Основы глубинного обучения

Лекция 2

Обратное распространение ошибки. Свёрточные сети.

Евгений Соколов

esokolov@hse.ru

НИУ ВШЭ, 2020

Обучение нейронных сетей

Опрос

Что из этого — формула для шага в градиентном спуске?

1. $w^t = w^{t-1} + \eta \, \nabla Q(w^t)$
2. $w^t = w^{t-1} - \eta \, \nabla Q(w^{t-1})$
3. $w^t = w^{t-1} - \eta \, \nabla Q(w^t)$
4. $w^t = w^{t-1} + \eta \, \nabla Q(w^0)$

Градиентный спуск

- Повторять до сходимости:

$$w^t = w^{t-1} - \eta \, \nabla Q(w^{t-1})$$

Новая точка

Размер шага

Градиент в предыдущей точке

Сходимость

- Останавливаем процесс, если

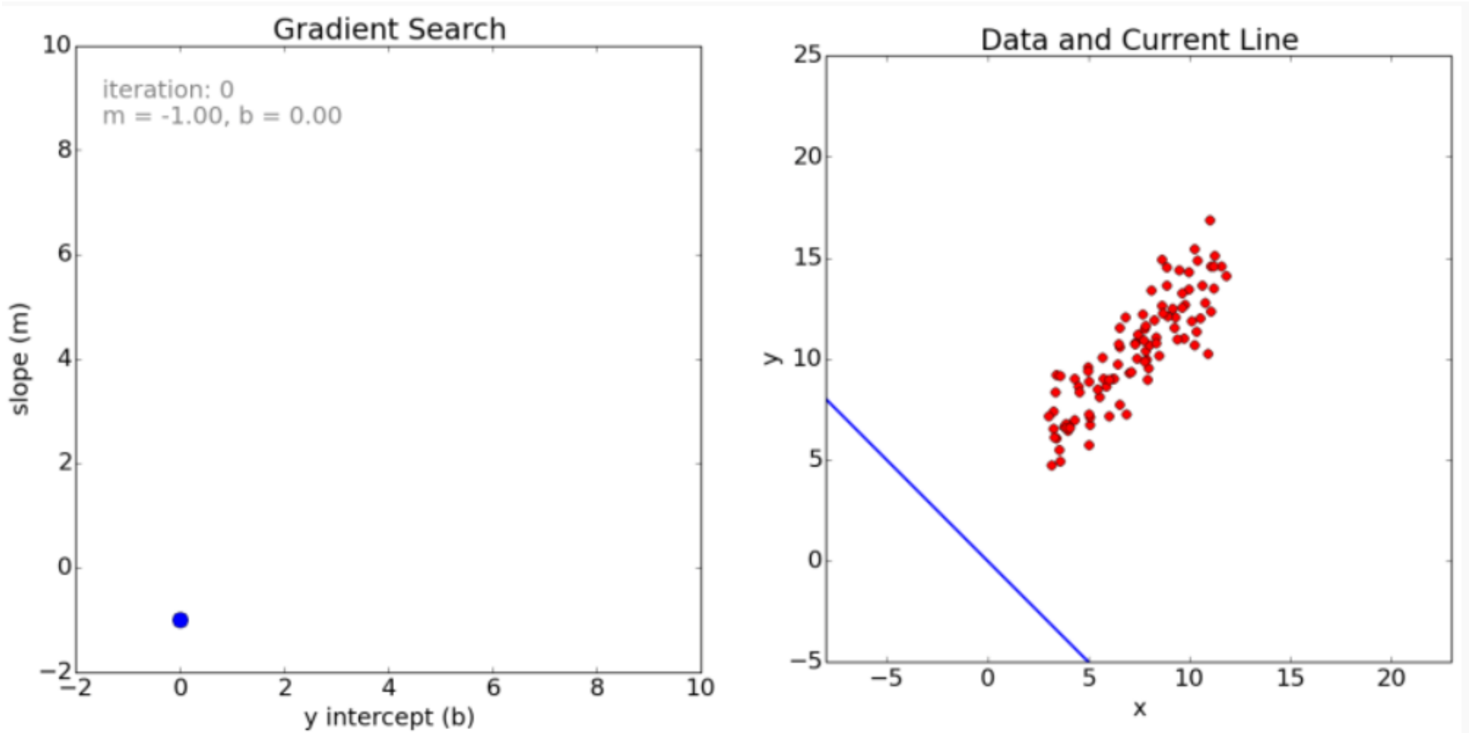
$$\|w^t - w^{t-1}\| < \varepsilon$$

- Другой вариант:

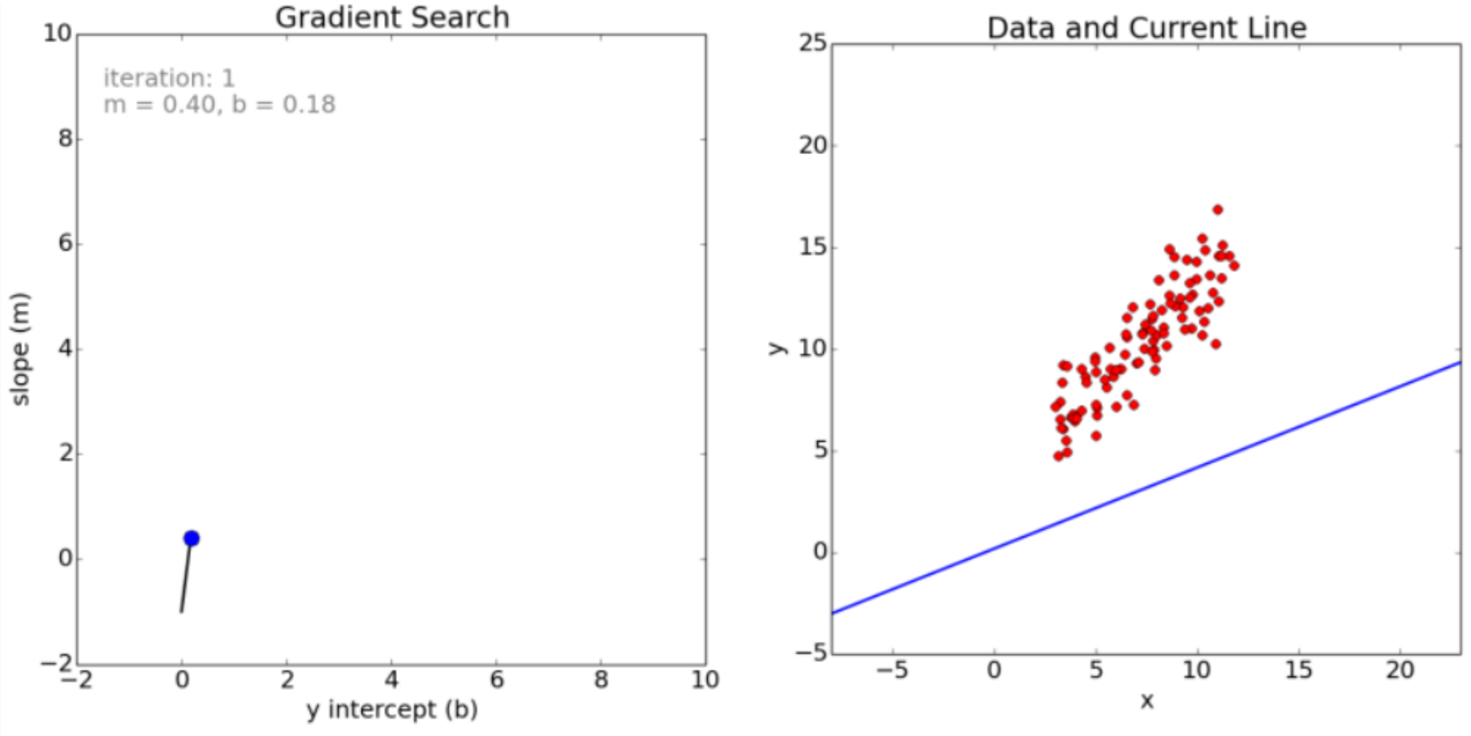
$$\|\nabla Q(w^t)\| < \varepsilon$$

- Обычно в глубинном обучении: останавливаемся, когда ошибка на тестовой выборке начинает расти

