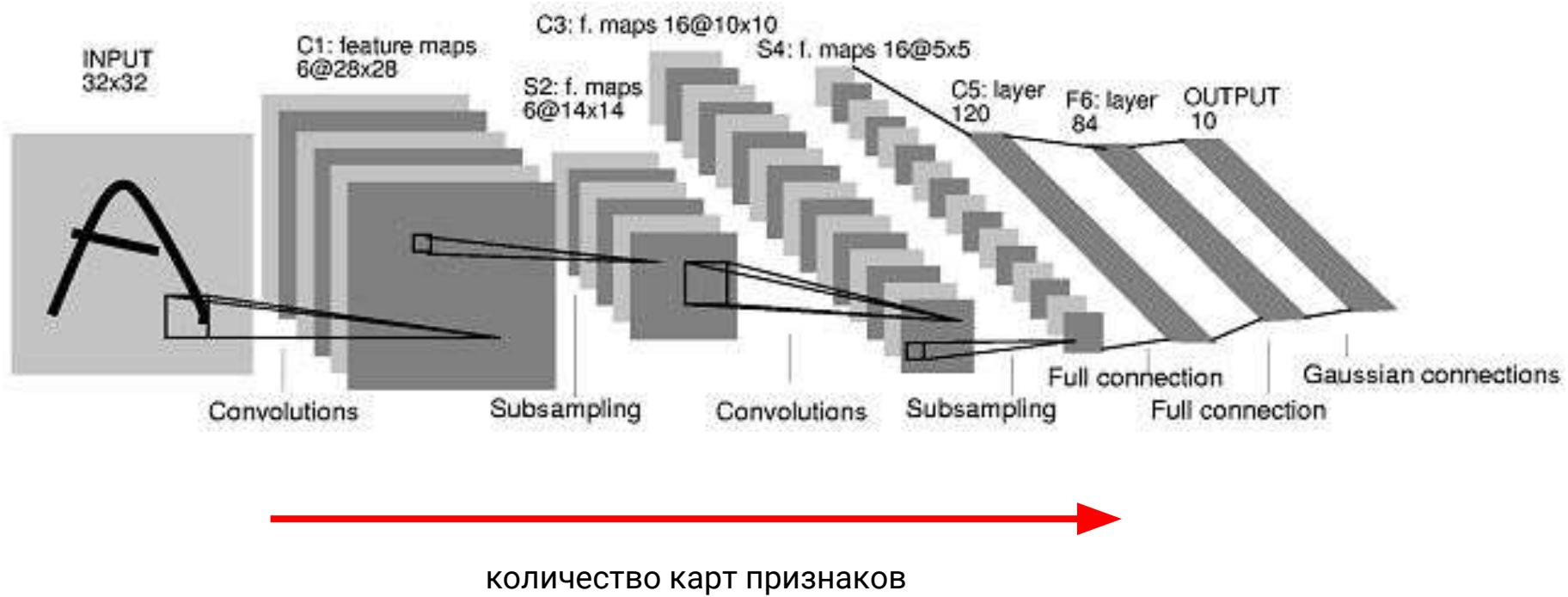
<https://arxiv.org/abs/1512.03385>

ResNet



<https://arxiv.org/abs/1512.03385>

Feature maps



More architectures

Модели для классификации на ImageNet:

- ResNeXt (<https://arxiv.org/abs/1611.05431>)
- SENet (<https://arxiv.org/abs/1709.01507>)
- DenseNet (<https://arxiv.org/abs/1608.06993>)
- Inception-ResNet-V2 (<https://arxiv.org/abs/1602.07261>)
- Inception-V4 (<https://arxiv.org/abs/1602.07261>)
- Xception (<https://arxiv.org/abs/1610.02357>)

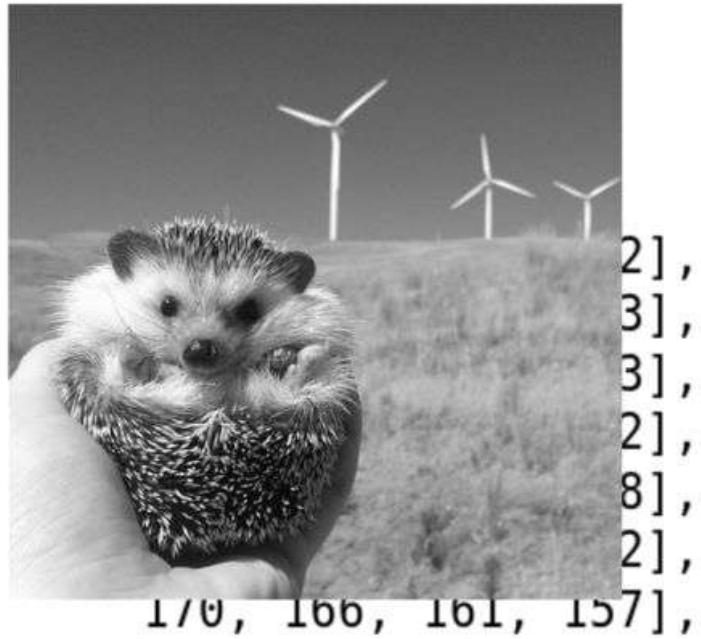
Легковесные модели (для мобильных устройств):

- NASNet (<https://arxiv.org/abs/1707.07012>)
- MobileNetV2 (<https://arxiv.org/abs/1801.04381>)
- ShuffleNet (<https://arxiv.org/abs/1707.01083>)
- SqueezeNet (<https://arxiv.org/abs/1602.07360>)

Свёрточные нейросети

ЧБ изображение

- Матрица чисел размера (H, W)
- Каждый пиксель:
целое число от 0 до 255
- Значение:
 - 0 -- чёрный
 - 255 -- белый

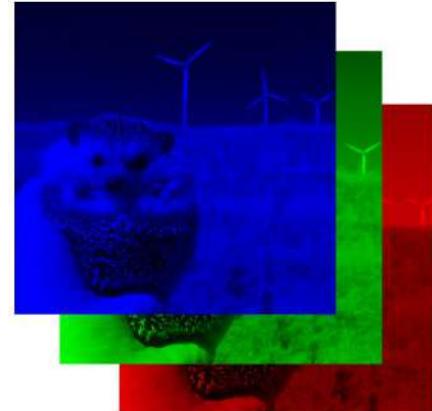


Цветное изображение

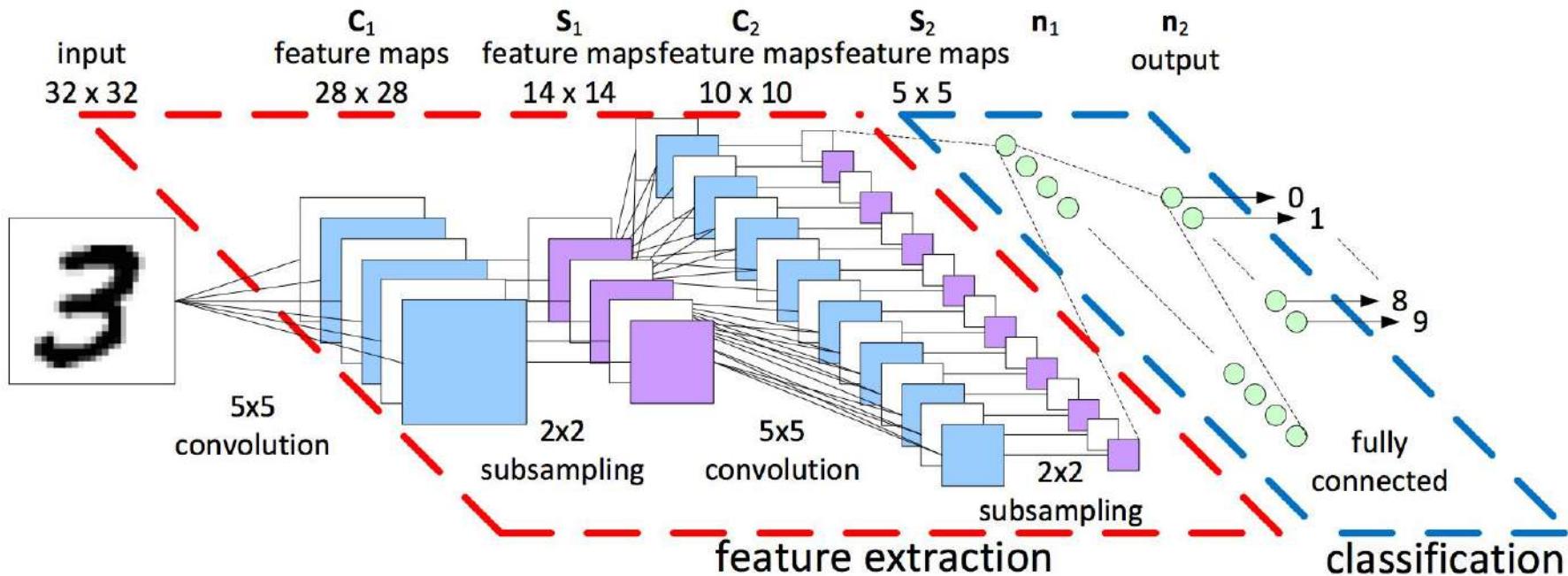
- Матрица чисел размера ($H, W, 3$)
- Каждый пиксель:
целое число от 0 до 255
- Значение:
 - 0 -- чёрный
 - 255 -- белый



||



Типичная сверточная сеть



Ключевые понятия

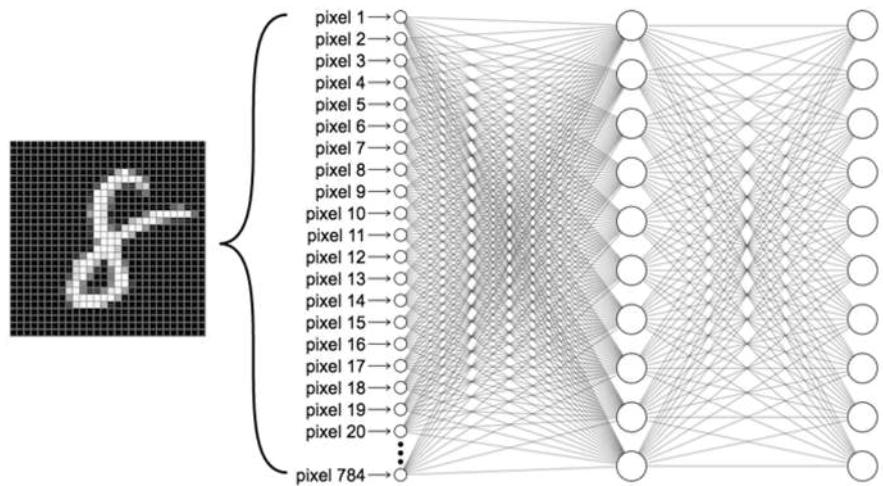
- Операция свёртки (convolution)
- Операция пулинга (pooling)

Операция свёртки

Свёрточные нейросети

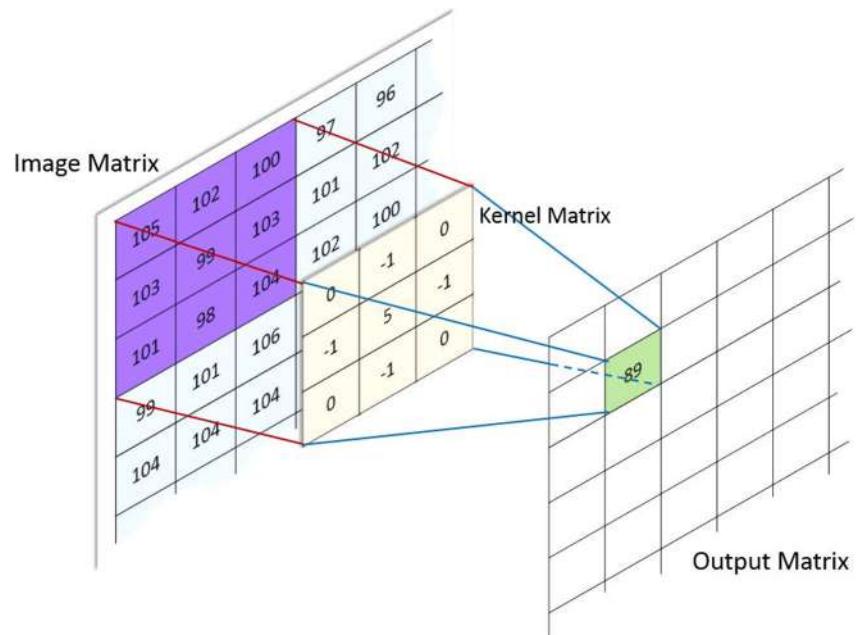
Мотивация

- MLP: взвесим каждый пиксель
- Не учитывается “пространственная” информация
- Не учитывается специфика изображений



Свёртка

- Идея: инвариантность к сдвигам объектов внутри картинки
- Нужен паттерн, реагирующий на регионы с объектами эффективно
- => Фильтр свёртки



