MML minor #4

Сверточные нейронные сети

Какие задачи хотим решать

Уточка

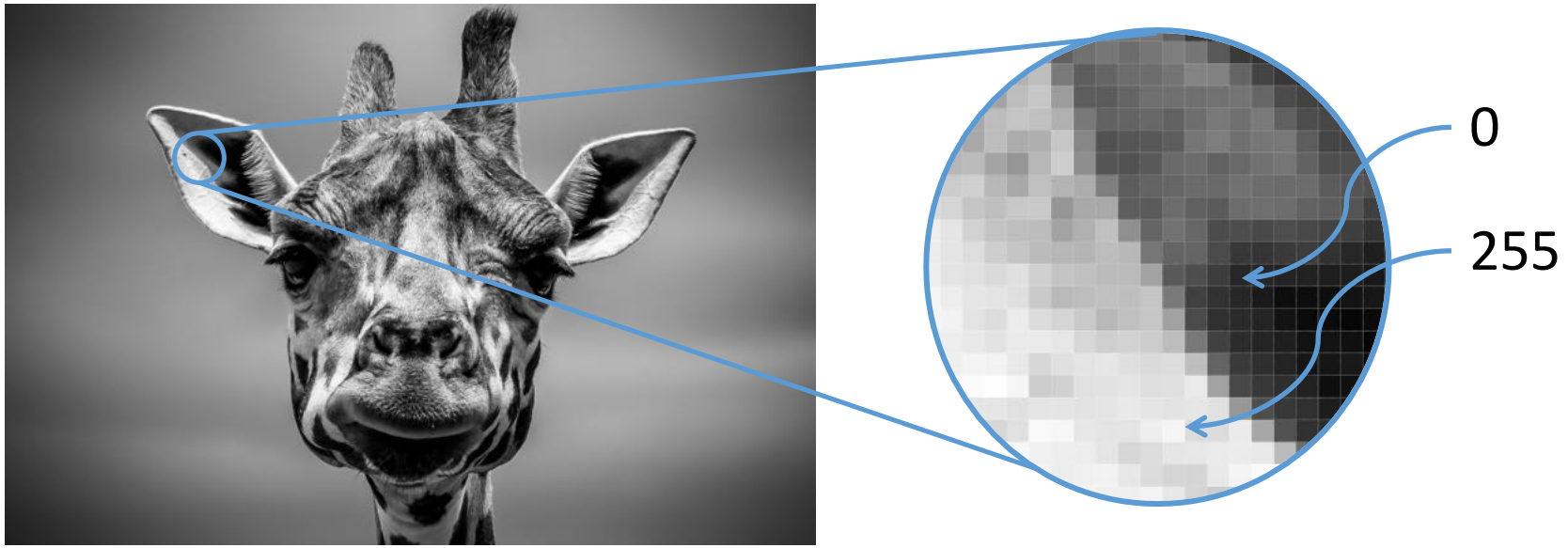
Какие задачи хотим решать

Уточка

52

Как компьютер видит картинку

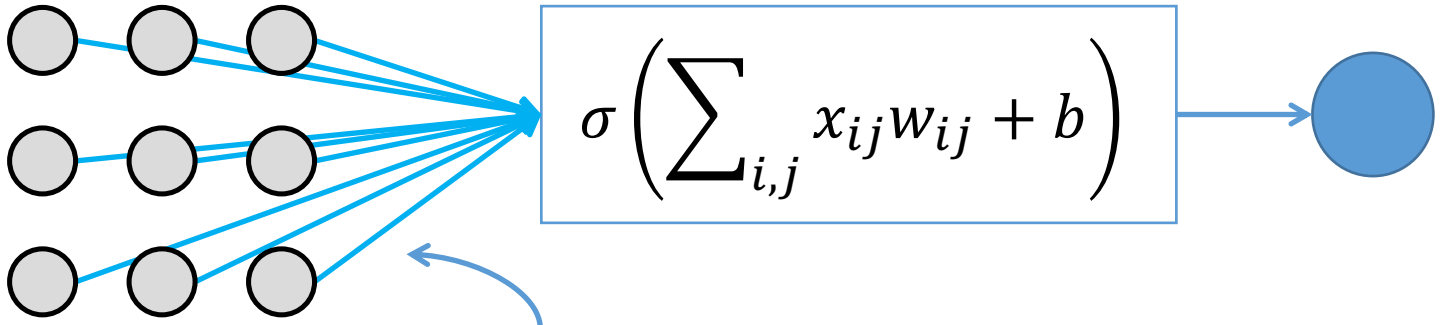
- Матрица яркостей пикселей (например размера 300 x 300)
- Яркости пикселей от 0 до 255



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

Как применить к картинке нейросеть?

Нейрон поверх всех пикселей?

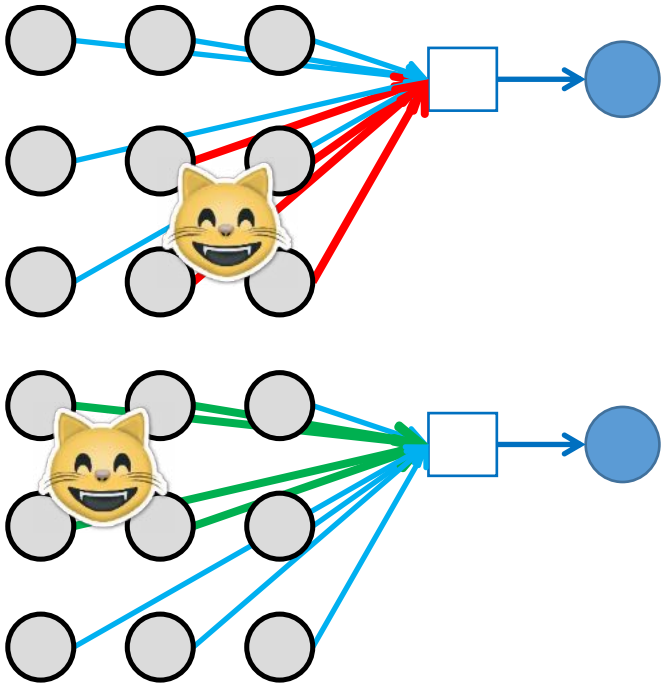


$$x_{norm} = \frac{x}{255} - 0.5$$

Не, так не работает!

Почему нет?

- Хотим выучить детектор кота:

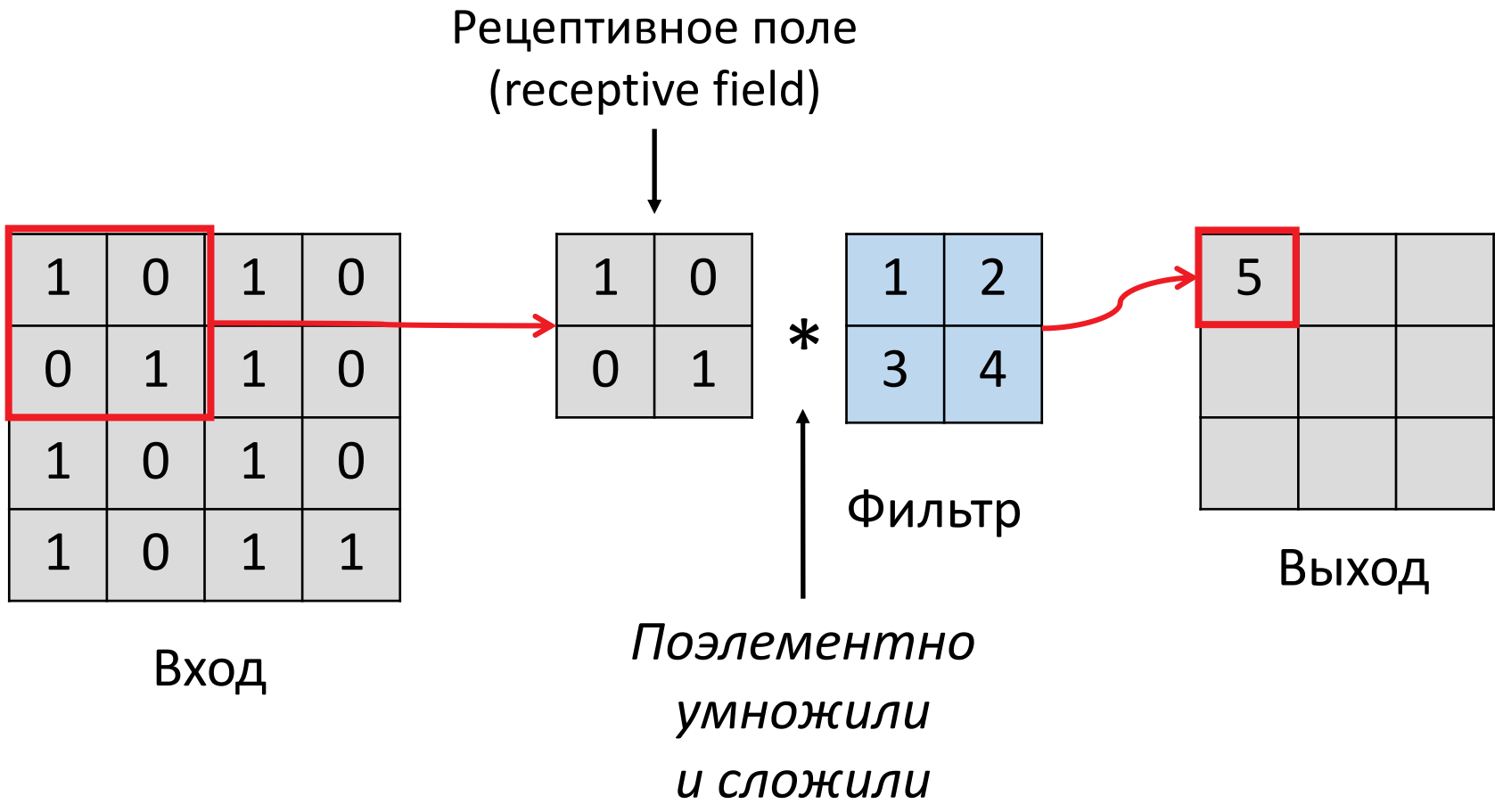


На этой картинке красные веса w_{ij} немного изменятся при градиентном спуске

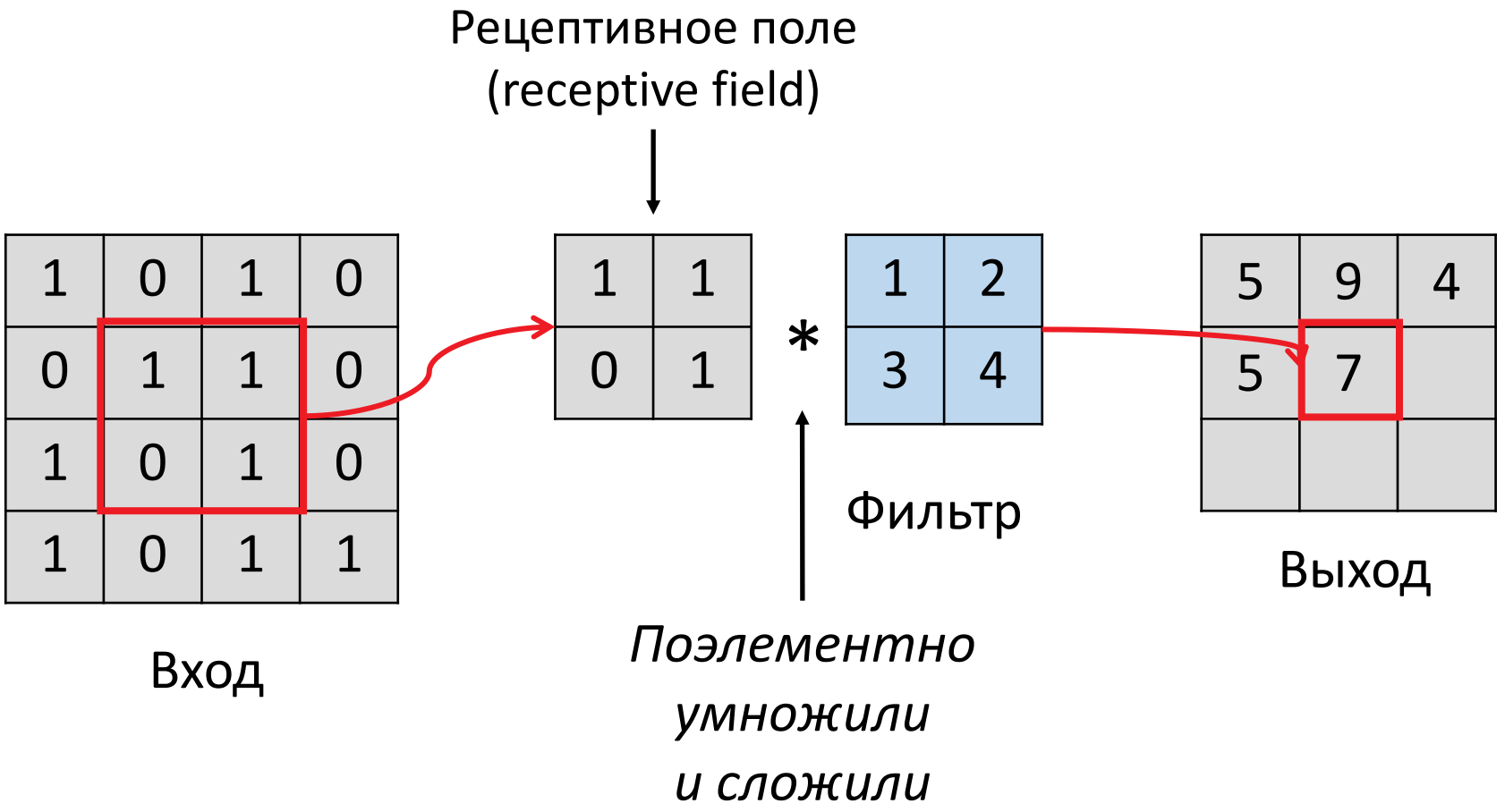
А на этой зеленые веса w_{ij} изменятся...

- 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

Фильтр

-1	-1	-1
-1	8	-1
-1	-1	-1

* =

Поиск краев

Входная картинка

На однотонной заливке дает ноль (черный)

Свертки применяют к картинкам давно

Фильтр

Входная картинка

*

-1	-1	-1
-1	8	-1
-1	-1	-1

=

Поиск краев

*

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

=

Повышение резкости

Добавляет к картинке края

Свертки применяют к картинкам давно

Входная картинка

Фильтр

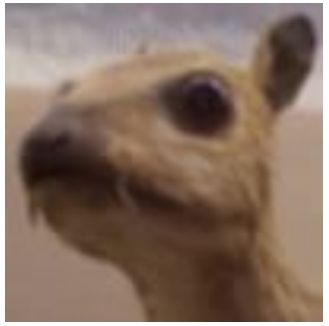
$$* \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} =$$

