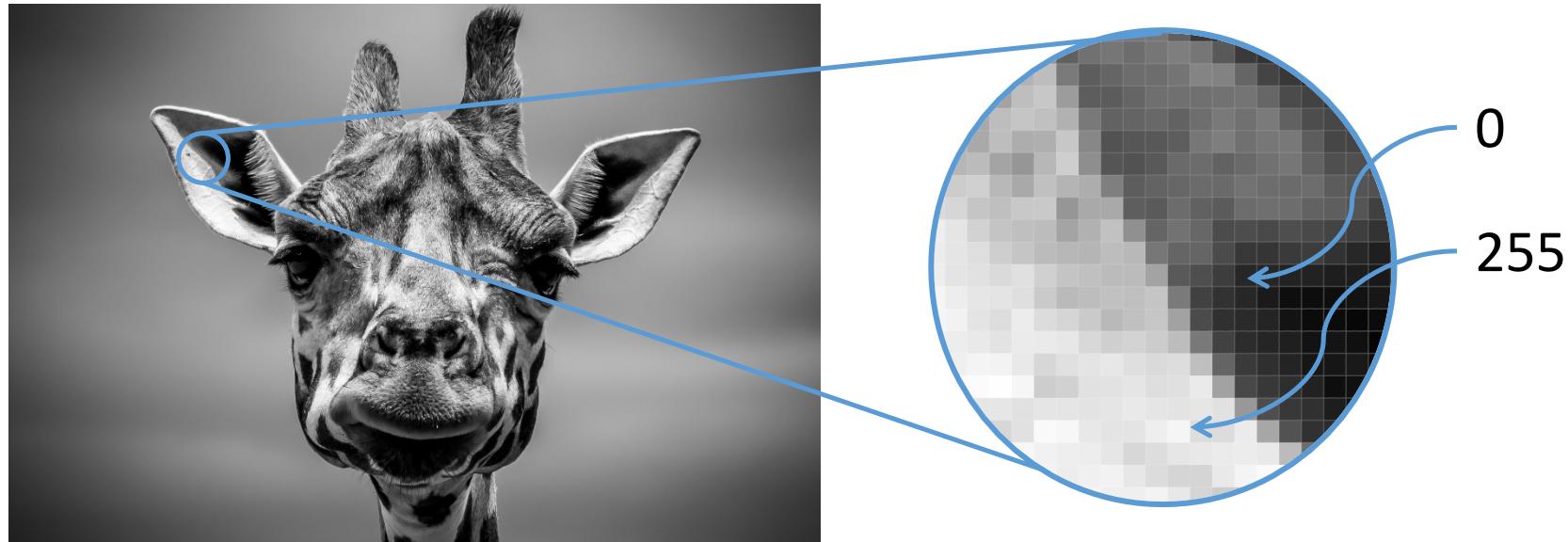


LSML #6

Сверточные сети

Как компьютер видит картинку

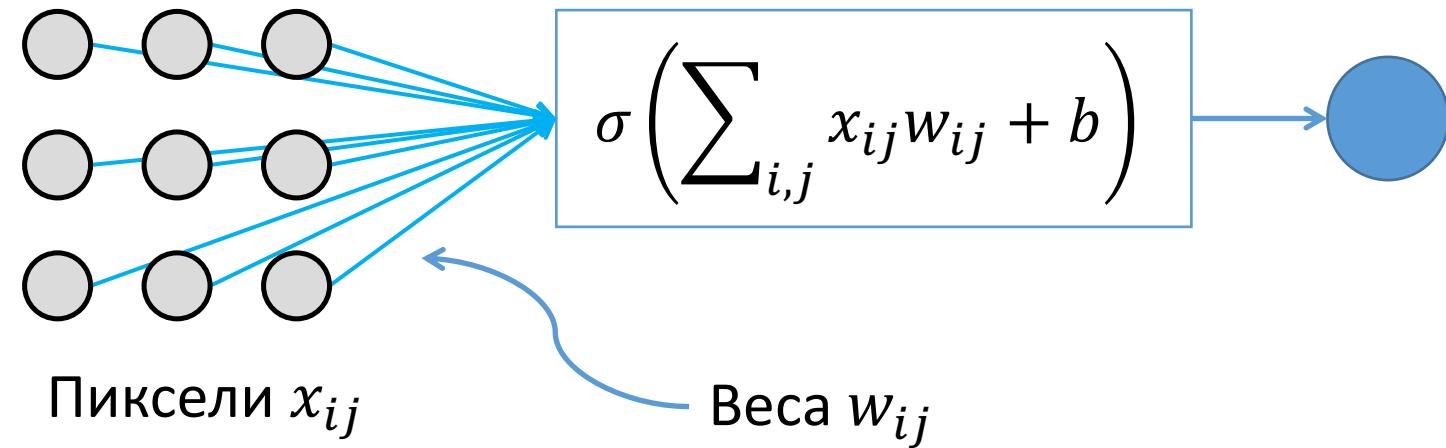
- Матрица яркостей пикселей (например размера 300 x 300)
- Яркости пикселей от 0 до 255



- В цветной картинке 3 канала яркости: красный, зеленый, синий

Как применить к картинке нейросеть?

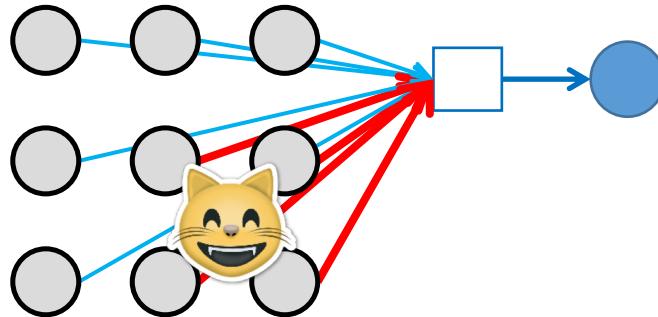
Нейрон поверх всех пикселей?



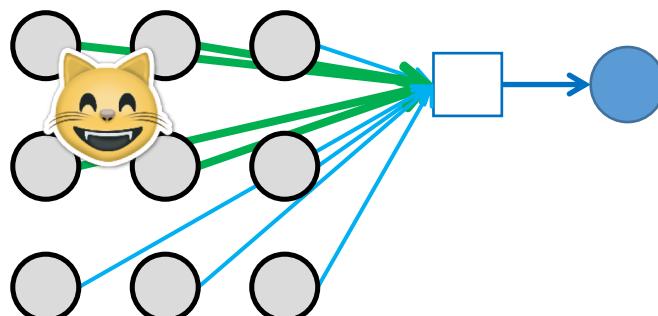
Не, так не работает!

Почему нет?

- Хотим выучить детектор кота:



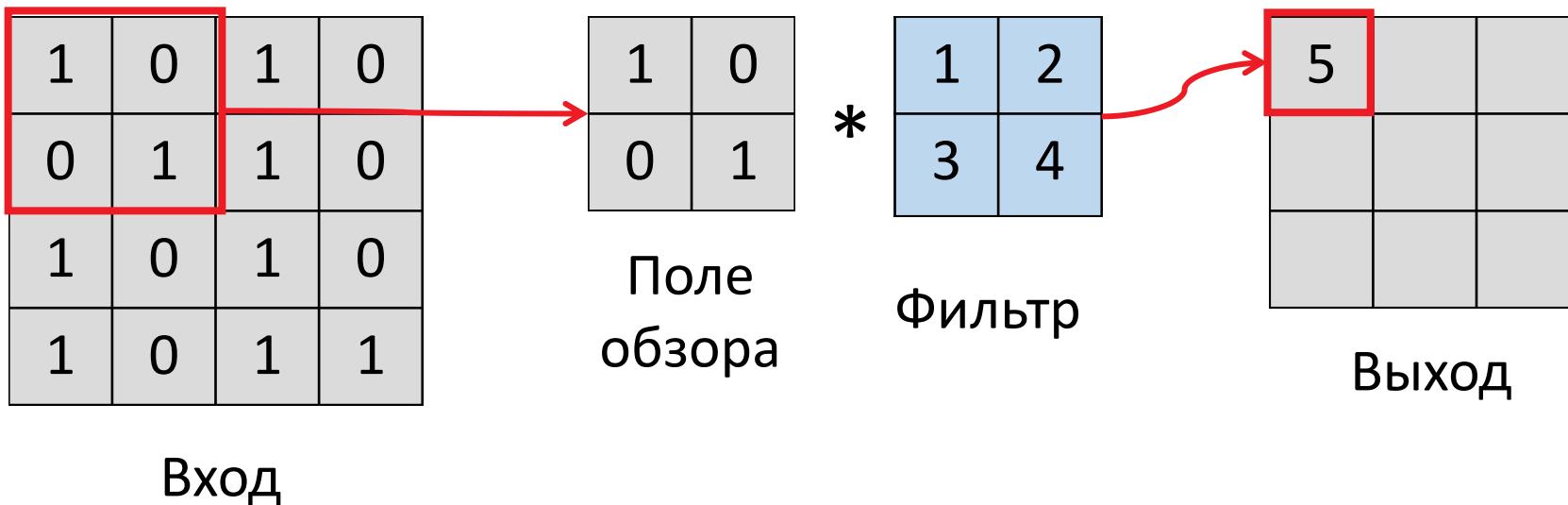
На этой картинке **красные** веса
 w_{ij} немного изменяются при
градиентном спуске



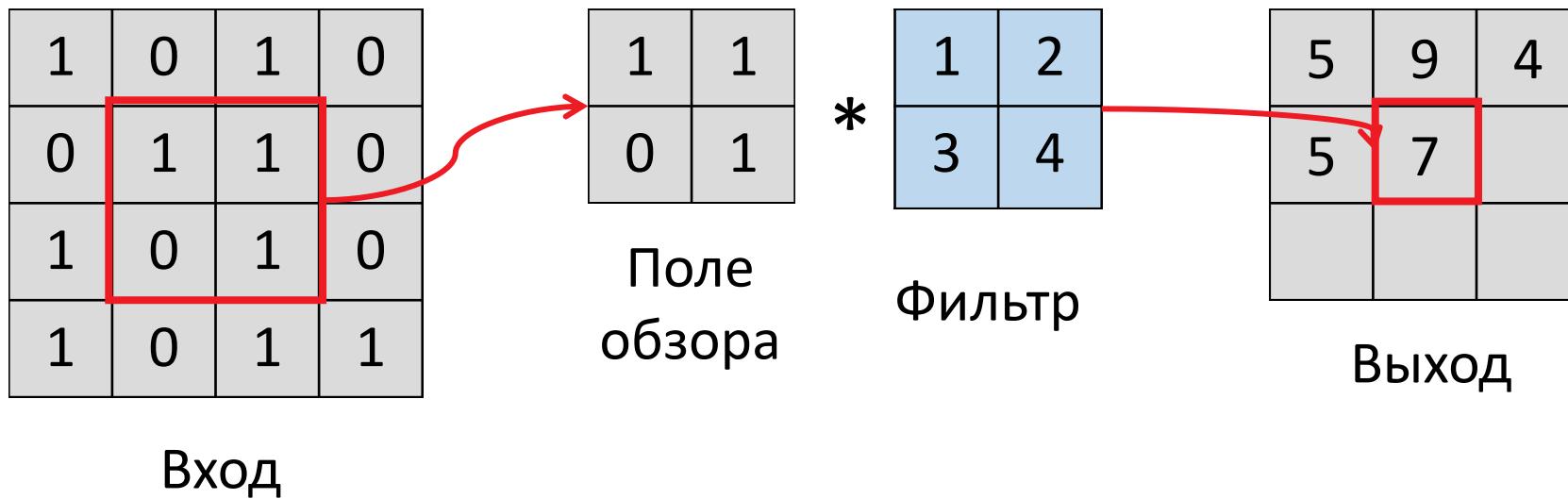
А на этой **зеленые** веса w_{ij}
изменяются...

- Мы учим те же самые «признаки кота» в разных областях картинки и не полностью используем наши прецеденты!
- А вдруг нам покажут кота в другой части картинки?

Операция свертки



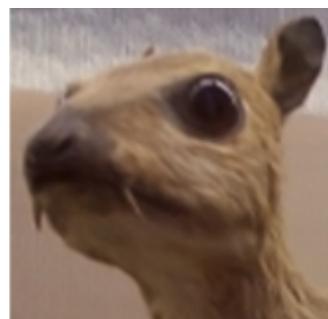
Операция свертки



Свертки применяют к картинкам давно



Свертки применяют к картинкам давно



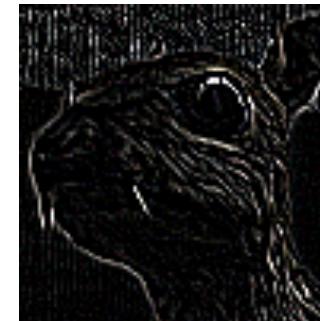
Входная
картинка

Фильтр

$$\begin{matrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{matrix}$$

*

=



Поиск краев

$$\begin{matrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{matrix}$$

*

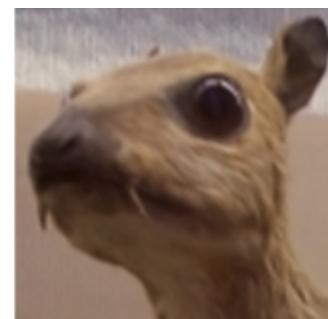
=



Повышение
резкости

Добавляет к картинке края

Свертки применяют к картинкам давно *как в Photoshop*



Входная
картинка

Фильтр

$$\begin{array}{|c|c|c|} \hline -1 & -1 & -1 \\ \hline -1 & 8 & -1 \\ \hline -1 & -1 & -1 \\ \hline \end{array}$$

*

=



Поиск краев

$$\begin{array}{|c|c|c|} \hline 0 & -1 & 0 \\ \hline -1 & 5 & -1 \\ \hline 0 & -1 & 0 \\ \hline \end{array}$$