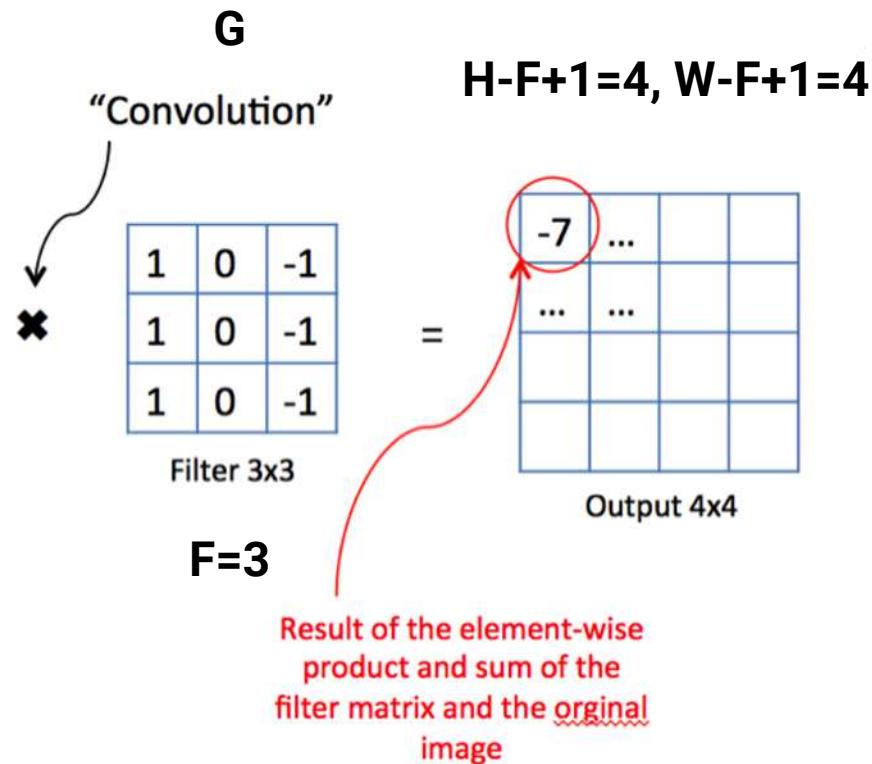
Свёртка ч/б изображения с фильтром FxF

I

3	1	1	2	8	4
1	0	7	3	2	6
2	3	5	1	1	3
1	4	1	2	6	5
3	2	1	3	7	2
9	2	6	2	5	1

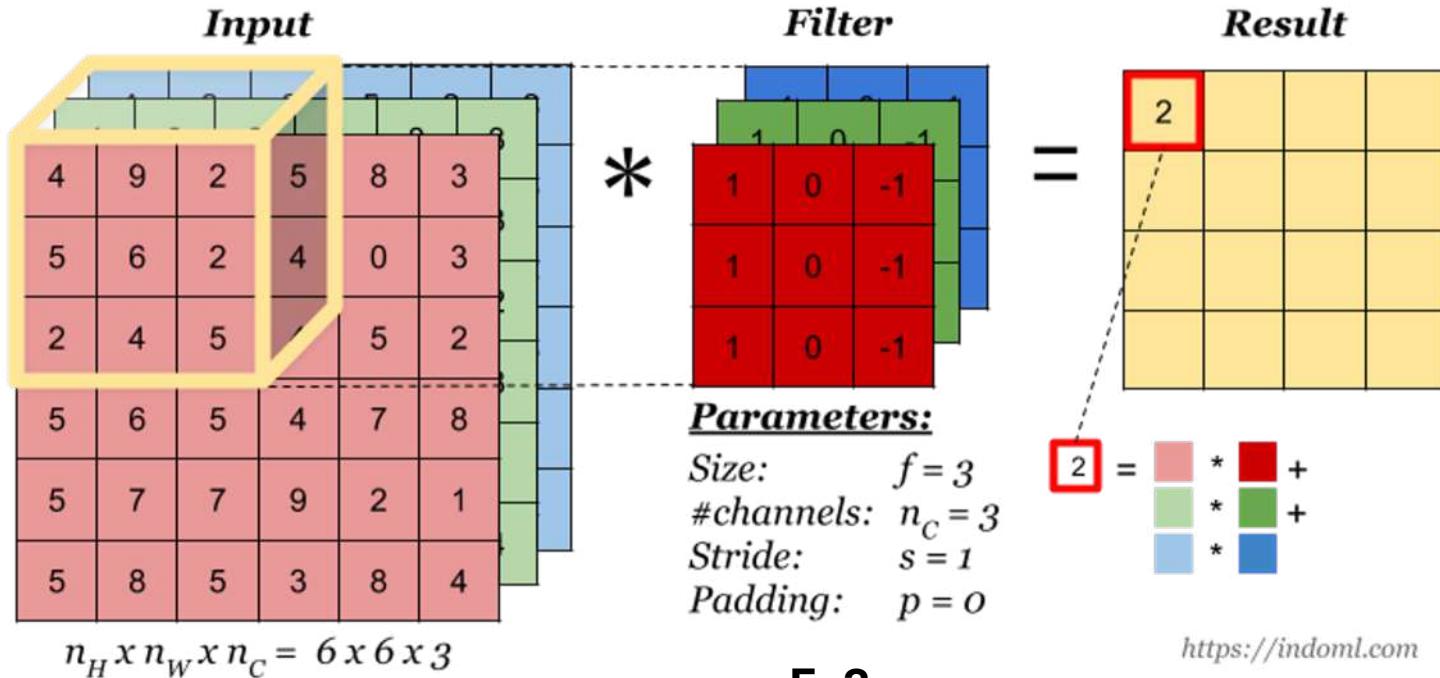
Original image 6x6

H=6, W=6



Свёртка цветного изображения с фильтром FxF

$$H-F+1=4, W-F+1=4$$



$H=6, W=6, C=3$
(H, W, C)

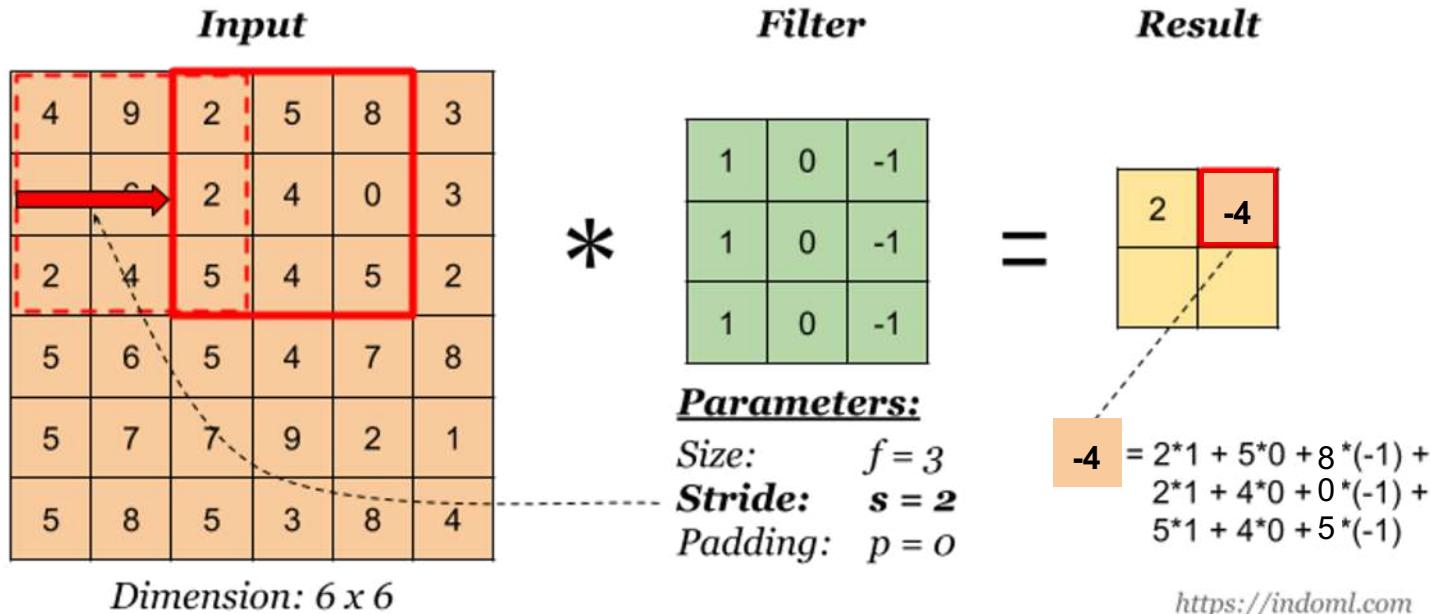
$F=3$
(F, F, C)

<https://indoml.com>

Шаг свёртки (stride)

Stride = 2

$$H_{out} = (H - F) // k + 1 = 2$$
$$W_{out} = (W - F) // k + 1 = 2$$



Dimension: 6×6

$H=6, W=6, C=1$
 (H, W, C)

<https://indoml.com>

$F=3$
 (F, F, C)

Обрамление нулями (padding)

0 ₂	0 ₀	0 ₁	0	0	0	0
0 ₁	2 ₀	2 ₀	3	3	3	0
0 ₀	0 ₁	1 ₁	3	0	3	0
0	2	3	0	1	3	0
0	3	3	2	1	2	0
0	3	3	0	2	3	0
0	0	0	0	0	0	0

1	6	5
7	10	9
7	10	8

Результат операции свёртки

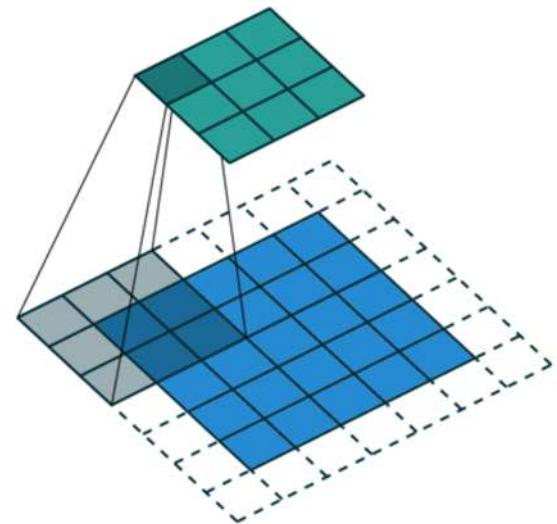
- Размер картинки: $(H, W, 3)$
- Размер фильтра: $(F, F, 3)$
- Stride = S
- Padding = P
- Размер тензора-результата $(H_{out}, W_{out}, C_{out})$:

$$H_{out} = (H - F + 2P) // S + 1$$

$$W_{out} = (W - F + 2P) // S + 1$$

$$C_{out} = 1$$

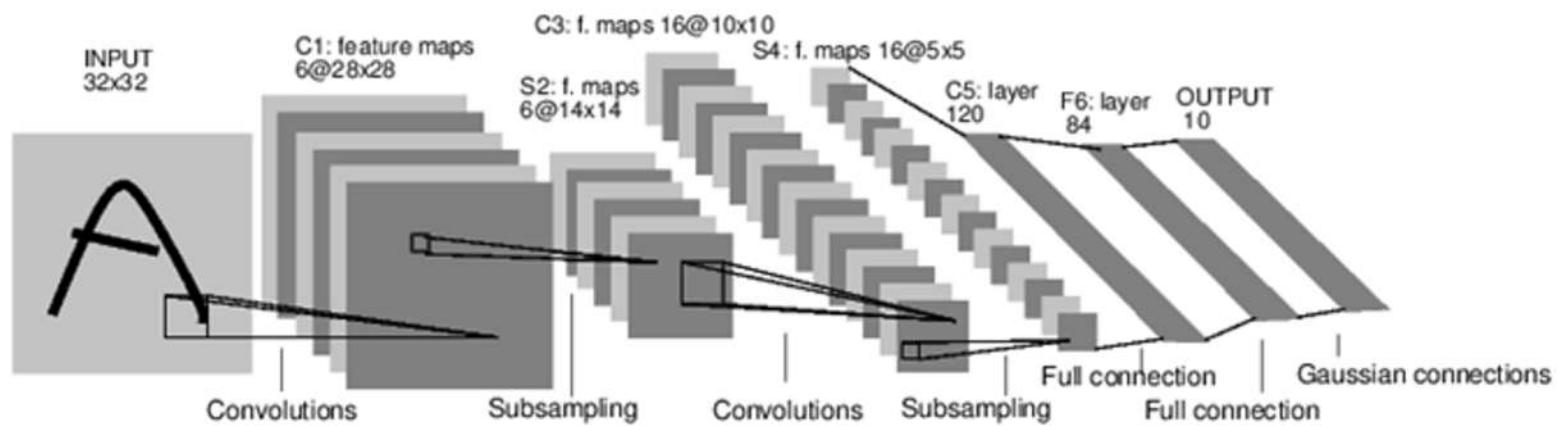
- Полный гайд по свёрткам:
<https://arxiv.org/abs/1603.07285>