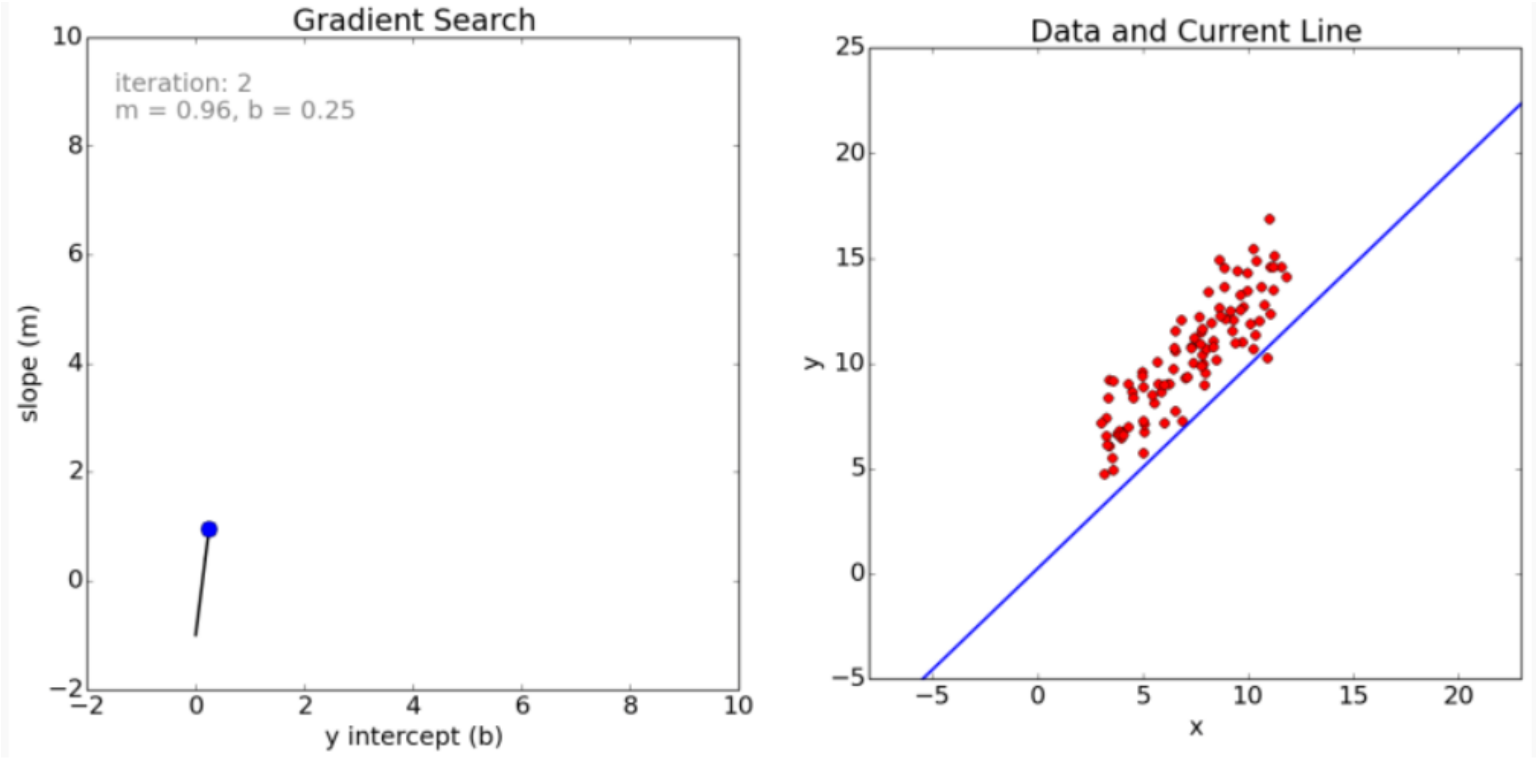
Парная регрессия



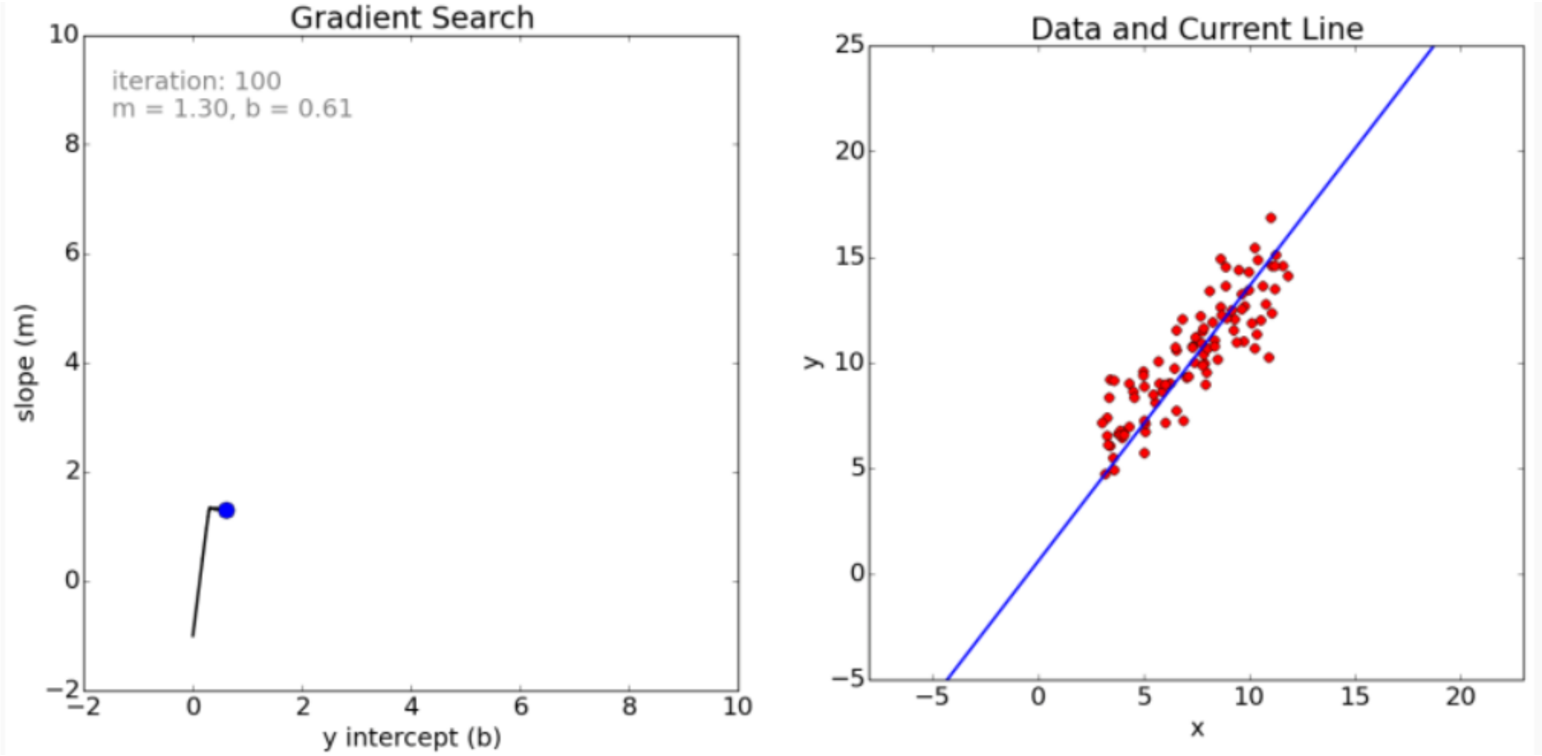
Парная регрессия



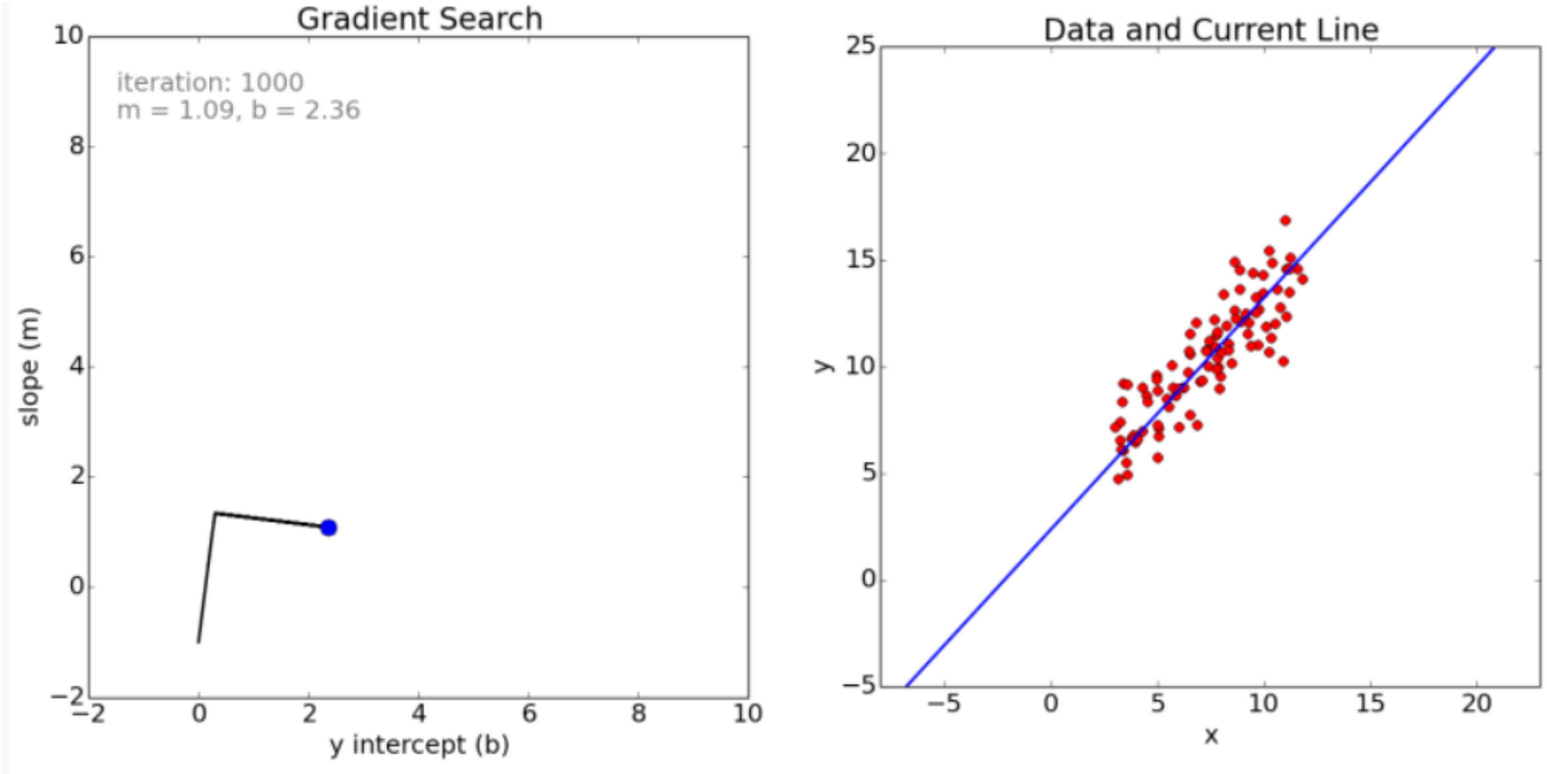
Парная регрессия



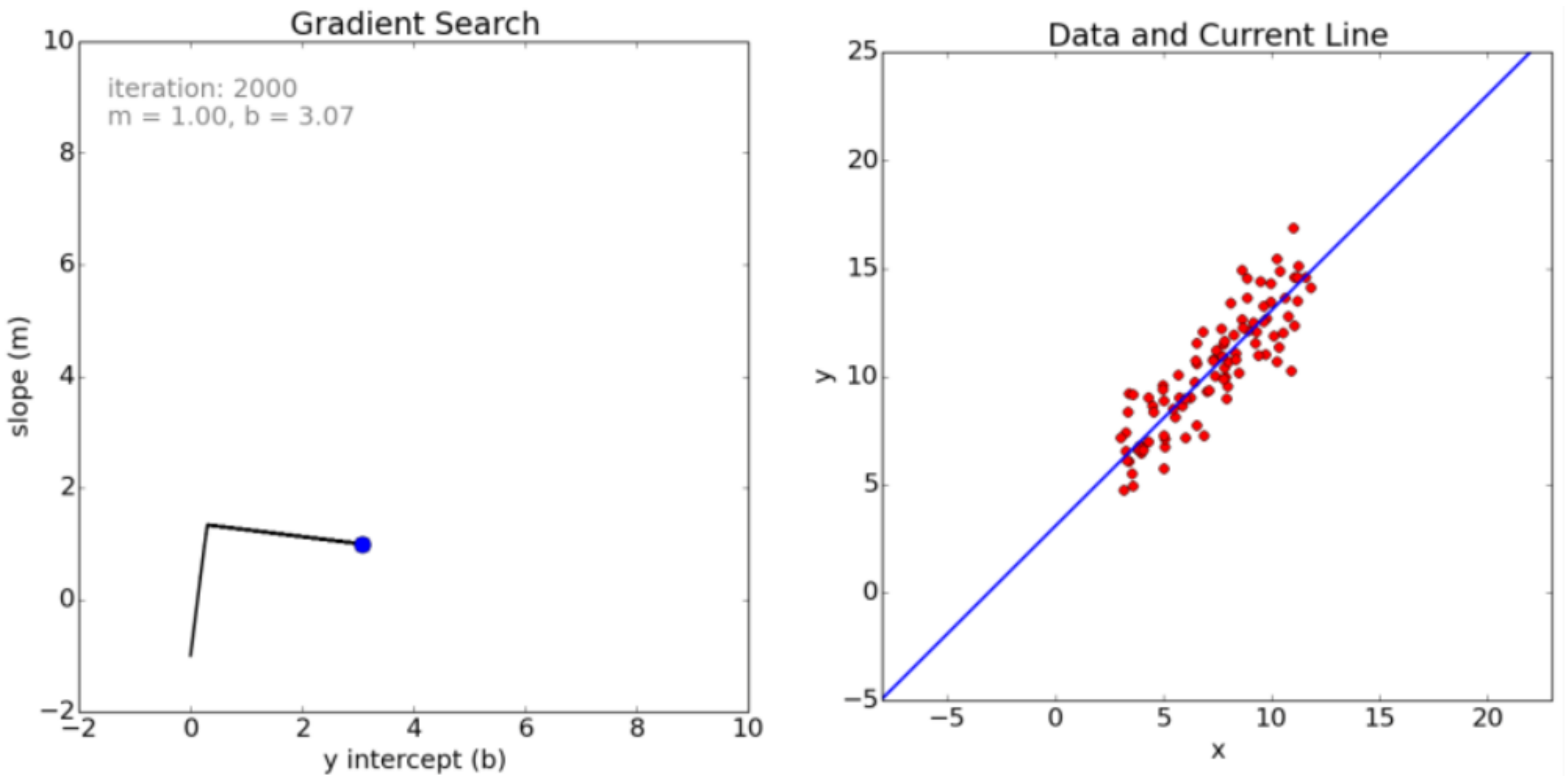
Парная регрессия



Парная регрессия



Парная регрессия



Градиентный спуск

1. Начальное приближение: w^0

2. Повторять:

$$w^t = w^{t-1} - \eta \nabla Q(w^{t-1})$$

3. Останавливаемся, если

$$\|w^t - w^{t-1}\| < \varepsilon$$

Обучение нейронных сетей

- Все слои обычно дифференцируемы, поэтому можно посчитать производные по всем параметрам

$$x \longrightarrow \mathrm{FC}_1 \longrightarrow f \longrightarrow \mathrm{FC}_2 \longrightarrow a(x)$$

- $a(x) = FC_2\left(f\left(FC_1(x)\right)\right)$
- Где здесь параметры?

Обучение нейронных сетей