*

=

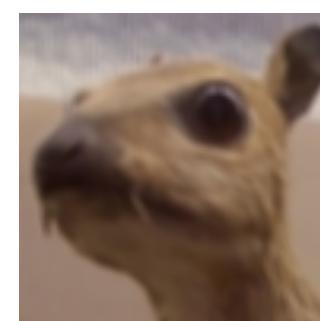


Повышение
резкости

$$\begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$

$\ast \frac{1}{9}$

=



Размытие

Свертка похожа на корреляцию с шаблоном

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

Вход

*

1	0
0	1

Фильтр

=

0	0	0
0	1	0
0	0	2

Выход

Свертка похожа на корреляцию с шаблоном

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

Вход

*

1	0
0	1

=

0	0	0
0	1	0
0	0	2

Фильтр

Выход

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

Вход

*

1	0
0	1

=

0	0	0
0	0	1
0	1	0

Фильтр

Выход

Свертка похожа на корреляцию с шаблоном

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

Вход

*

1	0
0	1

=

0	0	0
0	1	0
0	0	2

Max = 2

Фильтр

Выход

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

Вход

*

1	0
0	1

=

0	0	0
0	0	1
0	1	0

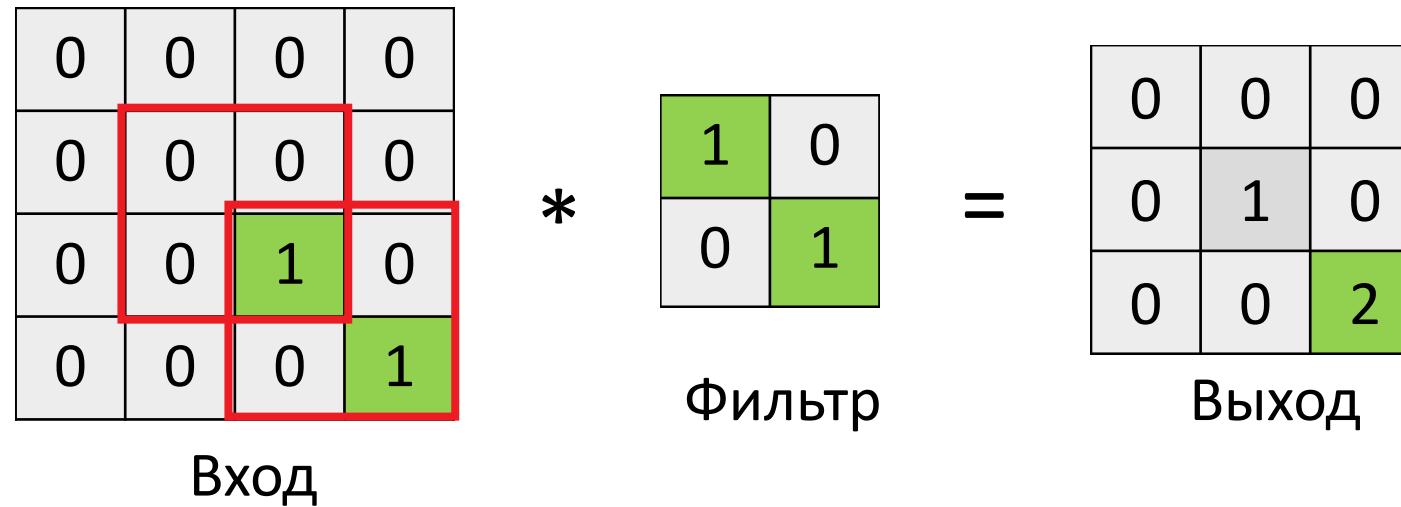
Max = 1

Фильтр

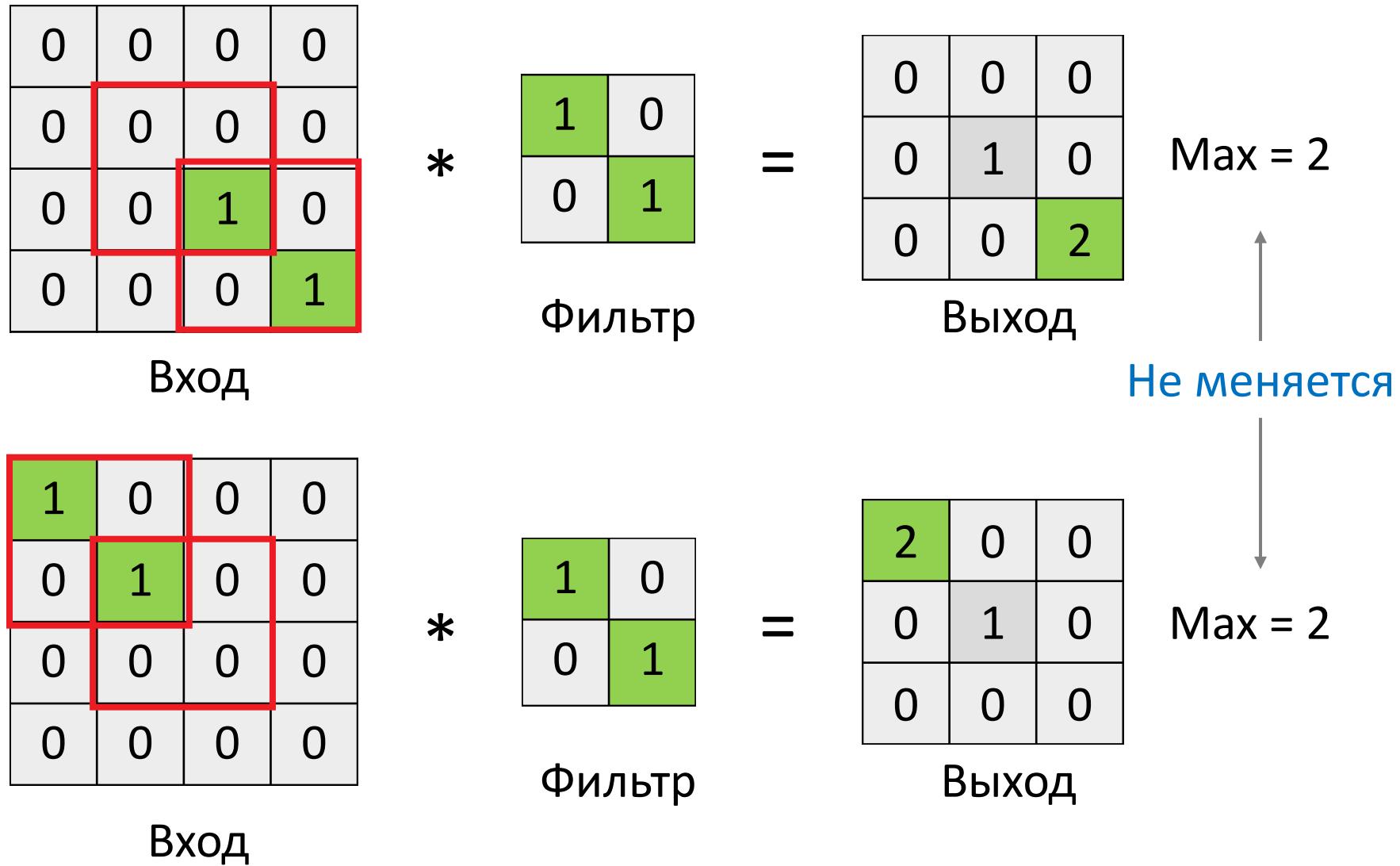
Выход

Простой
классификатор

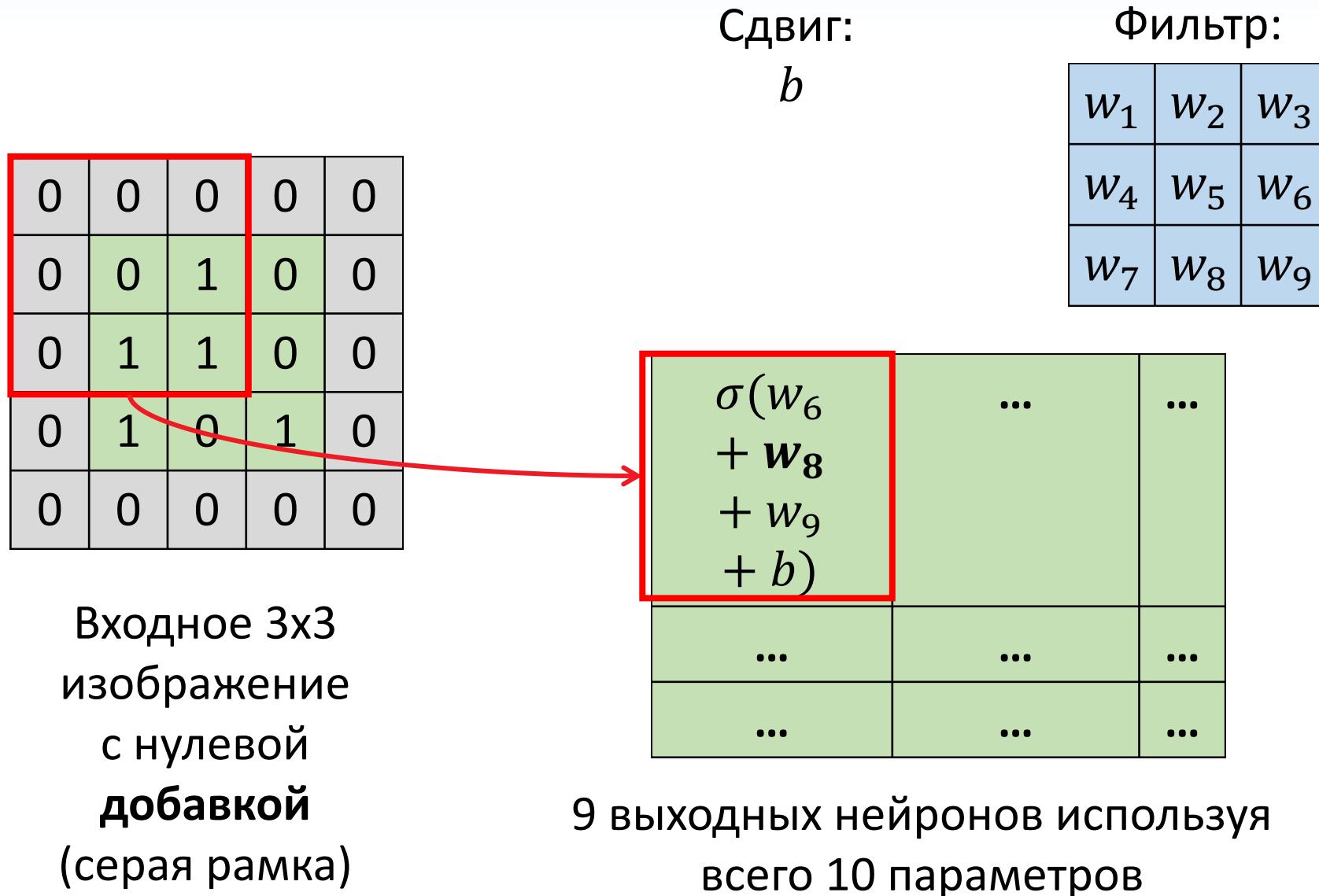
Свертка и сдвиг коммутативны



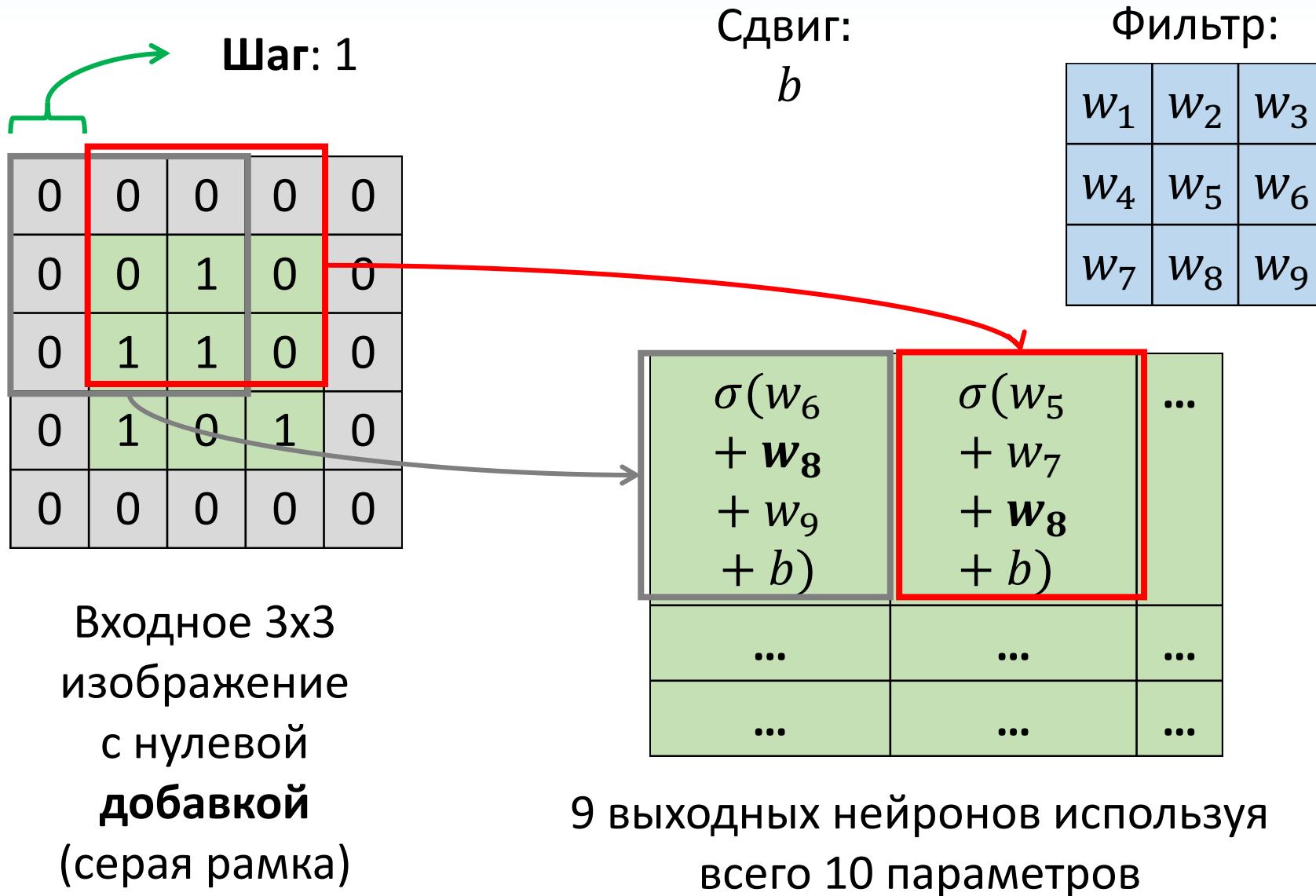
Свертка и сдвиг коммутативны



Сверточный слой в нейросети

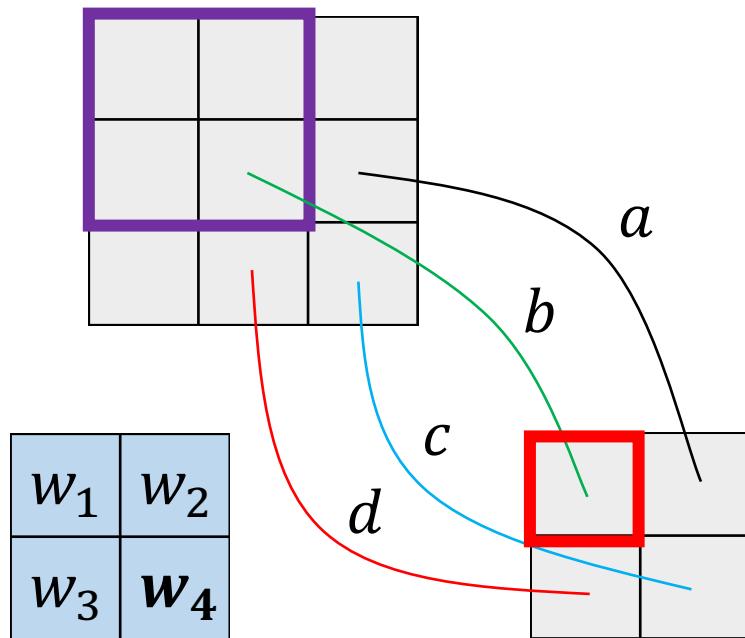


Сверточный слой в нейросети



Как считать градиенты

Представим, что все использования w_4 это разные веса:



$$a = a - \gamma \frac{\partial L}{\partial a} \quad b = b - \gamma \frac{\partial L}{\partial b}$$

$$c = c - \gamma \frac{\partial L}{\partial c} \quad d = d - \gamma \frac{\partial L}{\partial d}$$

$$w_4 = w_4 - \gamma \left(\frac{\partial L}{\partial a} + \frac{\partial L}{\partial b} + \frac{\partial L}{\partial c} + \frac{\partial L}{\partial d} \right)$$