Свёрточный и пулинг слои

Свёрточные нейросети

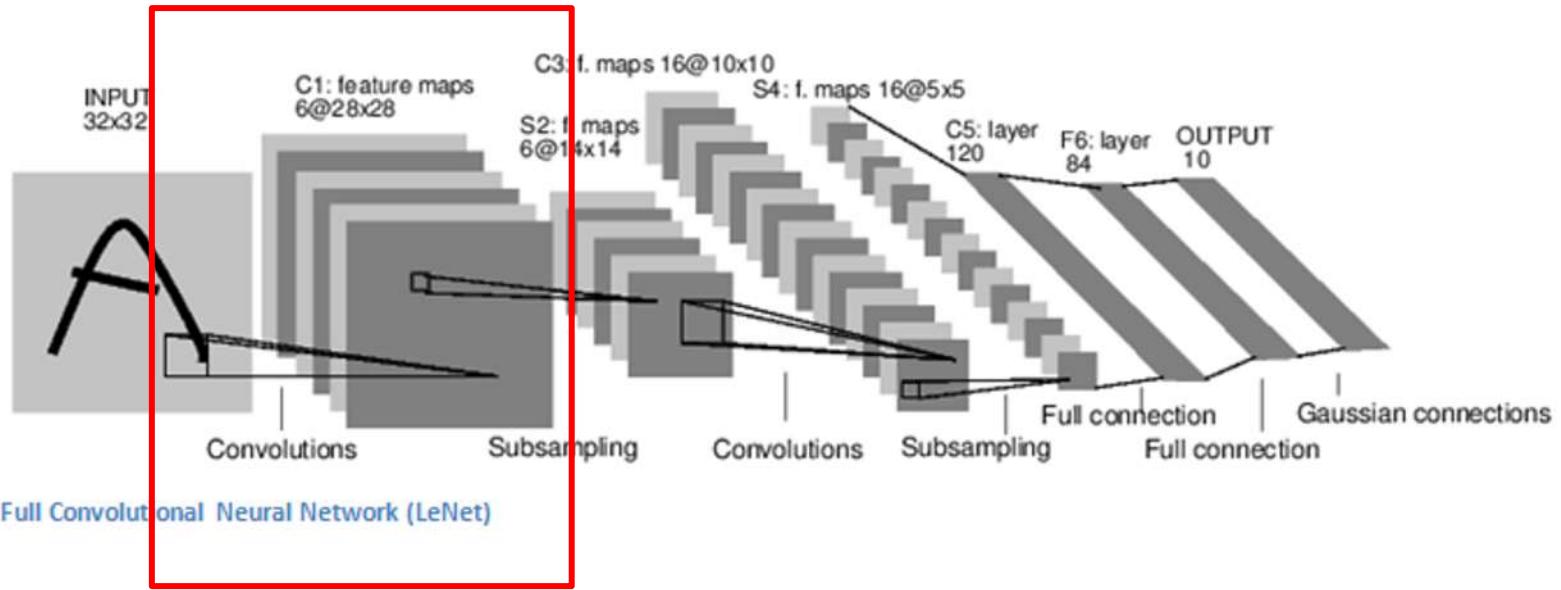
Свёрточная нейросеть



A Full Convolutional Neural Network (LeNet)

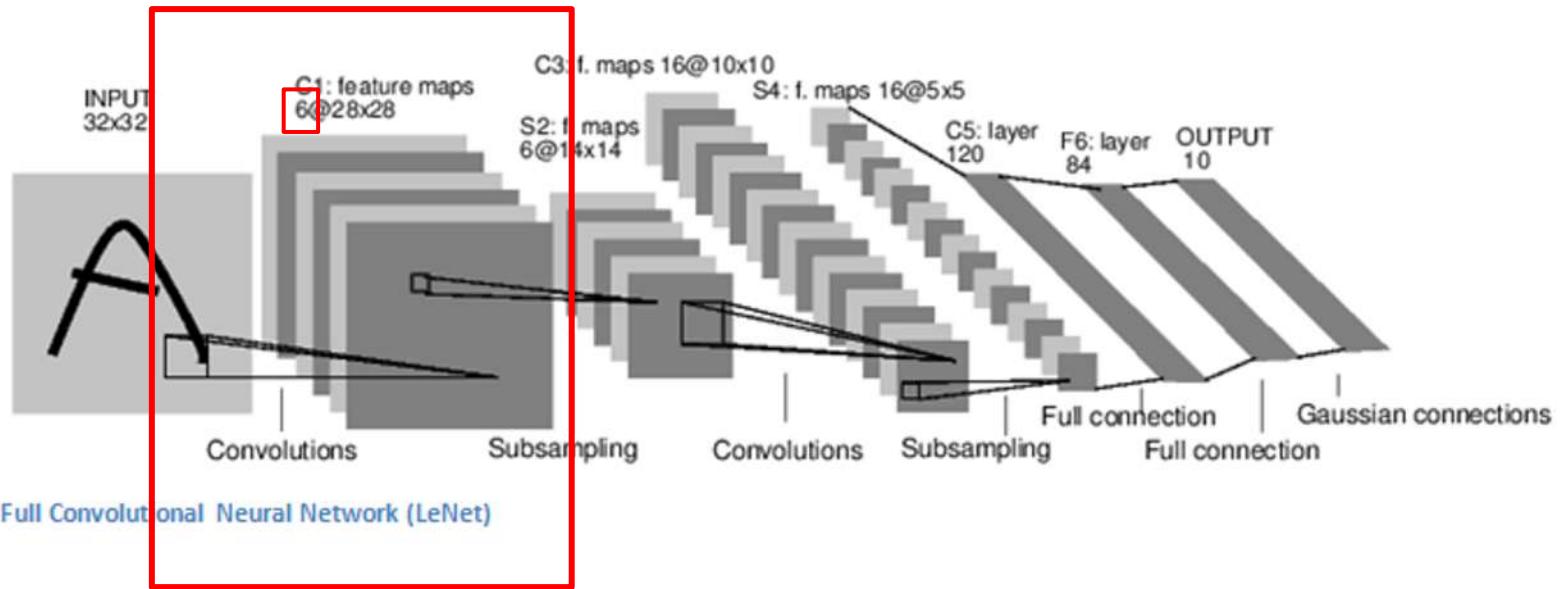
Свёрточный слой

Свёрточный слой



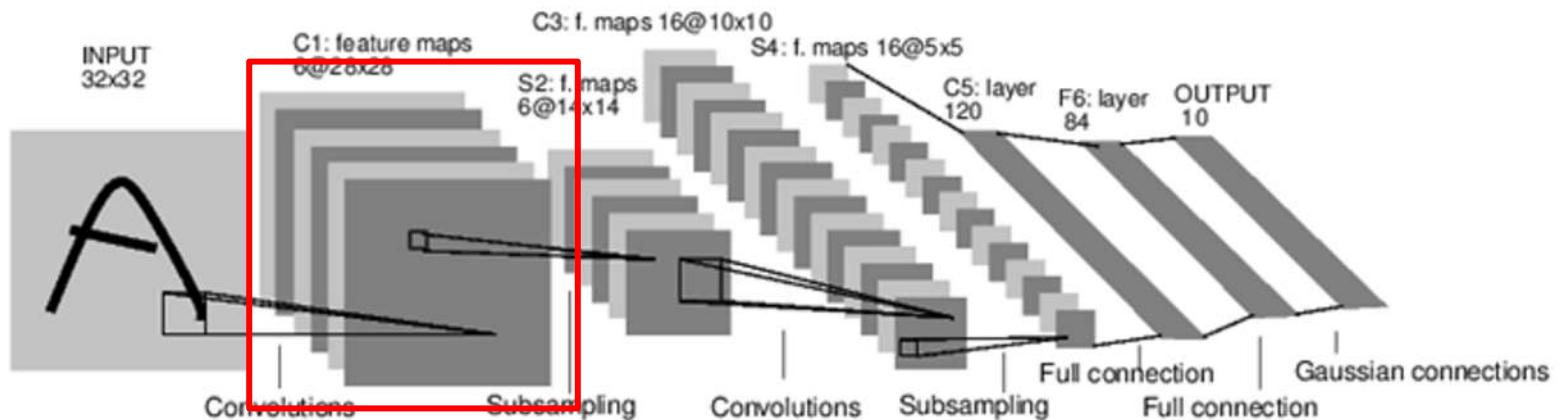
Свёрточный слой

Свёрточный слой



В данном случае
k=6 фильтров

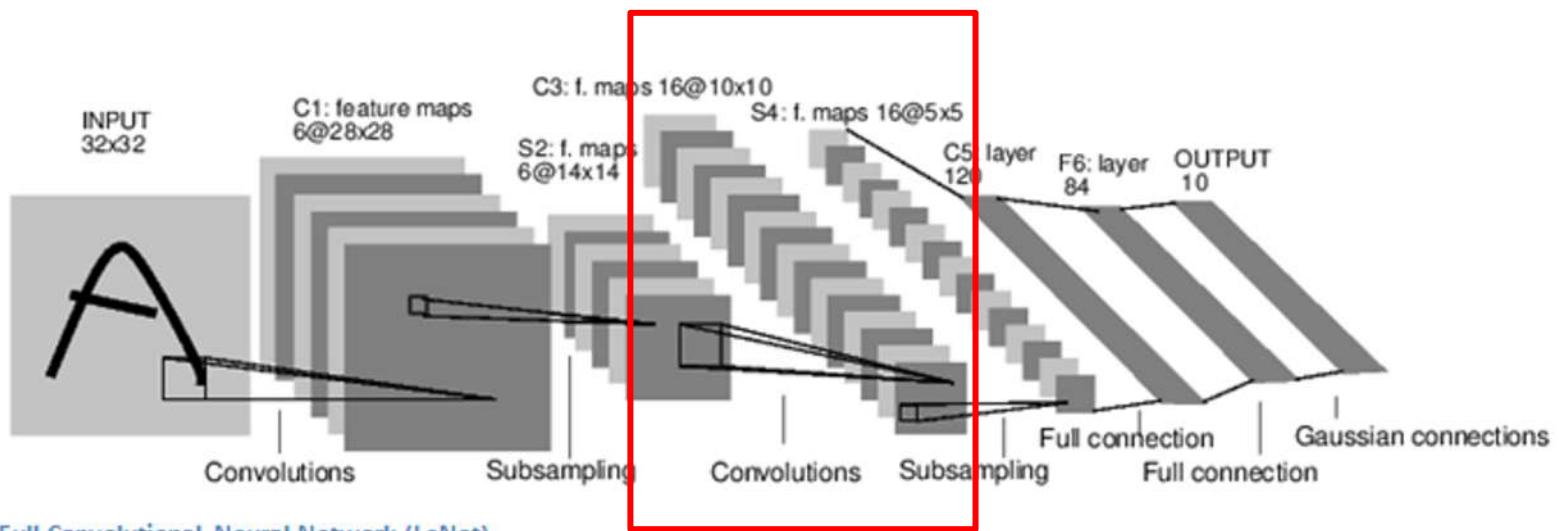
Свёрточный слой



A Full Convolutional Neural Network (LeNet)

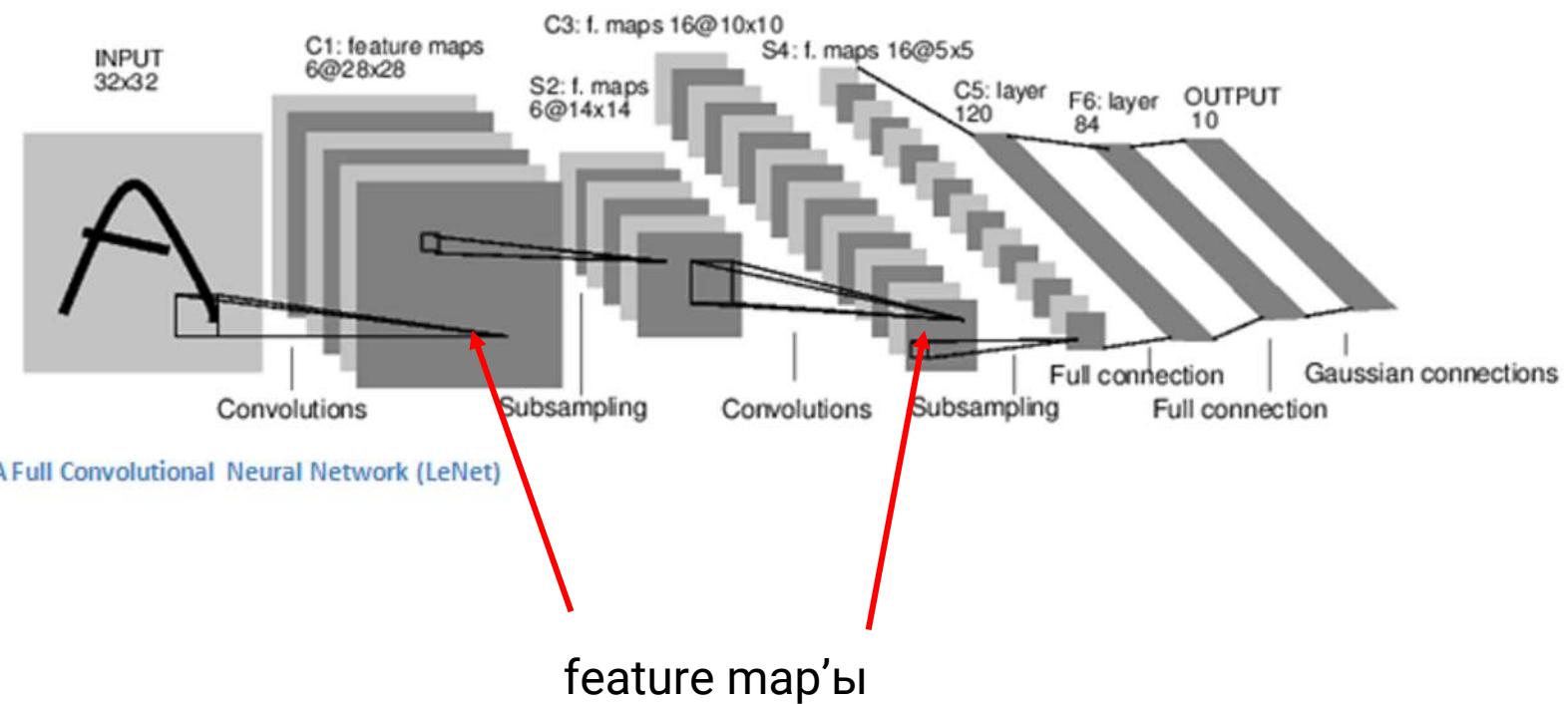
6 матриц размера 28x28
($28 = H - F + 1 = 32 - 5 + 1$)