Поиск краев

$$* \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix} =$$

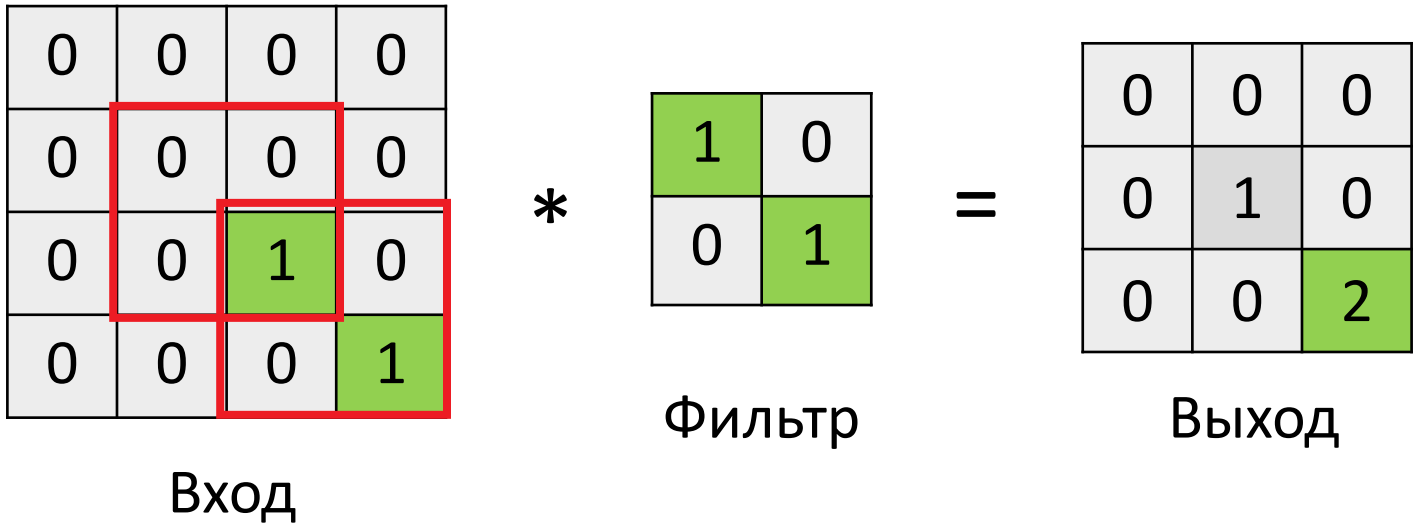
Повышение резкости

$$* \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} =$$



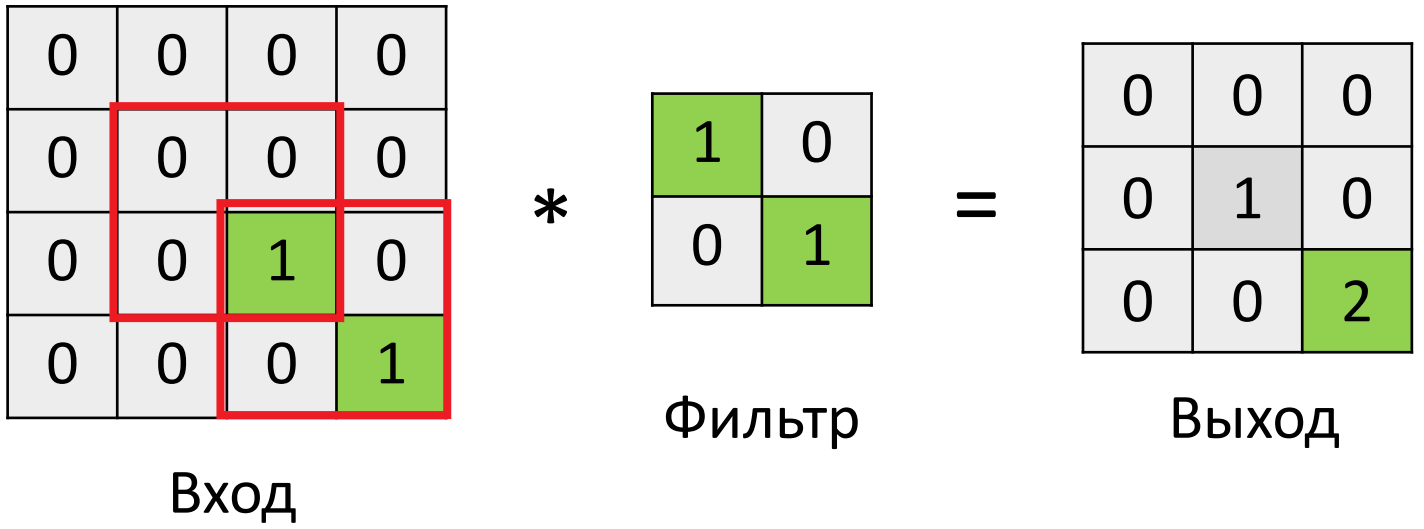
Размытие

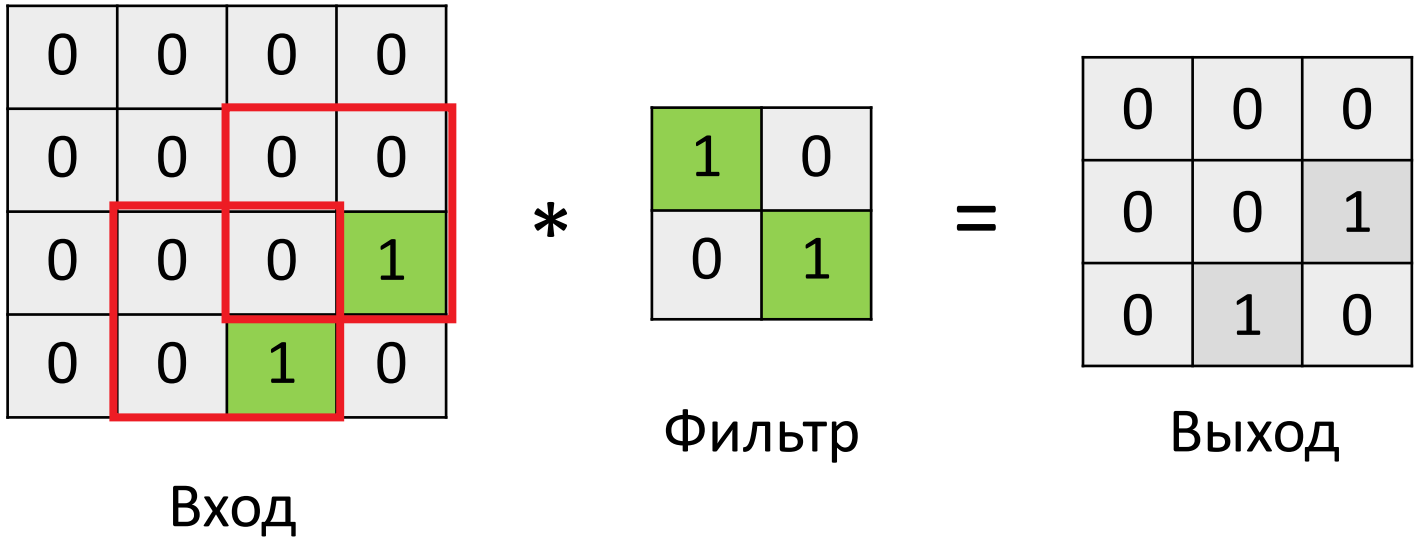
Свертка похожа на корреляцию с шаблоном



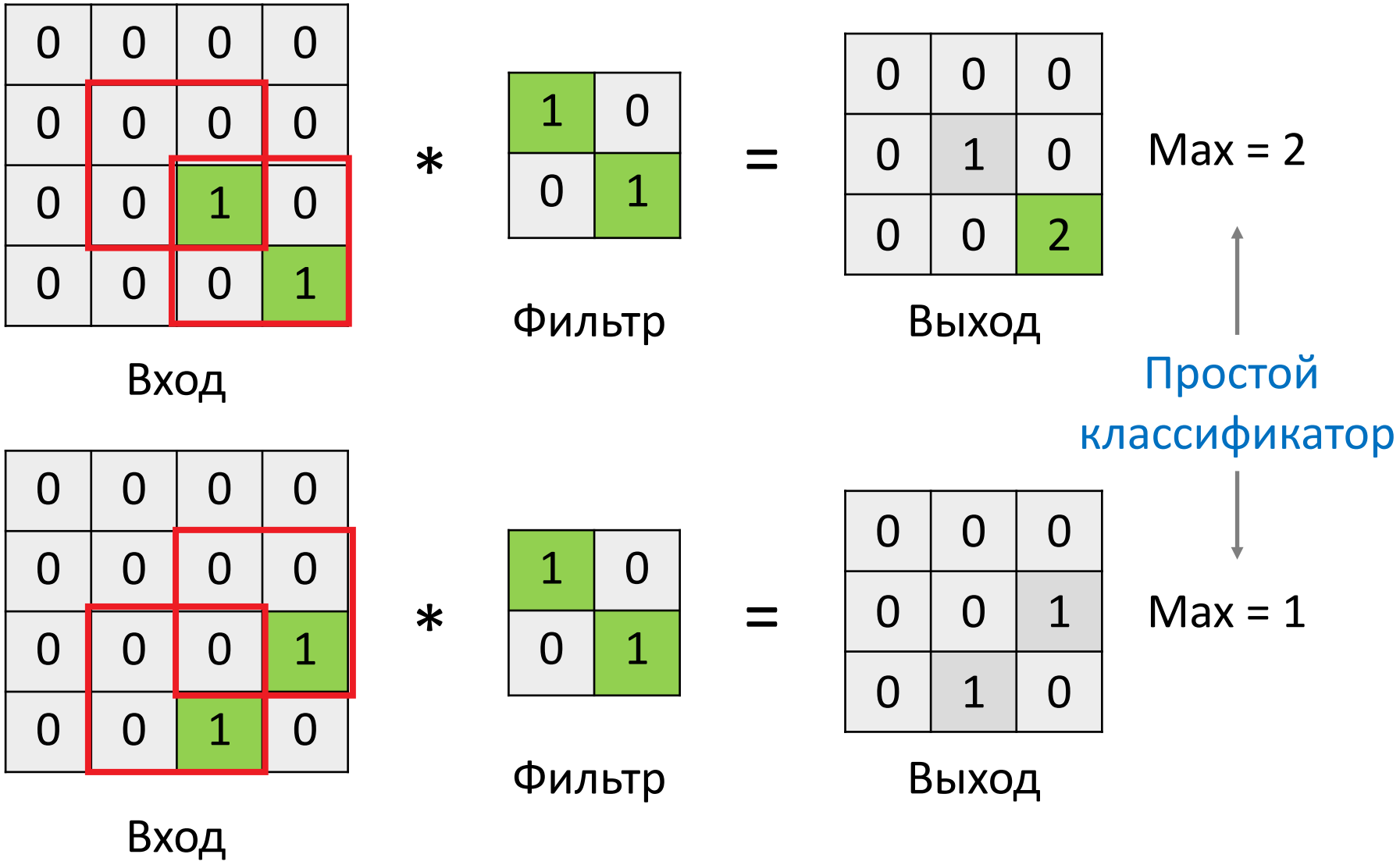
Вход Фильтр Выход

Свертка похожа на корреляцию с шаблоном

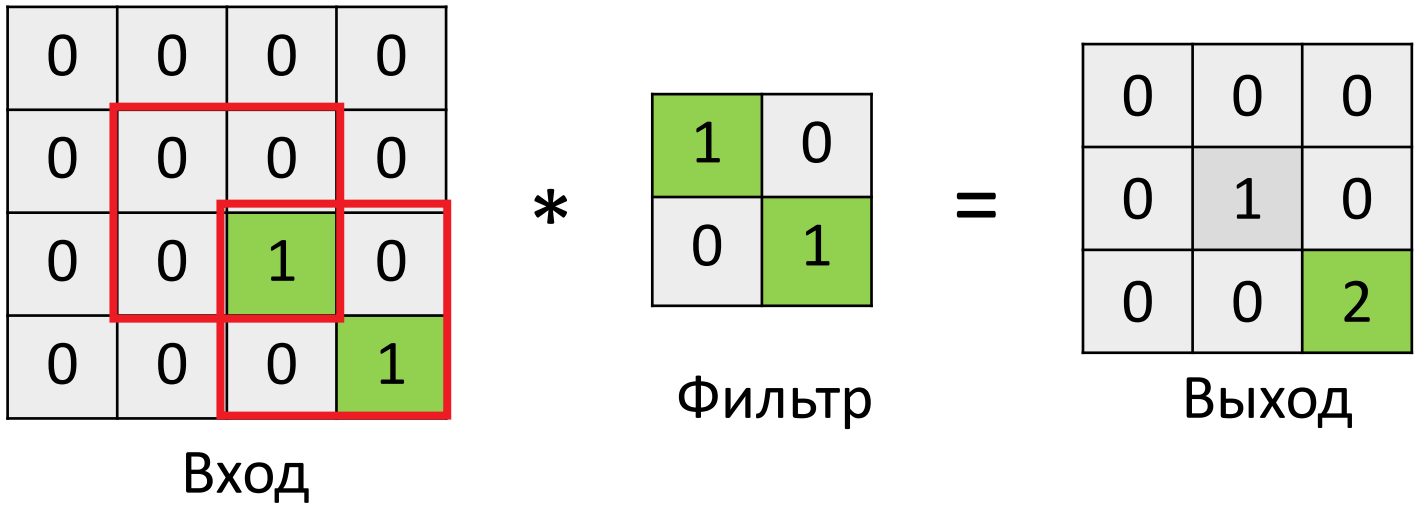




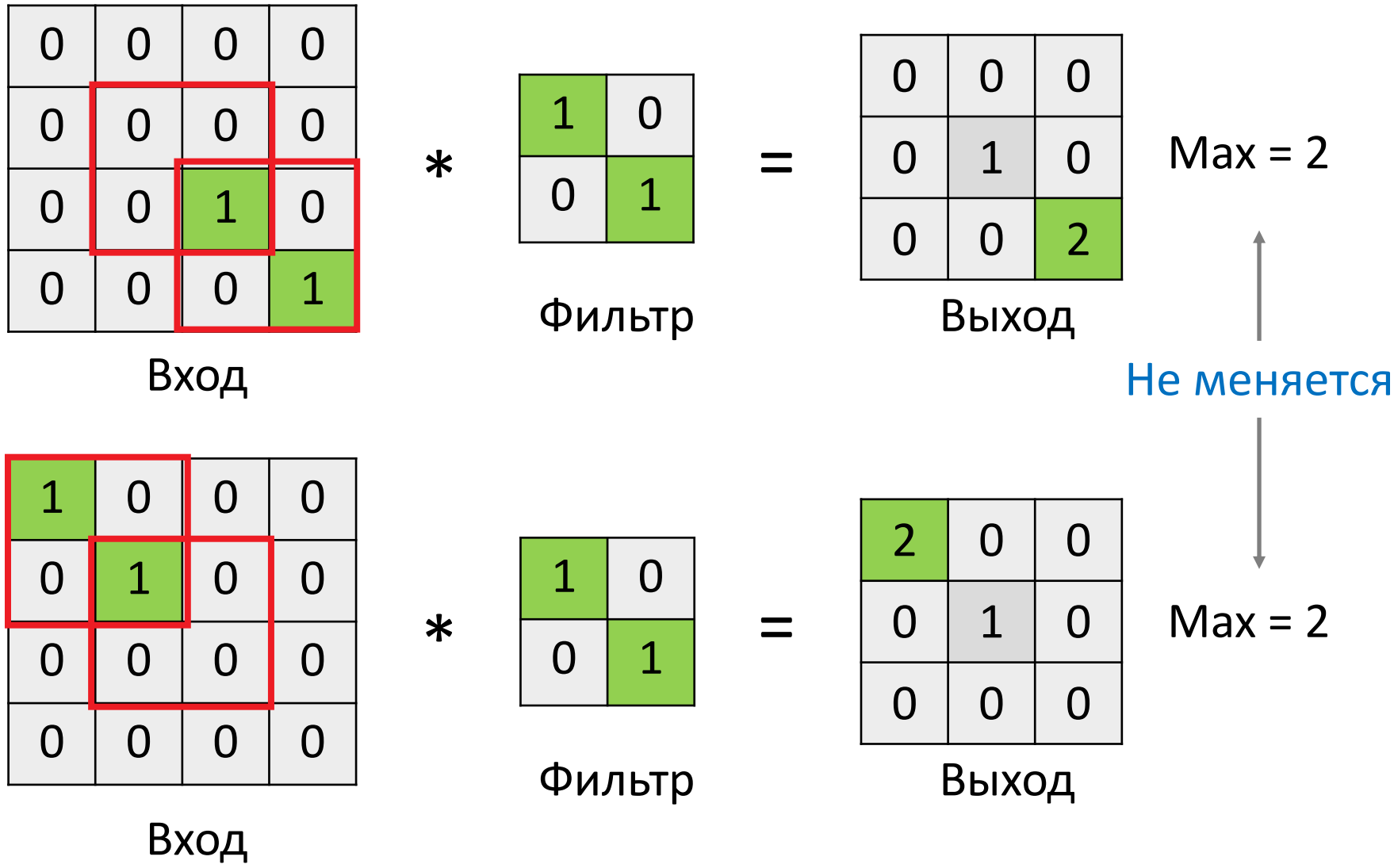
Свертка похожа на корреляцию с шаблоном



Свертка и сдвиг коммутативны



Свертка и сдвиг коммутативны



Сверточный слой в нейросети

Сдвиг:
b

Фильтр (kernel):