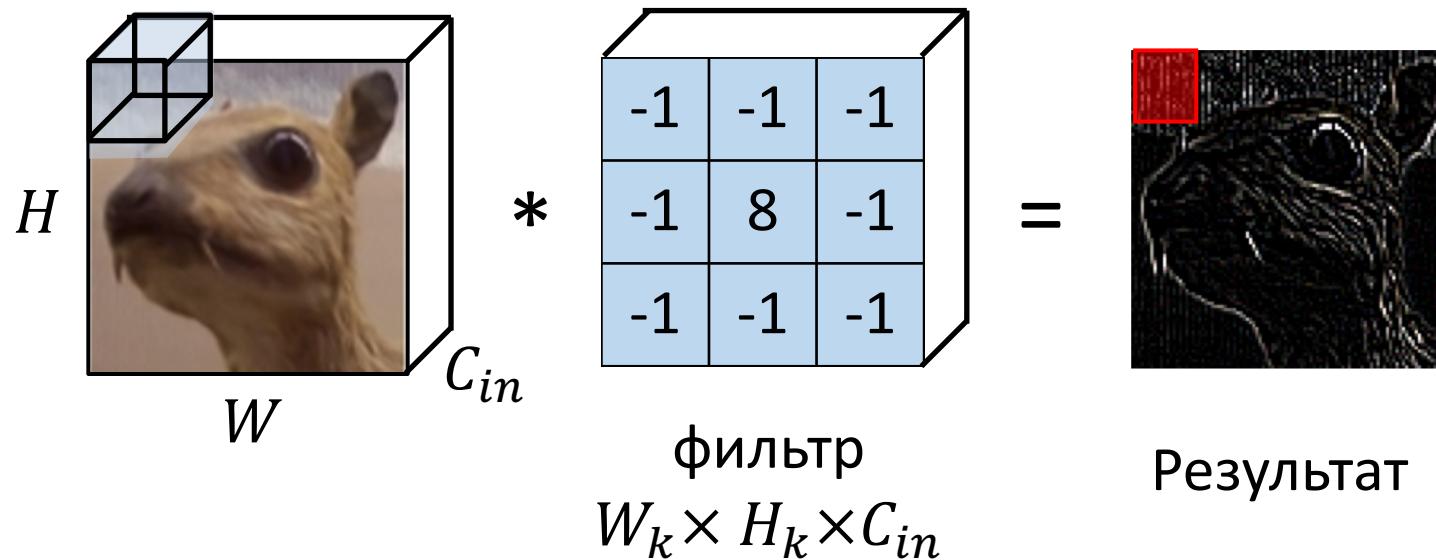
Суммируем градиенты по всем использованием веса w_4 !

В сверточном слое меньше параметров

- Картинка 300x300
- 300x300 выходных нейронов
- Свертка 5x5 – **26** параметров
- В полно-связном слое – **8.1×10^9** параметров

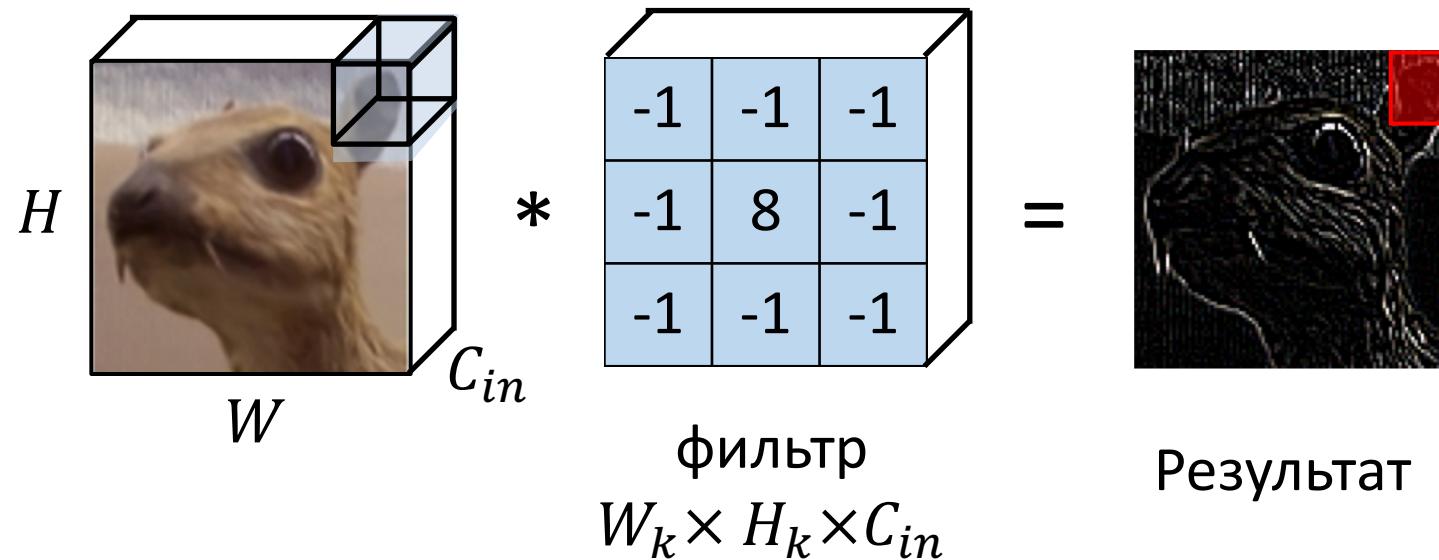
Цветная картинка

- Цветная картинка – это тензор $W \times H \times C_{in}$
- W – ширина,
- H – высота,
- C_{in} – количество каналов (3 RGB канала).



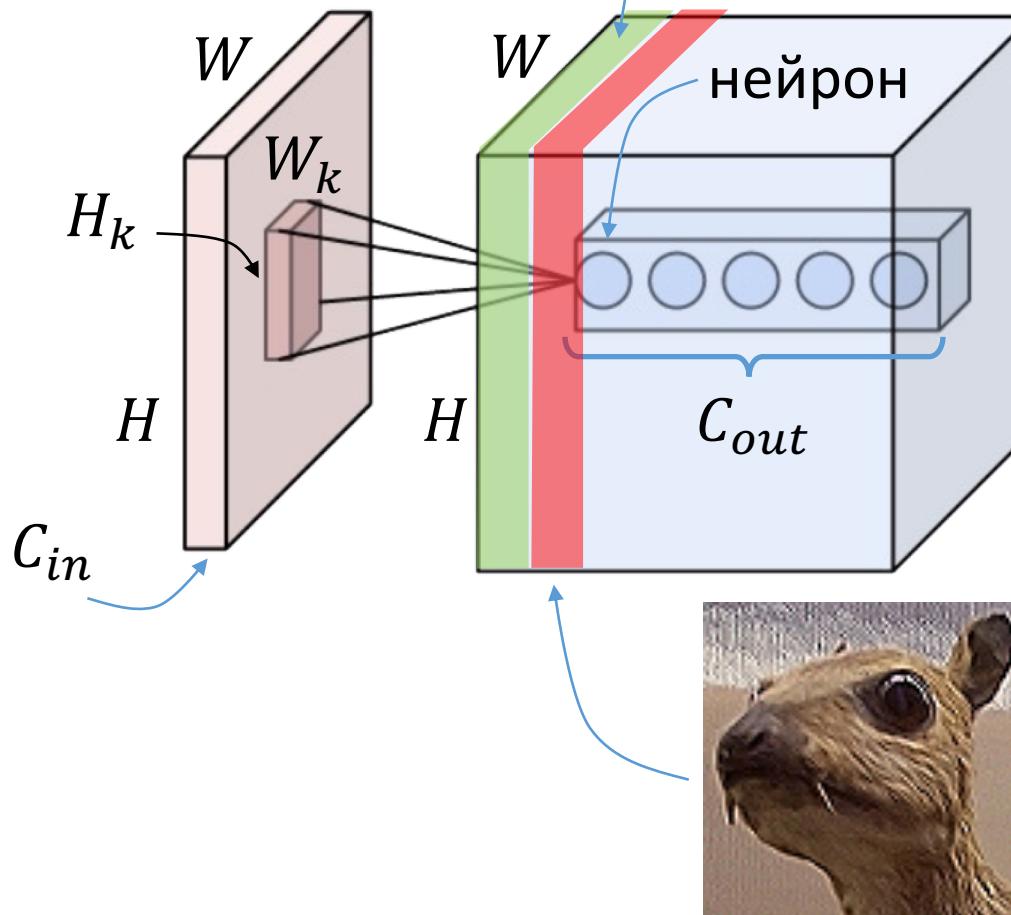
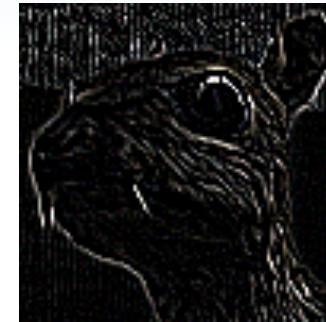
Цветная картинка

- Цветная картинка – это тензор $W \times H \times C_{in}$
- W – ширина,
- H – высота,
- C_{in} – количество каналов (3 RGB канала).



Одного фильтра мало!

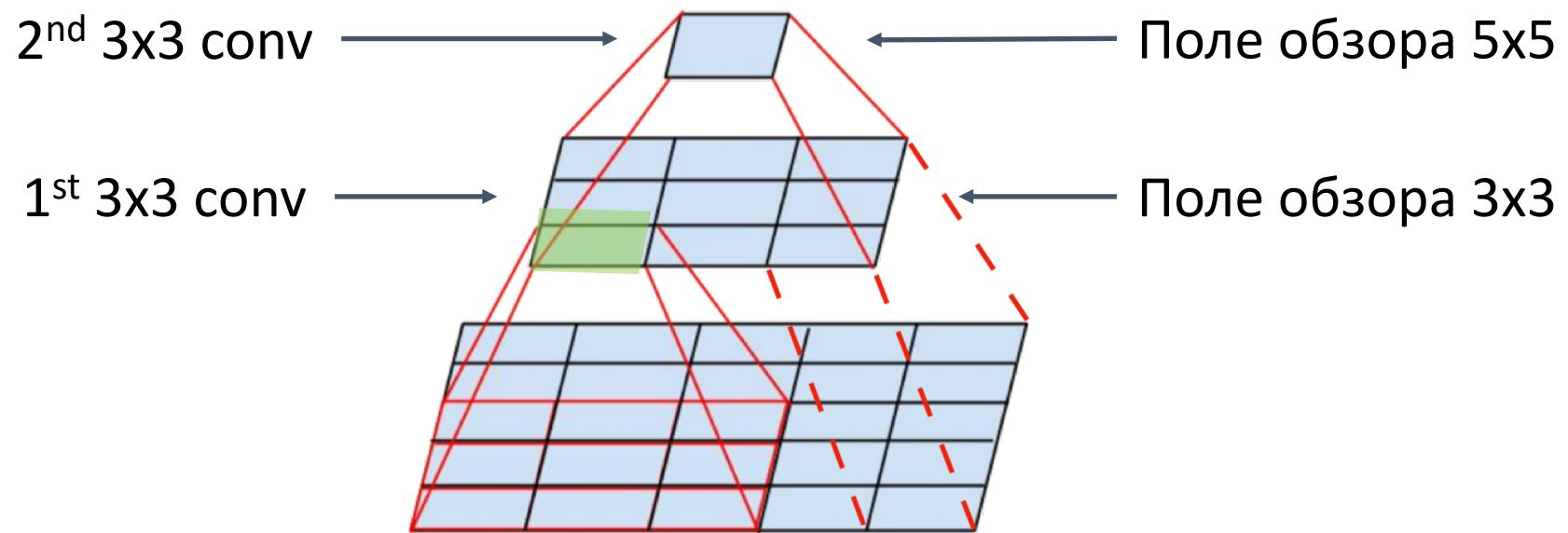
- На входе $W \times H \times C_{in}$
- На выходе $W \times H \times C_{out}$



В одном пикселе появляется **глубина** с разными характеристиками пикселя изображения

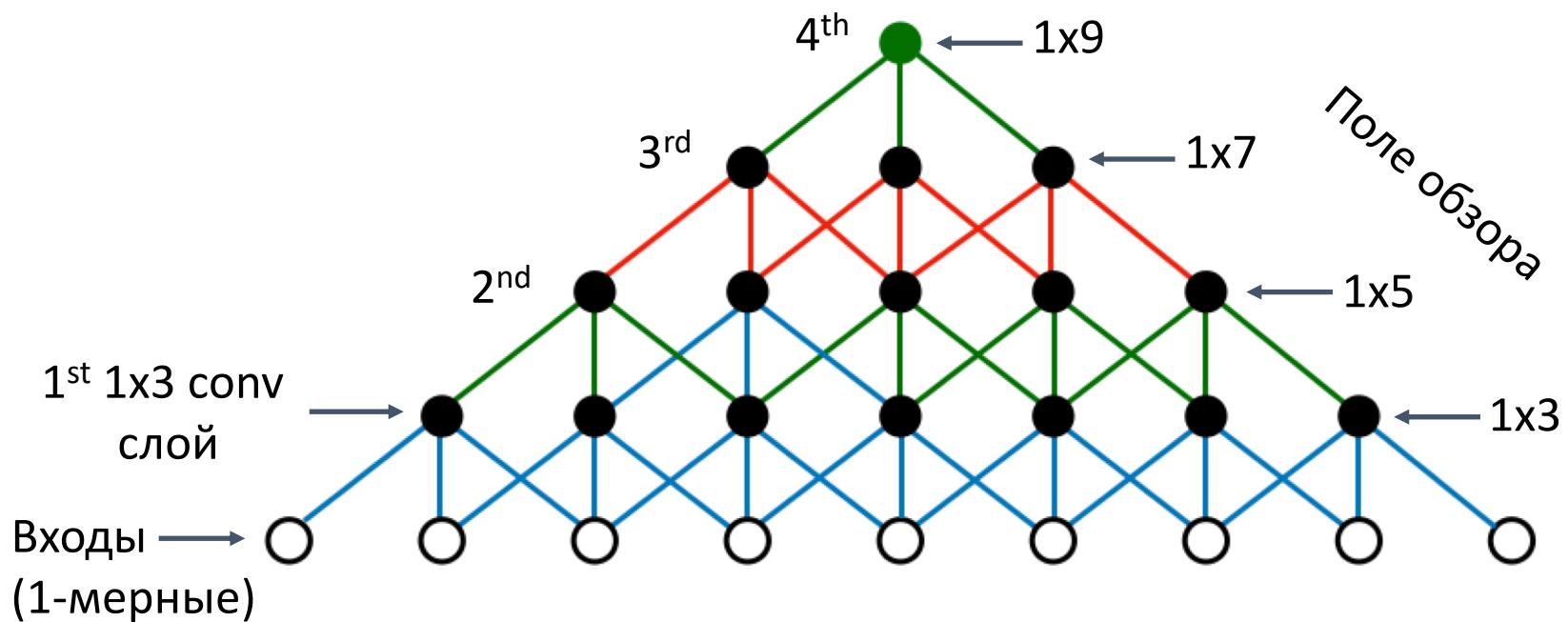
Да и одного сверточного слоя мало!