Второй свёрточный слой

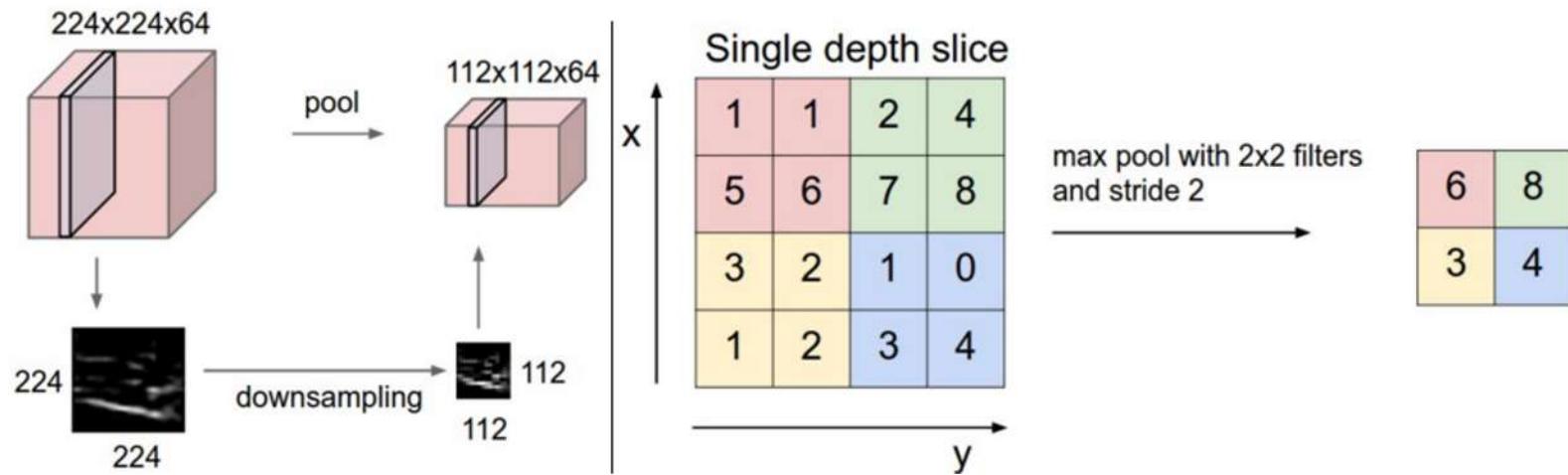


Второй свёрточный слой:
 $k_2=16$ фильтров, у каждого глубина $k=6$
(столько фильтров было в предыдущем
свёрточном слое)

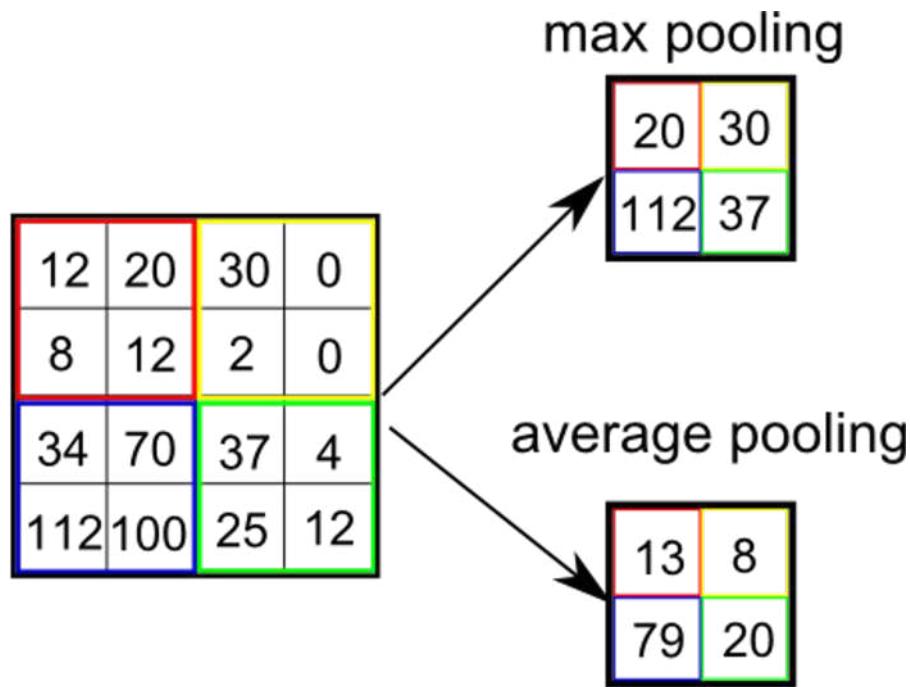
Feature maps



Pooling слой



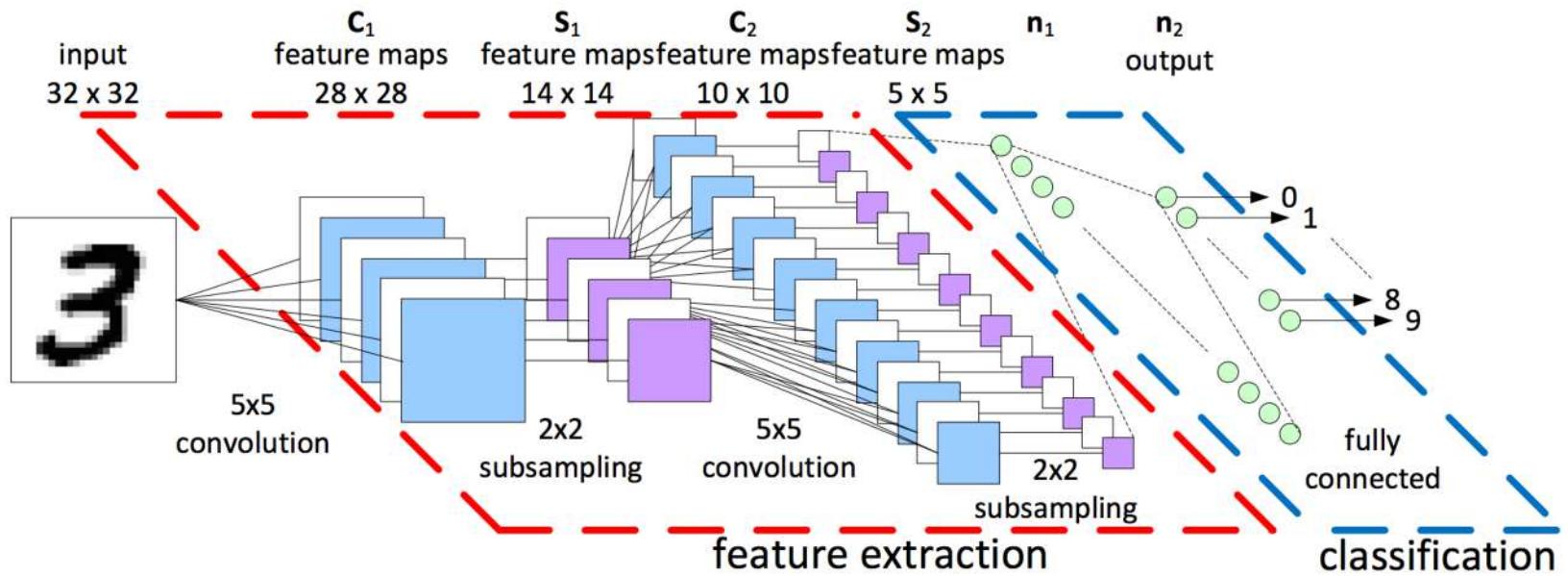
Pooling слой



Число параметров

- Веса свёрточного слоя = числа в фильтрах
- Фильтр ($F, F, 5$) имеет $F*F*5 + 1$ весов (ещё свободный член)
- Если свёрточный слой имеет k фильтров (F, F, C), то у него $k * (F*F*C) + k$ весов
(последнее “+ k ” -- так как у каждого фильтра есть ещё свободный член)
- Pooling-слои не имеют весов

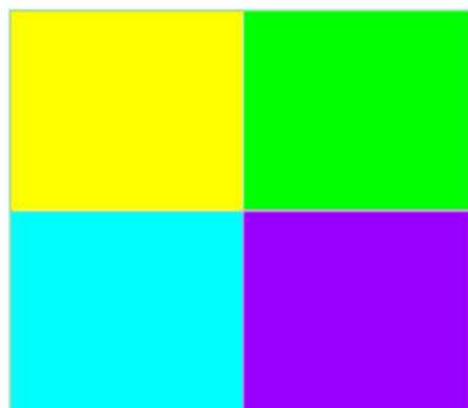
Forward pass: пример



Обучение CNN (backward pass)

- Backpropagation с учётом свёрток и пулингов
- Подробнее: <http://bit.ly/2DW6Odr>