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

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

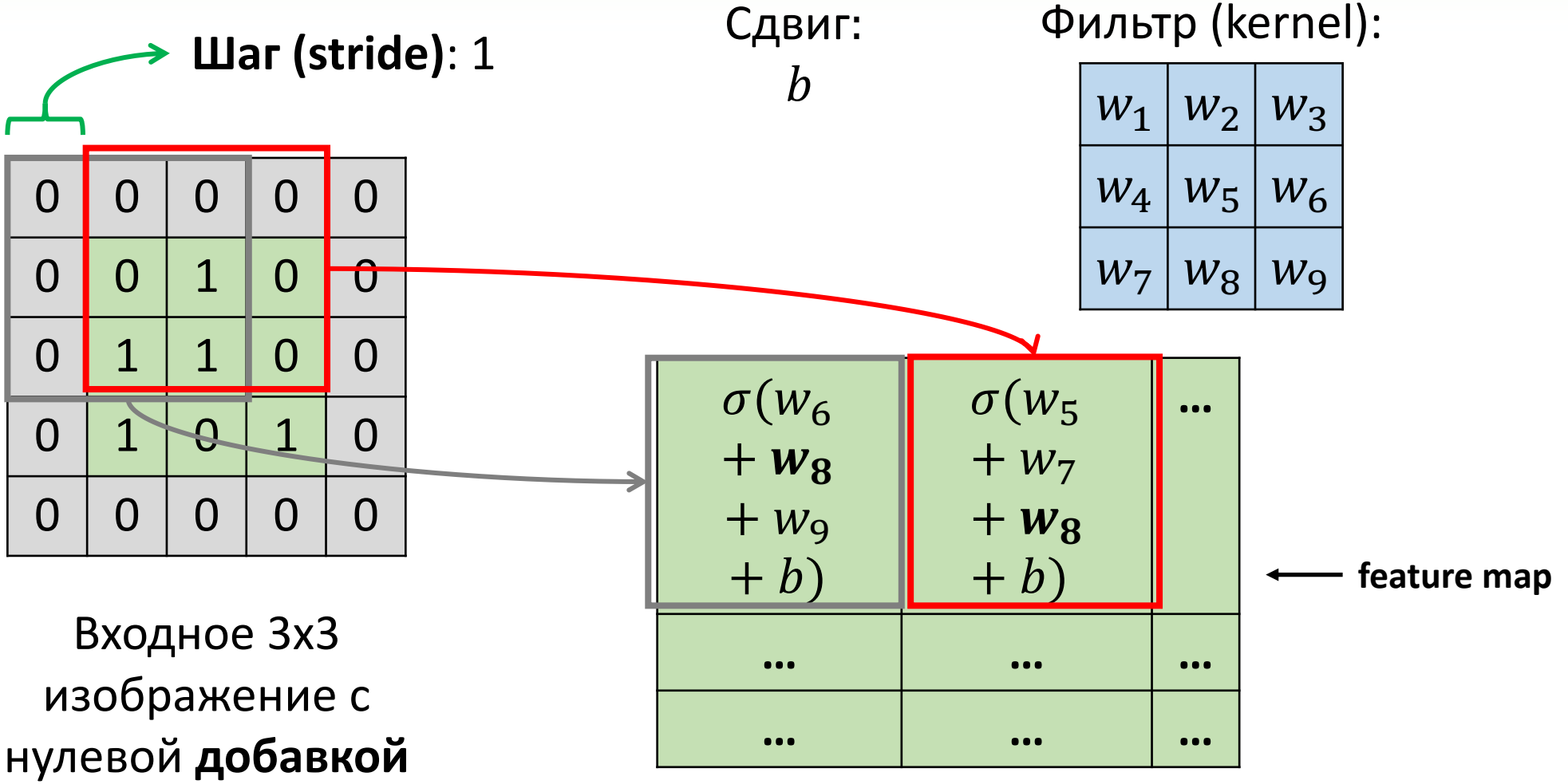
Входное 3x3 изображение с нулевой **добавкой** (padding, серая рамка)

$\sigma(w_6 + \boldsymbol{w_8} + w_9 + b)$	…	…
…	…	…
…	…	…

← **feature map**

9 выходных нейронов используя всего 10 параметров

Сверточный слой в нейросети

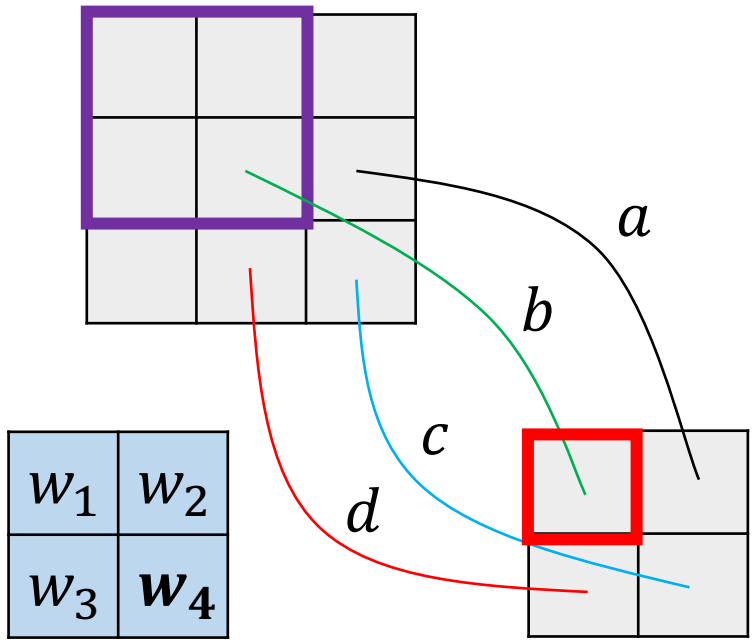


Входное 3x3 изображение с нулевой **добавкой** (padding, серая рамка)

9 выходных нейронов используя всего 10 параметров

Как считать градиенты

Представим, что все использования w_4 это разные веса:



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

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

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

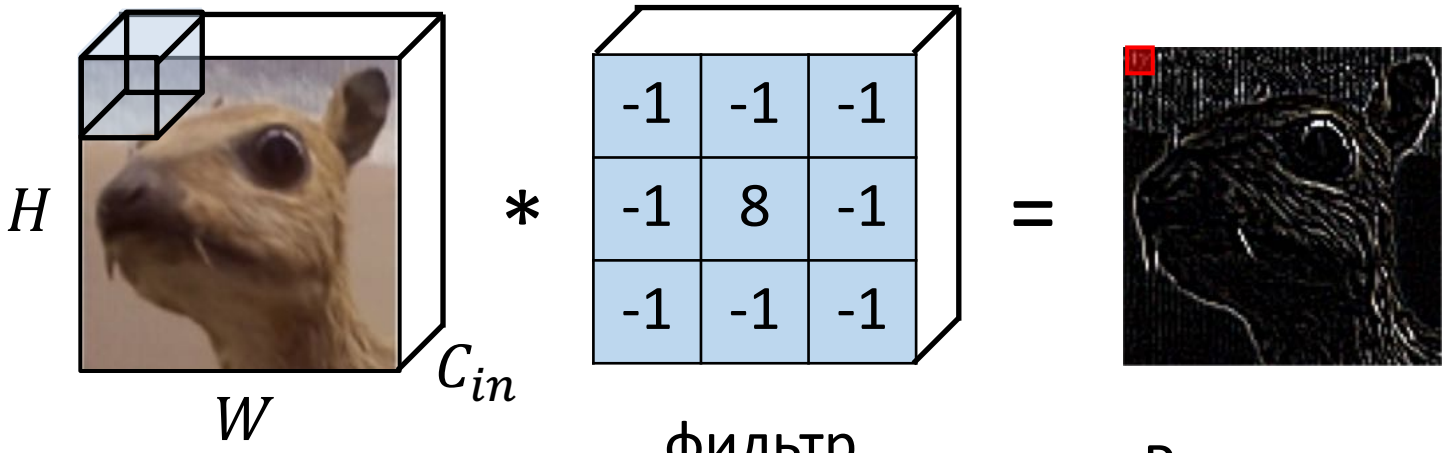
Суммируем градиенты по всем использованиям веса w_4!

В сверточном слое меньше параметров

- Картинка 300x300
- 300x300 выходных нейронов
- Свертка 5x5 – **26** параметров
- В полно-связном слое – $\mathbf{8.1 \times 10^9}$ параметров