- Нейроны первого слоя смотрят на кусочек 3×3 .
- А что если кошка больше? 😊
- Нужен второй слой!



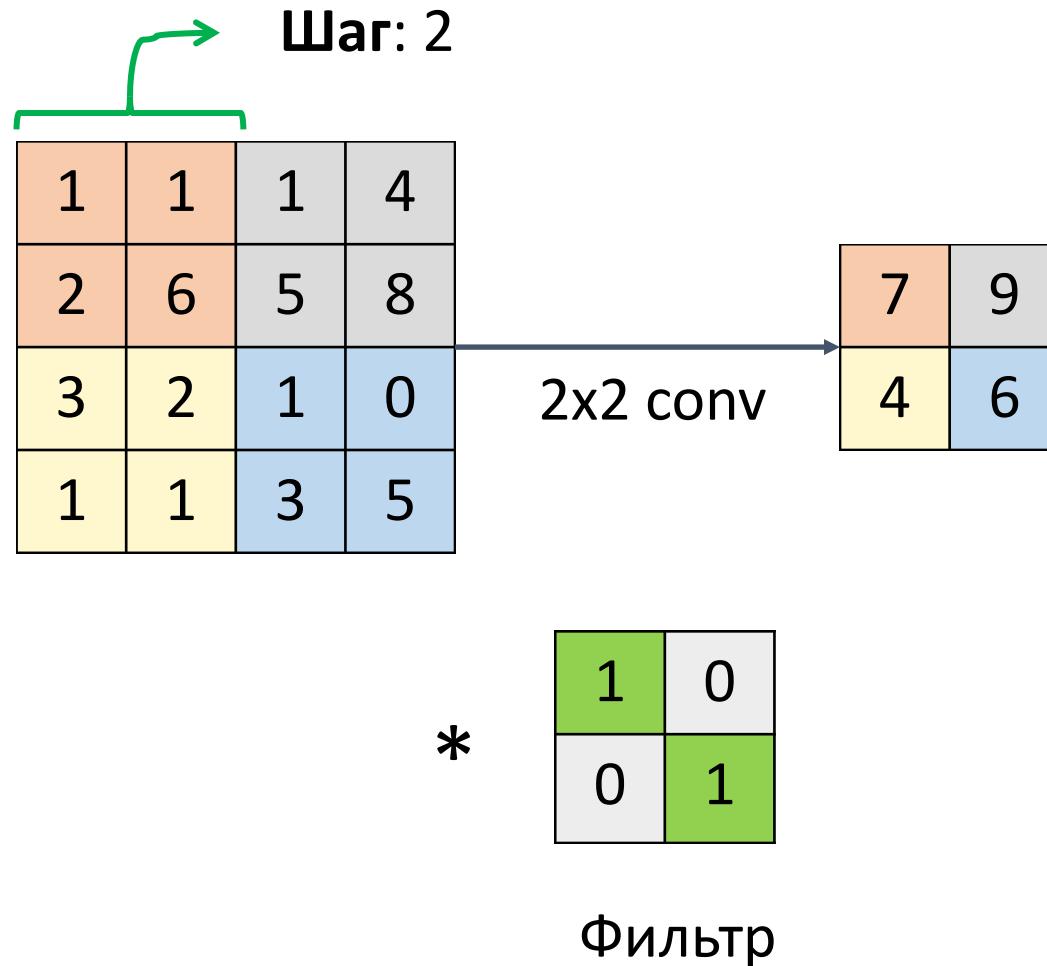
N сверточных слоев

- N слоев с фильтрами 3×3
- На N -ом слое поле обзора $(2N + 1) \times (2N + 1)$.
- Если кот 300×300 , то нам надо 150 слоев! Многовато...



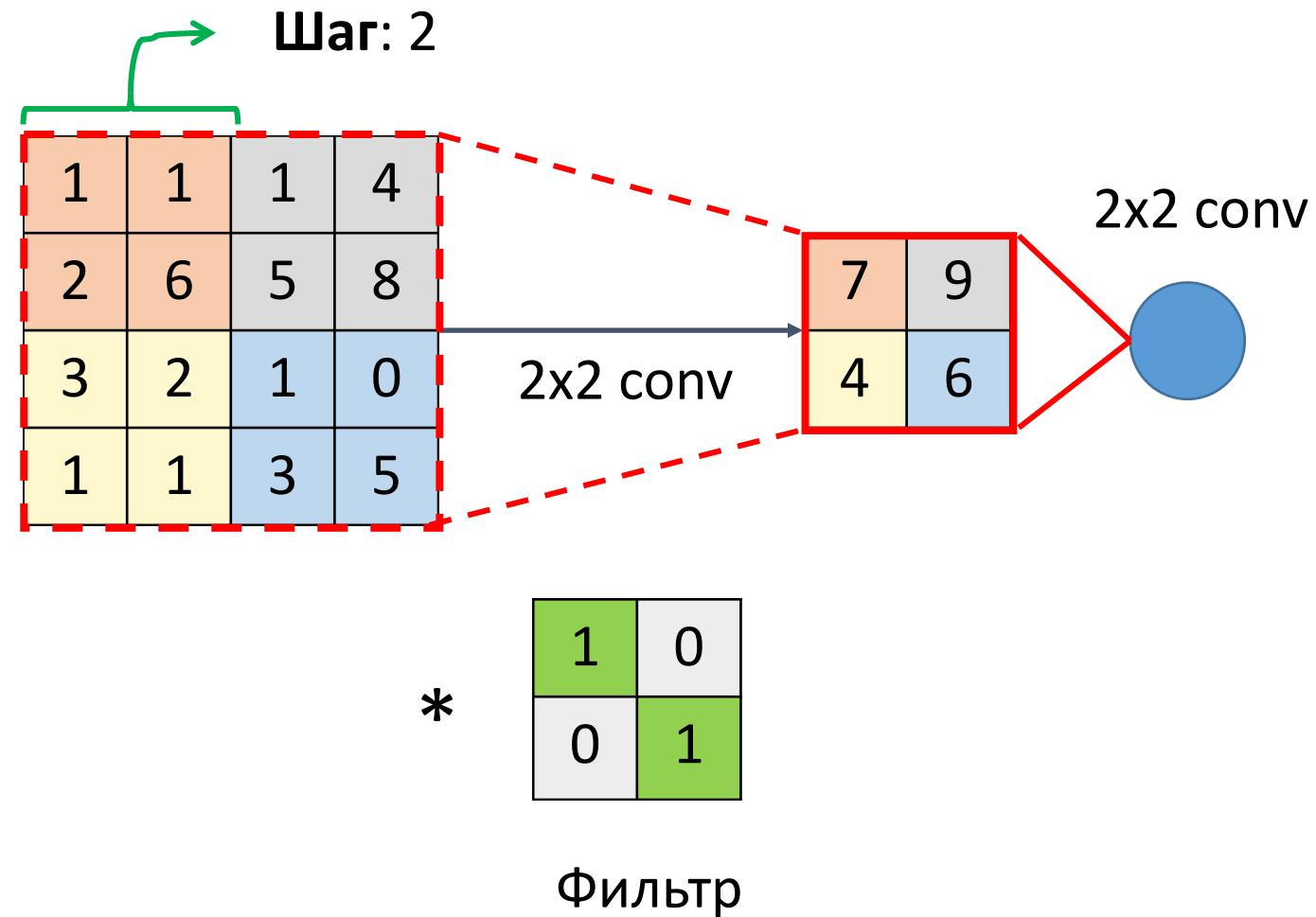
Нужно растить поле обзора быстрее!

Можно увеличить шаг свертки!



Нужно растить поле обзора быстрее!

Можно увеличить шаг свертки!



Давайте вспомним инвариантность к сдвигу!

$$\begin{array}{|c|c|c|c|} \hline 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 1 \\ \hline \end{array} * \begin{array}{|c|c|} \hline 1 & 0 \\ \hline 0 & 1 \\ \hline \end{array} = \begin{array}{|c|c|c|} \hline 0 & 0 & 0 \\ \hline 0 & 1 & 0 \\ \hline 0 & 0 & 2 \\ \hline \end{array}$$

Фильтр

Выход

$$\begin{array}{|c|c|c|c|} \hline 1 & 0 & 0 & 0 \\ \hline 0 & 1 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 \\ \hline \end{array} * \begin{array}{|c|c|} \hline 1 & 0 \\ \hline 0 & 1 \\ \hline \end{array} = \begin{array}{|c|c|c|} \hline 2 & 0 & 0 \\ \hline 0 & 1 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array}$$

Фильтр

Выход

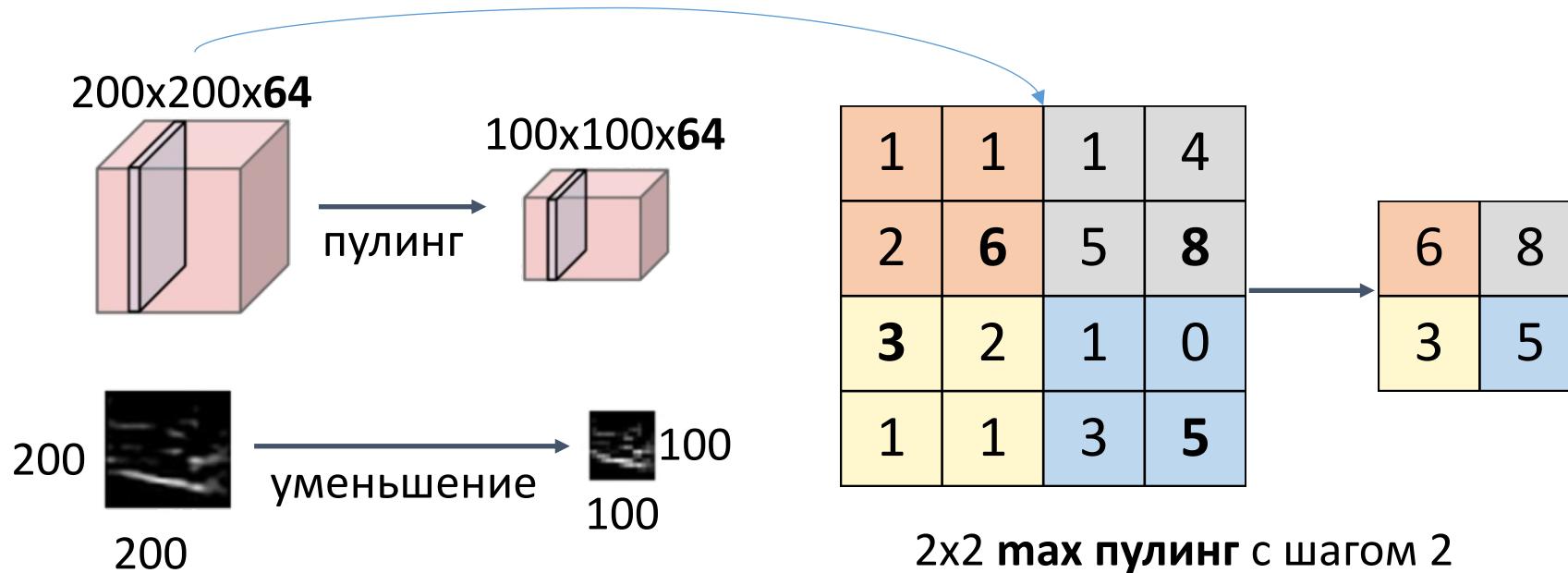
Max = 2

Не изменился

Max = 2

Пулинг слой

- Работает почти как свертка, только вместо фильтра берет **максимум**.



Как считать градиент для пулинга

- Строго говоря: максимум не дифференцируемая функция!

6	8
3	5

Maximum = 8

7	8
3	5

Maximum = 8

- Градиент 0 по немаксимальным входам, потому что при их изменении не меняется выход (максимум).