X_{11}	X_{12}	X_{13}
X_{21}	X_{22}	X_{23}
X_{31}	X_{32}	X_{33}



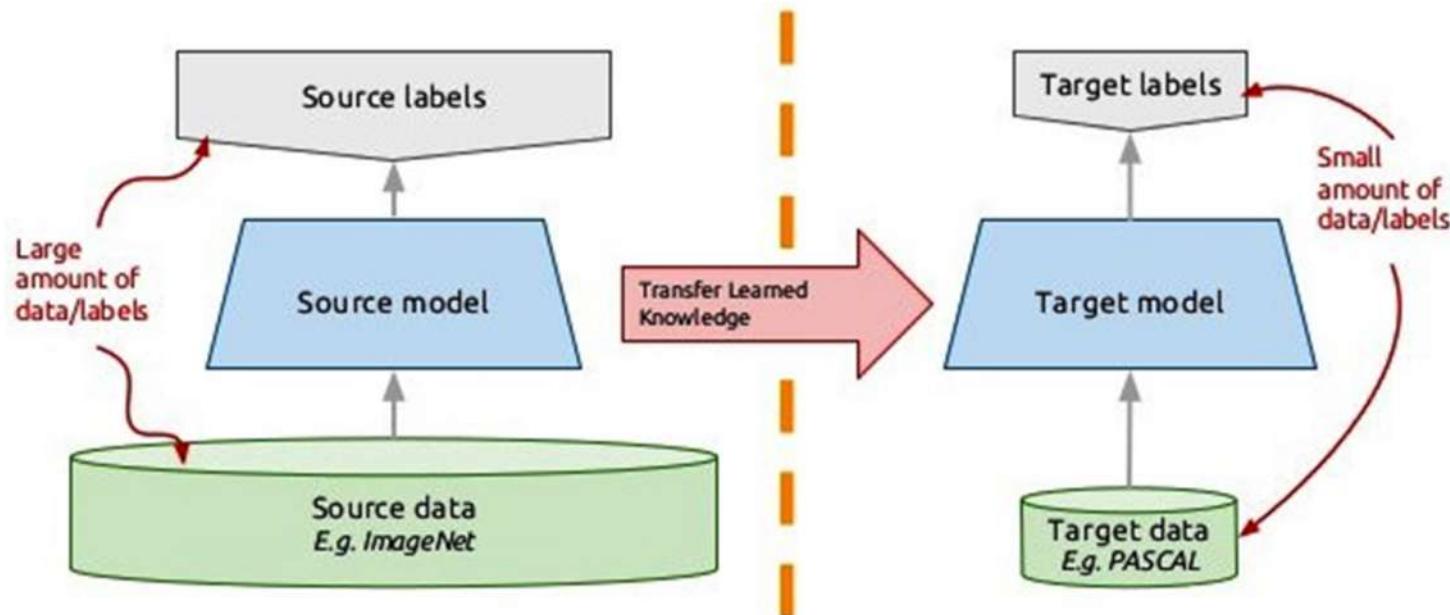
δh_{11}	δh_{12}
δh_{21}	δh_{22}

Transfer Learning

Свёрточные нейросети

Transfer Learning

Transfer learning: idea

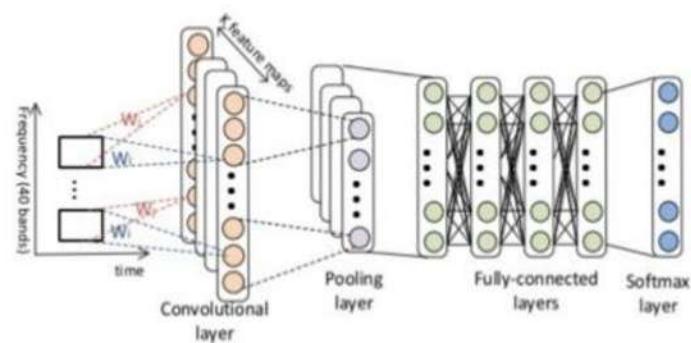


James Le

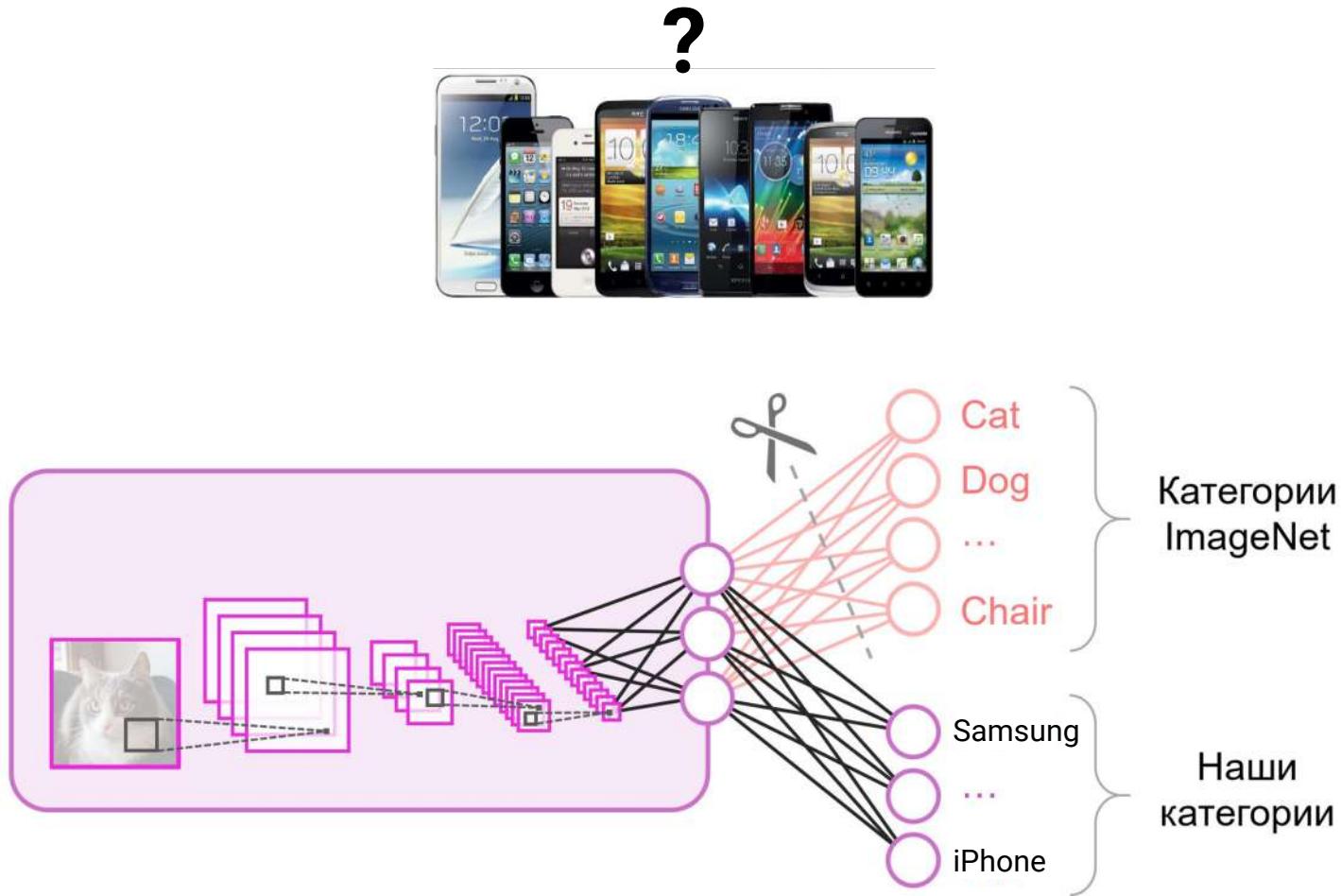
Transfer Learning

?

Convolutional Neural Network (CNN)



Transfer Learning



Рекуррентные нейронные сети

Зачем нужен еще один тип сетей?

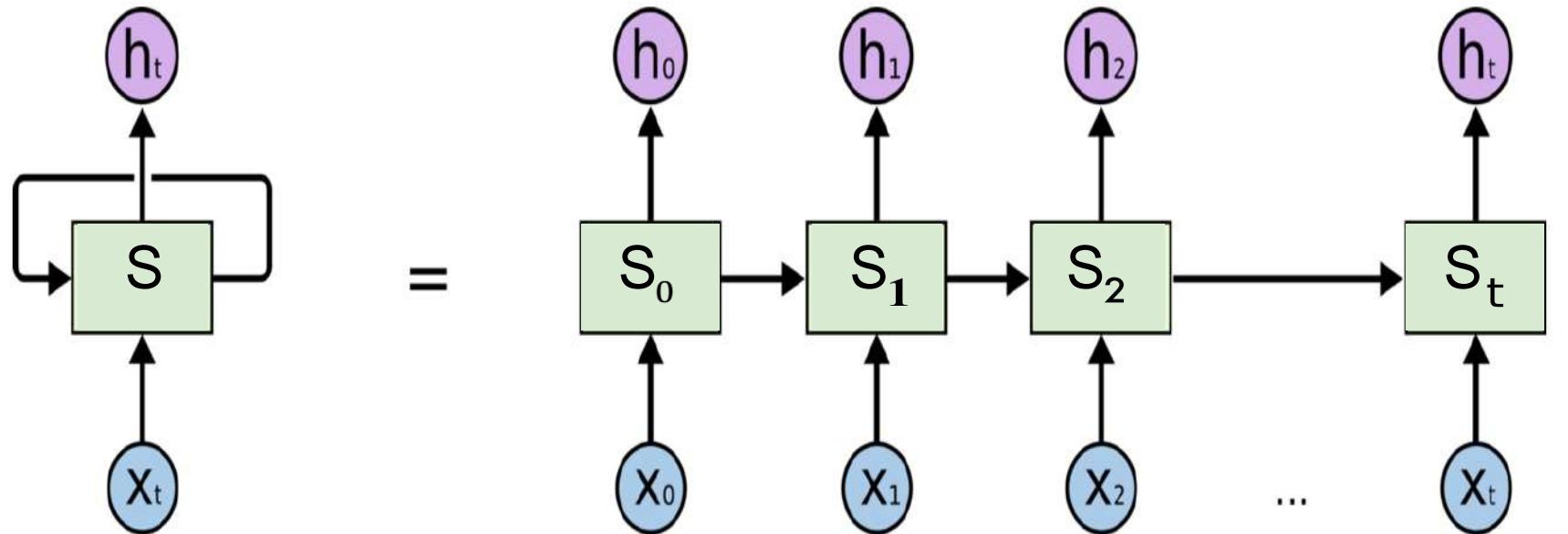
Примеры последовательностей в данных:

- Видео
- Аудио / музыка
- Текст
- Диалоги

Рекуррентные сети

- Нужны для работы с данными в виде последовательностей
- Реализуют “память” нейронной сети

Схема рекуррентного нейрона



I love you very much

Рекуррентные нейронные сети

forward pass

Рекуррентный нейрон

x_t – вход нейрона

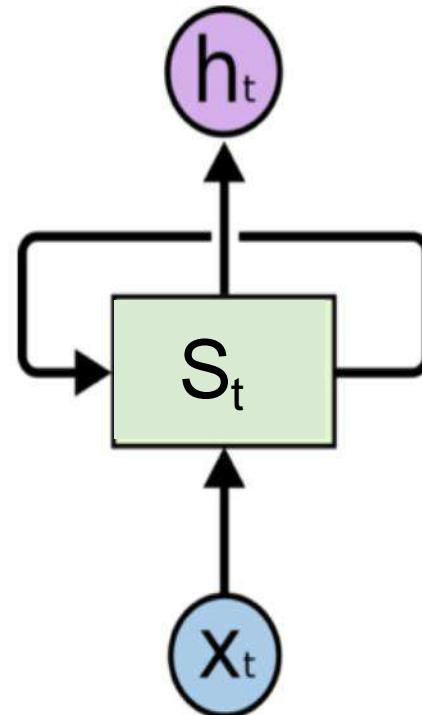
в момент времени t

s_t – скрытое состояние нейрона

в момент времени t (память)

h_t – выход нейрона

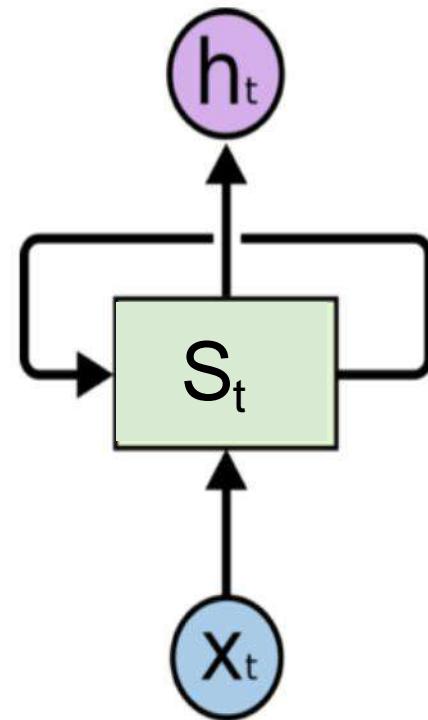
в момент времени t



Рекуррентный нейрон

$$s_t = f(\textcolor{red}{U}x_t + \textcolor{red}{W}s_{t-1})$$

$$h_t = softmax(\textcolor{red}{V}s_t)$$



Рекуррентные нейронные сети

backward pass

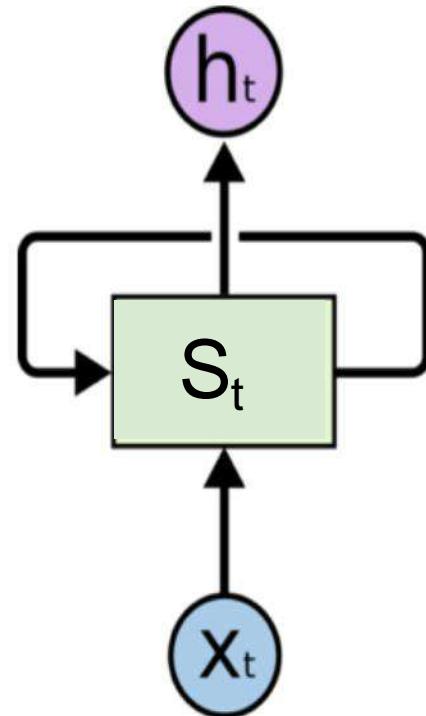
Backpropagation

$$s_t = f(\mathbf{U}x_t + \mathbf{W}s_{t-1})$$

$$h_t = softmax(\mathbf{V}s_t)$$

Нужно уметь вычислять градиенты

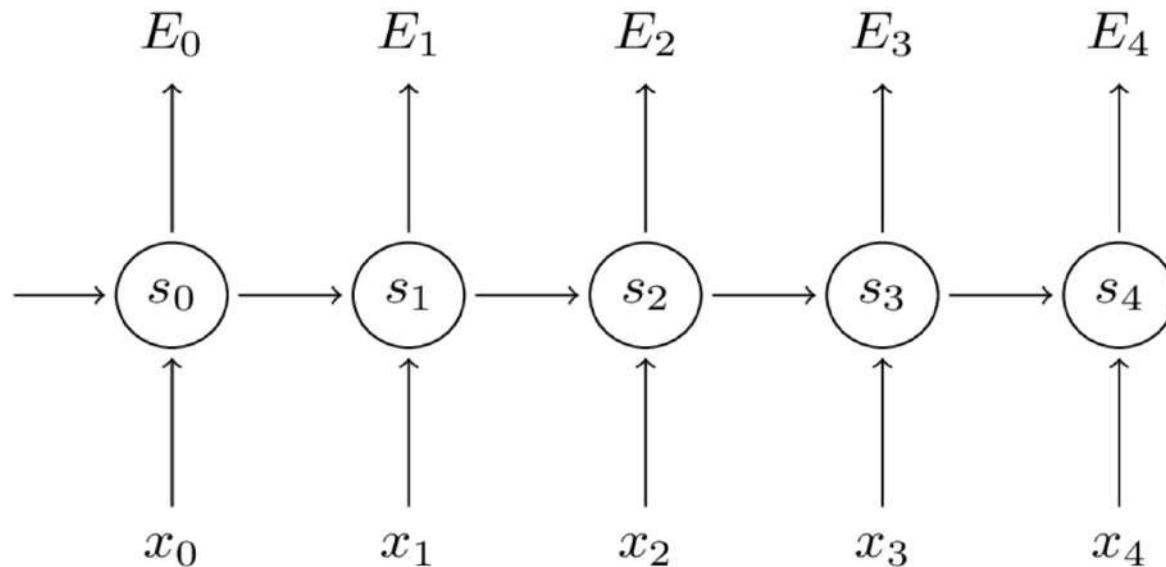
$$\mathbf{U}, \mathbf{W}, \mathbf{V}$$



Backpropagation

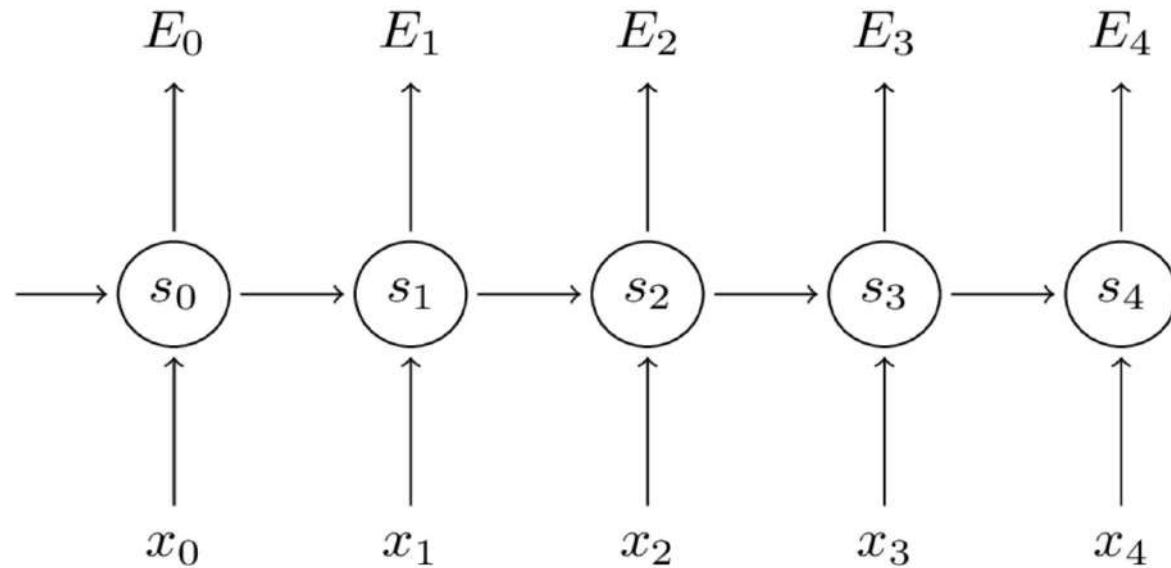
Идеи:

- Ошибка сети есть сумма ее ошибок для всех t
- Общий градиент по переменной есть сумма ее градиентов для всех t