- Все слои обычно дифференцируемы, поэтому можно посчитать производные по всем параметрам

$$x \to \mathrm{FC}_1 \to f \to \mathrm{FC}_2 \to a(x)$$

- $a(x) = FC_2\left(f\left(FC_1(x)\right)\right)$
- Где здесь параметры?

Обучение нейронных сетей

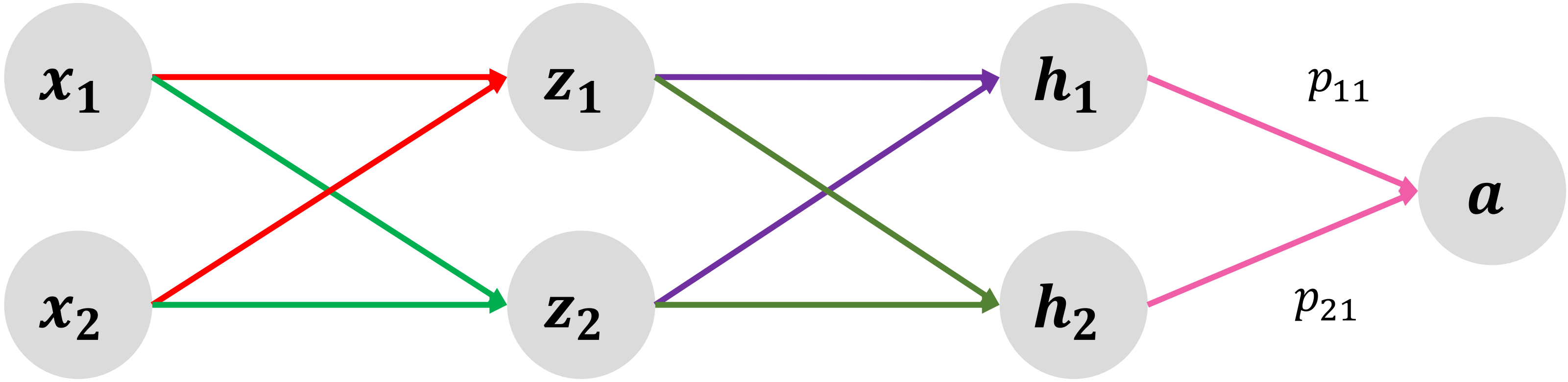
- Все слои обычно дифференцируемы, поэтому можно посчитать производные по всем параметрам

$$\boxed{x} \to \boxed{\mathrm{FC}_1} \to \boxed{f} \to \boxed{\mathrm{FC}_2} \to \boxed{a(x)}$$

- $a(x) = FC_2\Big(f\big(FC_1(x)\big)\Big)$

$$\frac{1}{\ell}\sum_{i=1}^{\ell} L\big(y_i, a(x_i)\big) \to \min_{\mathrm{a}}$$

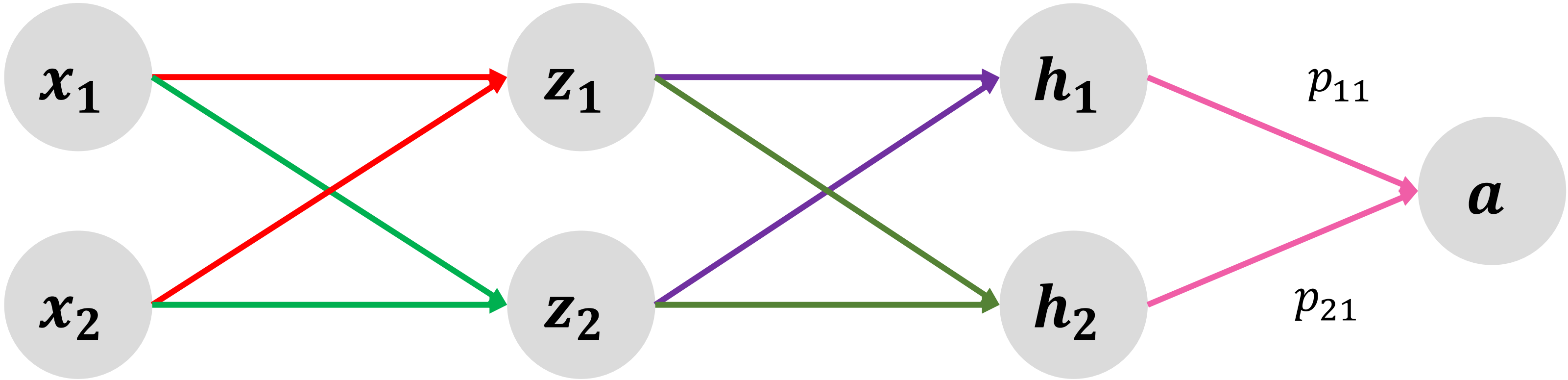
Как считать производные?



$$a(x) = p_{11}h_1(x) + p_{21}h_2(x)$$

$$\frac{\partial a}{\partial p_{11}} = ?$$

Как считать производные?