6	8
3	5

Maximum = 8

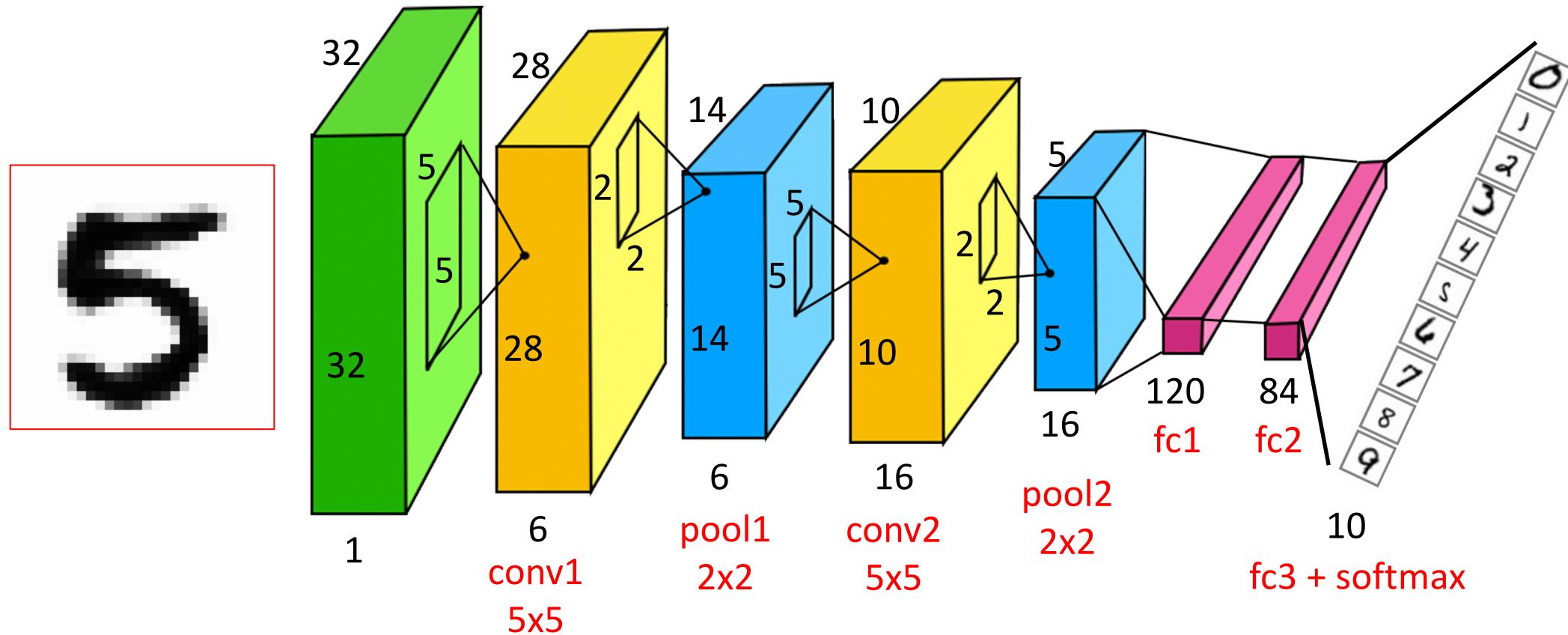
6	9
3	5

Maximum = 9

- Для максимального входа градиент 1.

Соберем это все в сверточную сеть

- LeNet-5 архитектура (1998) для распознавания рукописных цифр (датасет MNIST):



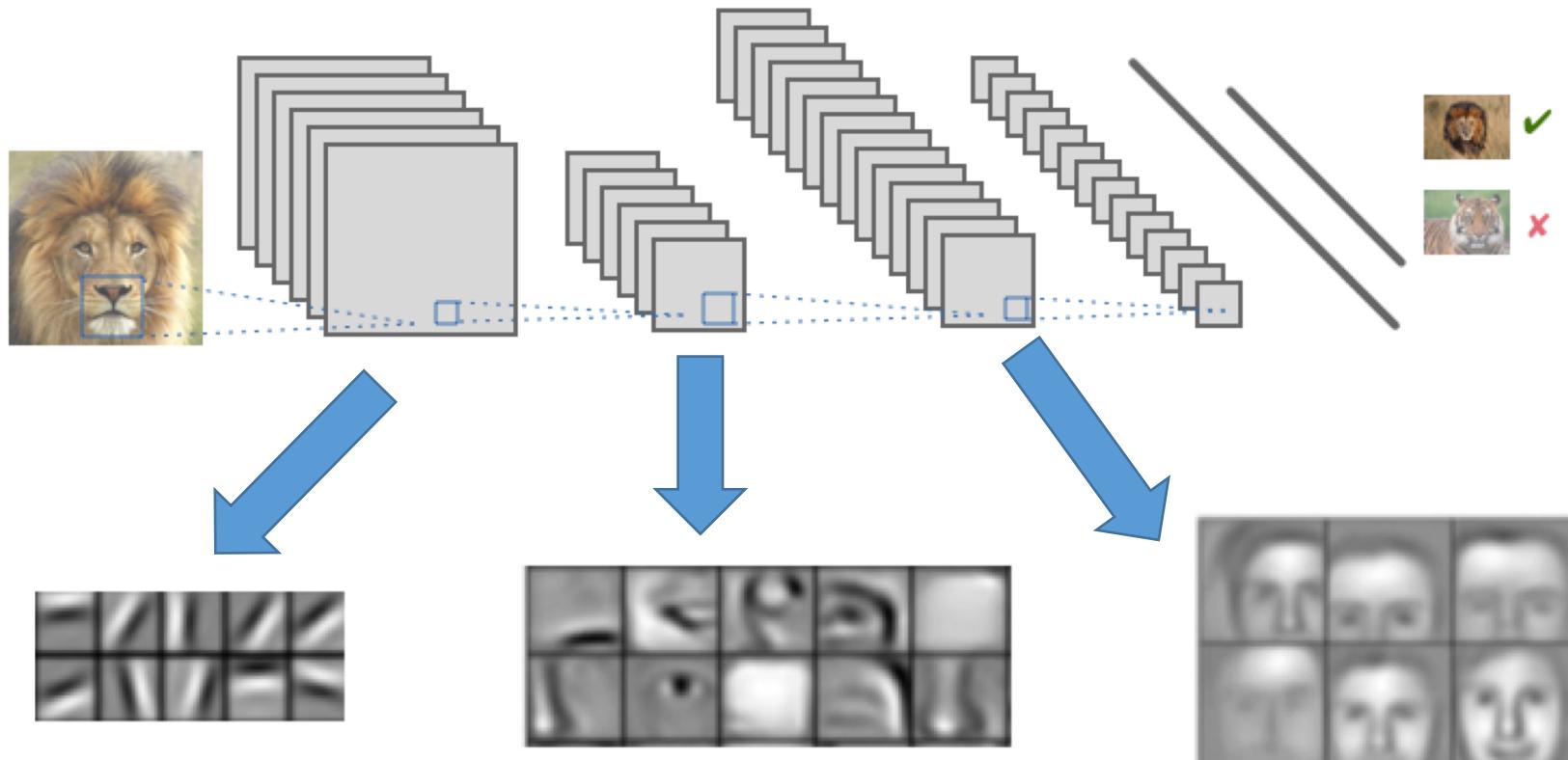
Softmax

- Softmax — это обобщение логистической функции для многомерного случая.
- Преобразует вектор значений в такие, что каждое из них на отрезке [0,1] и в сумме они дают 1:

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

Нейросеть учит иерархические шаблоны

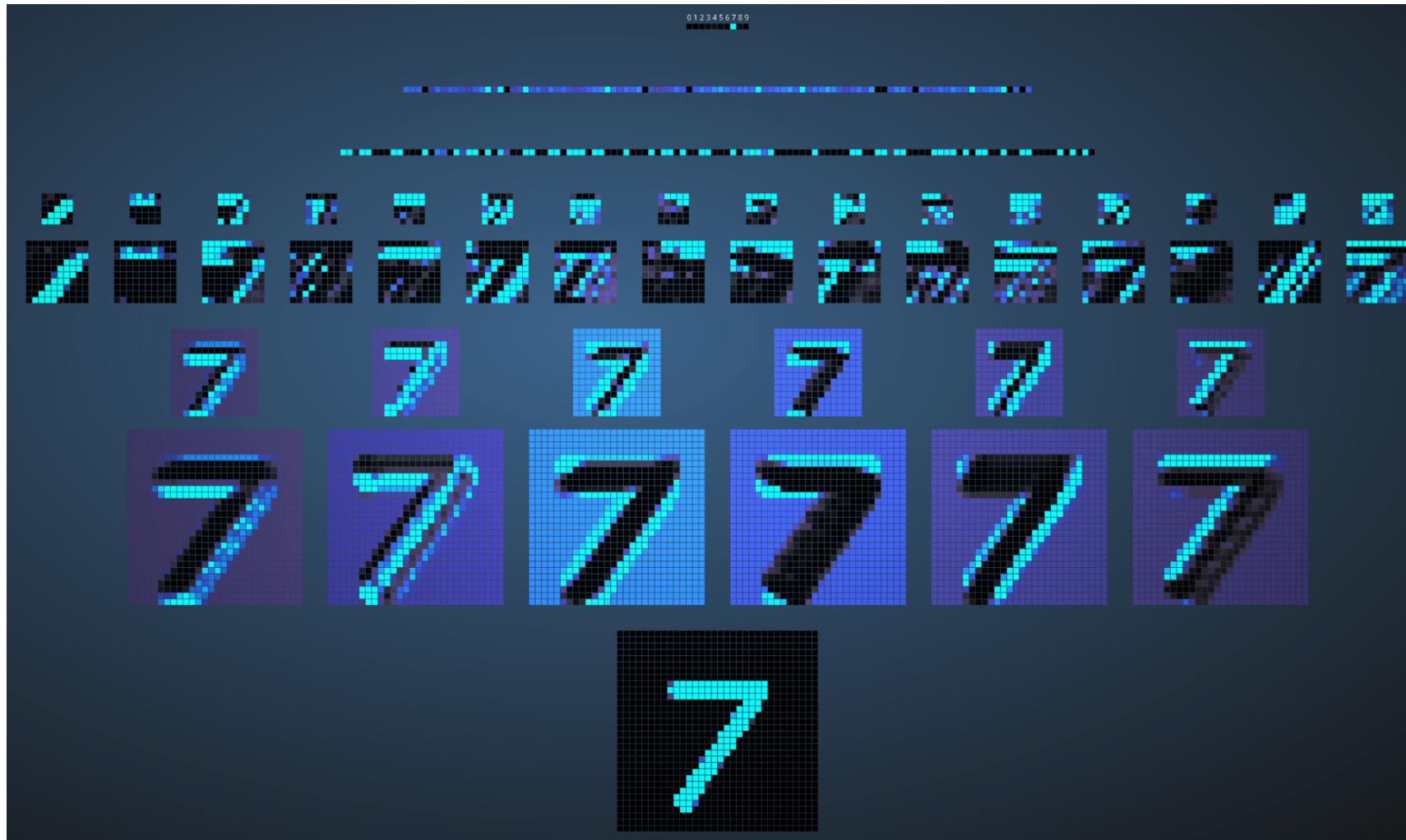
Глубокая нейронная сеть:



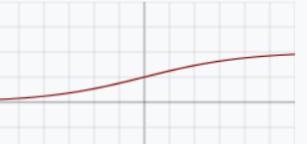
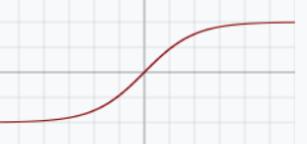
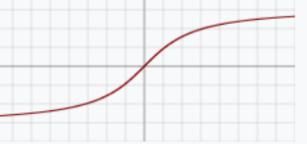
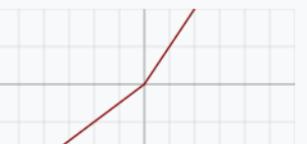
Имея достаточное количество обучающих примеров
машина сама найдет все эти шаблоны в данных.

Демо: визуализация обученной сети для MNIST

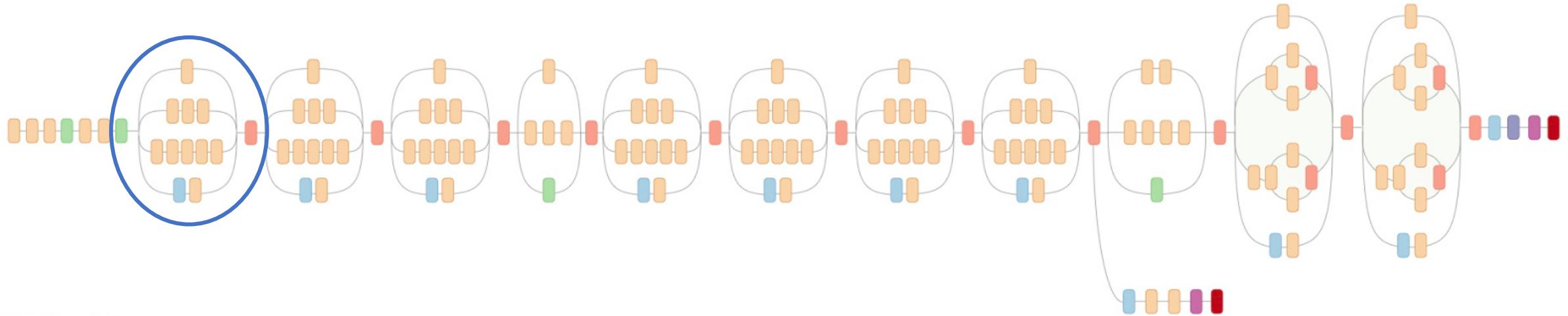
- <http://scs.ryerson.ca/~aharley/vis/conv/flat.html>



Не забываем про функцию активации

Logistic (a.k.a. Soft step)		$f(x) = \frac{1}{1 + e^{-x}}$
TanH		$f(x) = \tanh(x) = \frac{2}{1 + e^{-2x}} - 1$
ArcTan		$f(x) = \tan^{-1}(x)$
Softsign [7][8]		$f(x) = \frac{x}{1 + x }$
Rectified linear unit (ReLU) ^[9]		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$
Leaky rectified linear unit (Leaky ReLU) ^[10]		$f(x) = \begin{cases} 0.01x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$

