

MML minor #4

Сверточные нейронные сети

Какие задачи хотим решать



Уточка

Какие задачи хотим решать



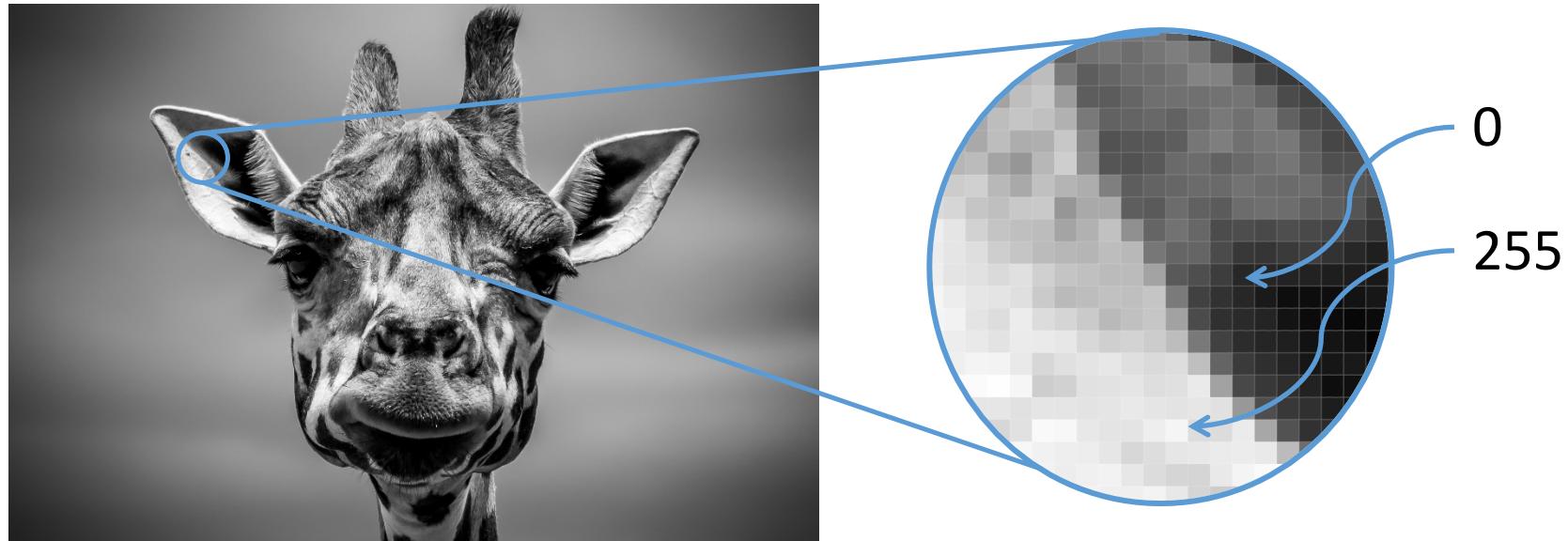
Уточка



52

Как компьютер видит картинку

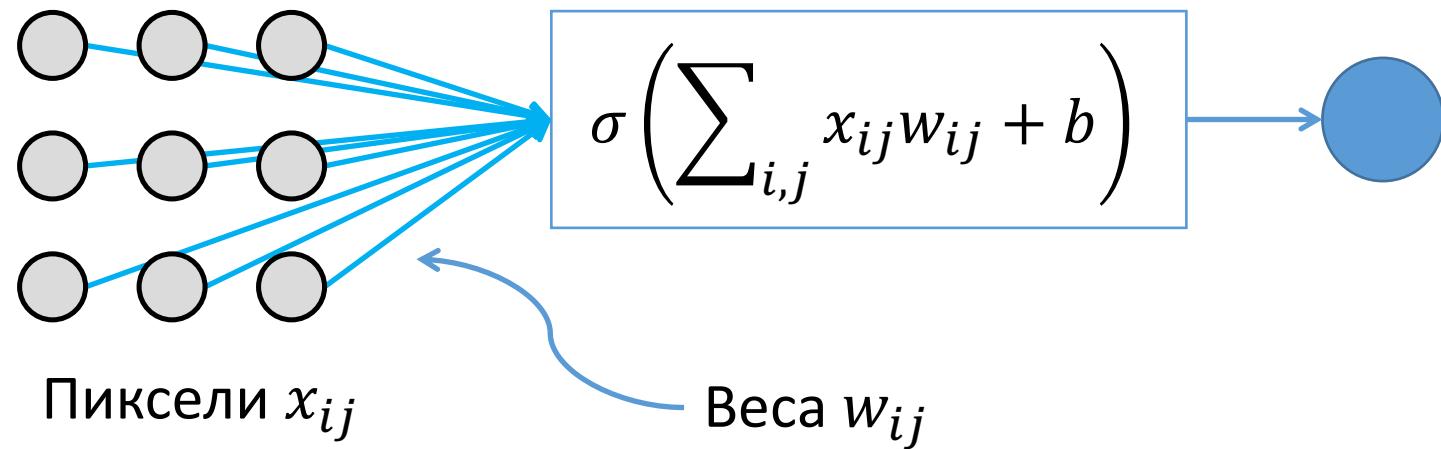
- Матрица яркостей пикселей (например размера 300 x 300)
- Яркости пикселей от 0 до 255



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

Как применить к картинке нейросеть?

Нейрон поверх всех пикселей?

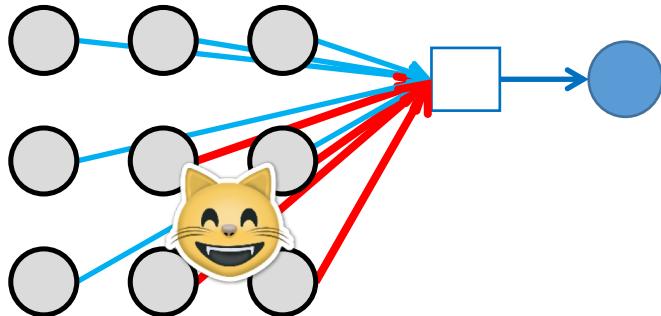


$$x_{norm} = \frac{x}{255} - 0.5$$

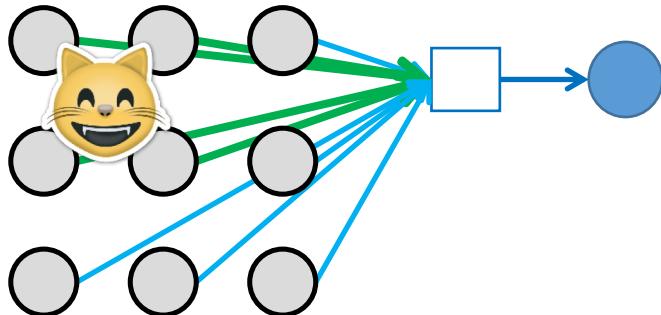
Не, так не работает!

Почему нет?

- Хотим выучить детектор кота:



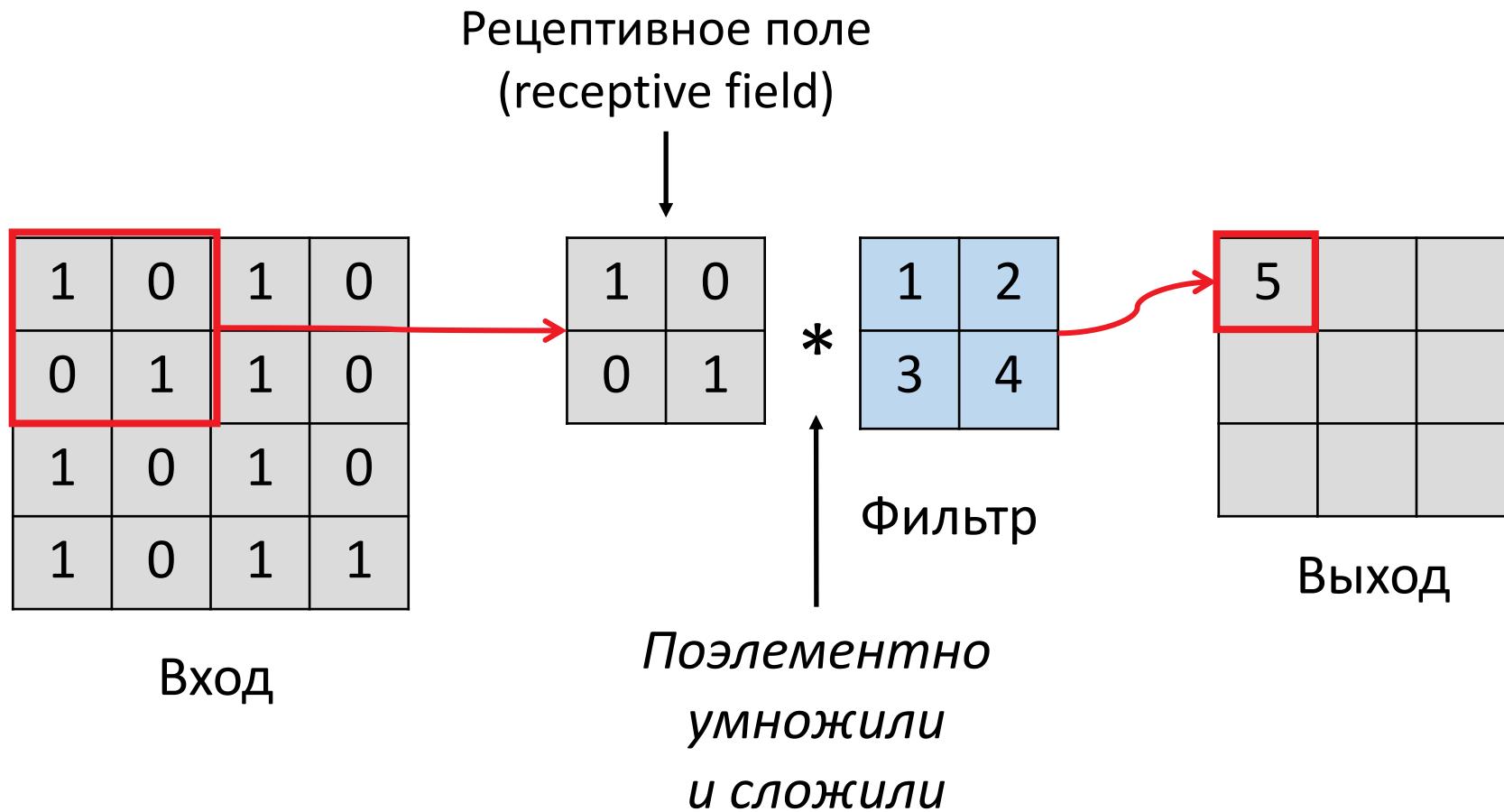
На этой картинке **красные** веса w_{ij} немного изменяются при градиентном спуске