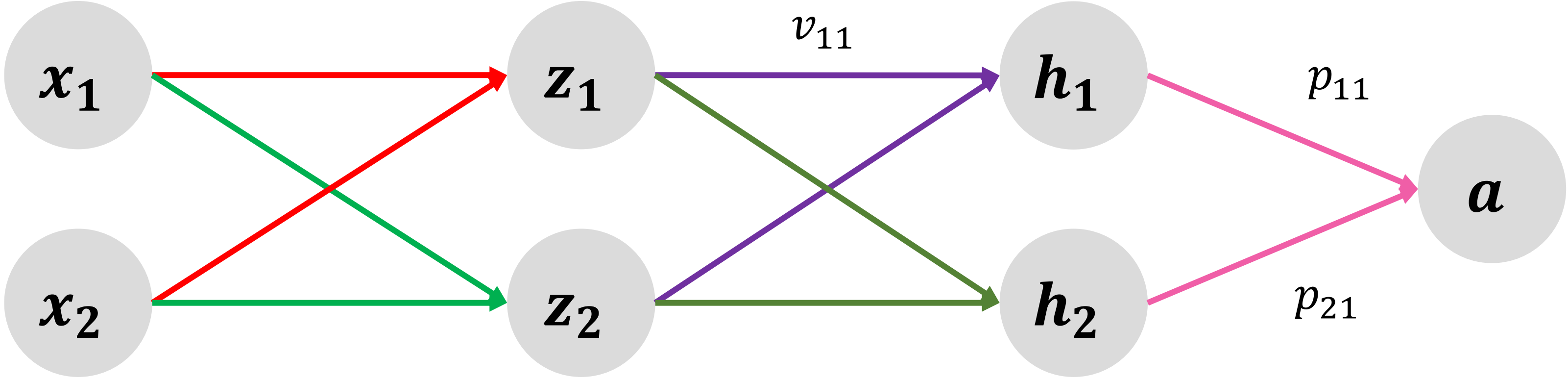


$$a(x) = p_{11}h_1(x) + p_{21}h_2(x)$$

$$\frac{\partial a}{\partial p_{11}} = h_1(x)$$

- Чем больше $h_1(x)$, тем сильнее p_{11} влияет на a

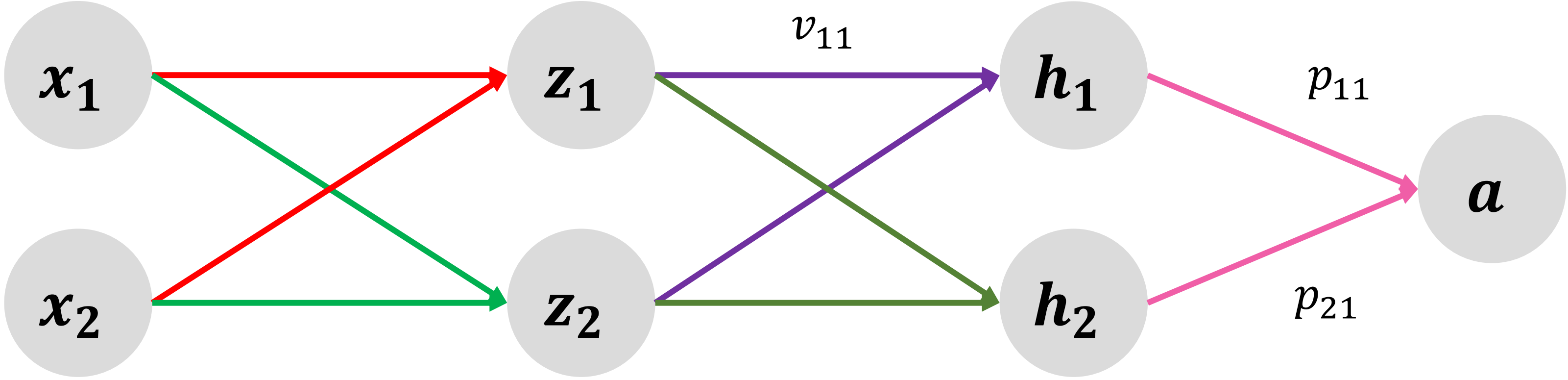
Как считать производные?



$$a(x) = p_{11} f\big(v_{11} z_1(x) + v_{21} z_2(x)\big) + p_{21} h_2(x)$$

$$\frac{\partial a}{\partial v_{11}} = ?$$

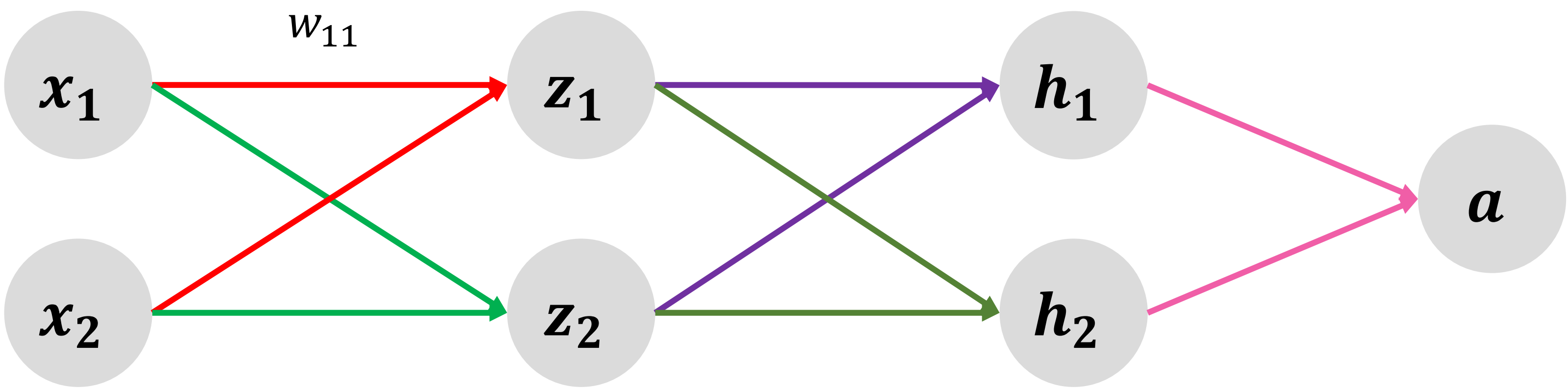
Как считать производные?



$$a(x) = p_{11} f\big(v_{11} z_1(x) + v_{21} z_2(x)\big) + p_{21} h_2(x)$$

$$\frac{\partial a}{\partial v_{11}} = \frac{\partial a}{\partial h_1} \frac{\partial h_1}{\partial v_{11}}$$

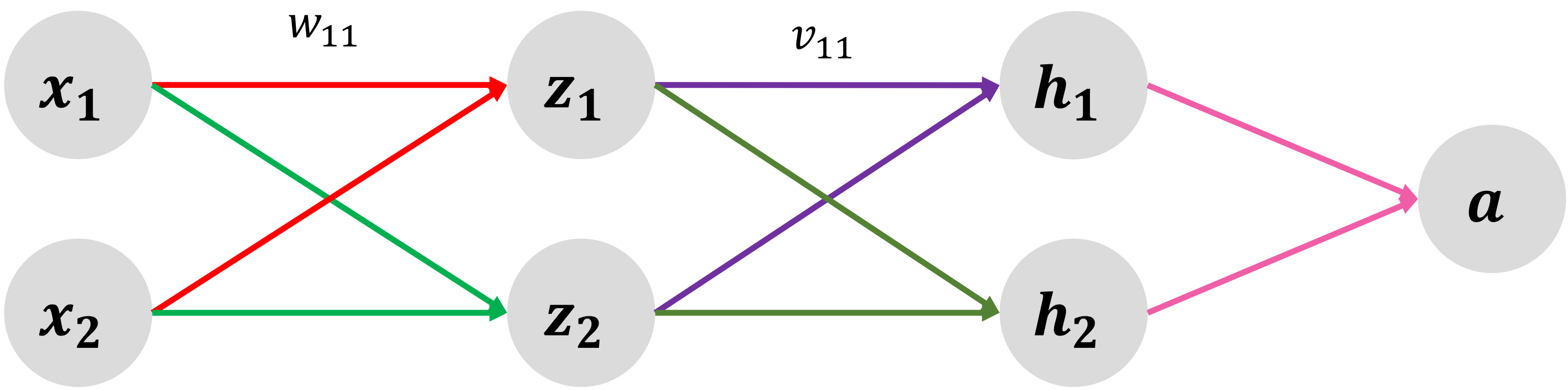
Как считать производные?