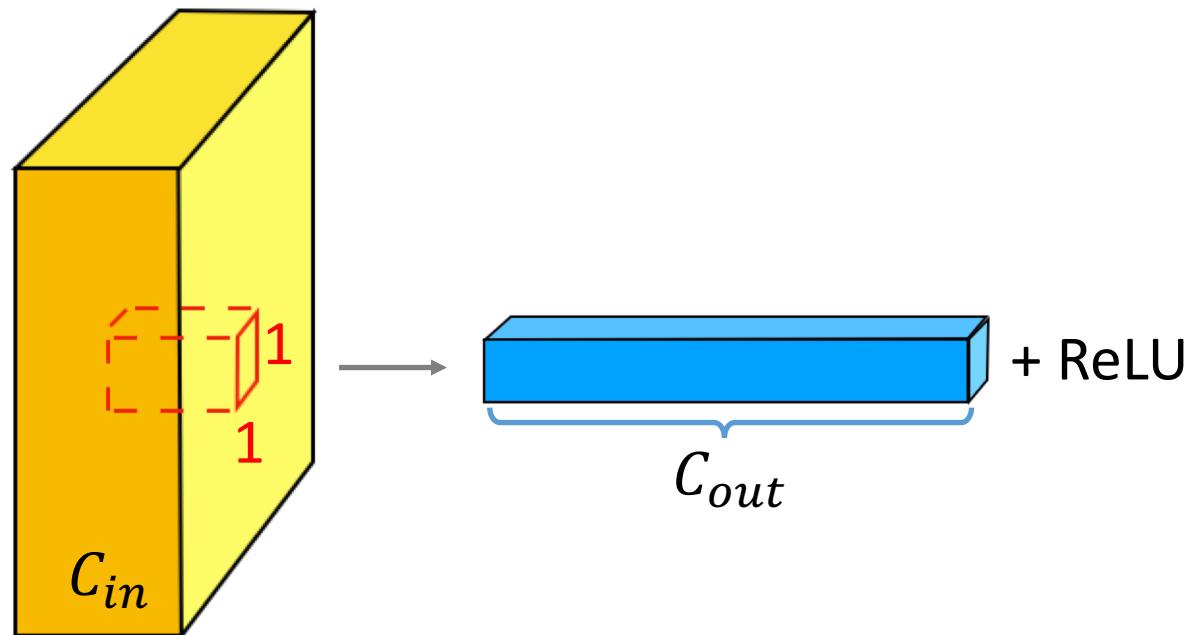
Inception V3 сложнее



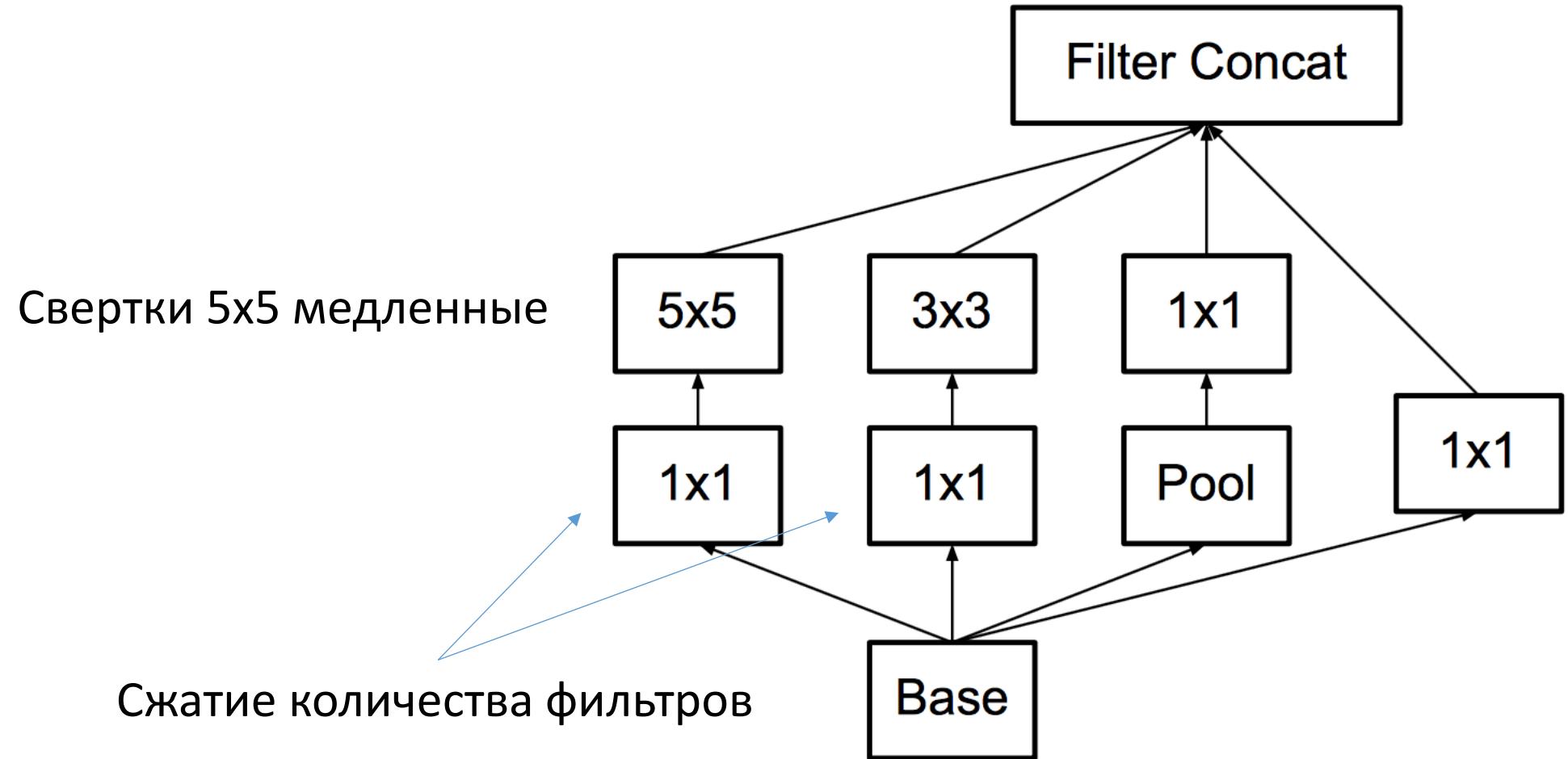
- Convolution
- AvgPool
- MaxPool
- Concat
- Dropout
- Fully connected
- Softmax

Свертка 1×1

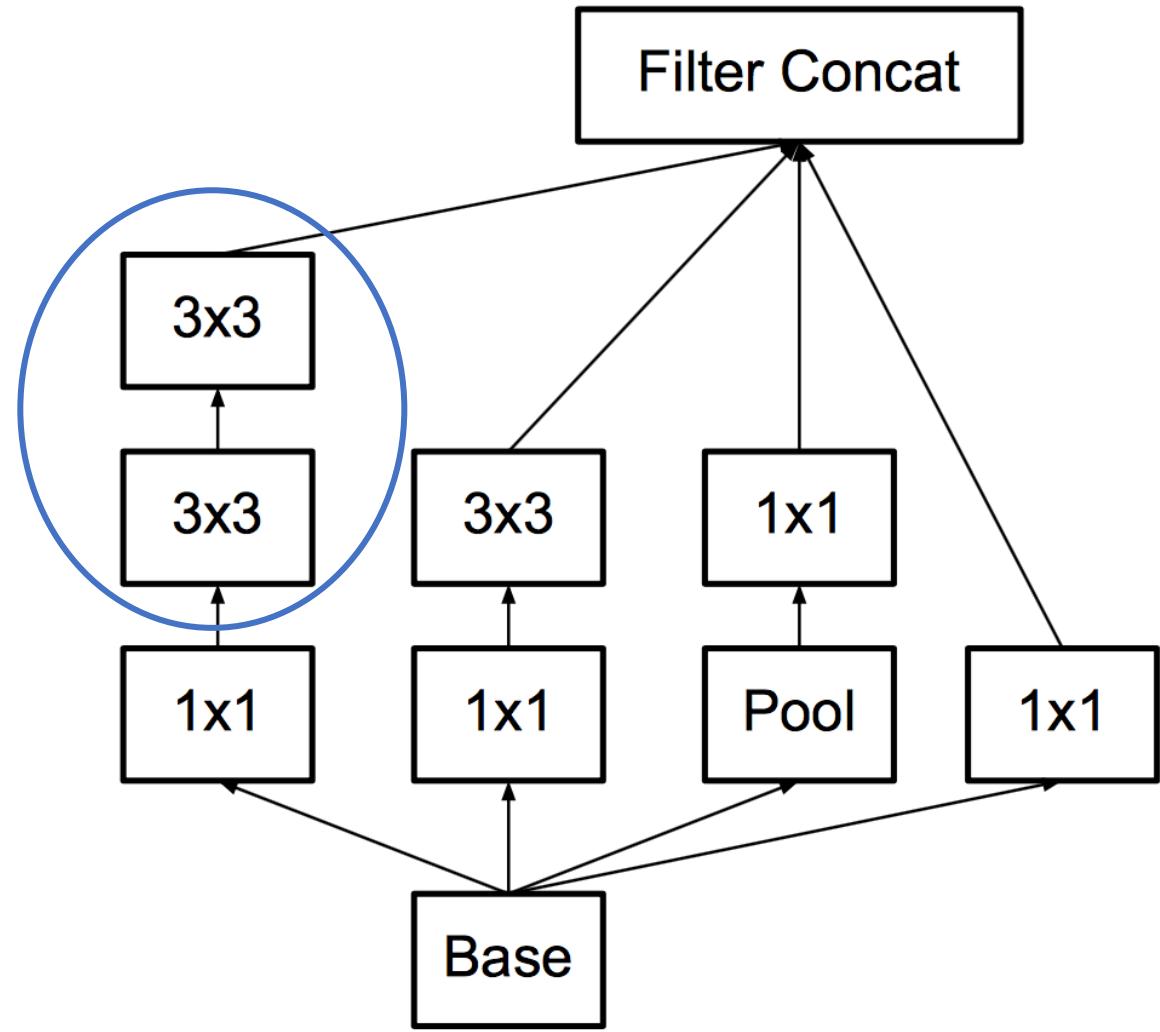
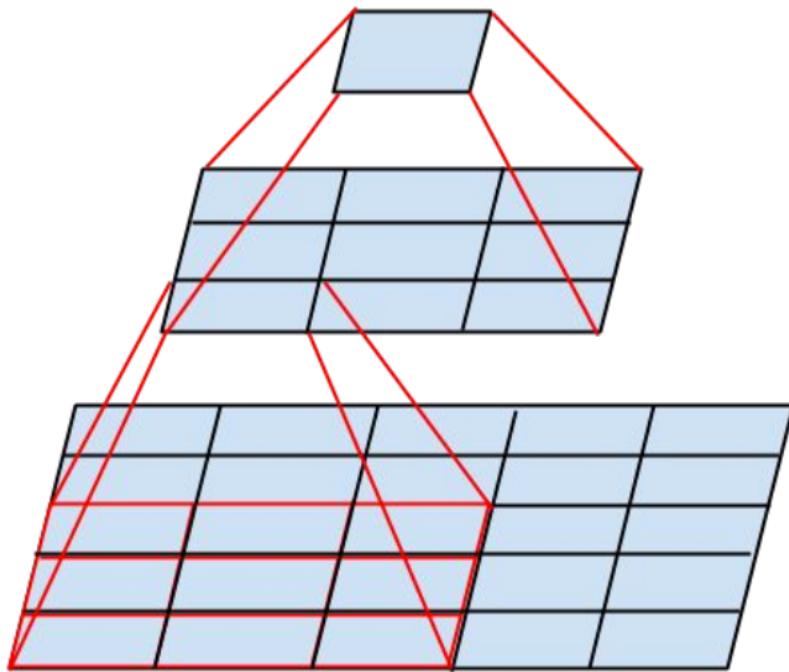
- Учитывает взаимосвязи разных входных каналов в одном пикселе
- Можно использовать для сжатия представления (фильтров может быть избыточное число) как в PCA



Inception block

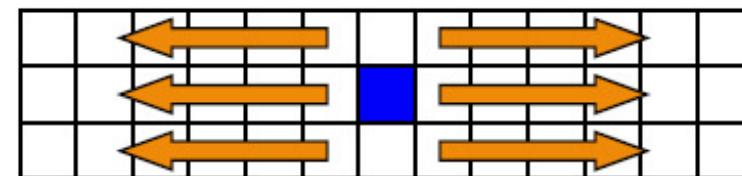


Свертки 5x5 дорогие, факторизуем их

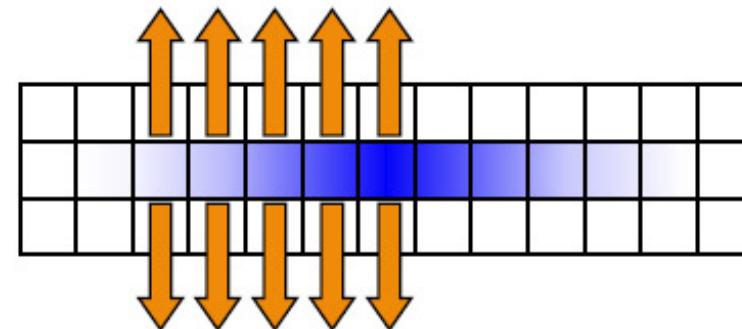


Размытие по Гауссу это сепарабельный фильтр

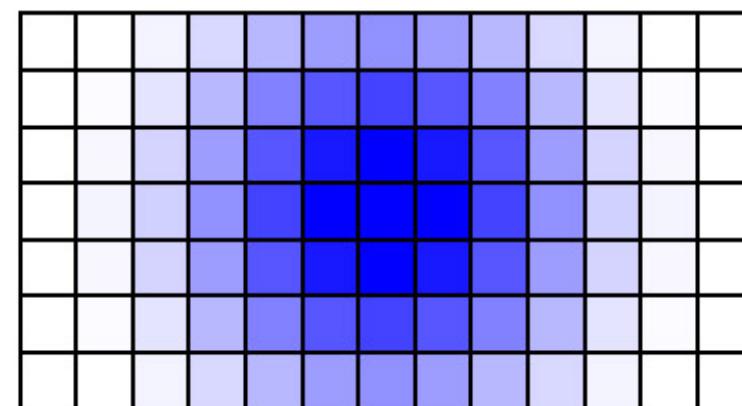
- Размытие можно сделать сначала по горизонтали
- Затем размыть результат по вертикали
- Это эквивалентно двумерной свертке, но сильно быстрее
- Вдруг сработает для других фильтров?