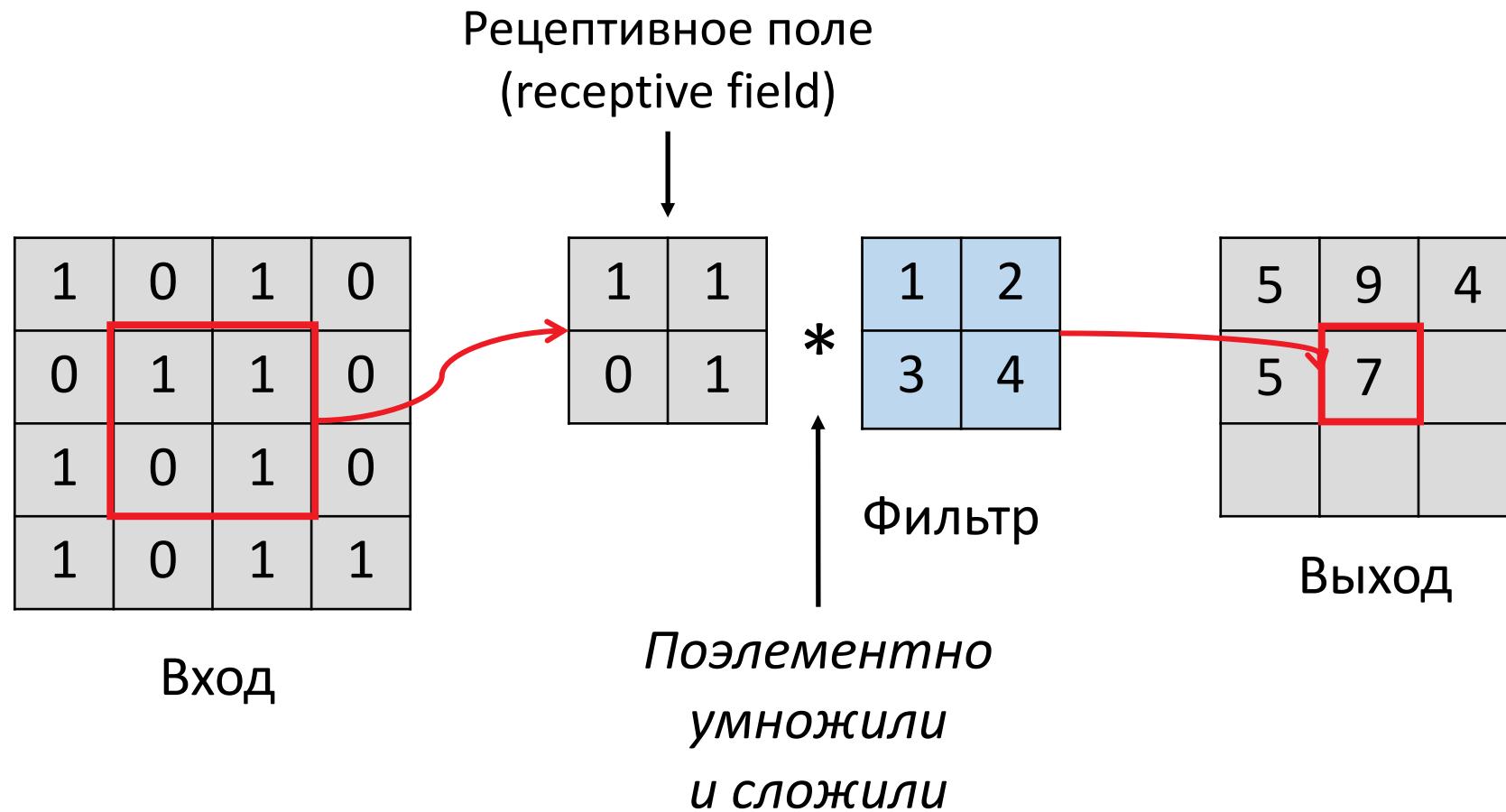
А на этой **зеленые** веса w_{ij} изменяются...

- 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

$$\text{Входная картинка} \quad * \quad \begin{array}{|c|c|c|} \hline -1 & -1 & -1 \\ \hline -1 & 8 & -1 \\ \hline -1 & -1 & -1 \\ \hline \end{array} \quad = \quad \text{Поиск краев}$$

Фильтр

На однотонной заливке дает ноль (черный)



Свертки применяют к картинкам давно



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

Добавляет к картинке края

Свертки применяют к картинкам давно



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

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

*

=

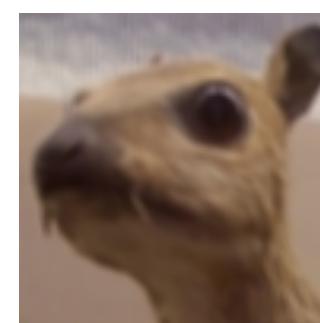


Повышение
резкости

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

$$*\frac{1}{9}$$

=



Размытие

Свертка похожа на корреляцию с шаблоном

The diagram illustrates a convolution operation. On the left, labeled "Вход" (Input), is a 4x4 grid of zeros. A 3x3 subgrid in the bottom-right corner is highlighted with a red border. In the center, labeled "Фильтр" (Filter), is a 3x3 grid with values 1, 0, 0; 0, 1, 0; and 1, 0, 0. To the right of the filter is a multiplication sign (*). On the far right, labeled "Выход" (Output), is a 3x3 grid where the bottom-right element is 2, while all other elements are 0. This indicates that the filter has been applied to the highlighted input subgrid, resulting in a value of 2 at the output position corresponding to the filter's receptive field.

Свертка похожа на корреляцию с шаблоном

The diagram illustrates a convolution operation. On the left, labeled "Вход" (Input), is a 4x4 matrix with values:

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

A 3x3 kernel, labeled "Фильтр" (Filter), is applied to the input. The kernel has values:

1	0
0	1

The result of the convolution is a 1x3 vector, labeled "Выход" (Output), with values:

0	0	0
0	1	0
0	0	2

The operation is represented by the formula: $\text{Вход} * \text{Фильтр} = \text{Выход}$.

The diagram shows a convolution operation. On the left, a 4x4 input matrix (labeled "Вход") is shown with its bottom-right 2x2 submatrix highlighted in green. To its right is a multiplication sign (*). Next is a 2x2 filter matrix (labeled "Фильтр") with its top-left element (1) highlighted in green. To its right is an equals sign (=). On the far right, a 3x3 output matrix (labeled "Выход") is shown, with its second row and third column highlighted in green.