Цветная картинка

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

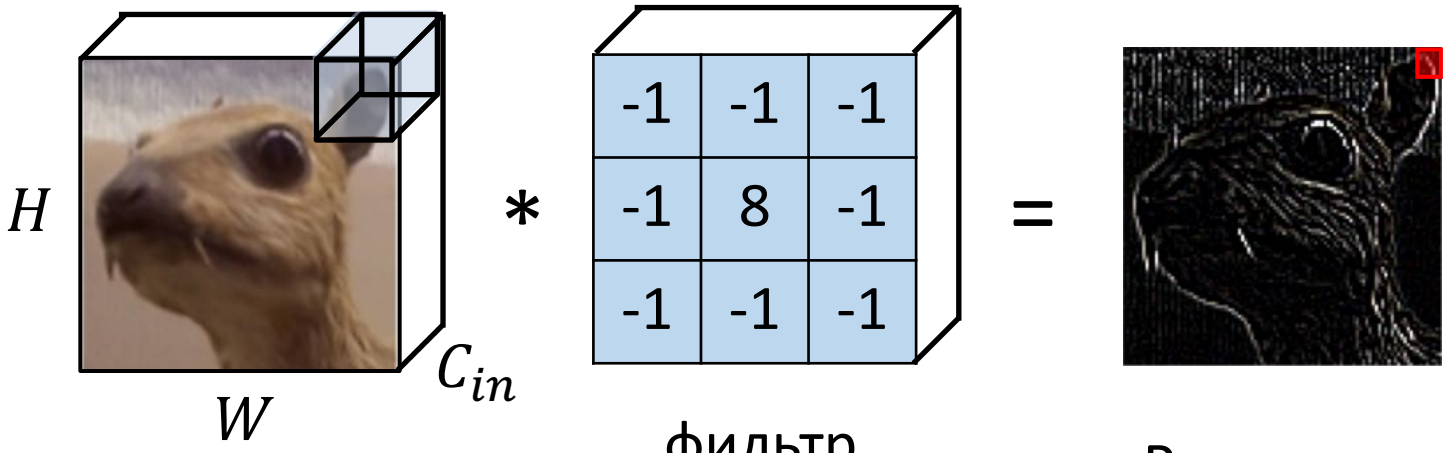


фильтр
W_k× H_k×C_{in}

Результат

Цветная картинка

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

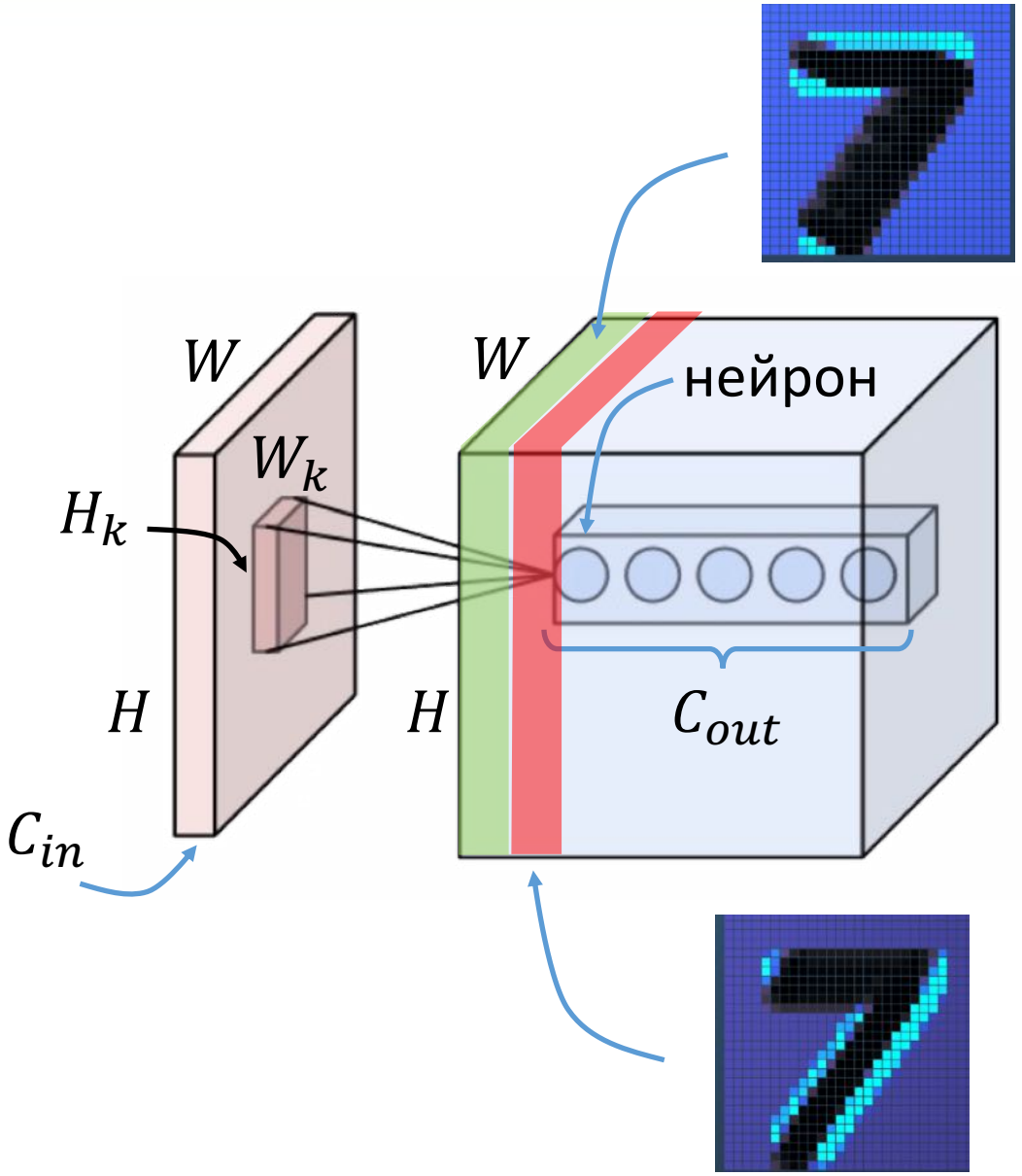


фильтр
W_k× H_k×C_{in}

Результат

Одного фильтра мало!

- На входе W× H×C_{in}
- На выходе W× H×C_{out}



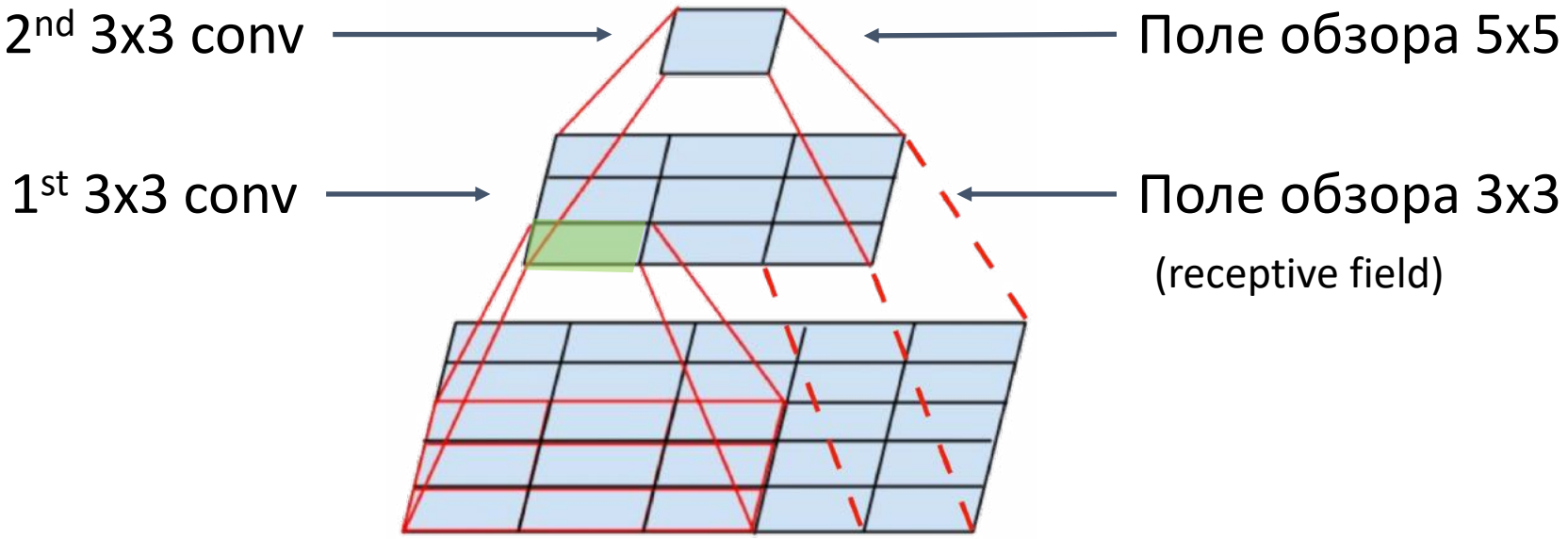
В одном пикселе появляется **глубина** с разными характеристиками пикселя изображения

Сколько параметров?

$$(W_k * H_k * C_{in} + 1) * C_{out}$$

Да и одного сверточного слоя мало!

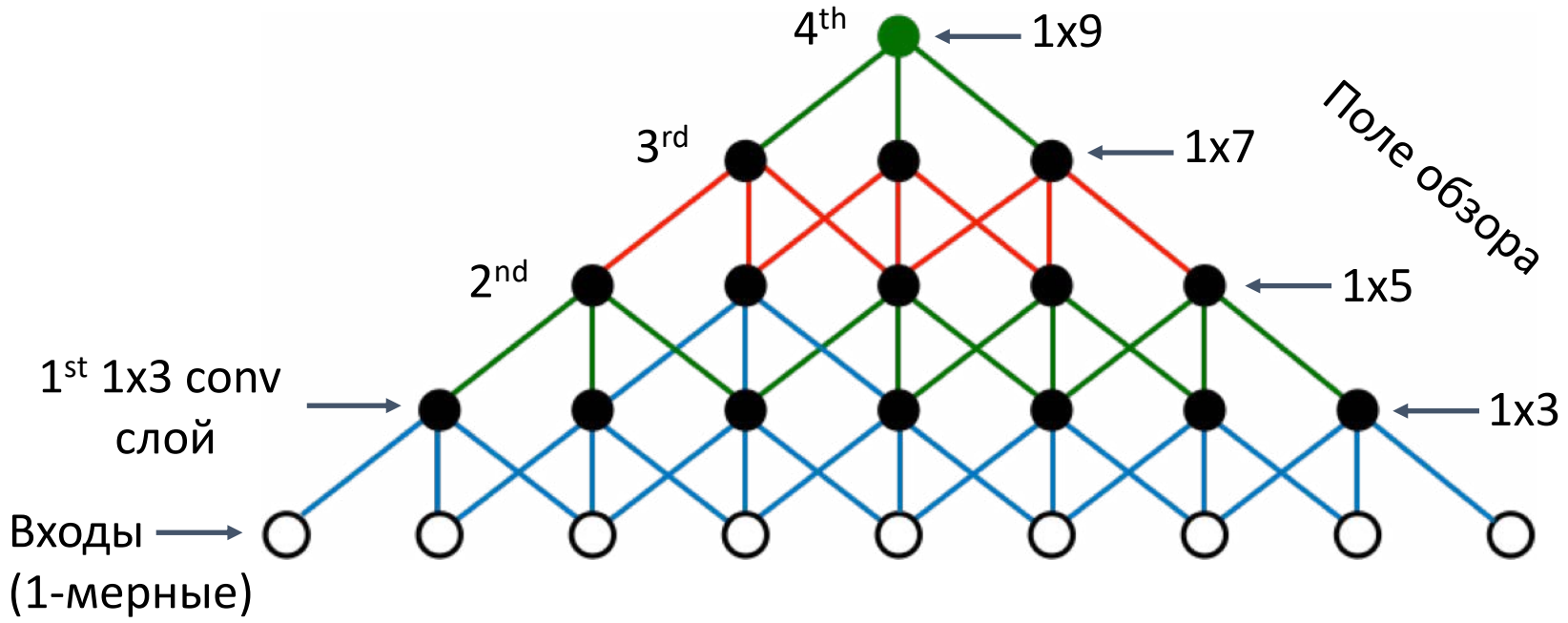
- Нейроны первого слоя смотрят на кусочек 3x3.
- А что если кошка больше? ☺
- Нужен второй слой!



Сколько слоев нужно для поля обзора 300x300?