Blur the source
horizontally

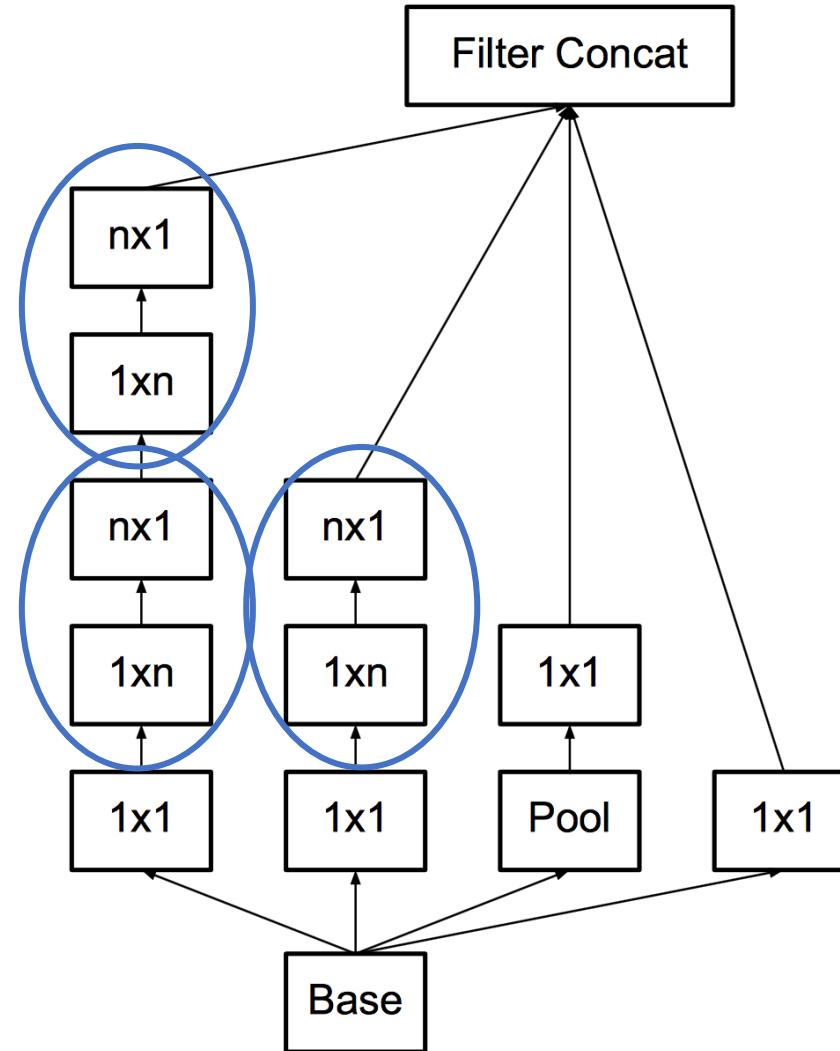


Blur the blur
vertically



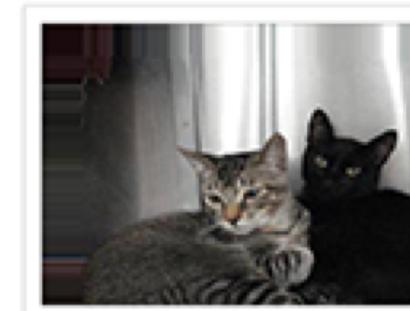
Result

Продолжим факторизовать свертки



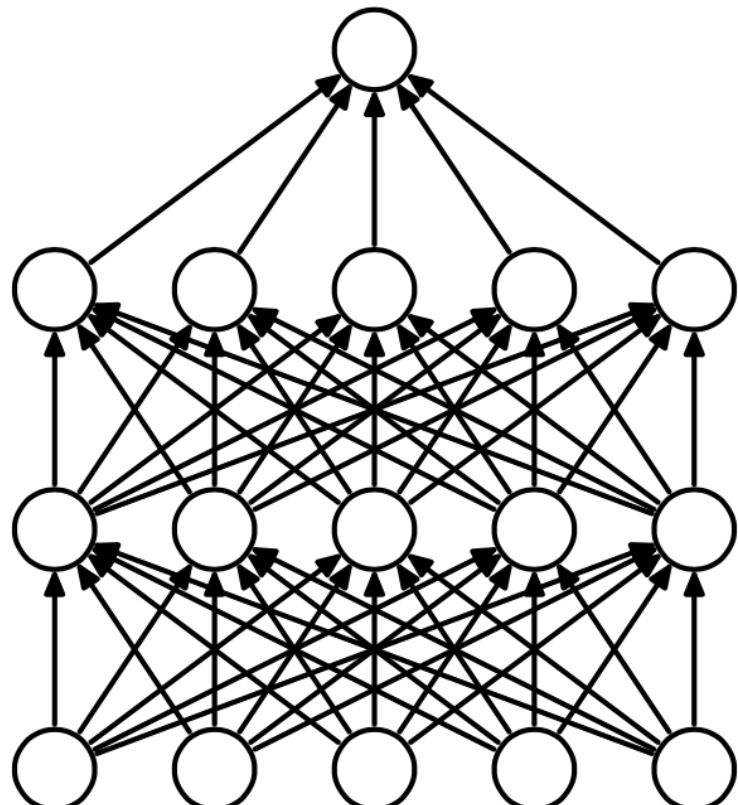
Data augmentation

- В нейросетях много миллионов параметров, а выборки ограниченные
- Хочется получить больше данных из имеющихся – добавим повороты, растяжения, отражения, и т.д. как новые картинки

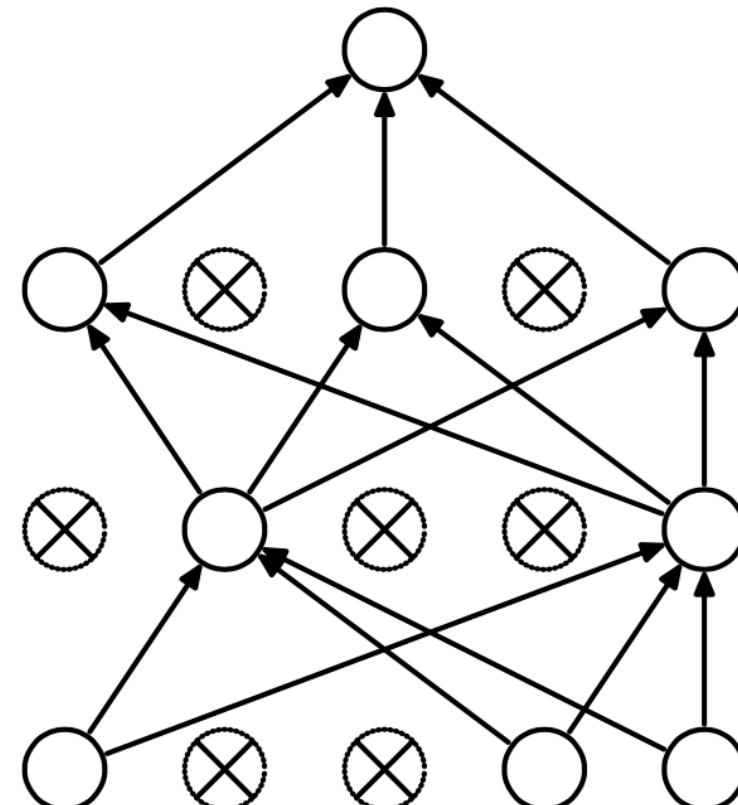


Dropout

- С вероятностью p выкидываем нейрон из сети



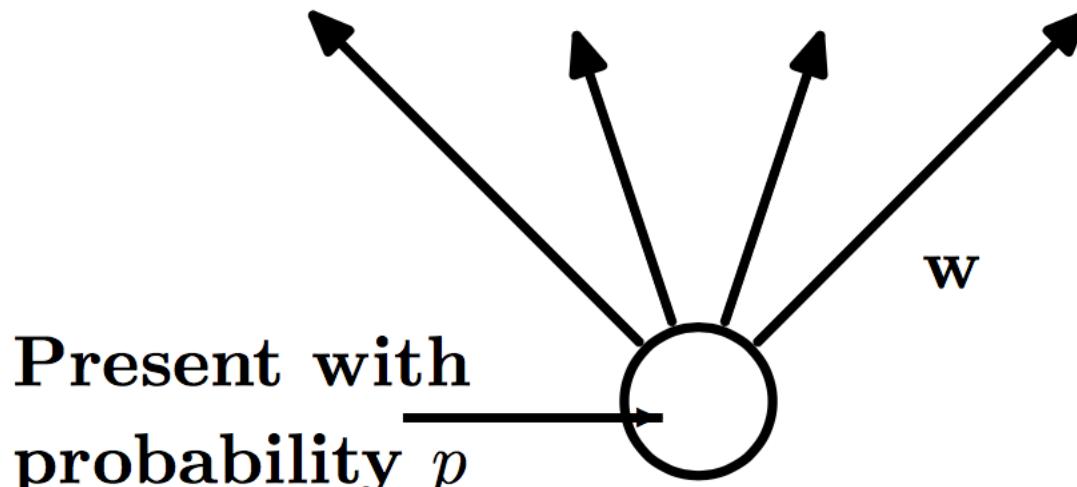
(a) Standard Neural Net



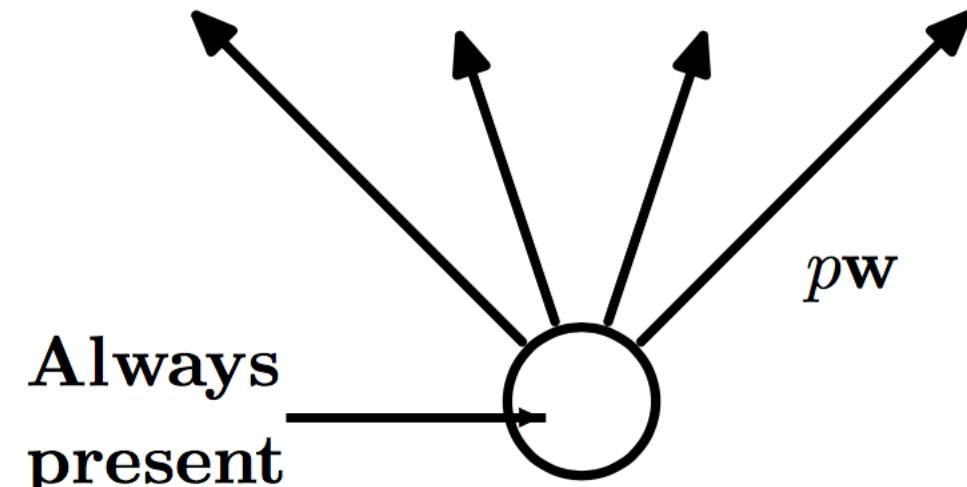
(b) After applying dropout.

Dropout

- Можно рассматривать как ансамбль из большого числа более простых сетей (байесовские методы объясняют лучше)



При обучении



При тестировании

Batch normalization

- Всем известно, что линейные модели лучше сходятся, если признаки нормированы
- В нейросетях у нас много линейных преобразований + активаций
- Выходы нейронов будут использоваться на следующем слое
- Было бы здорово научиться как-то нормировать выходы каждого нейрона

Batch normalization

- Нормализуем выход каждого нейрона:

$$h_i = \frac{h_i - \mu_i}{\sqrt{\sigma_i^2}}$$

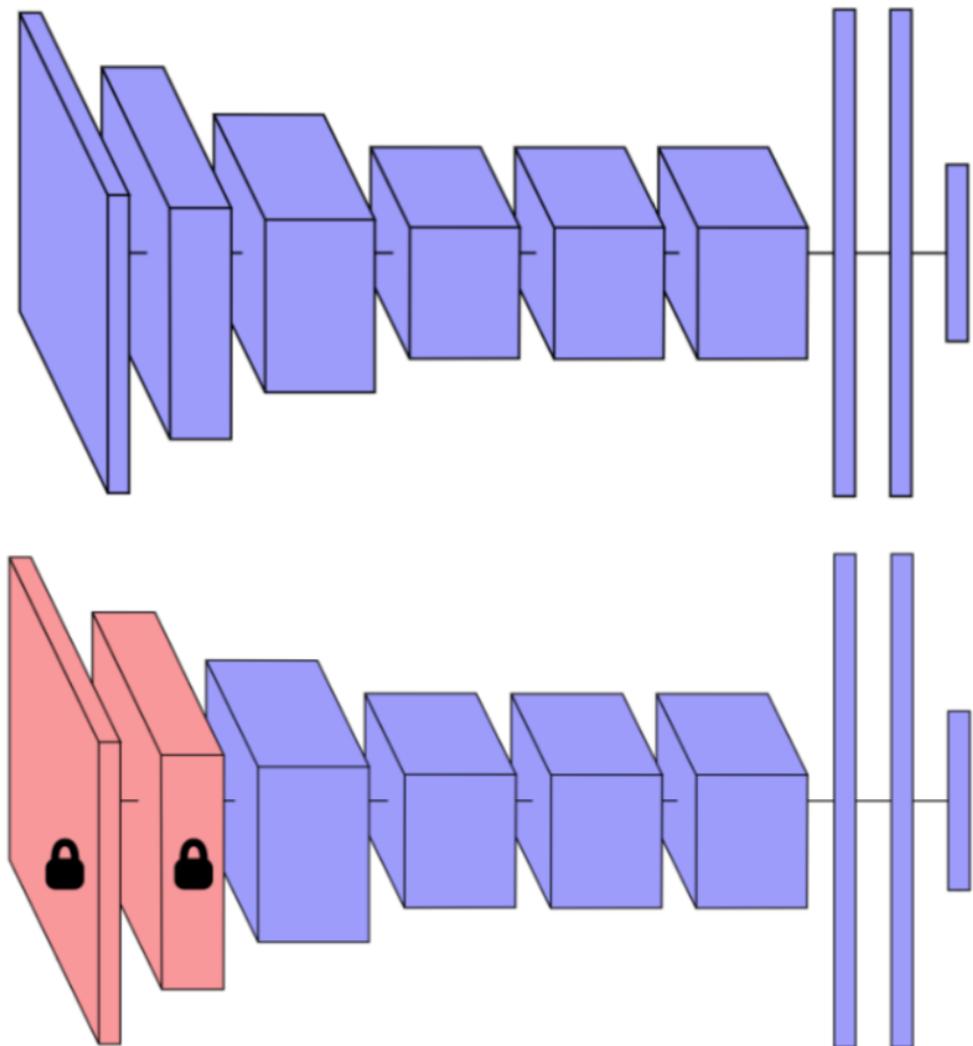
- Откуда брать μ_i и σ_i^2 ?
Экспоненциальное сглаживание оценок по batch!