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

Max = 2

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

Вход

*

1	0
0	1

Фильтр

=

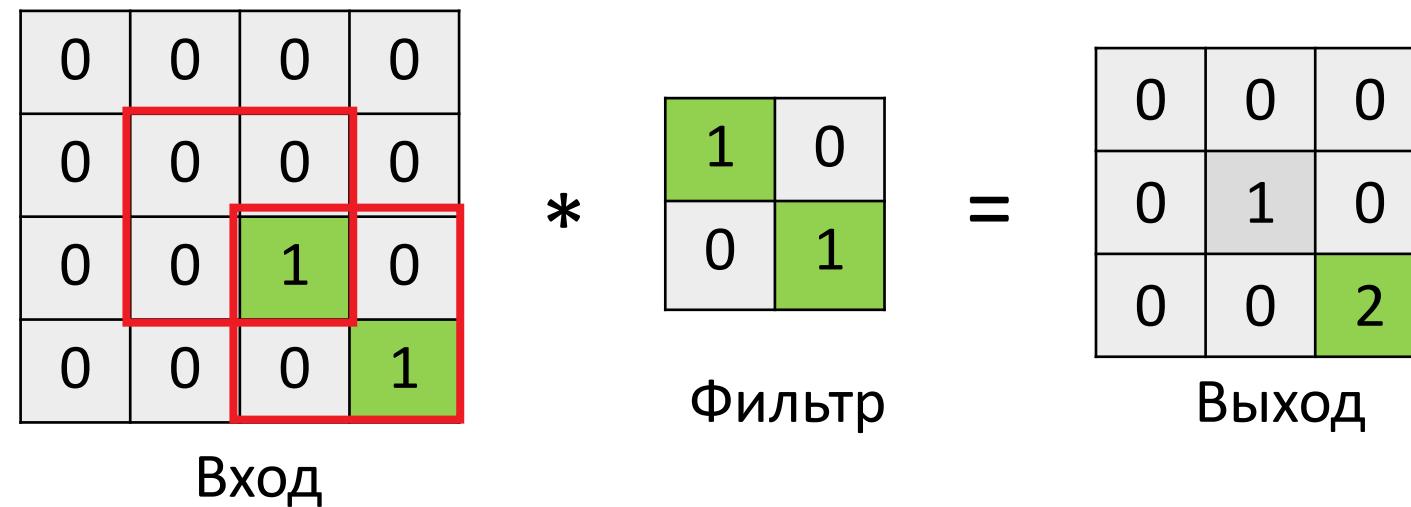
0	0	0
0	0	1
0	1	0

Выход

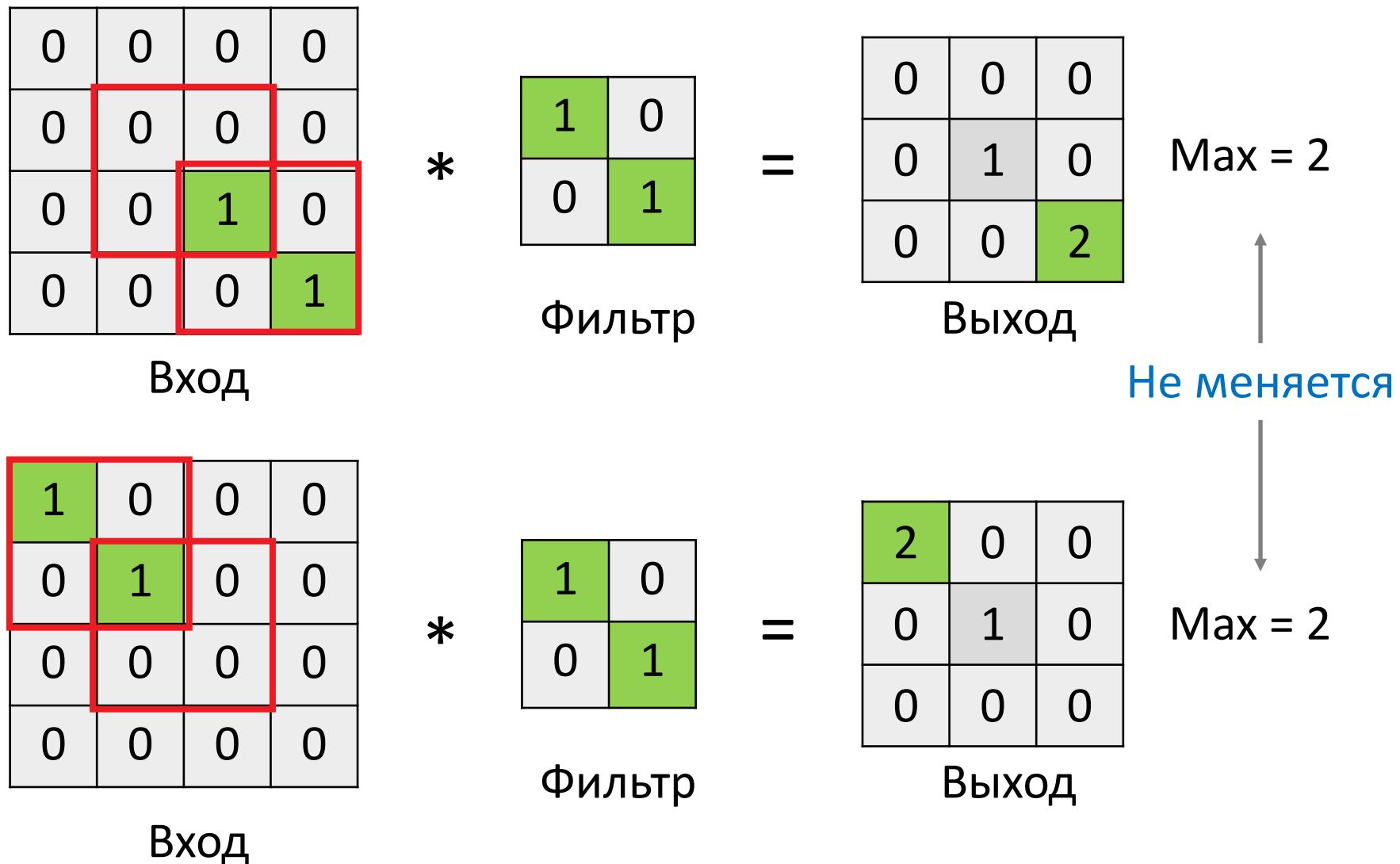
Max = 1

Простой
классификатор

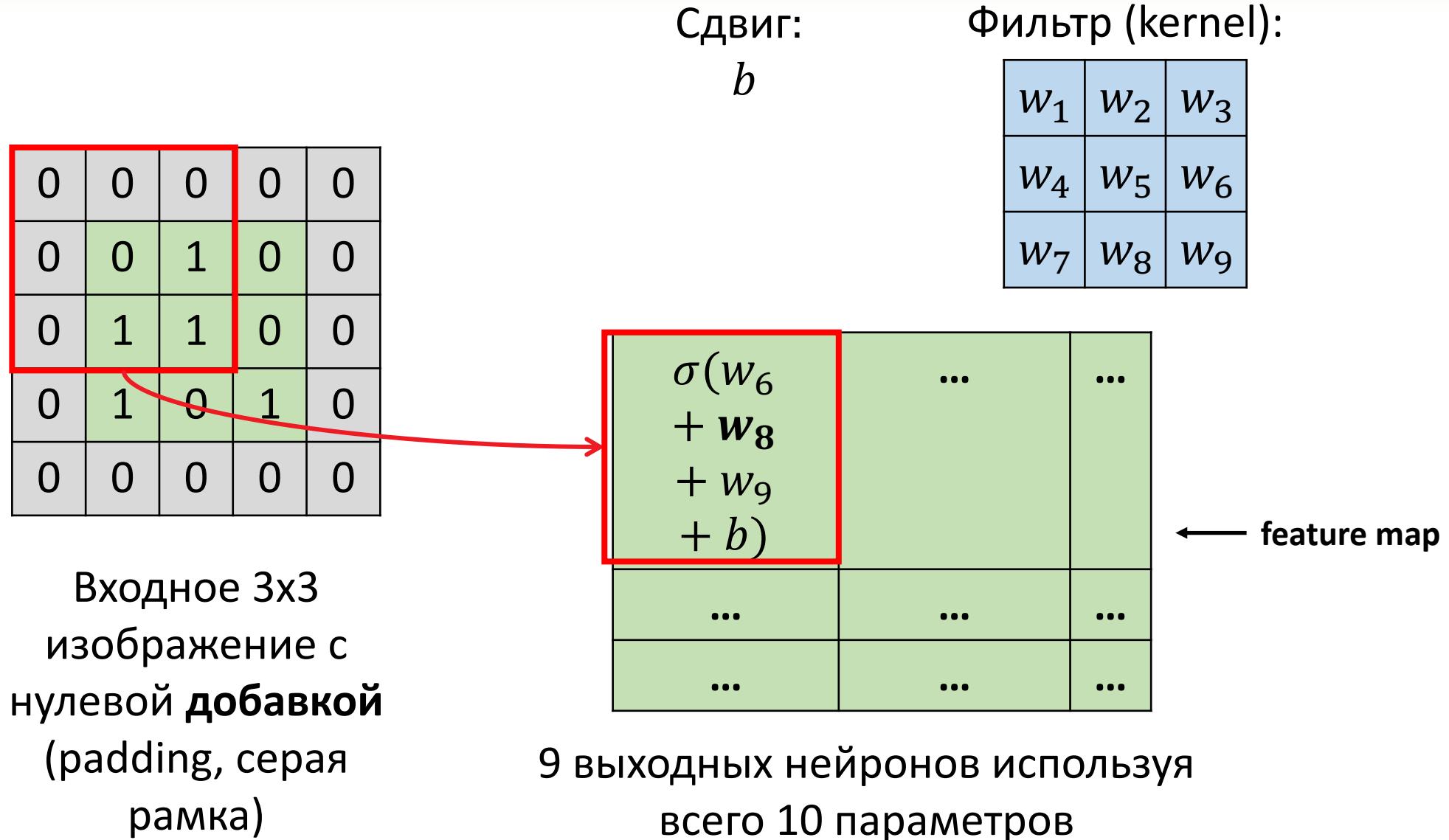
Свертка и сдвиг коммутативны



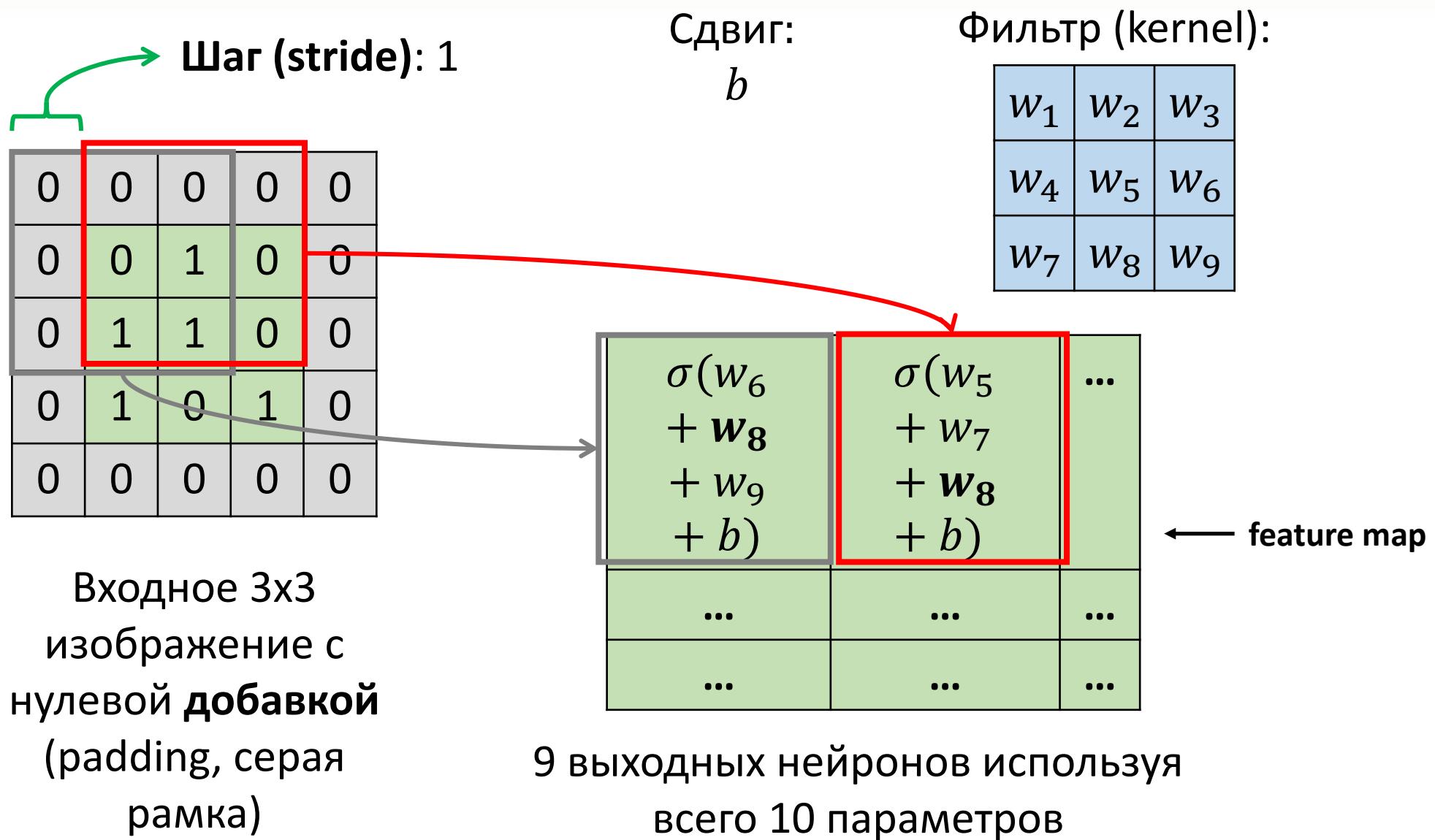
Свертка и сдвиг коммутативны