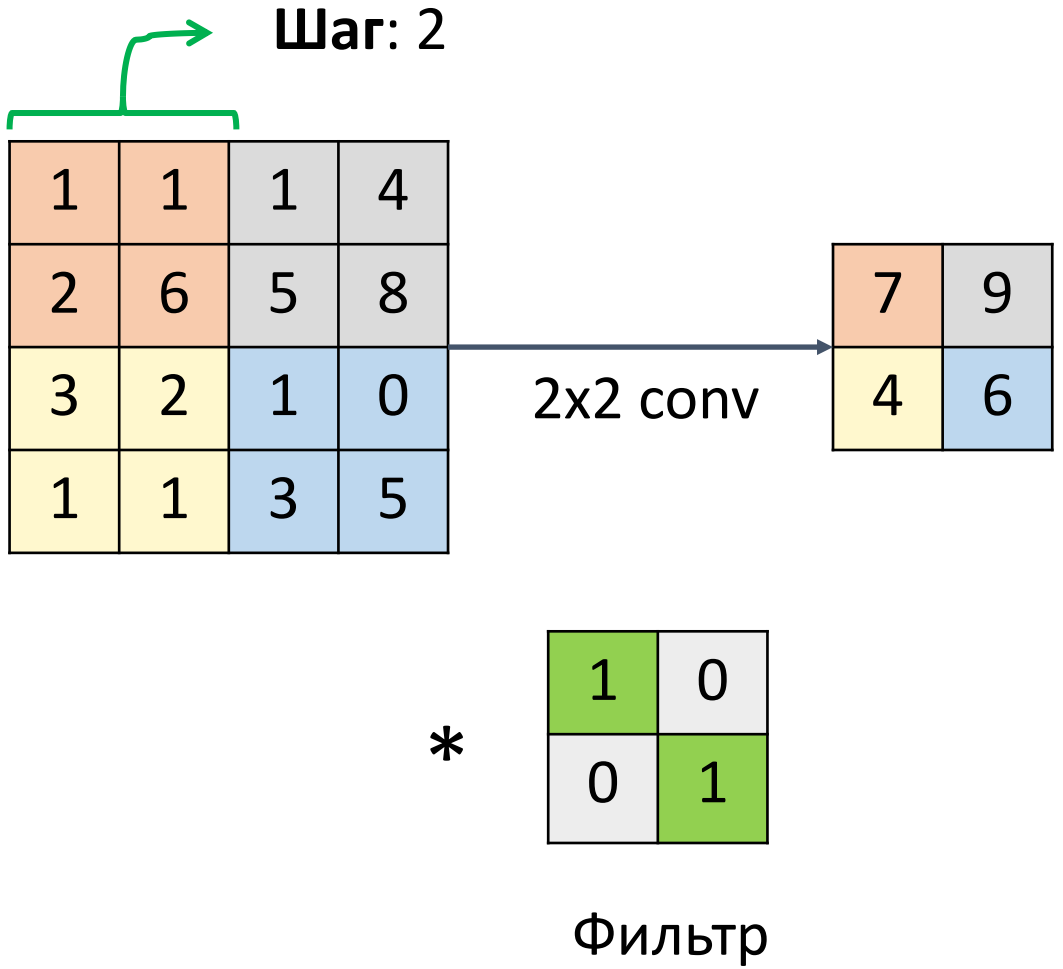
N сверточных слоев

- N слоев с фильтрами 3x3
- На N-ом слое поле обзора $(\boldsymbol{2N+1}) \times (\boldsymbol{2N+1})$.
- Если кот 300x300, то нам надо 150 слоев! Многовато…

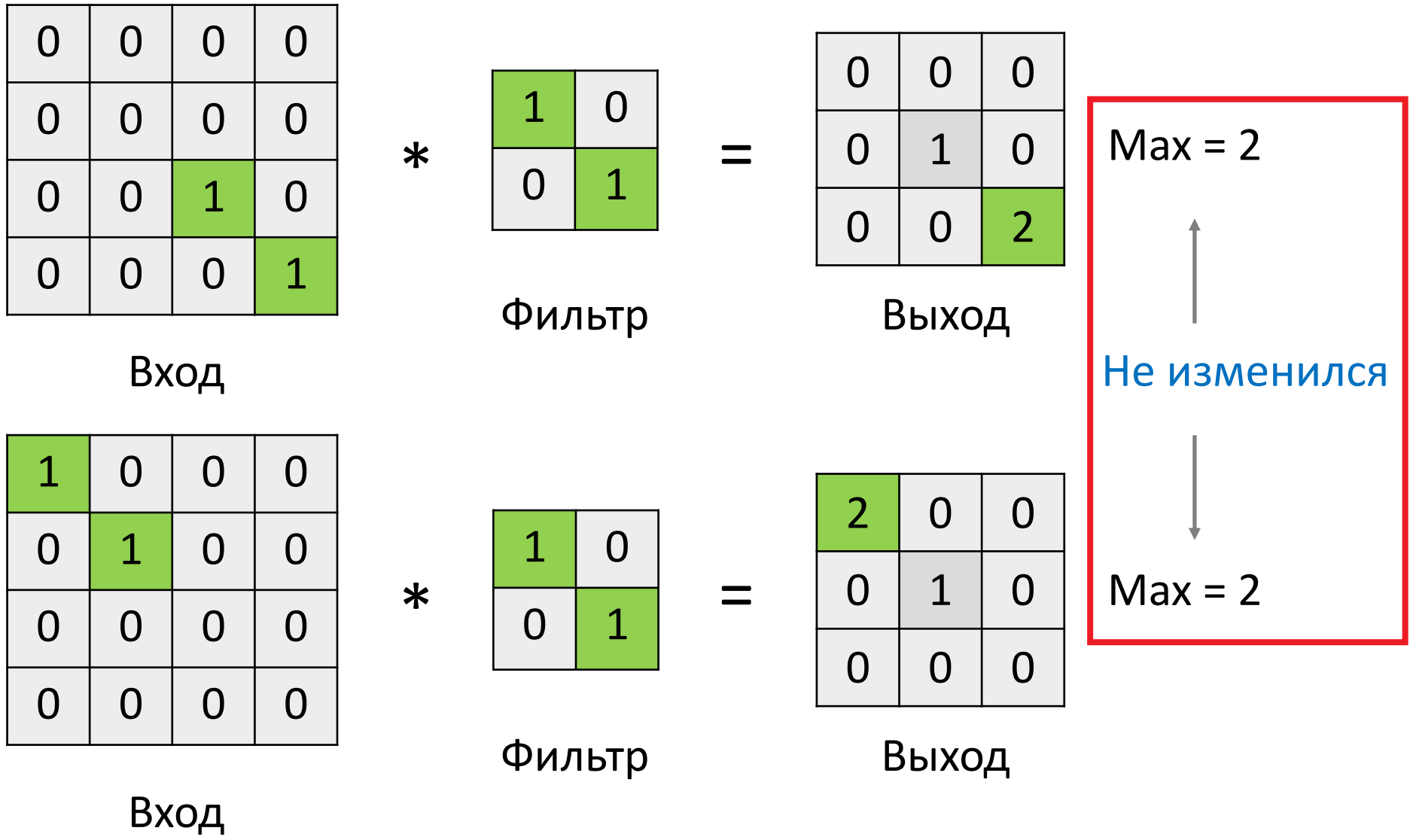


Нужно растить поле обзора быстрее!

Можно увеличить шаг свертки!

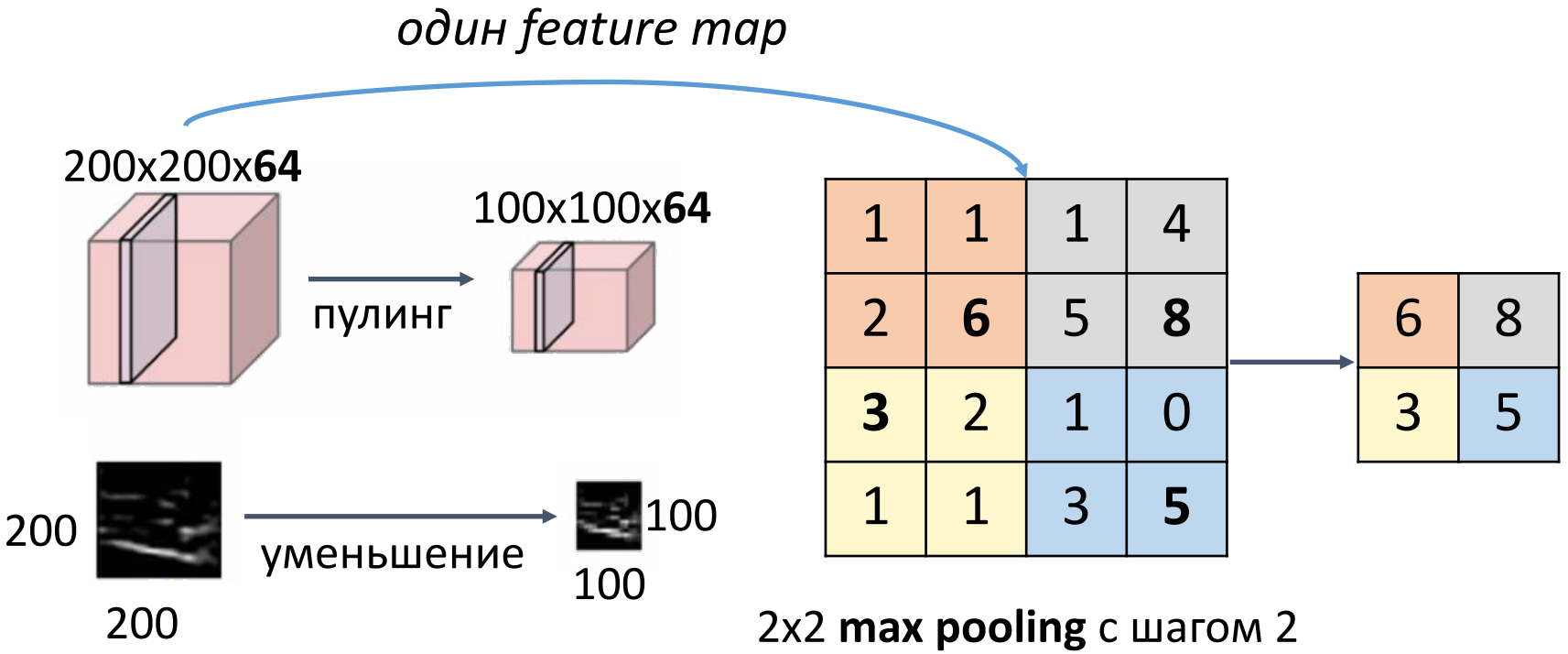


Давайте вспомним инвариантность к сдвигу!



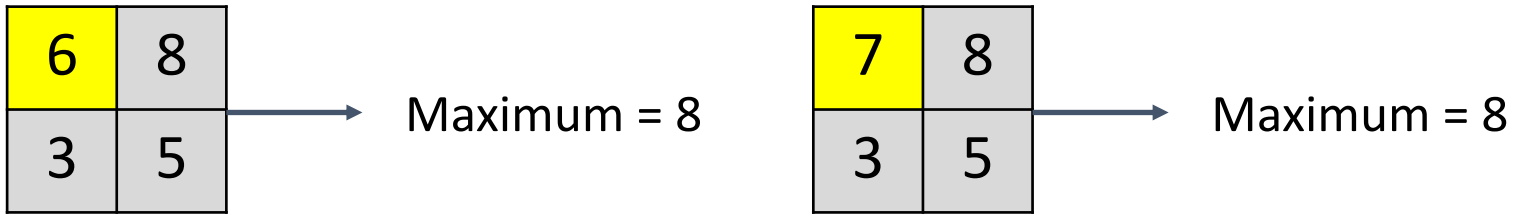
Пулинг слой

- Работает почти как свертка, только вместо свертки берет **максимум** (или среднее).