$$\frac{\partial a}{\partial w_{11}} = ?$$

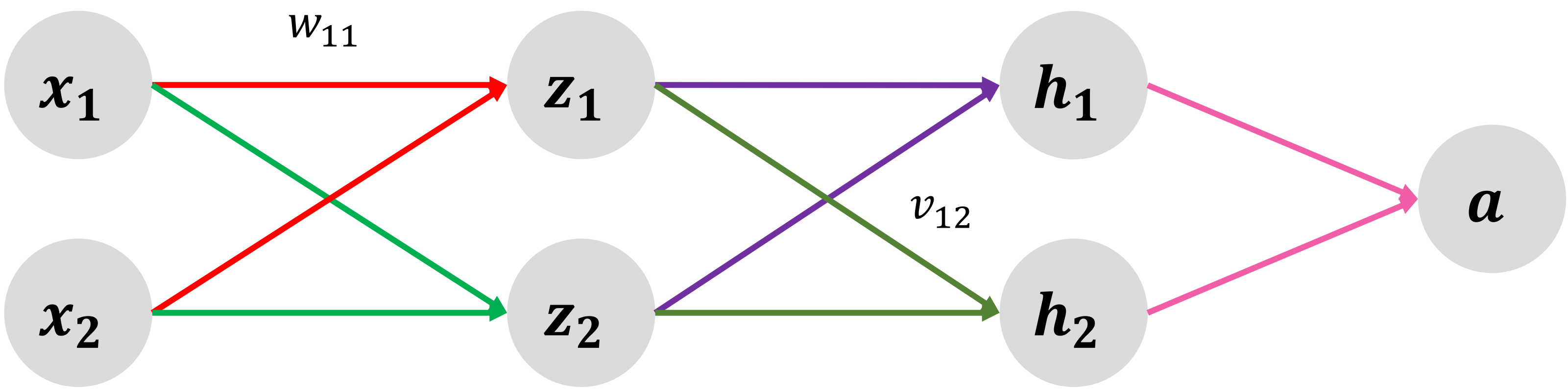
- Показывает, как сильно изменится a при изменении w_{11}

Как считать производные?



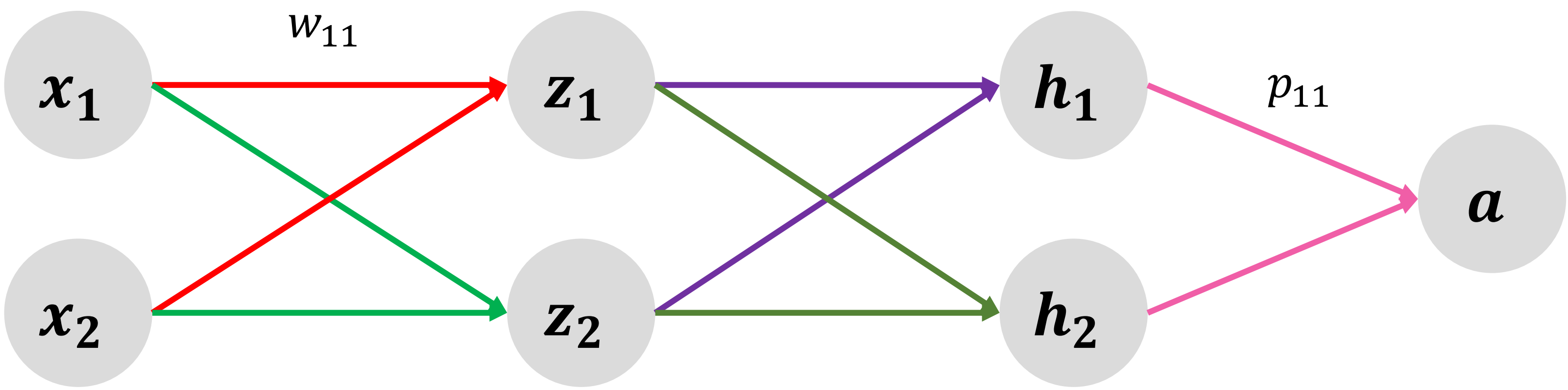
- Как сильно изменится a при изменении w_{11}?
- Влияет ли на это v_{11}?

Как считать производные?



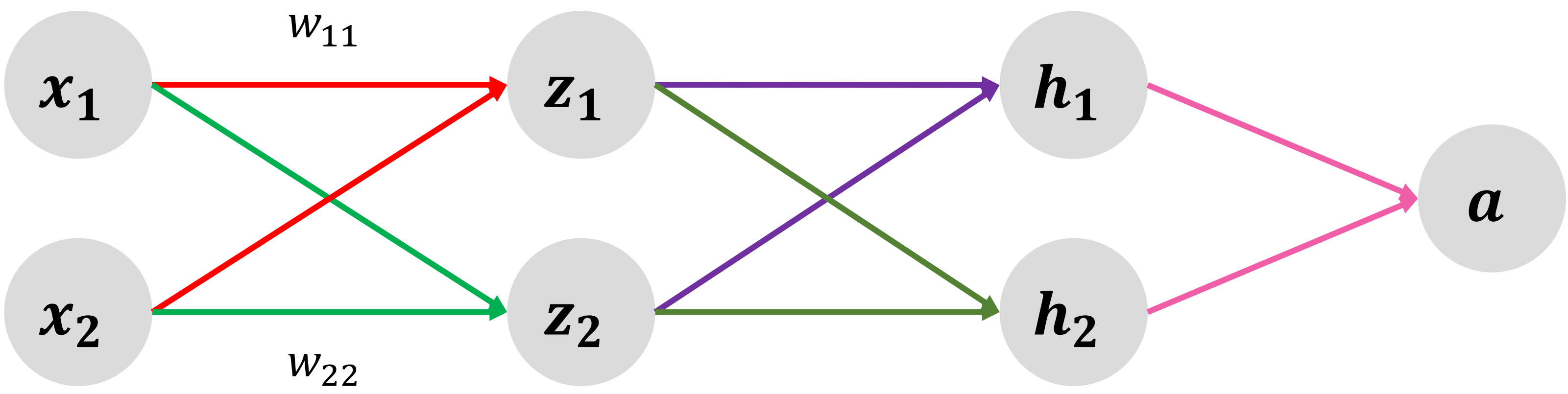
- Как сильно изменится a при изменении w_{11}?
- Влияет ли на это v_{12}?

Как считать производные?



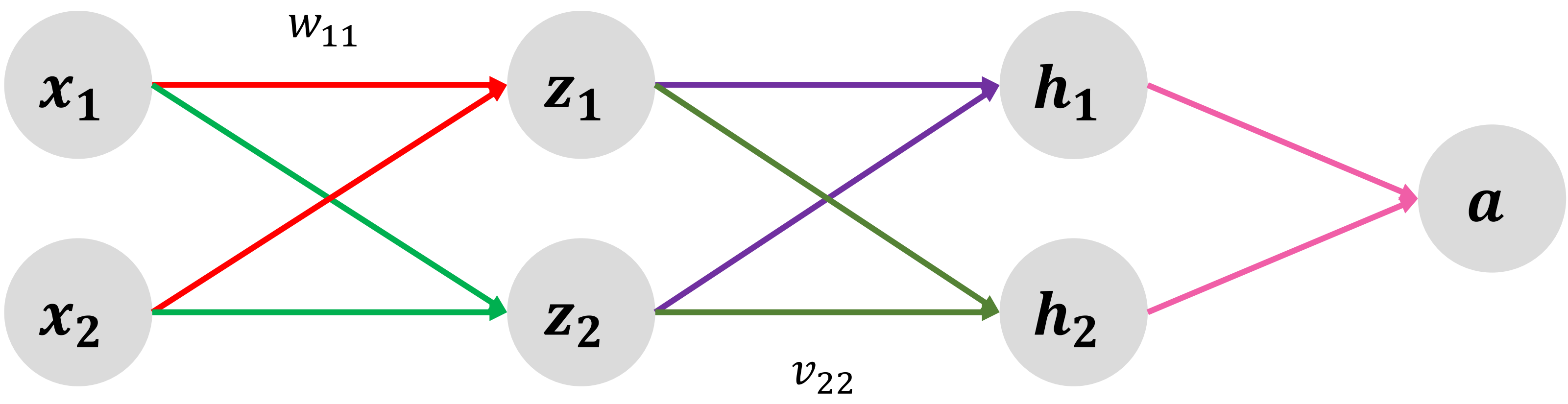
- Как сильно изменится a при изменении w_{11}?
- Влияет ли на это p_{11}?

Как считать производные?