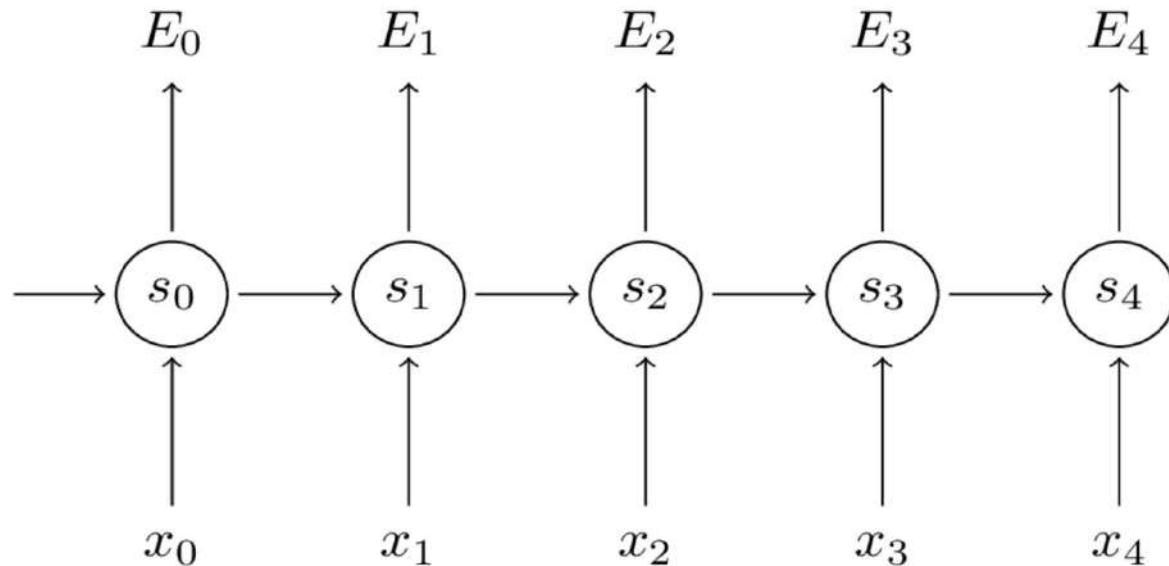
Backpropagation

$$\frac{\partial E}{\partial W} = \sum_t \frac{\partial E_t}{\partial W}$$



Backpropagation

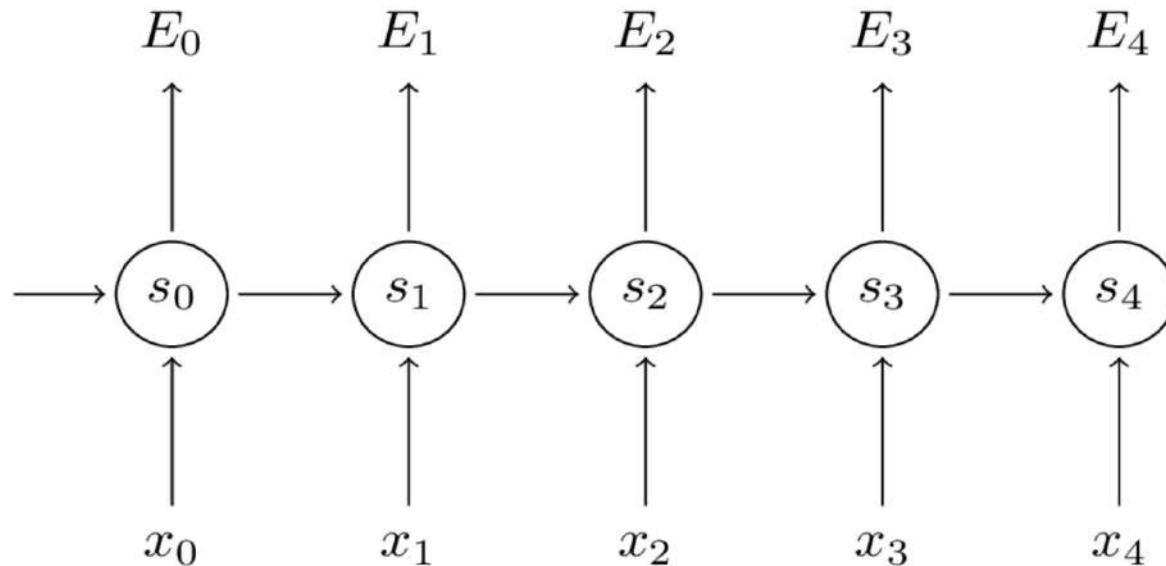
$$\frac{\partial E_3}{\partial W} = \frac{\partial E_3}{\partial \hat{y}_3} \frac{\partial \hat{y}_3}{\partial s_3} \frac{\partial s_3}{\partial W}$$



Backpropagation

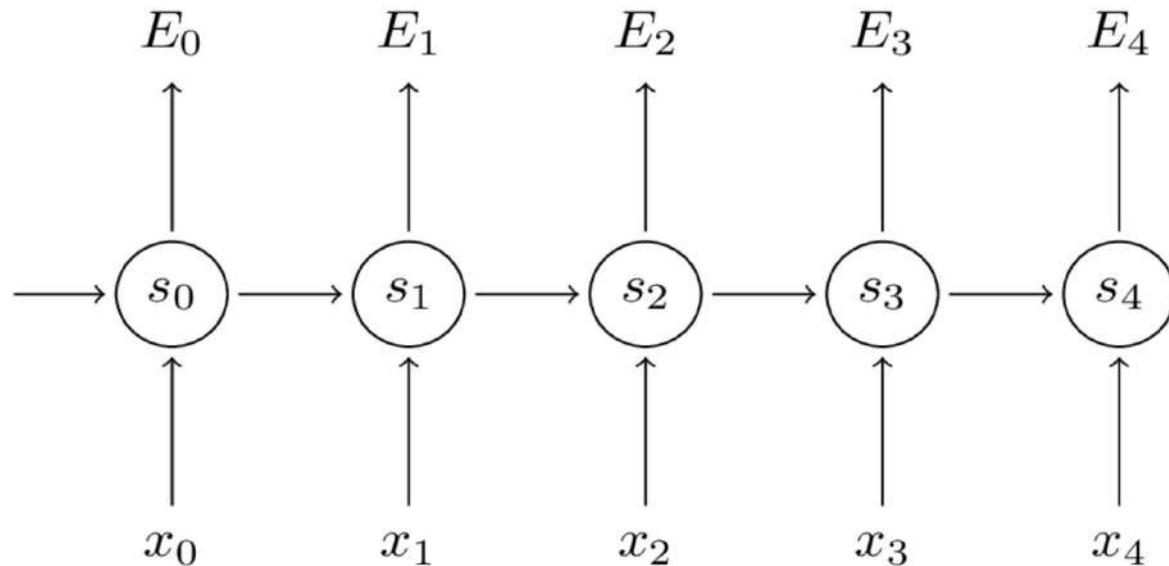
$$\frac{\partial E_3}{\partial W} = \frac{\partial E_3}{\partial \hat{y}_3} \frac{\partial \hat{y}_3}{\partial s_3} \frac{\partial s_3}{\partial W}$$

$$s_3 = f(\mathbf{U}x_3 + \mathbf{W}s_2)$$



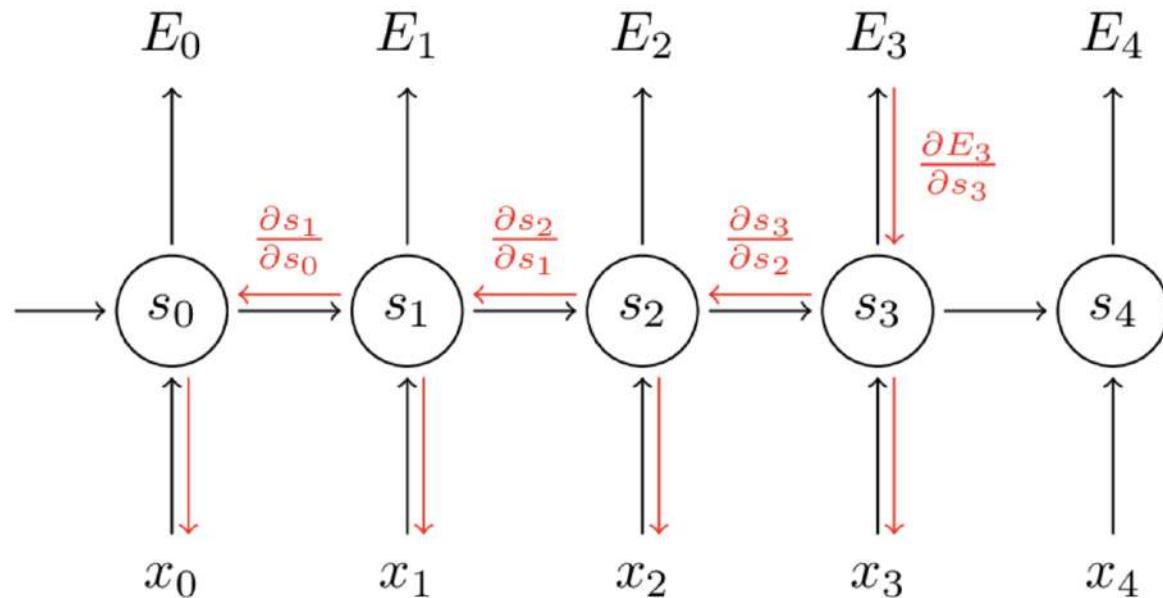
Backpropagation

$$\frac{\partial E_3}{\partial W} = \frac{\partial E_3}{\partial \hat{y}_3} \frac{\partial \hat{y}_3}{\partial s_3} \frac{\partial s_3}{\partial W} = \sum_{k=0}^3 \frac{\partial E_3}{\partial \hat{y}_3} \frac{\partial \hat{y}_3}{\partial s_3} \frac{\partial s_3}{\partial s_k} \frac{\partial s_k}{\partial W}$$



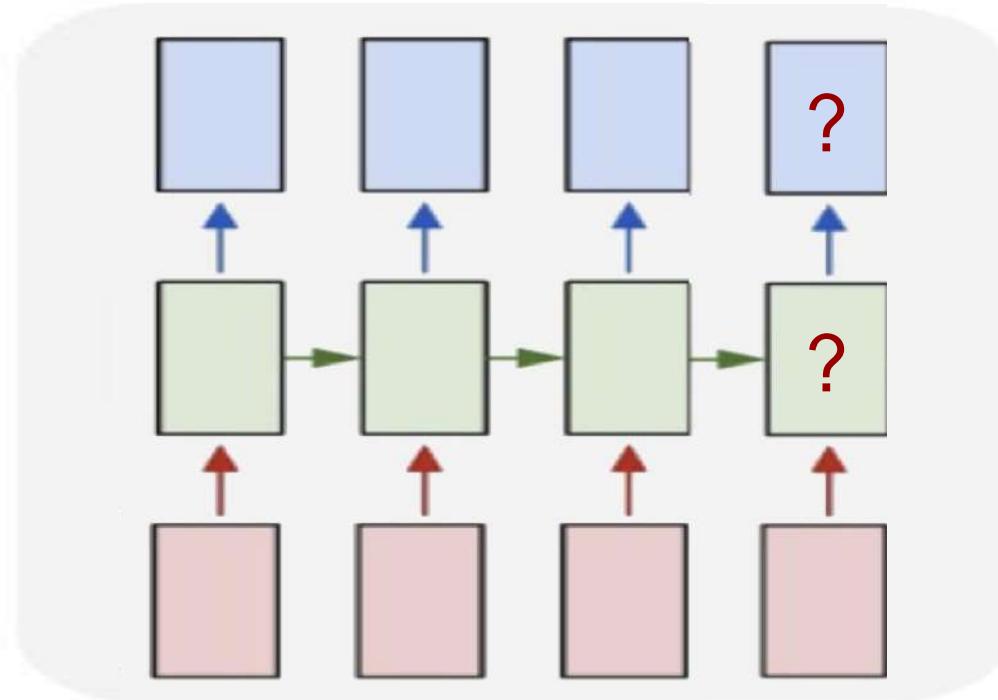
Backpropagation

$$\frac{\partial E_3}{\partial W} = \frac{\partial E_3}{\partial \hat{y}_3} \frac{\partial \hat{y}_3}{\partial s_3} \frac{\partial s_3}{\partial W} = \sum_{k=0}^3 \frac{\partial E_3}{\partial \hat{y}_3} \frac{\partial \hat{y}_3}{\partial s_3} \frac{\partial s_3}{\partial s_k} \frac{\partial s_k}{\partial W}$$



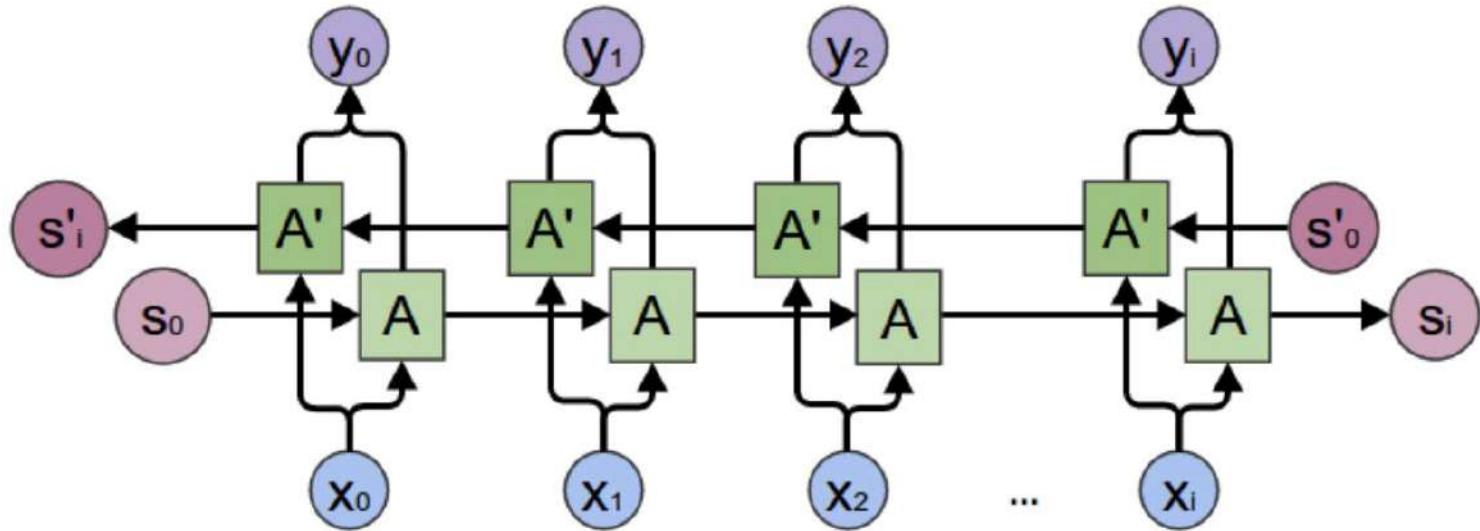
Bidirectional RNN

Bidirectional RNN



Он считает, что Лев ...