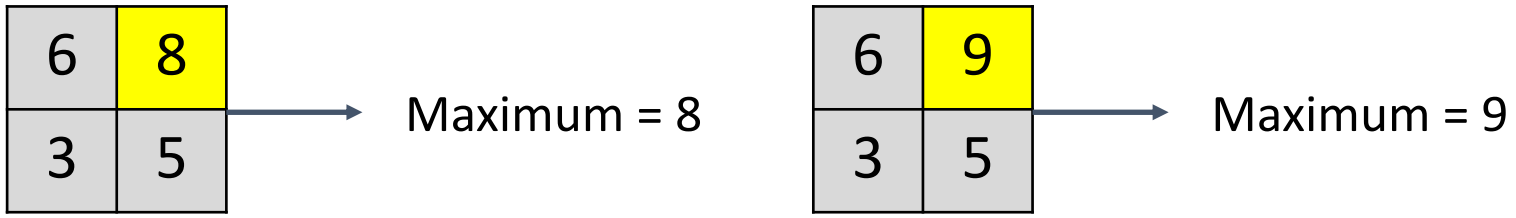


Как считать градиент для пулинга

- Строго говоря: максимум не дифференцируемая функция!



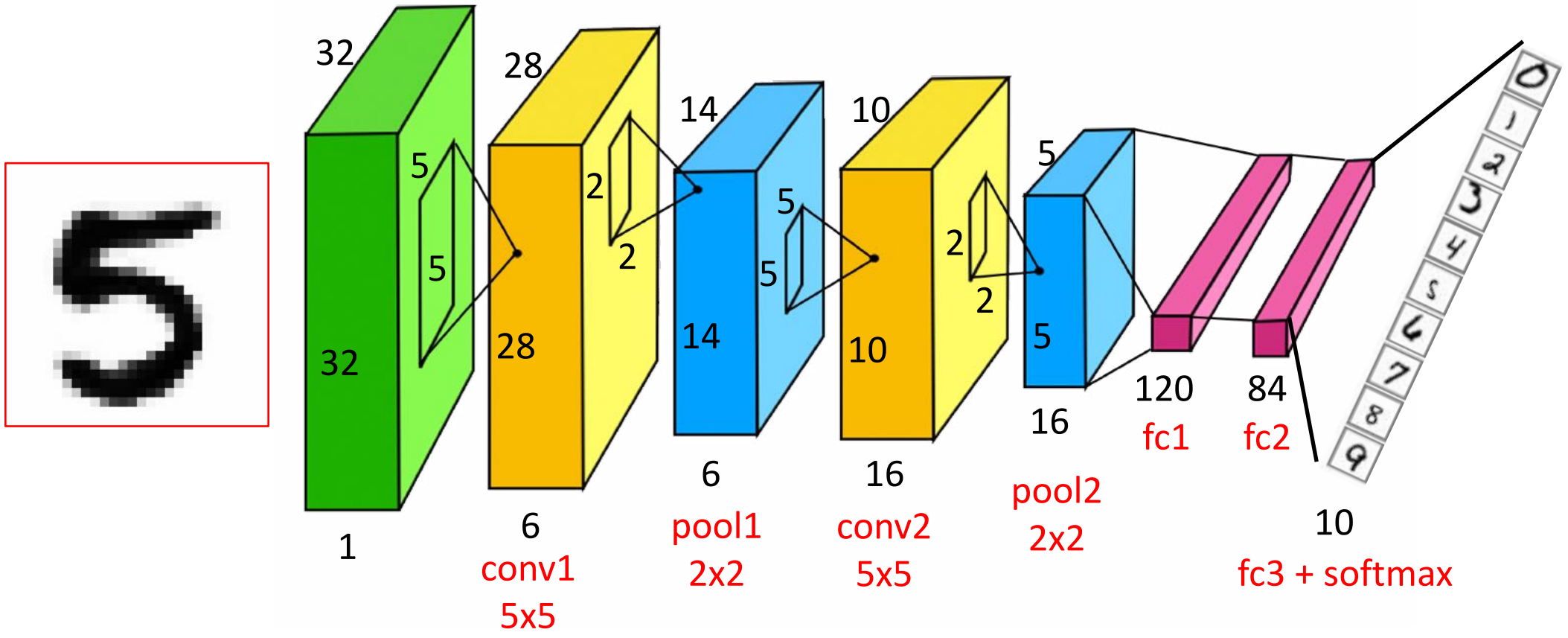
- Градиент 0 по немаксимальным входам, потому что при их изменении не меняется выход (максимум).



- Для максимального входа градиент 1.

Соберем это все в сверточную сеть

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



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

Softmax

- Softmax — это обобщение логистической функции для многомерного случая.

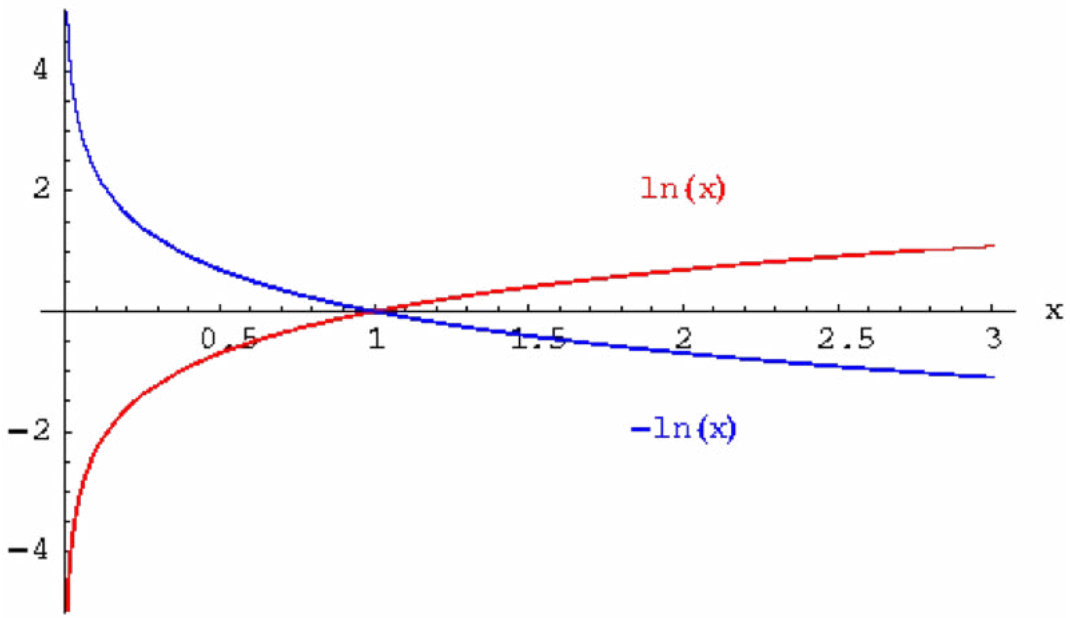
- Преобразует вектор значений в такие, что каждое из них на отрезке [0,1] и в сумме они дают 1:

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

Log loss (cross-entropy)

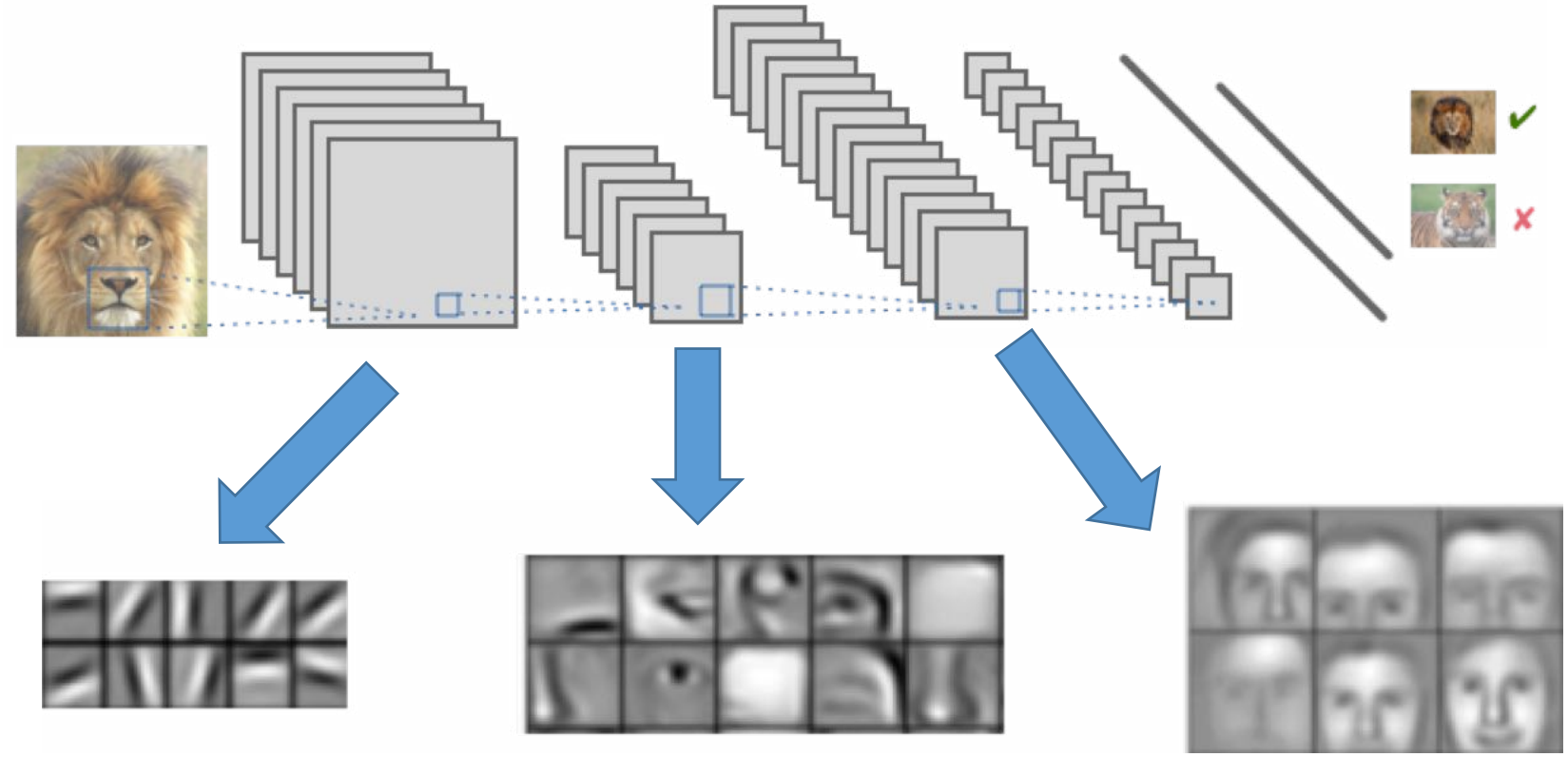
- Потери для классификации на K классов:

$$-\sum_{k=1}^{K} \log(p_k)\,[y = k]$$



Нейросеть учит иерархические шаблоны

Глубокая нейронная сеть:

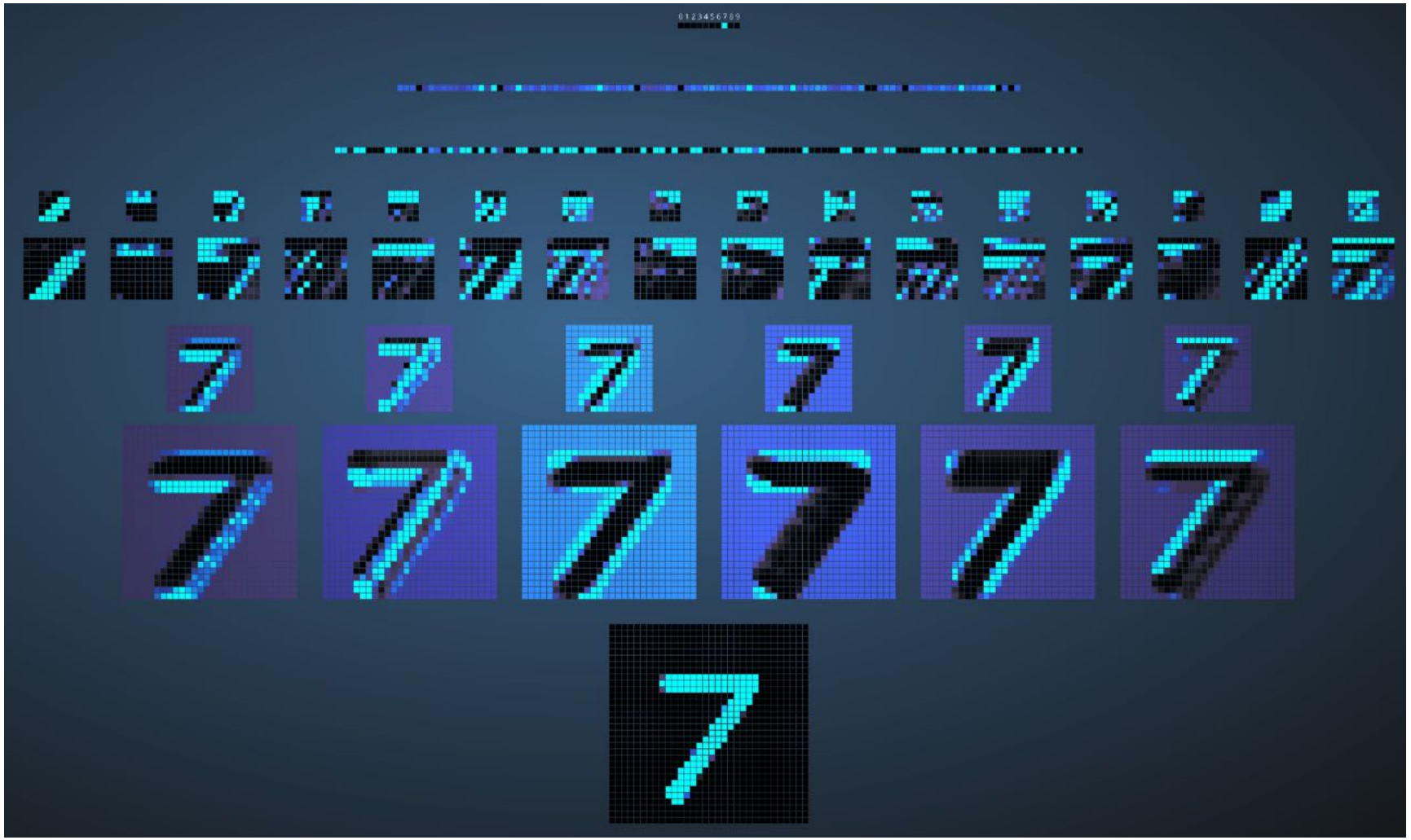


Входы, на которые реагируют нейроны

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

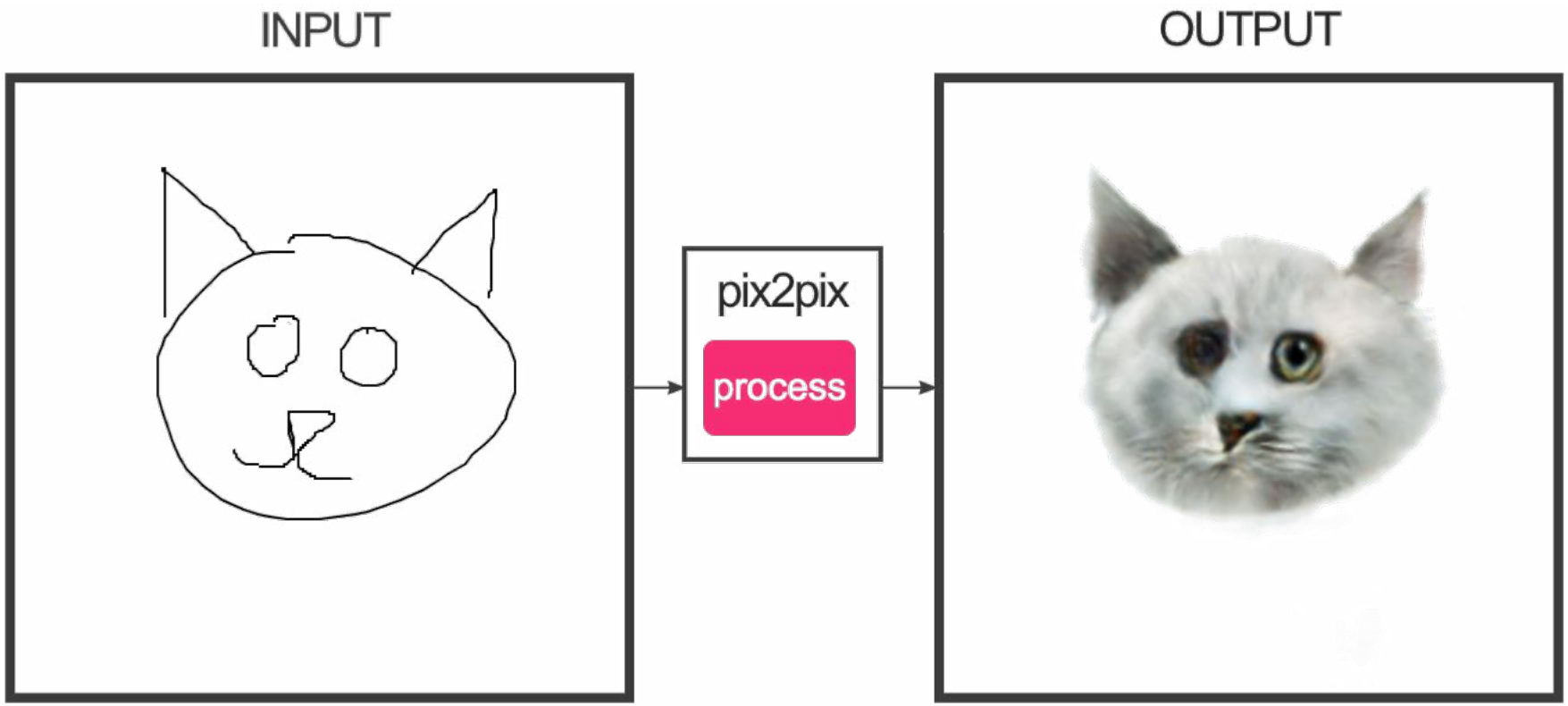


Применение: image to image

- https://affinelayer.com/pixsrv/

Применение: image to image

- https://affinelayer.com/pixsrv/

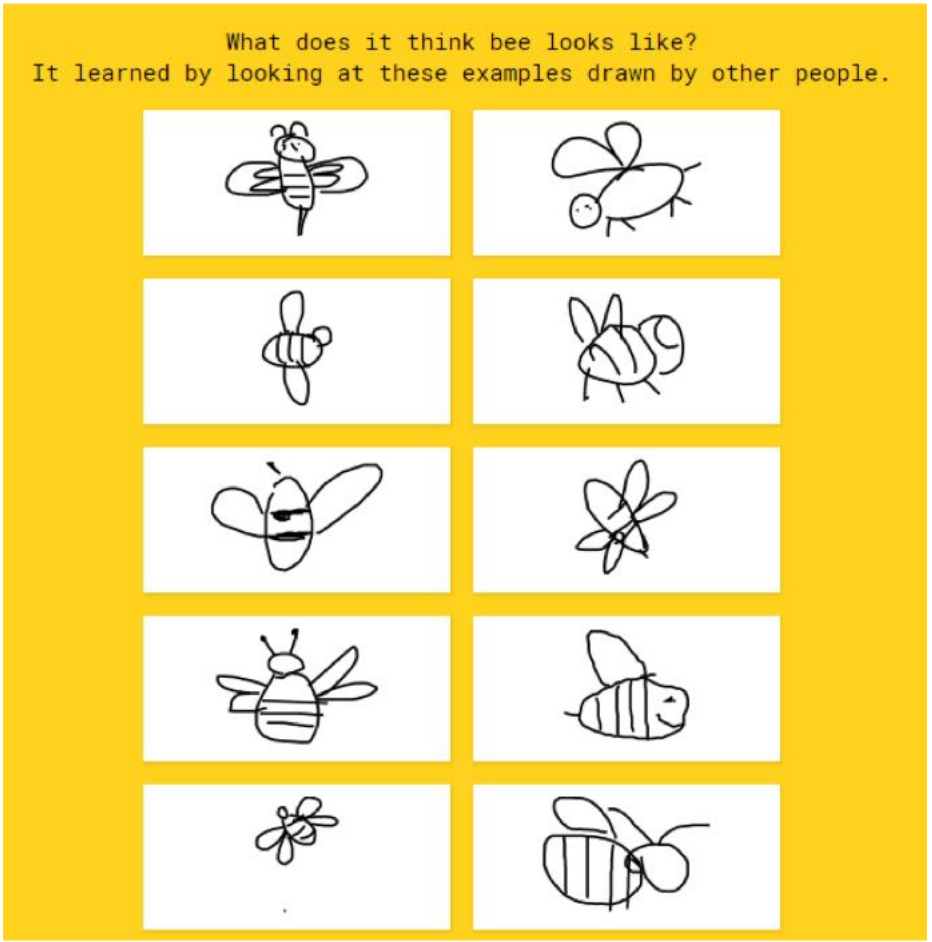


Применение: рисунки от руки

- https://quickdraw.withgoogle.com
- На Kaggle: https://www.kaggle.com/c/quickdraw-doodle-recognition

Применение: рисунки от руки

- https://quickdraw.withgoogle.com
- На Kaggle: https://www.kaggle.com/c/quickdraw-doodle-recognition



Ссылки

- http://cs231n.stanford.edu/
- http://cs231n.github.io/convolutional-networks/
- https://brohrer.github.io/how_convolutional_neural_networks_work.html
- https://blog.keras.io/how-convolutional-neural-networks-see-the-world.html