$$\mu_i := \alpha \cdot \text{mean}_{batch} + (1 - \alpha) \cdot \mu_i$$

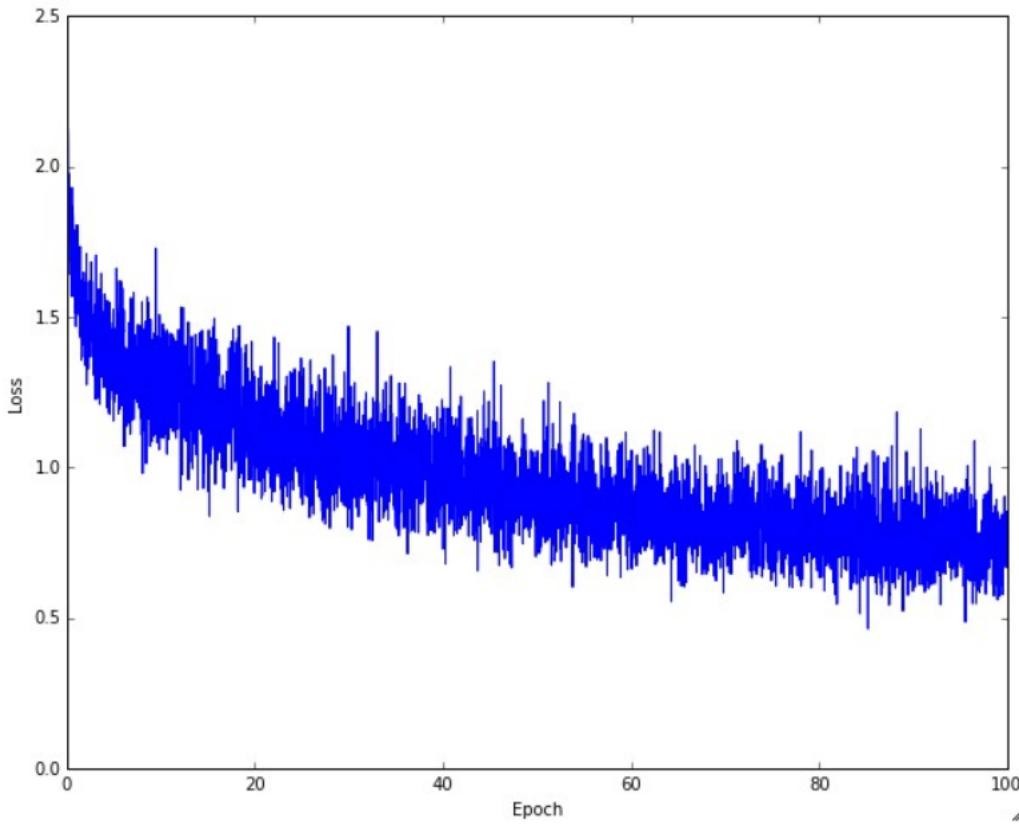
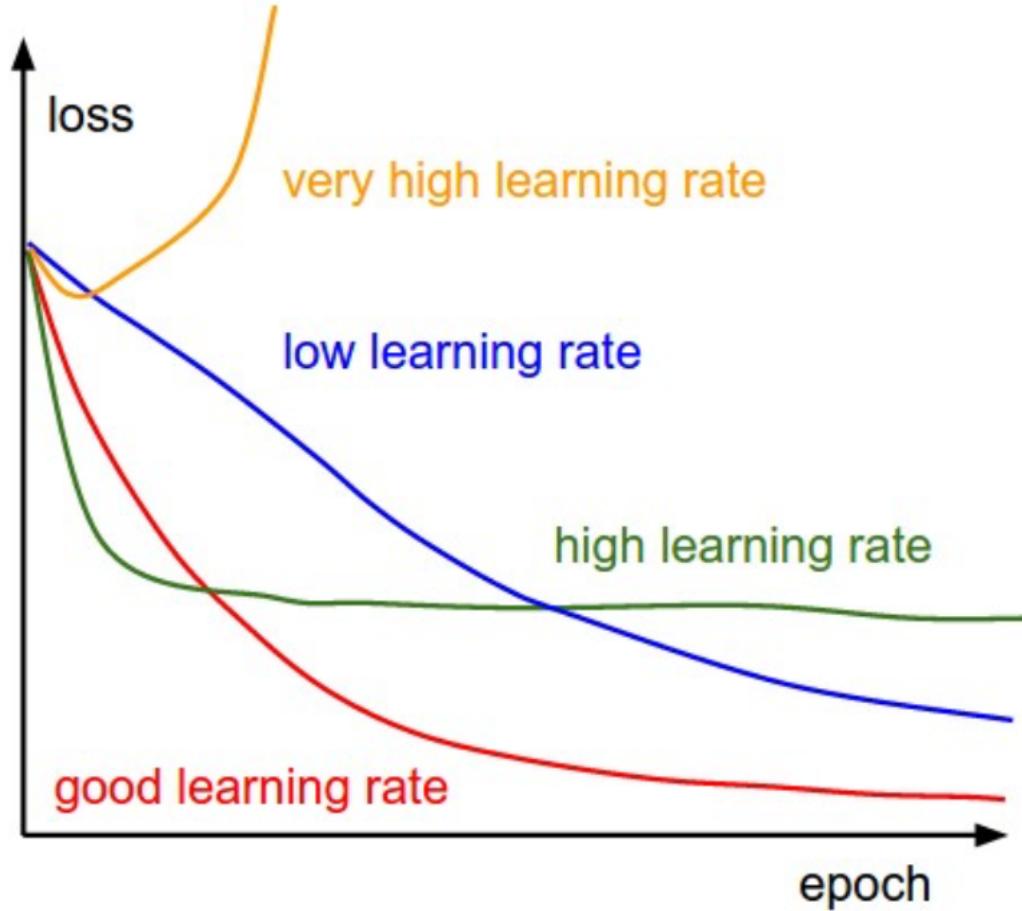
$$\sigma_i^2 := \alpha \cdot \text{variance}_{batch} + (1 - \alpha) \cdot \sigma_i^2$$

Transfer learning и fine-tuning

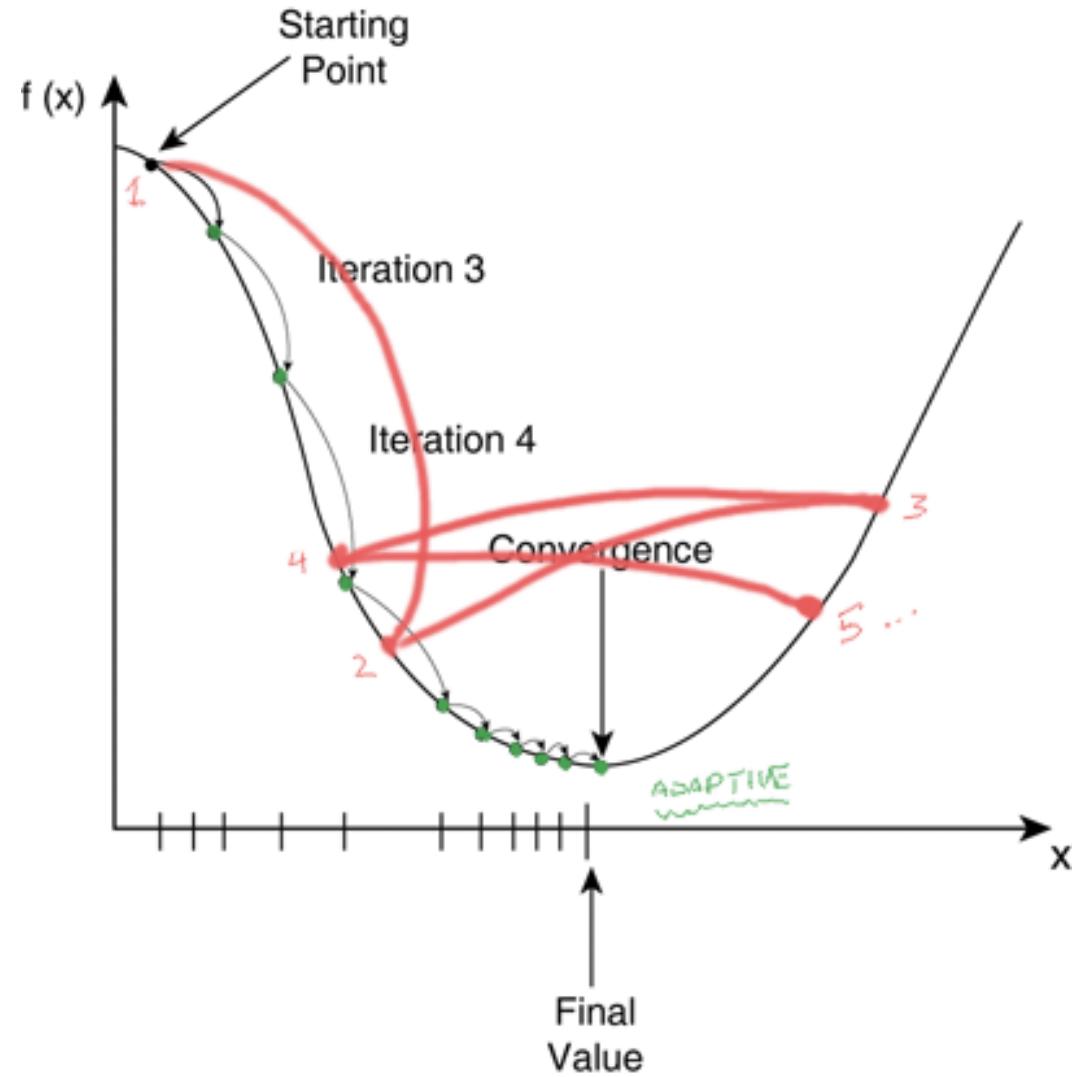
- Берем сеть, обученную на другую задачу
- Фильтры могут быть полезны и для нашей задачи
- Настраиваем только более глубокие слои
- Затем настраиваем с маленьким шагом всю сеть



Learning rate



Learning rate decay



Разновидности SGD

- Обычный SGD

$$w^{k+1} = w^k - \alpha \nabla f(w^k)$$

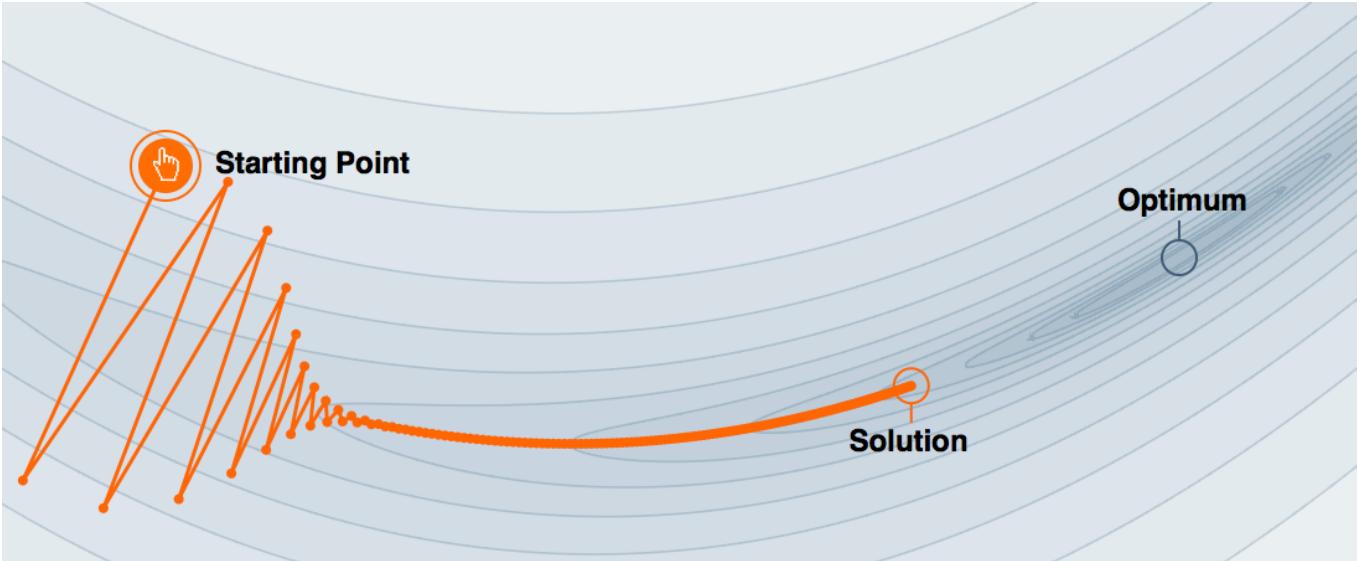
- SGD с моментами

Почти экспоненциальное
сглаживание градиента

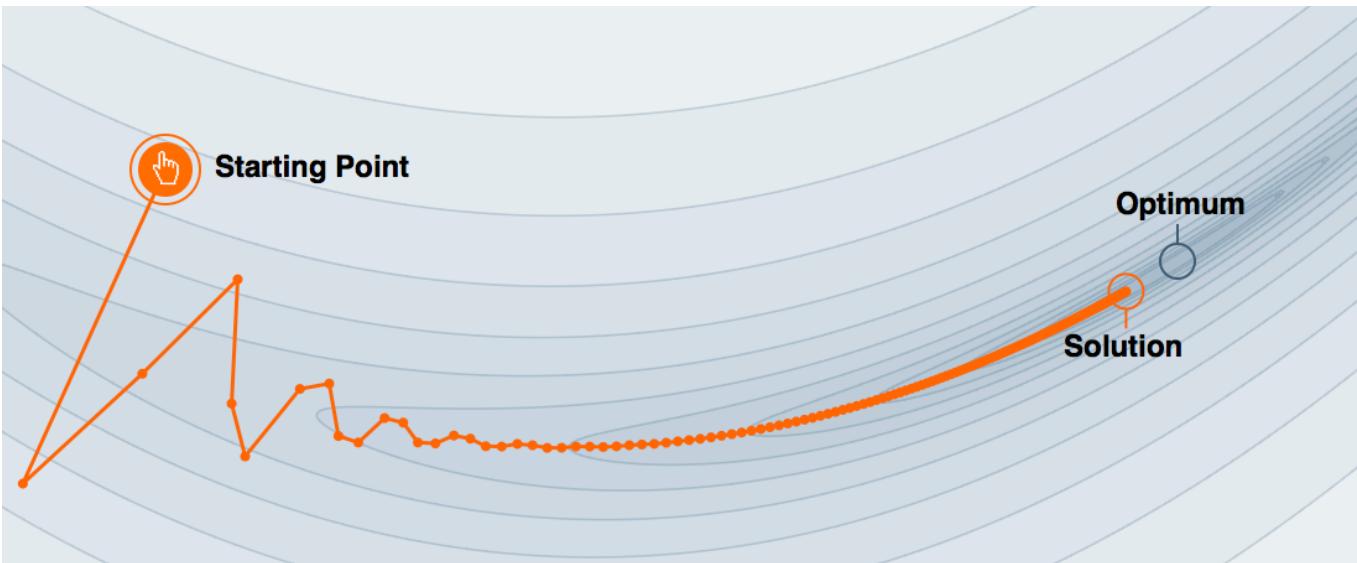

$$\begin{aligned} z^{k+1} &= \beta z^k + \nabla f(w^k) \\ w^{k+1} &= w^k - \alpha z^{k+1} \end{aligned}$$

SGD с моментами

Обычный SGD



SGD с моментами



RMSprop

Экспоненциальное
сглаживание квадрата
градиента

$$E[g^2]_t = 0.9E[g^2]_{t-1} + 0.1g_t^2.$$

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

Ссылки

- <http://cs231n.github.io/convolutional-networks/>
- <https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/>