Bidirectional RNN

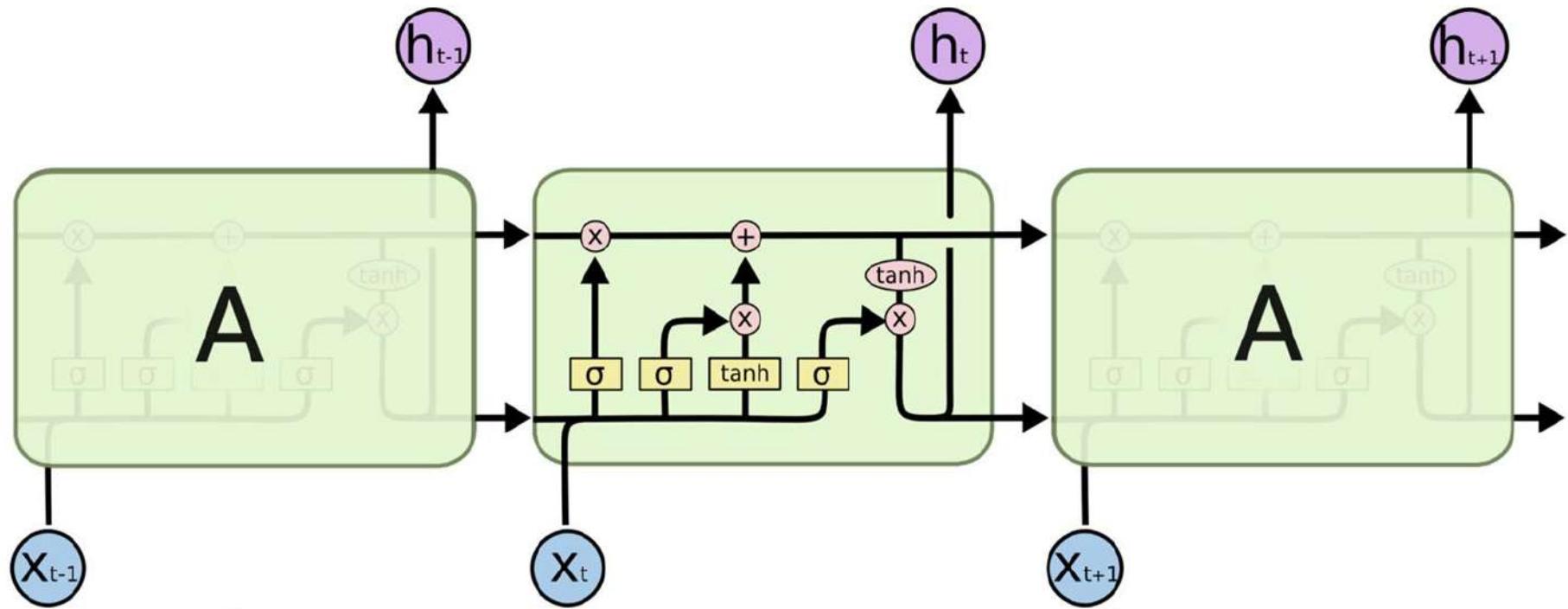


Он считает, что ... писатель

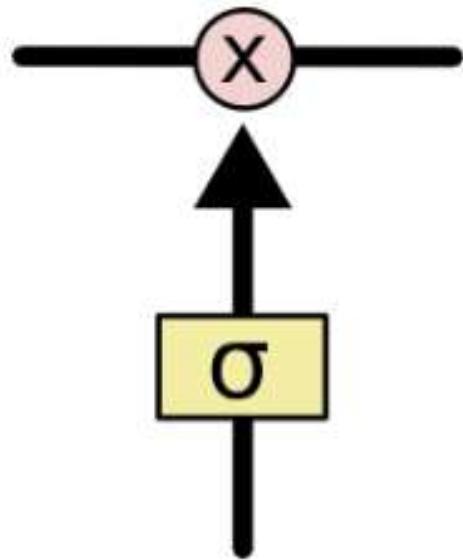
LSTM

Long-short term memory Unit

LSTM



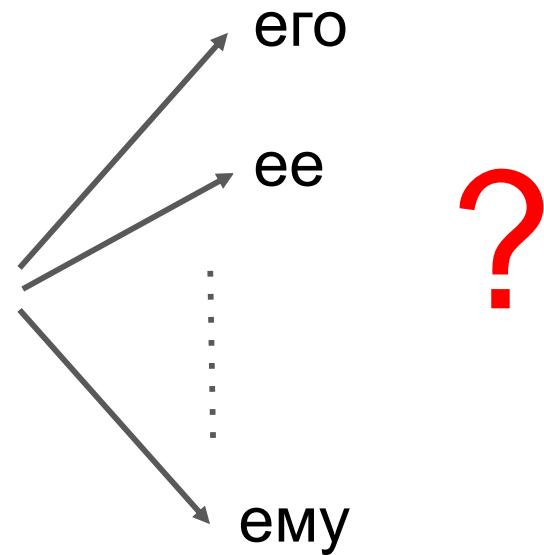
LSTM Gates



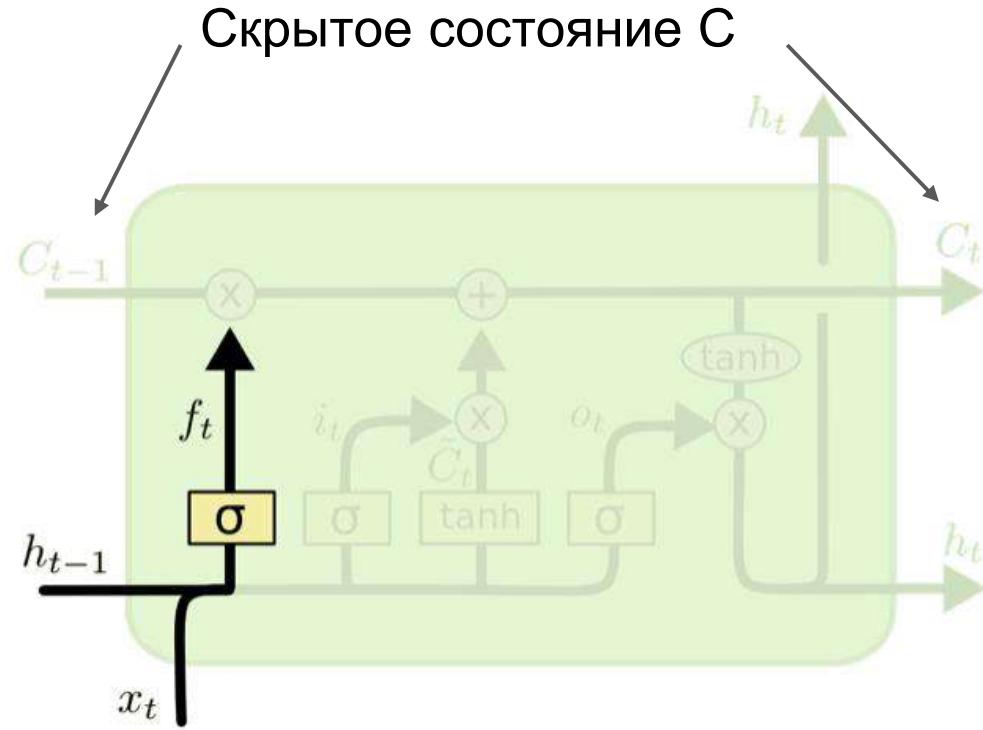
$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

LSTM Gates

Это мой пес Шарик. Я люблю...