Сверточный слой в нейросети

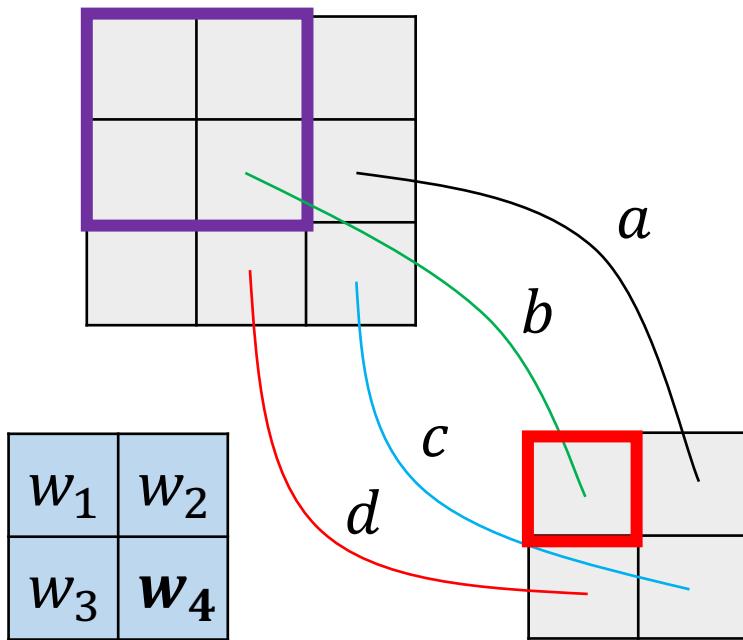


Сверточный слой в нейросети



Как считать градиенты

Представим, что все использования w_4 это разные веса:



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

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

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

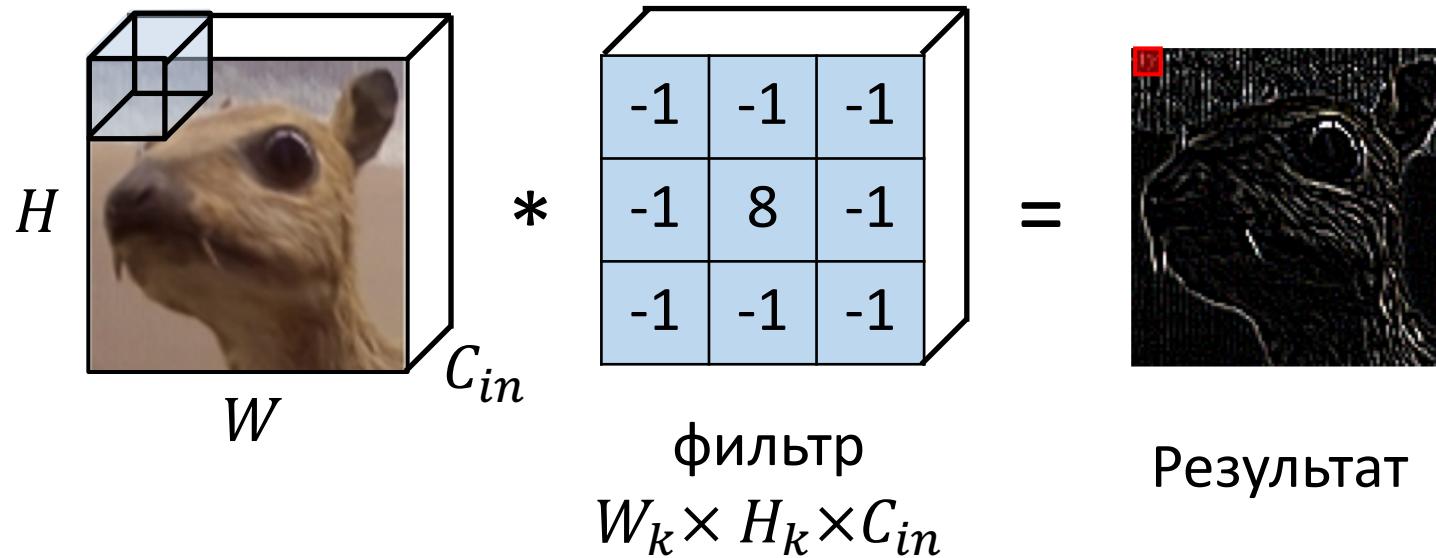
Суммируем градиенты по всем использованием веса w_4 !