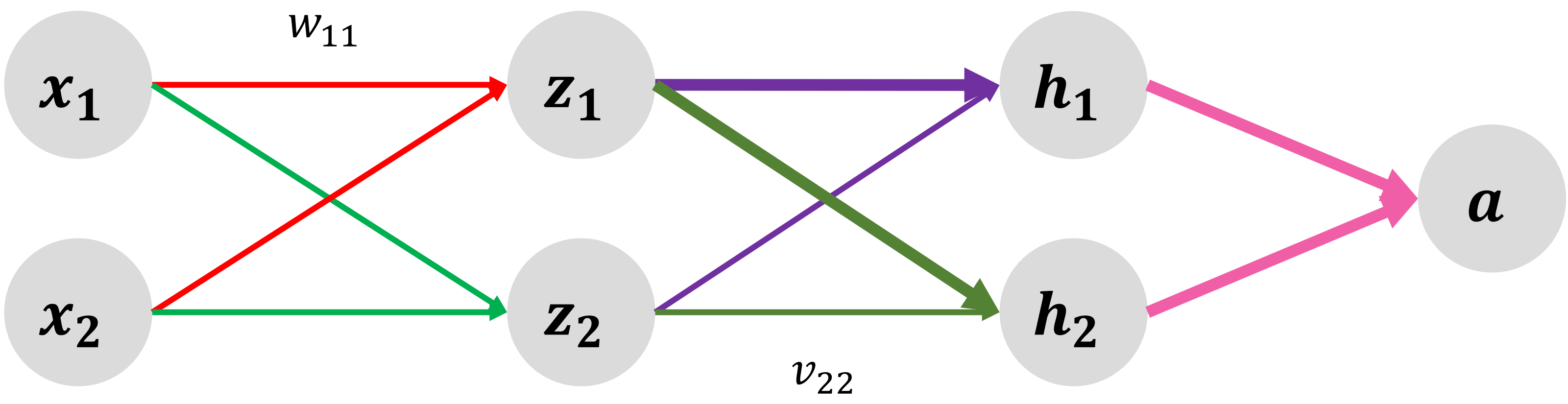
- Как сильно изменится a при изменении w_{11}?
- Влияет ли на это w_{22}?

Как считать производные?



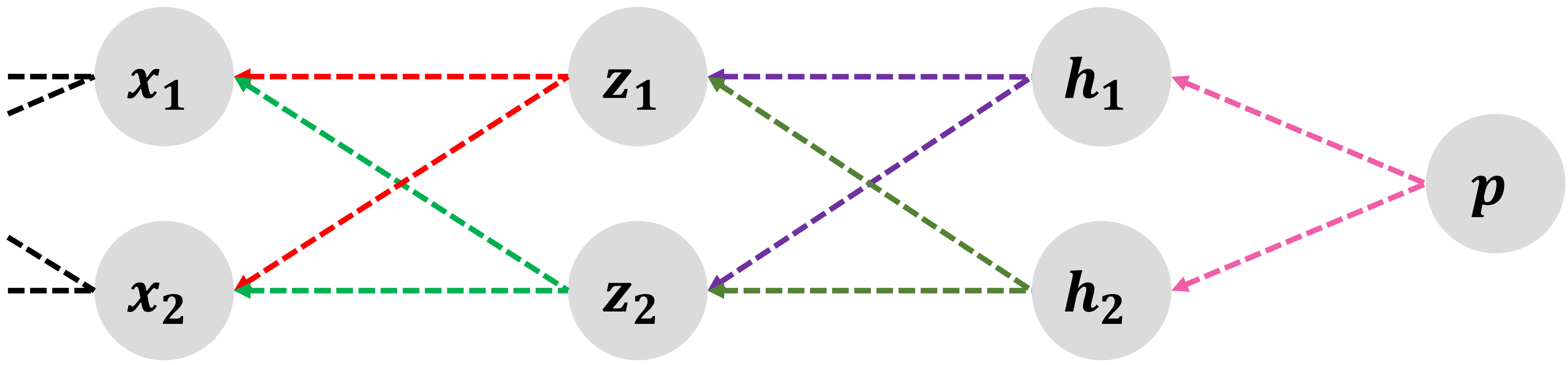
- Как сильно изменится a при изменении w_{11}?
- Влияет ли на это v_{22}?

Как считать производные?



$$\frac{\partial a}{\partial w_{11}} = \frac{\partial a}{\partial h_1}\frac{\partial h_1}{\partial z_1}\frac{\partial z_1}{\partial w_{11}} + \frac{\partial a}{\partial h_2}\frac{\partial h_2}{\partial z_1}\frac{\partial z_1}{\partial w_{11}}$$

Как считать производные?



- Мы как бы идём в обратную сторону по графу и считаем производные
- Метод обратного распространения ошибки (backpropagation)

Резюме

- Полносвязные сети мощные, но сложные в обучении
- Можно удобно считать производные через backprop

Полносвязные сети для изображений

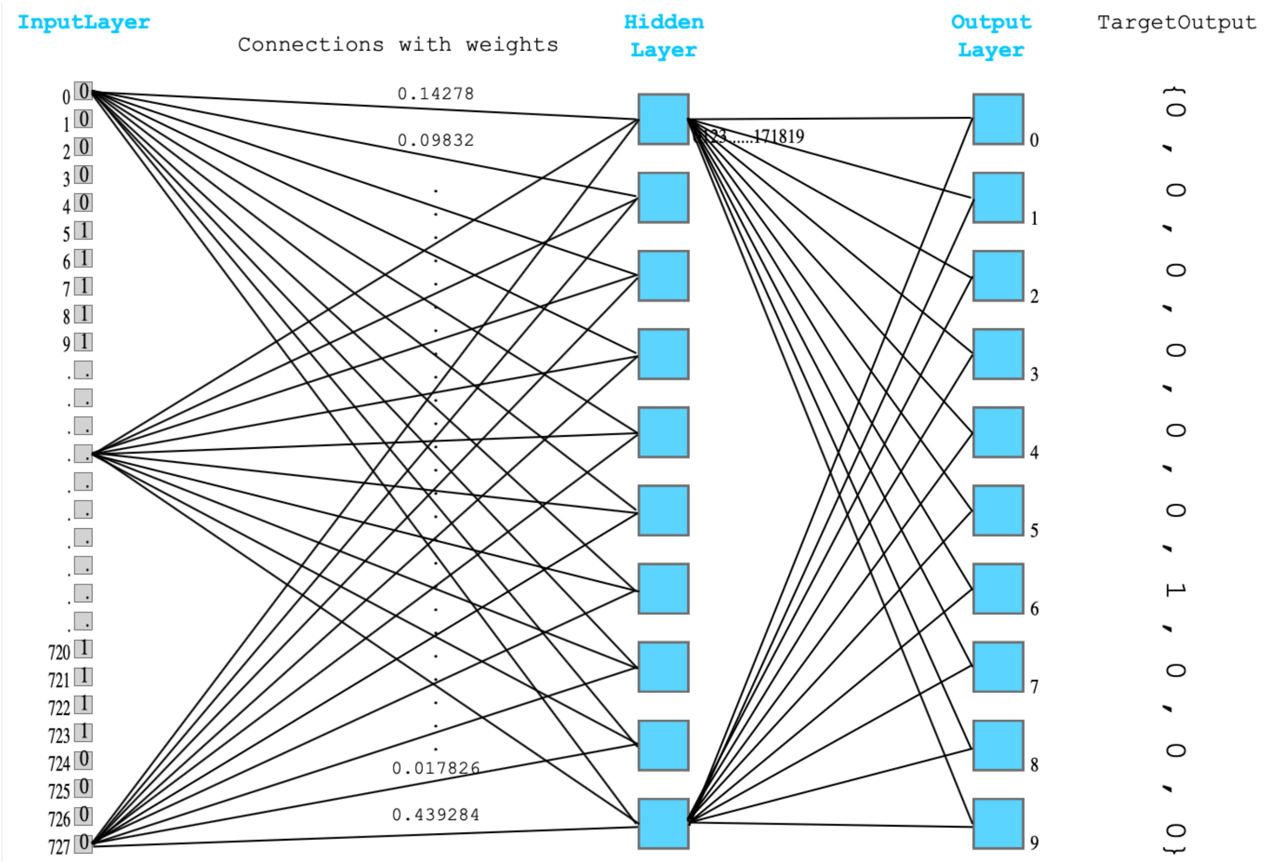
MNIST

MNIST

- Изображения 28 x 28
- Изображения центрированы
- 60.000 объектов в обучающей выборке

MNIST

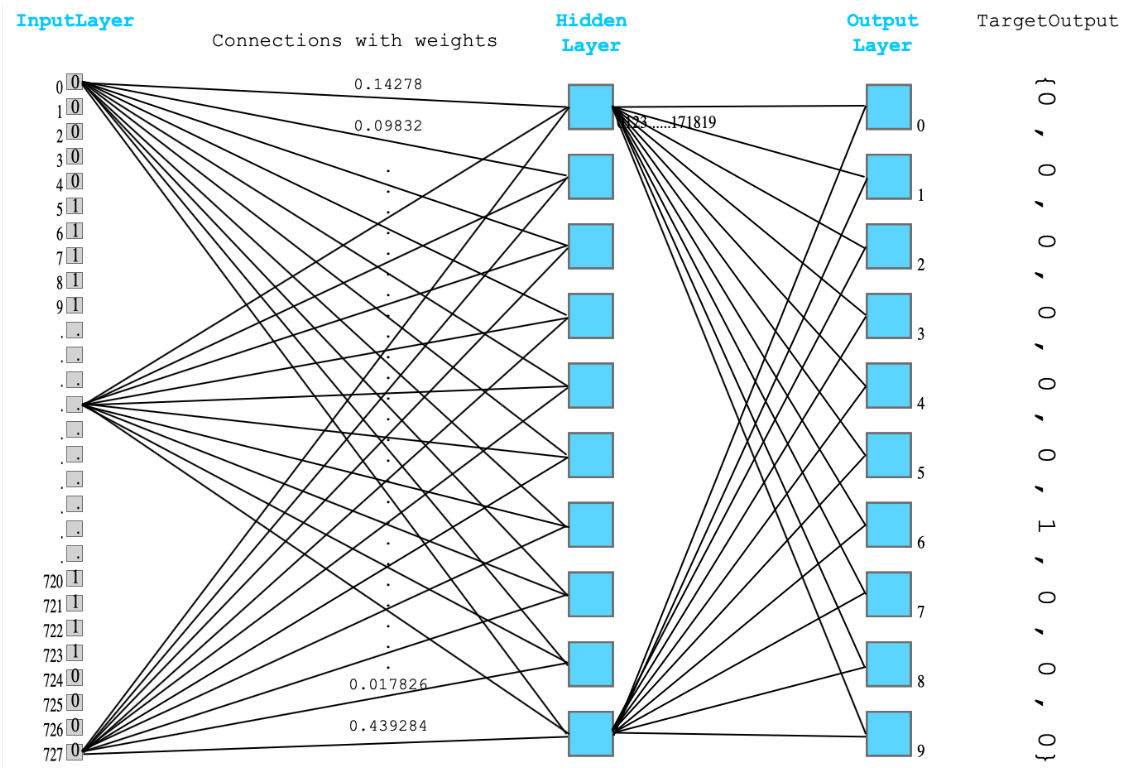
- Что может выучить полносвязная сеть?