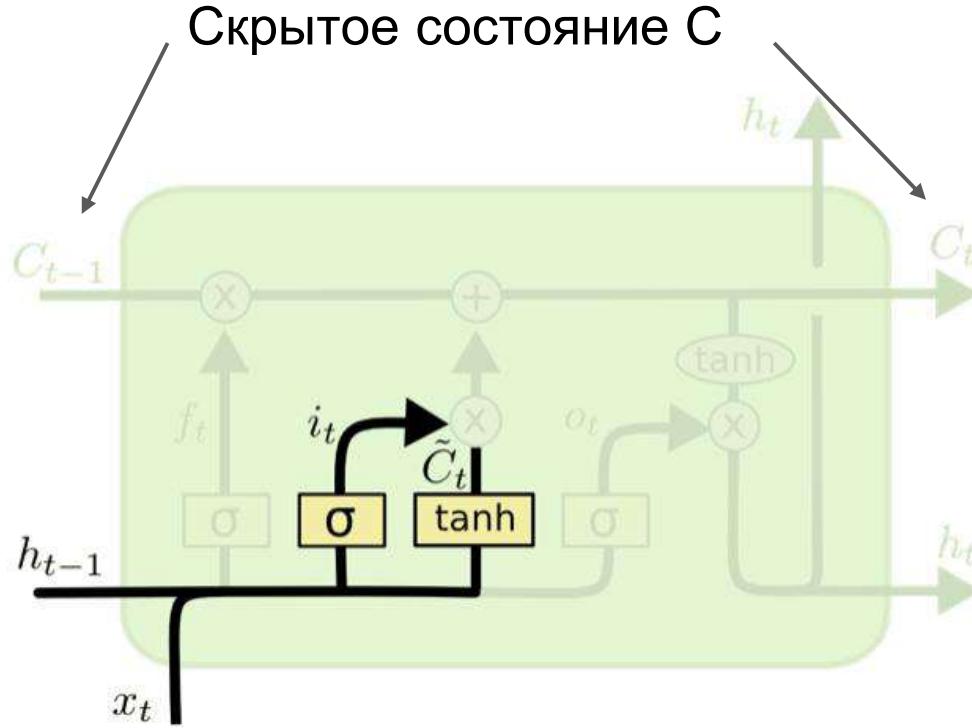


LSTM Forget gate



$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

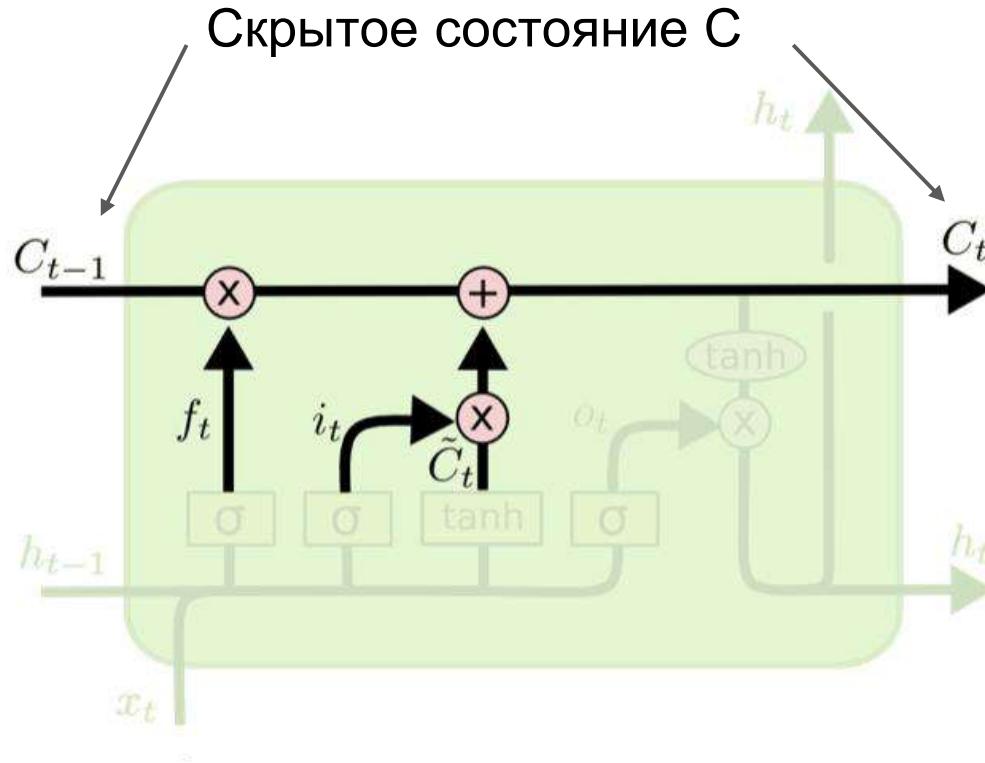
LSTM Update gate



update gate: $i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$

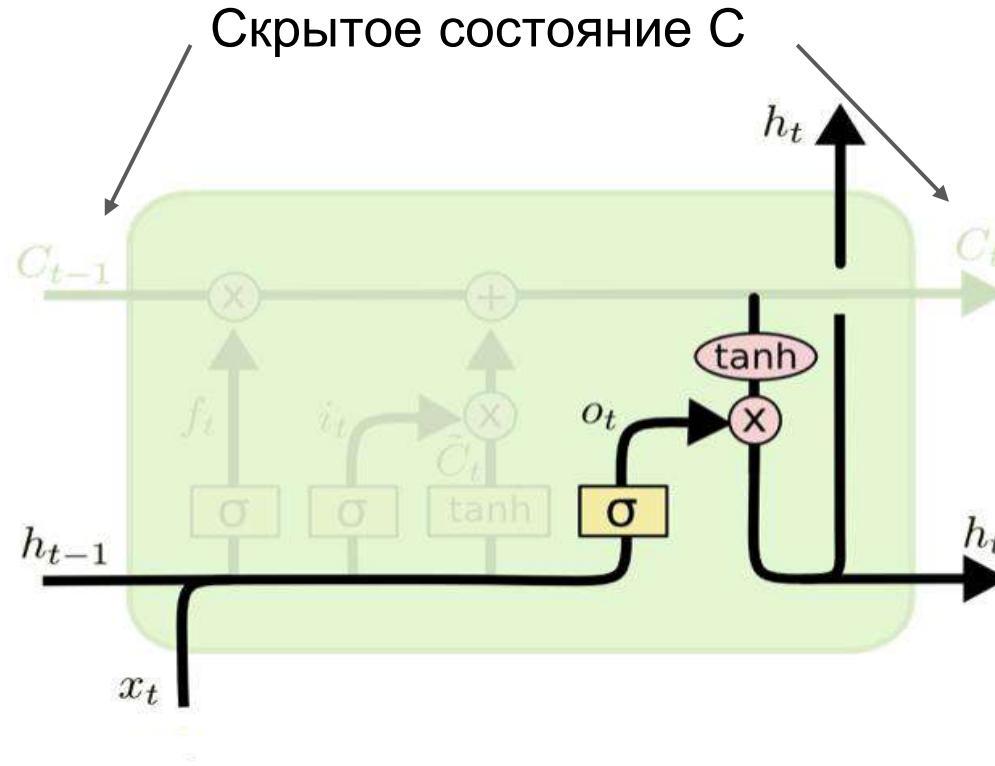
new information: $\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$

LSTM Hidden state



$$C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t$$

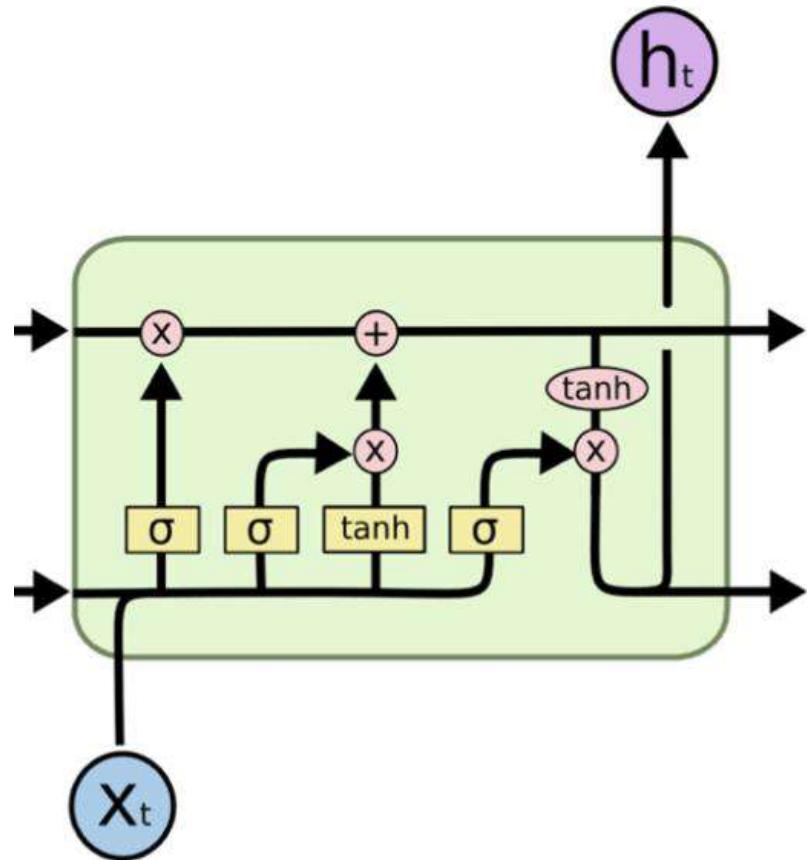
LSTM Output



$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

$$h_t = o_t \cdot \tanh(C_t)$$

LSTM



$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t$$

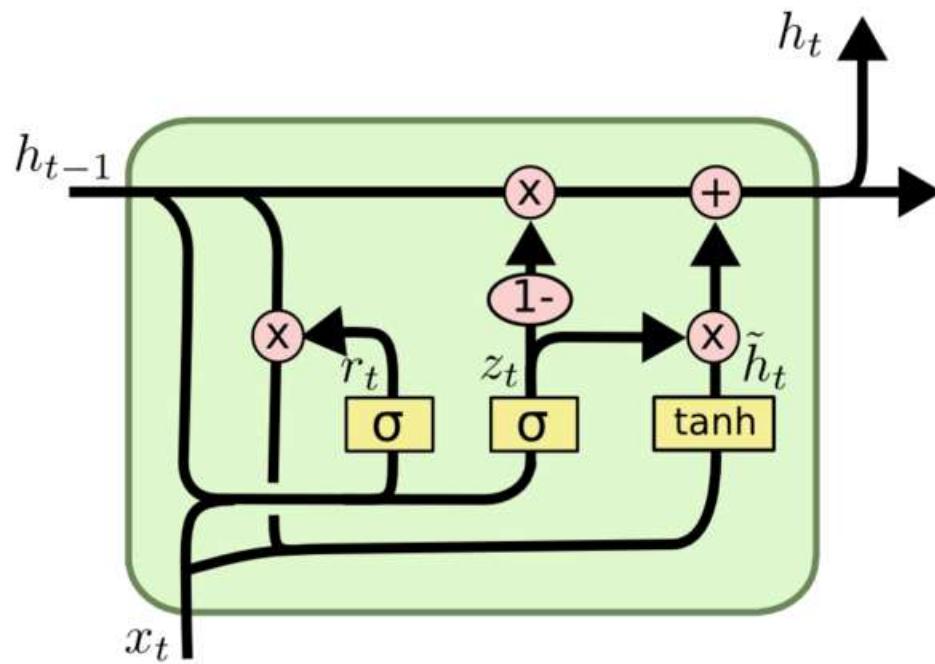
$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

$$h_t = o_t \cdot \tanh(C_t)$$

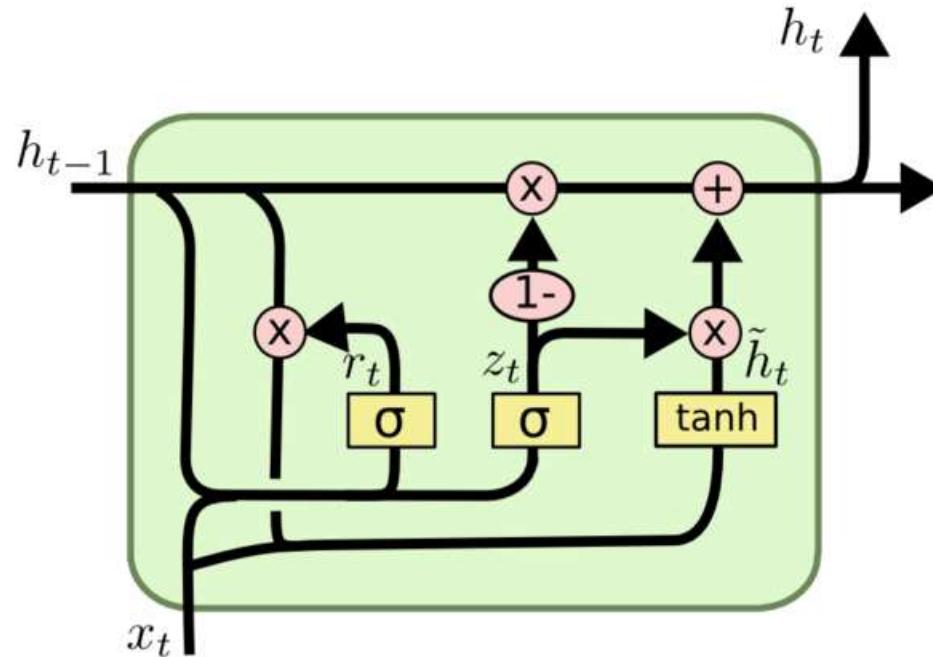
GRU

Gated recurrent Unit

GRU

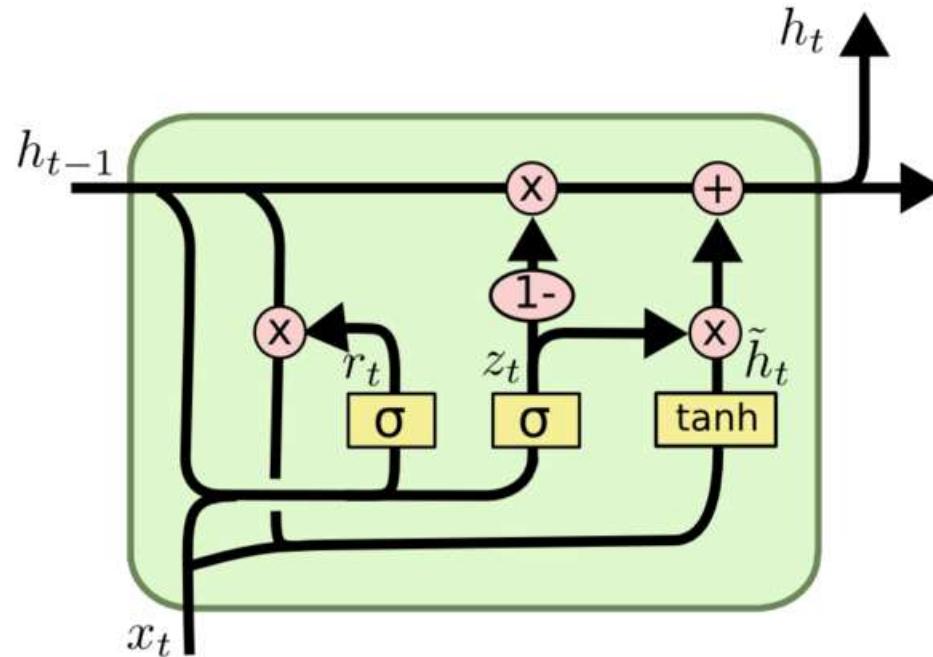


GRU



$$r_t = \sigma(\mathbf{W}_r \cdot [h_{t-1}, x_t] + \mathbf{b}_r)$$

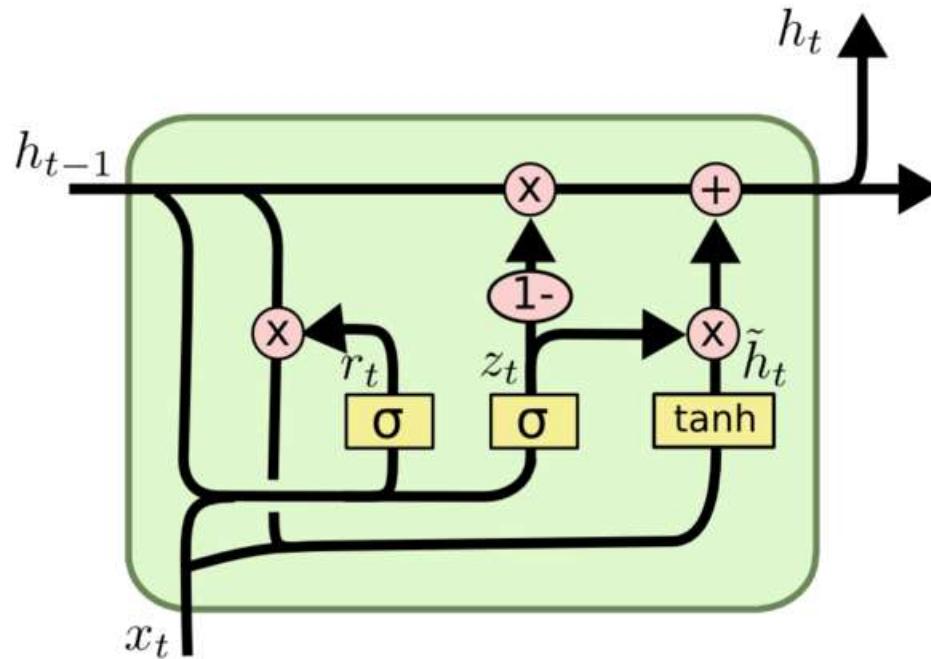
GRU



$$r_t = \sigma(\mathbf{W}_r \cdot [h_{t-1}, x_t] + \mathbf{b}_r)$$

$$\tilde{h}_t = \tanh(\mathbf{W}_h \cdot [r_t \cdot h_{t-1}, x_t] + \mathbf{b}_h)$$

GRU



$$r_t = \sigma(\mathbf{W}_r \cdot [h_{t-1}, x_t] + \mathbf{b}_r)$$

$$\tilde{h}_t = \tanh(\mathbf{W}_h \cdot [r_t \cdot h_{t-1}, x_t] + \mathbf{b}_h)$$

$$z_t = \sigma(\mathbf{W}_z \cdot [h_{t-1}, x_t] + \mathbf{b}_z)$$

$$h_t = (1 - z_t) \cdot h_{t-1} + z_t \cdot (\tilde{h}_t)$$