В сверточном слое меньше параметров

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

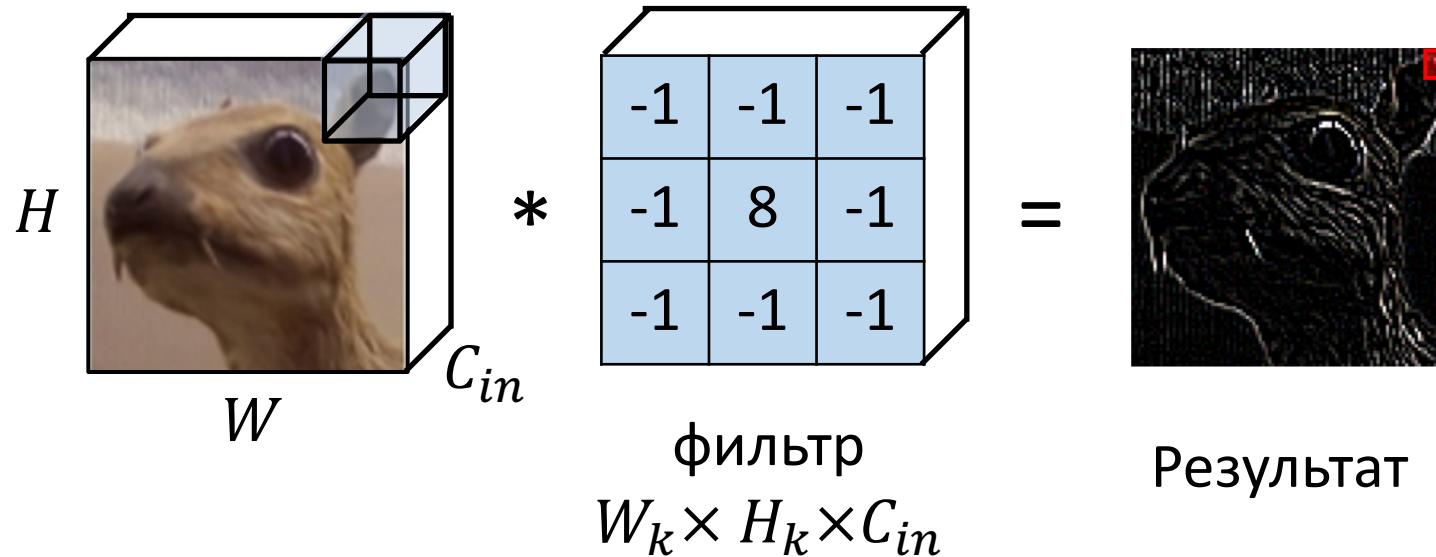
Цветная картинка

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



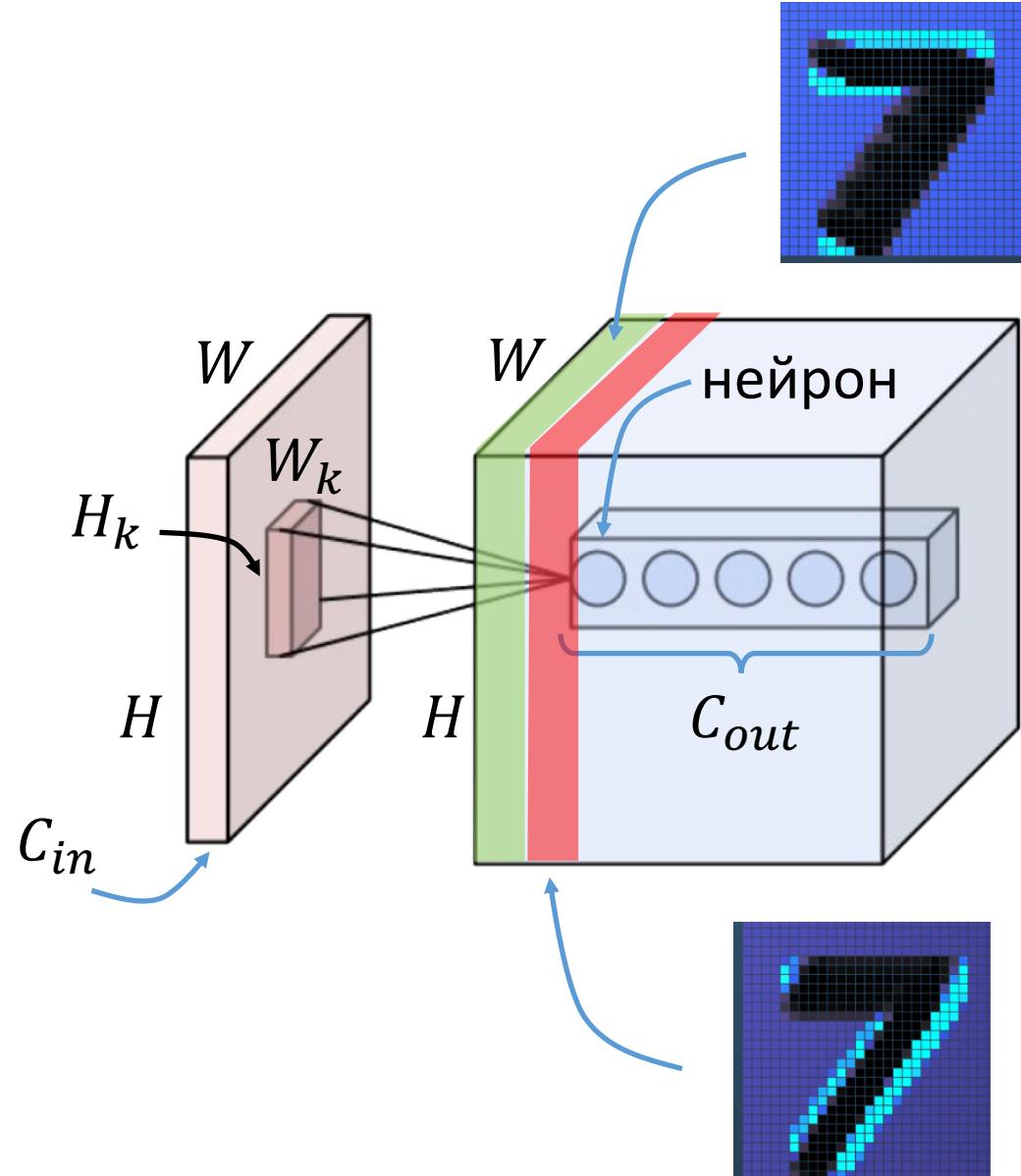
Цветная картинка

- 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 $W \times H \times C_{out}$



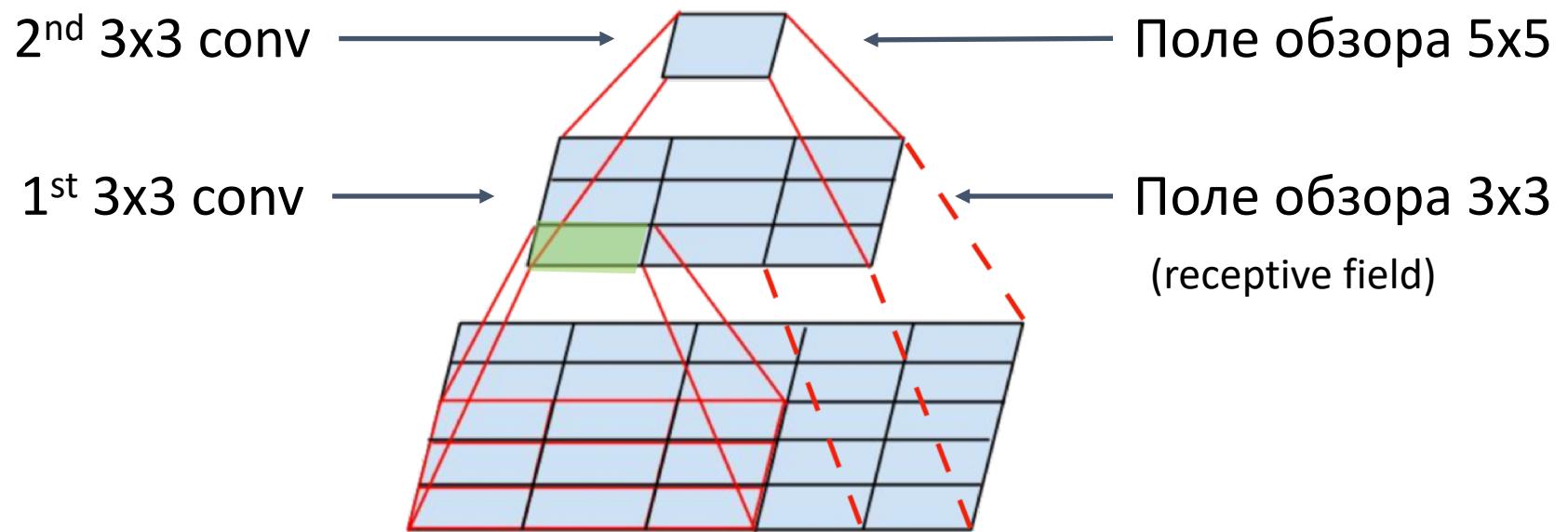
Сколько параметров?

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

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

Да и одного сверточного слоя мало!

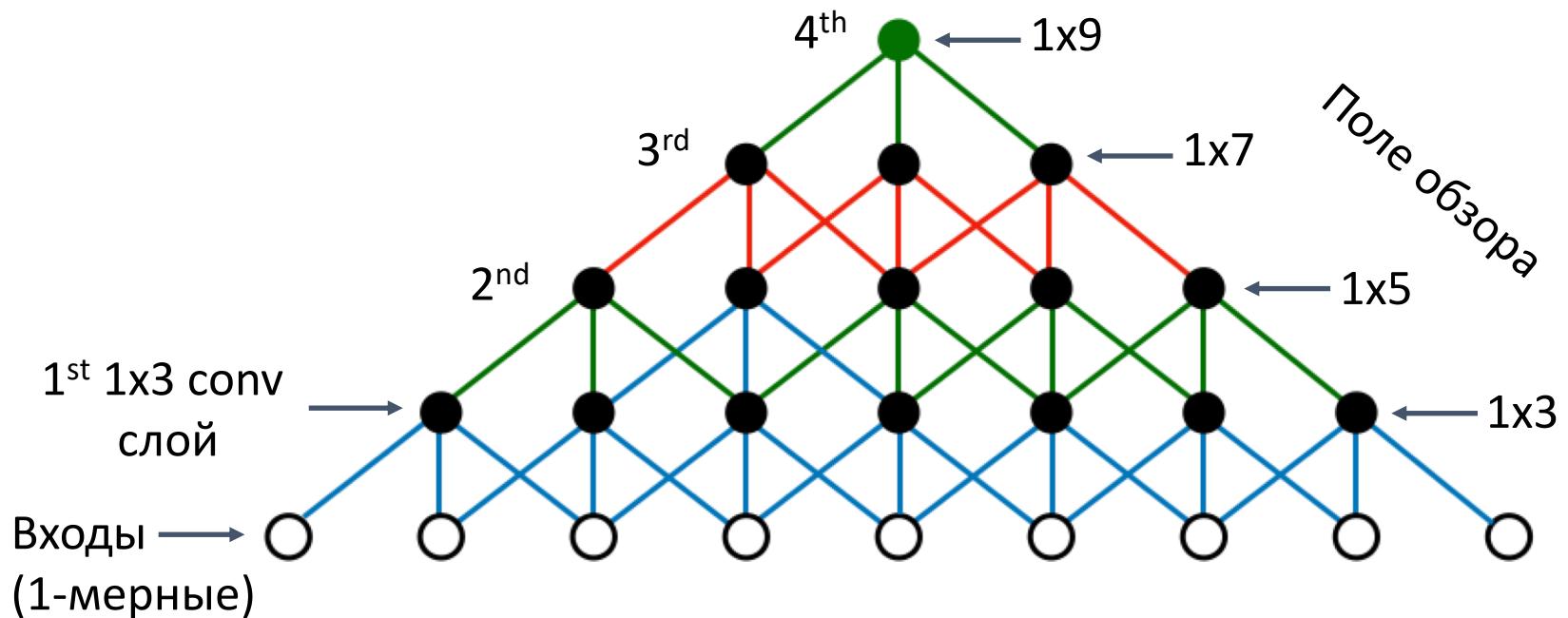
- Нейроны первого слоя смотрят на кусочек 3×3 .
- А что если кошка больше? 😊
- Нужен второй слой!