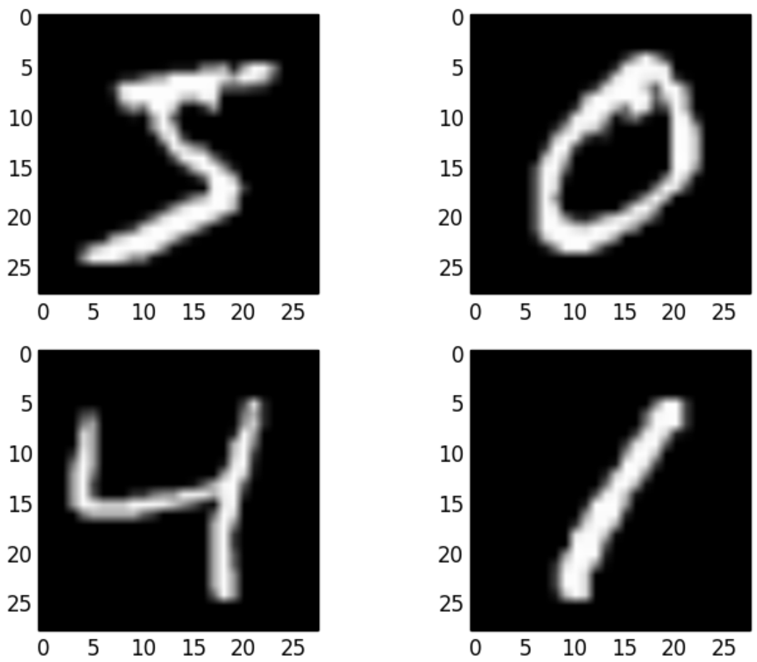
https://mmlind.github.io/Simple_3-Layer_Neural_Network_for_MNIST_Handwriting_Recognition/

MNIST

- Каждый нейрон может детектировать заполненность конкретного набора пикселей

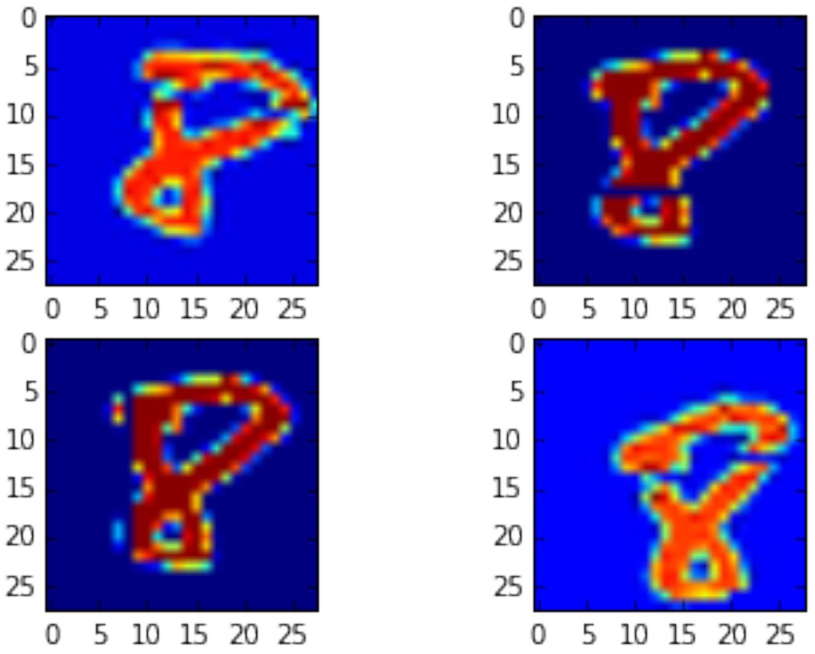


MNIST



MNIST

- Если немного сдвинуть цифру, то нейрон уже не будет на неё реагировать



https://srome.github.io/Jitter,-Convolutional-Neural-Networks,-and-a-Kaggle-Framework/

Число параметров

- 784 входа
- Полносвязный слой: 1000 нейронов
- Выходной слой: 10 нейронов (по одному на каждый класс)
- Весов между входным и полносвязным слоями:

$$(784 + 1)*1000 = 785.000$$

- Весов между полносвязным и выходным слоями:

$$(1000 + 1) * 10 = 10.010$$

Число параметров

- Можно добиться хорошего качества полносвязными сетями (с аугментацией)
- https://arxiv.org/abs/1003.0358

Table 1: Error rates on MNIST test set.

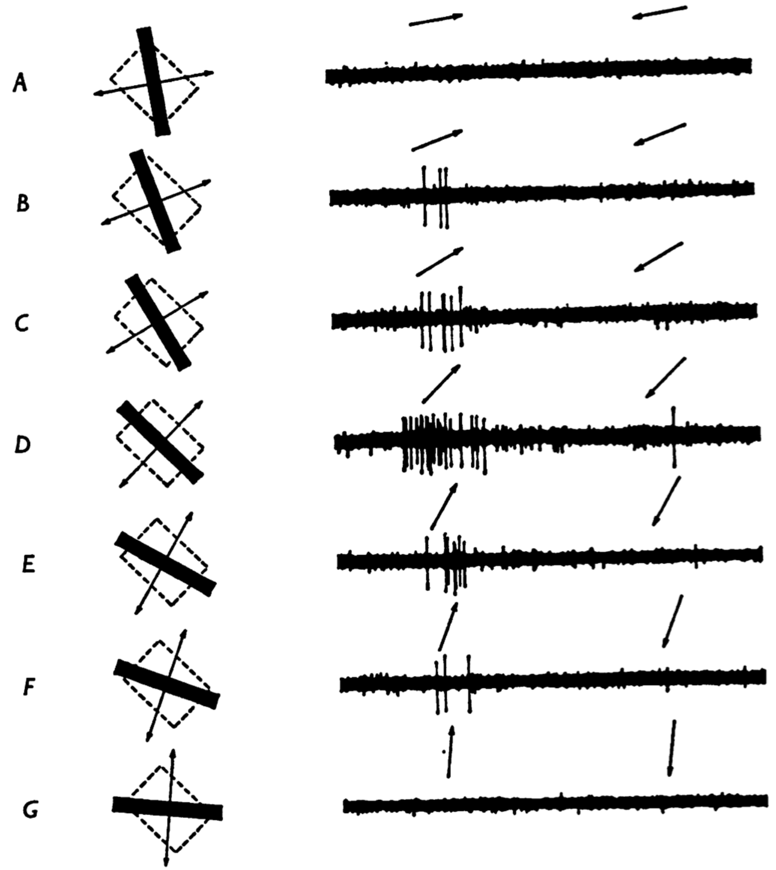
ID	architecture (number of neurons in each layer)	test error for best validation [%]	best test error [%]	simulation time [h]	weights [milions]
1	1000, 500, 10	**0.49**	0.44	23.4	1.34
2	1500, 1000, 500, 10	**0.46**	0.40	44.2	3.26
3	2000, 1500, 1000, 500, 10	**0.41**	0.39	66.7	6.69
4	2500, 2000, 1500, 1000, 500, 10	**0.35**	0.32	114.5	12.11
5	9×1000, 10	**0.44**	0.43	107.7	8.86

Полносвязные слои для изображений

- Очень много параметров
- Легко могут переобучиться
- Не учитывают специфику изображений (сдвиги, небольшие изменения формы и т.д.)
- Один из лучших способов борьбы с переобучением — снижение числа параметров