Сколько слоев нужно для поля обзора 300×300 ?

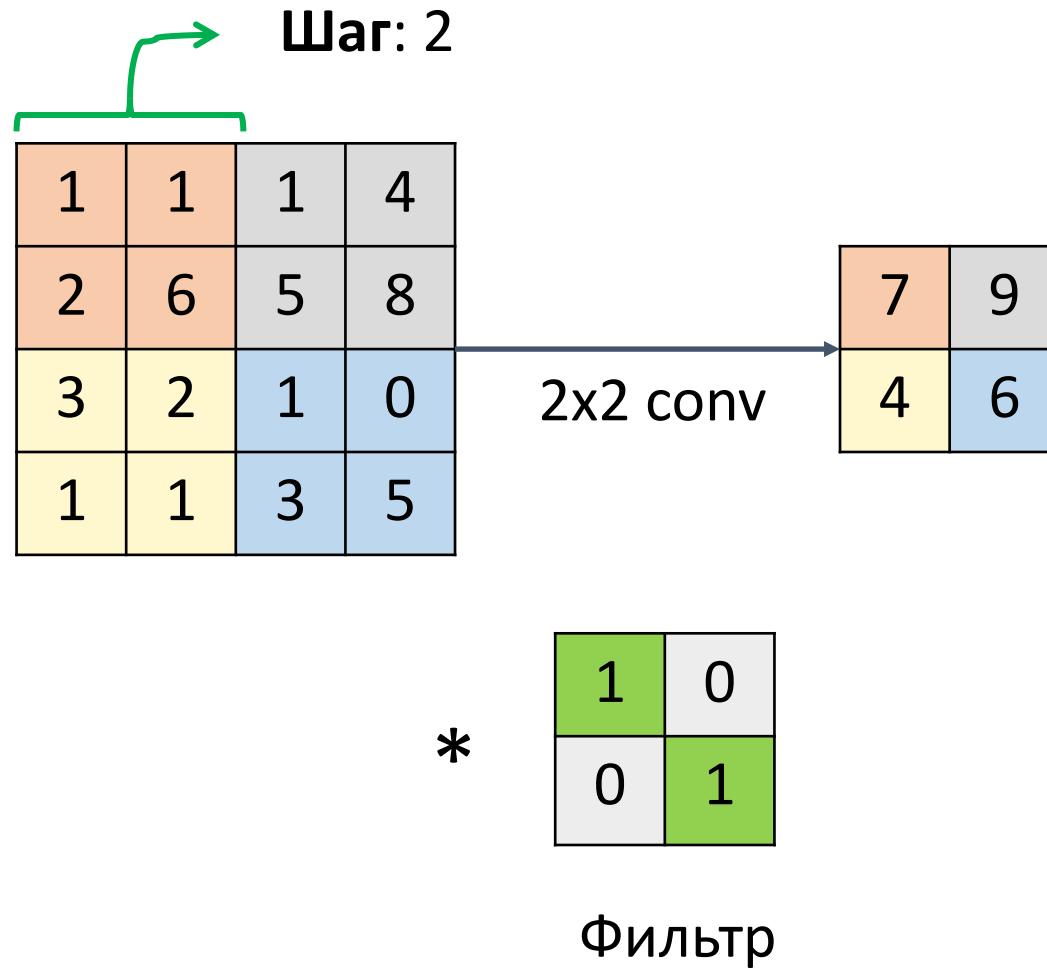
N сверточных слоев

- 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!



Давайте вспомним инвариантность к сдвигу!

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

Вход

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

Вход

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

Фильтр

Выход

Max = 2

Не изменился

Max = 2

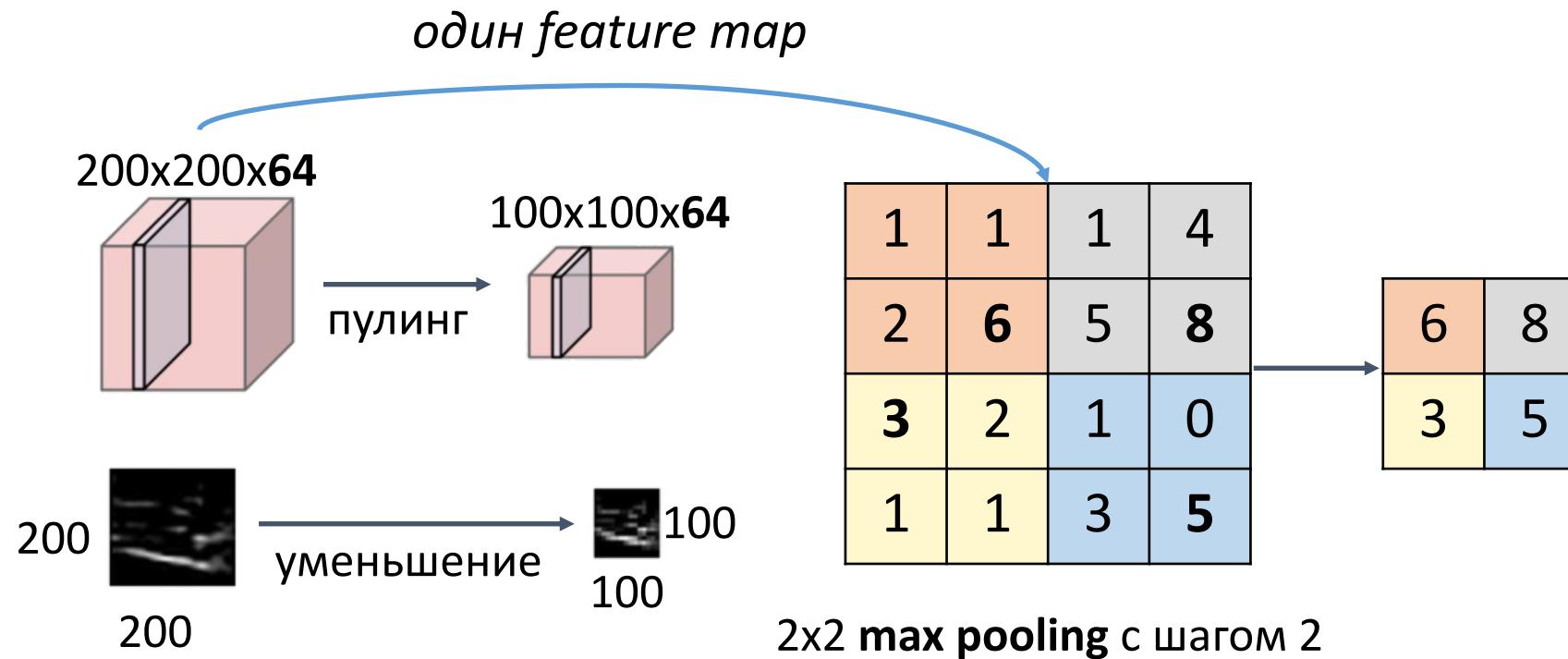
$$\begin{array}{c}
 \begin{array}{|c|c|} \hline
 1 & 0 \\ \hline
 0 & 1 \\ \hline
 \end{array} \quad = \quad
 \begin{array}{|c|c|c|} \hline
 2 & 0 & 0 \\ \hline
 0 & 1 & 0 \\ \hline
 0 & 0 & 0 \\ \hline
 \end{array}
 \end{array}$$

Фильтр

Выход

Пулинг слой

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



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

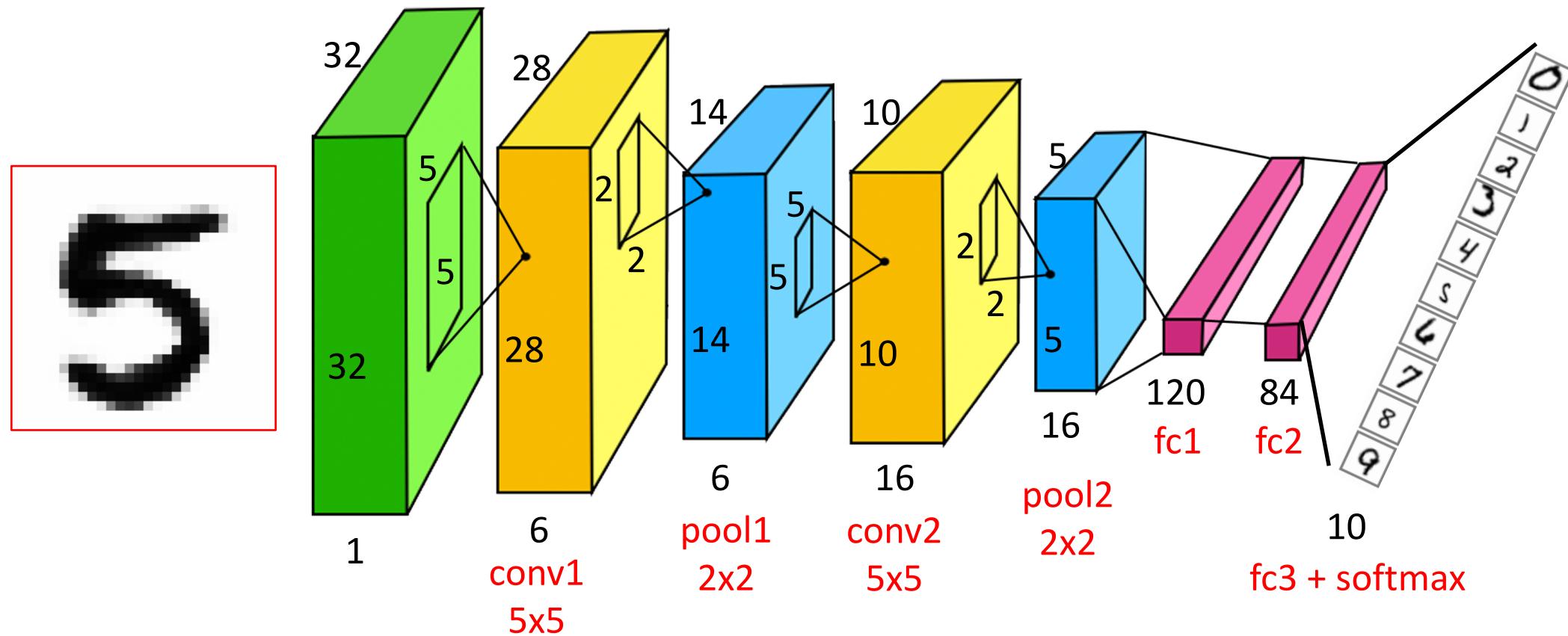
6	9
3	5

Maximum = 9

- Для максимального входа градиент 1.

Соберем это все в сверточную сеть

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



Softmax

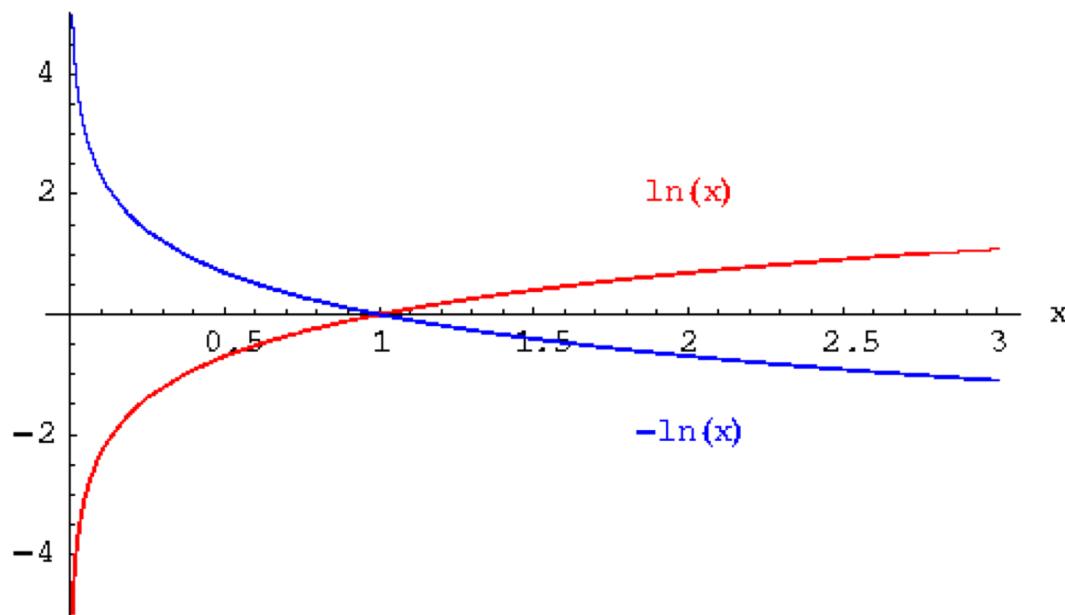
- Softmax — это обобщение логистической функции для многомерного случая.
- Преобразует вектор значений в такие, что каждое из них на отрезке [0,1] и в сумме они дают 1:

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

Log loss (cross-entropy)

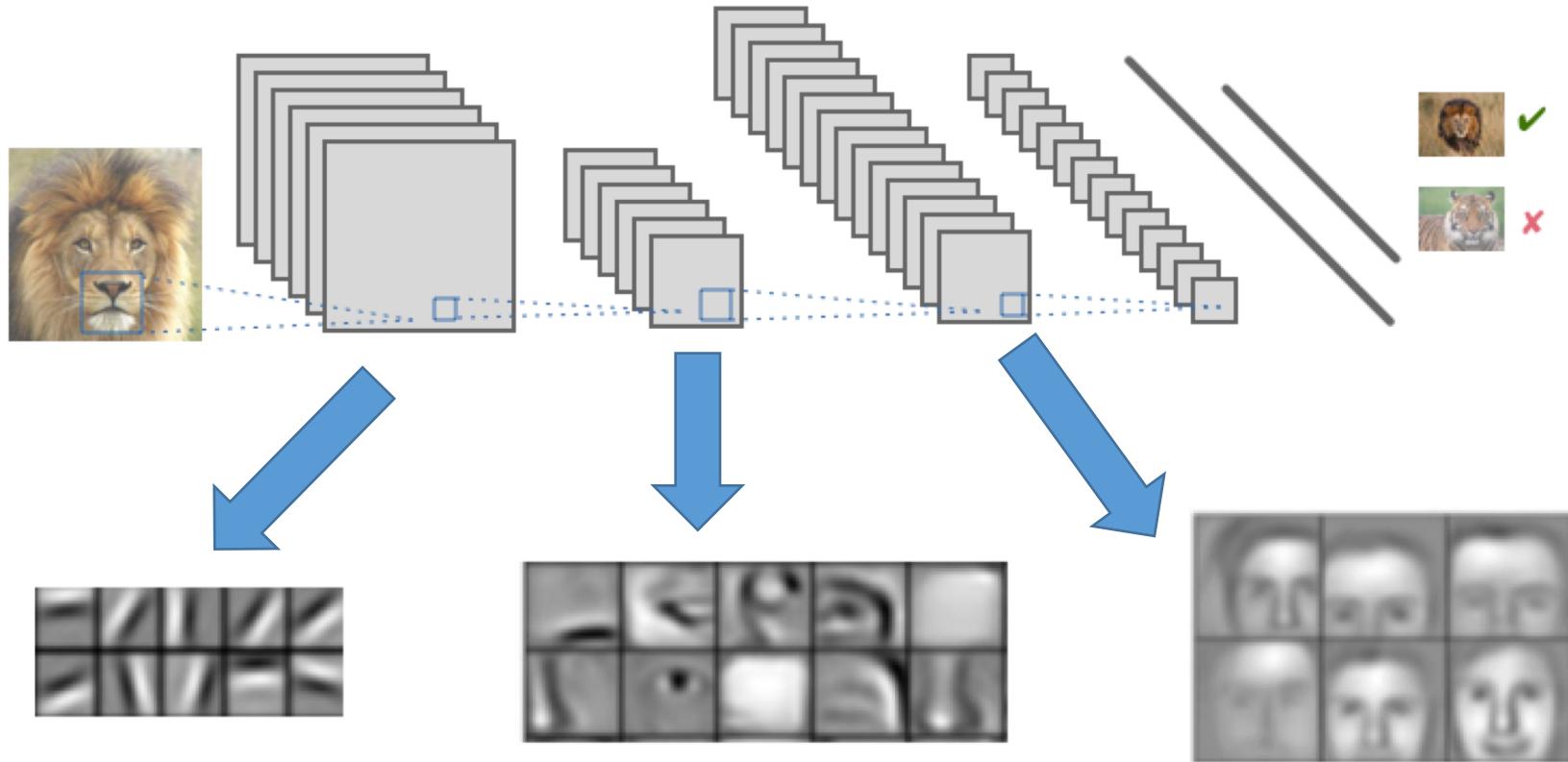
- Потери для классификации на K классов:

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



Нейросеть учит иерархические шаблоны

Глубокая нейронная сеть:

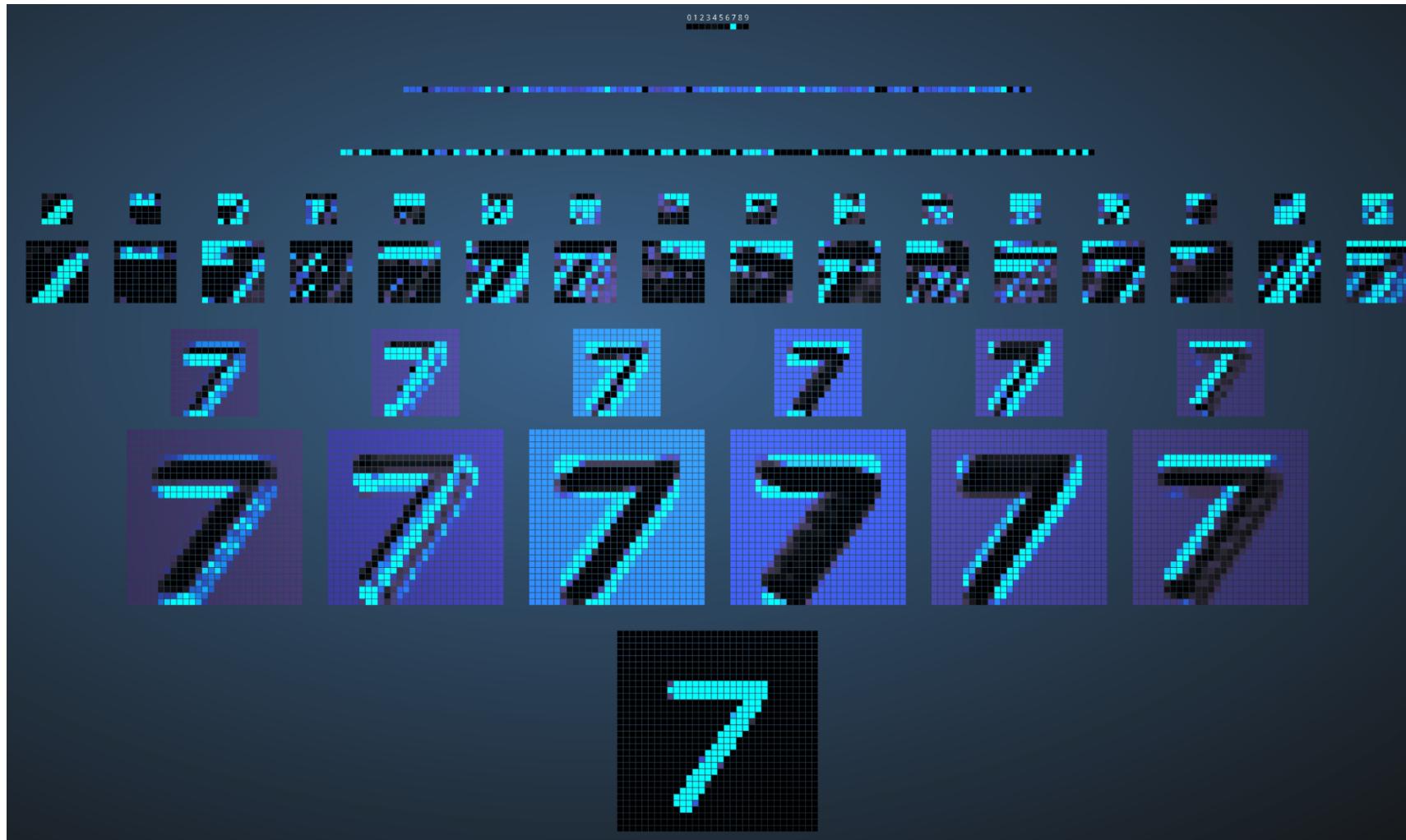


Входы, на
которые
реагируют
нейроны

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>

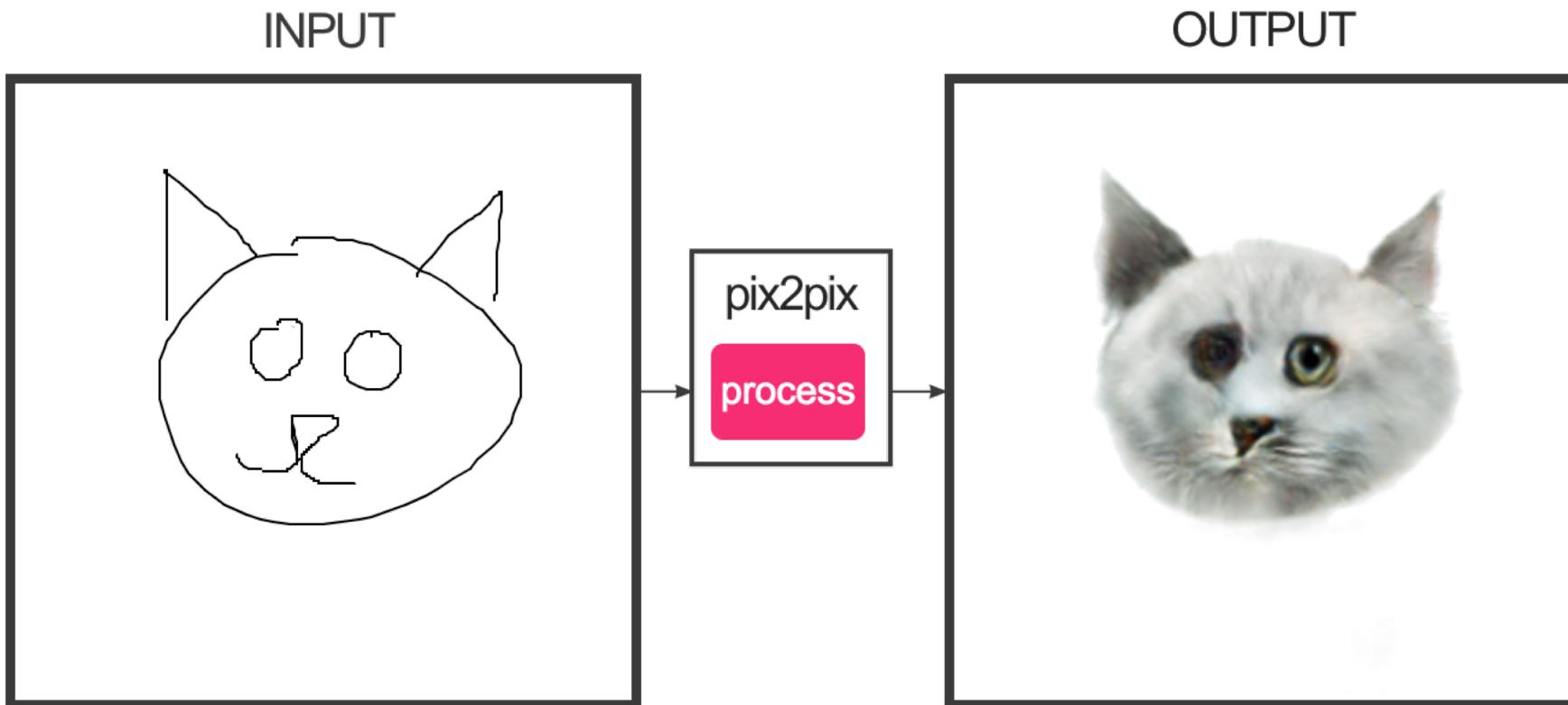


Приложение: image to image

- <https://affinelayer.com/pixsrv/>

Приложение: image to image

- <https://affinelayer.com/pixsrv/>

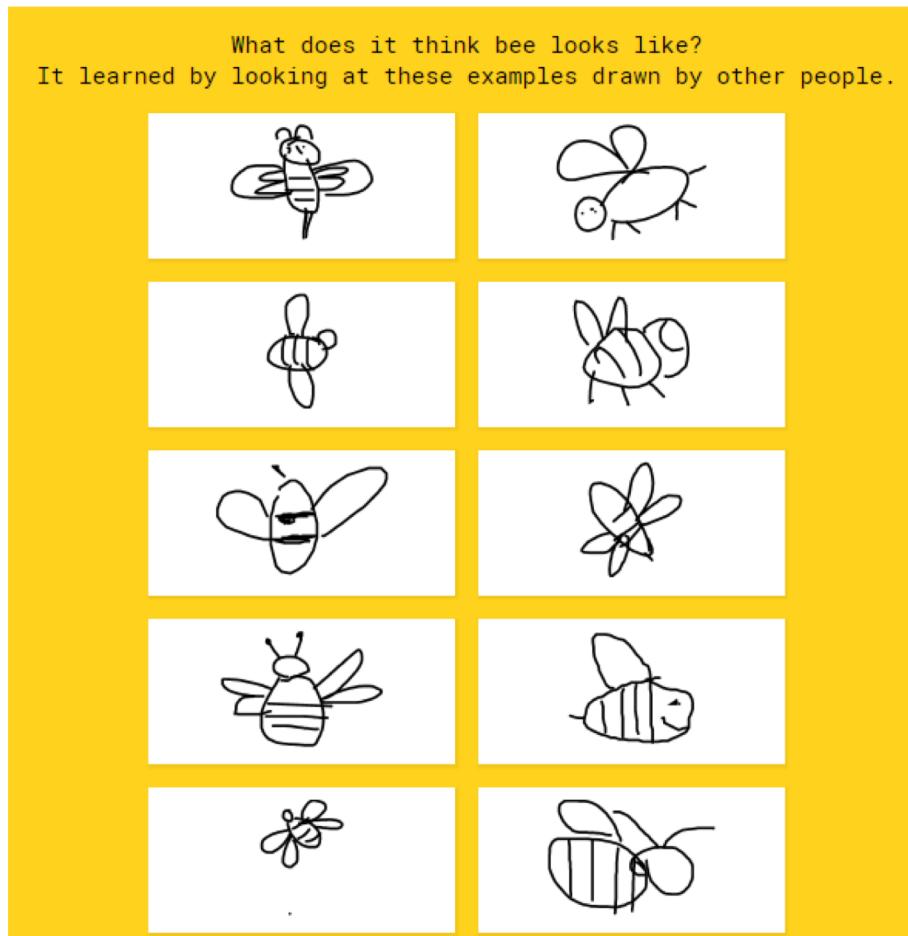


Приложение: рисунки от руки

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

Приложение: 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://brohrer.github.io/how_convolutional_neural_networks_work.html)
- <https://blog.keras.io/how-convolutional-neural-networks-see-the-world.html>