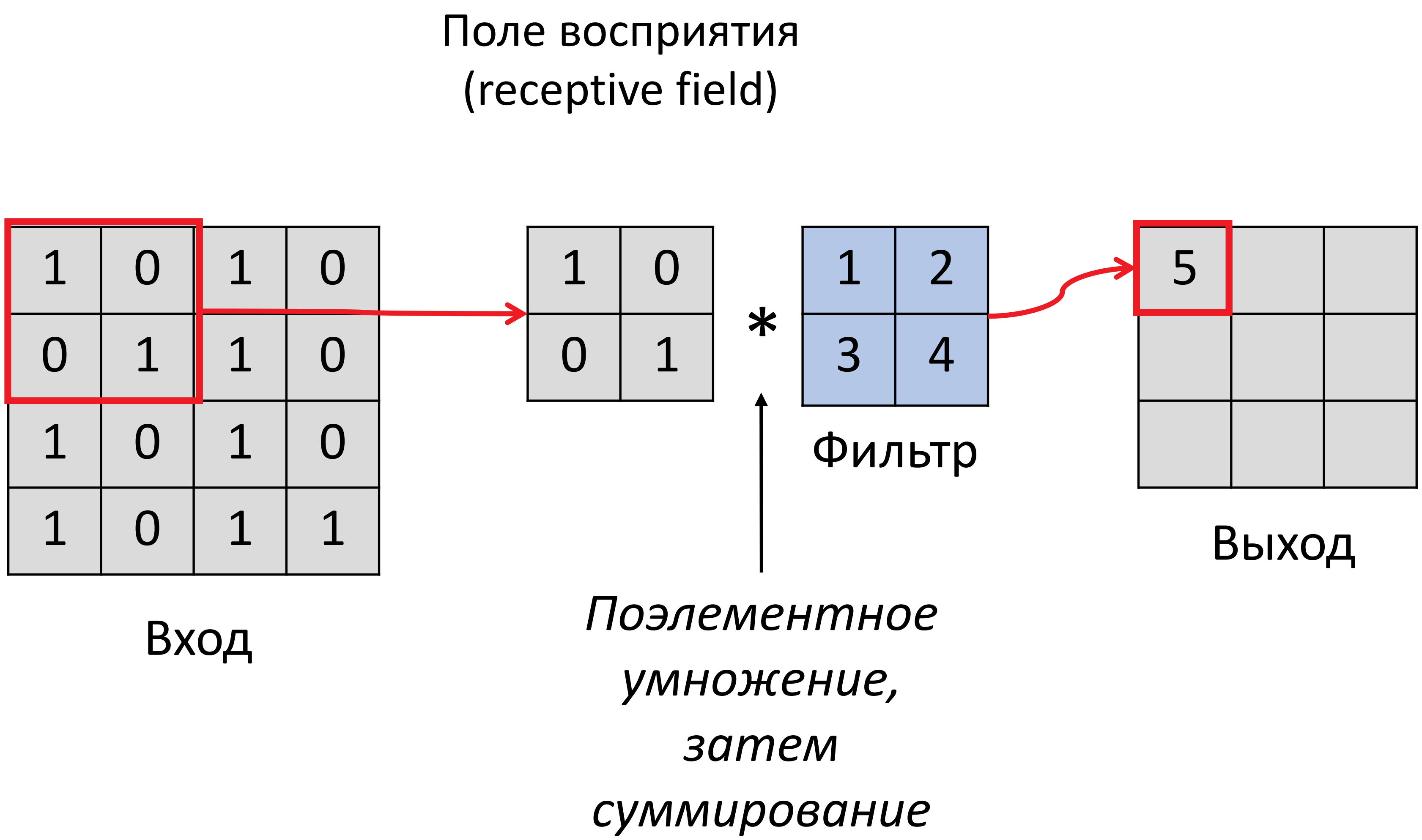
Свёртки

Эксперименты со зрительной корой

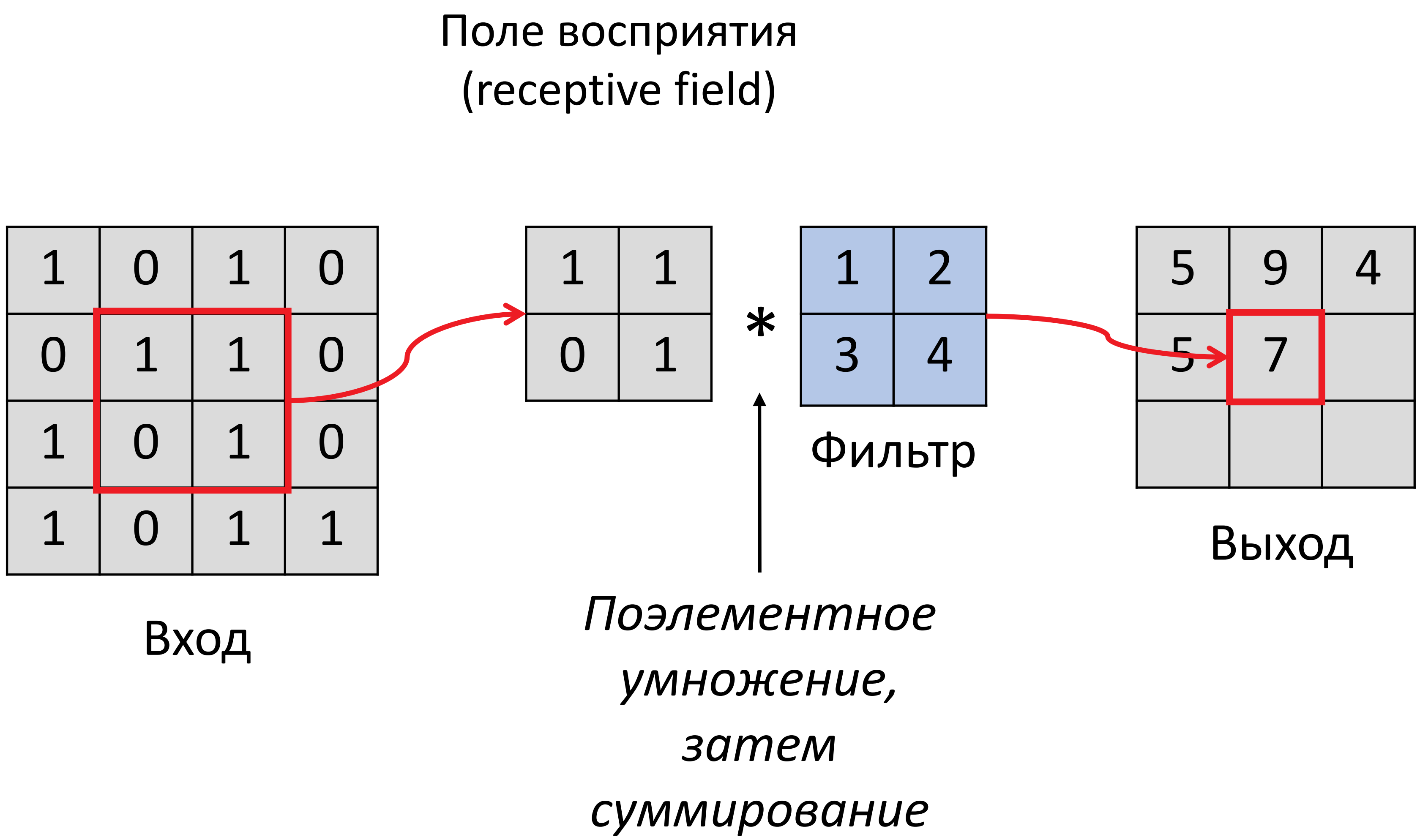


https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1557912/

Свёртка


Поле восприятия
(receptive field)
1 0 1 0
0 1 1 0
1 0 1 0
1 0 1 1
Вход
1 0
0 1
*
1 2
3 4
Фильтр
5
Выход
Поэлементное умножение, затем суммирование

Свёртка



Поле восприятия
(receptive field)
1 0 1 0
0 1 1 0
1 0 1 0
1 0 1 1
Вход
1 1
0 1
*
1 2
3 4
Фильтр
5 9 4
5 7
Выход
Поэлементное
умножение,
затем
суммирование

Свёртка

$$\begin{bmatrix}1 & 1\\0 & 1\end{bmatrix} * \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix} = \boxed{2}$$

$$\begin{bmatrix}1 & 1\\1 & 1\end{bmatrix} * \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix} = \boxed{2}$$

$$\begin{bmatrix}3 & 0\\0 & 3\end{bmatrix} * \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix} = \boxed{6}$$

$$\begin{bmatrix}1 & 2\\3 & 0\end{bmatrix} * \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix} = \boxed{1}$$

$$\begin{bmatrix}5 & 0\\0 & 5\end{bmatrix} * \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix} = \boxed{10}$$

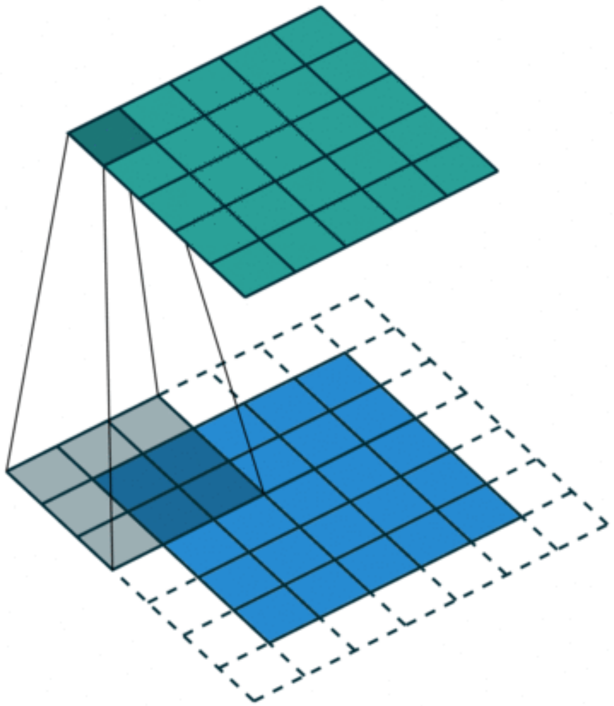
$$\begin{bmatrix}0 & 2\\3 & 0\end{bmatrix} * \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix} = \boxed{0}$$

Свёртка

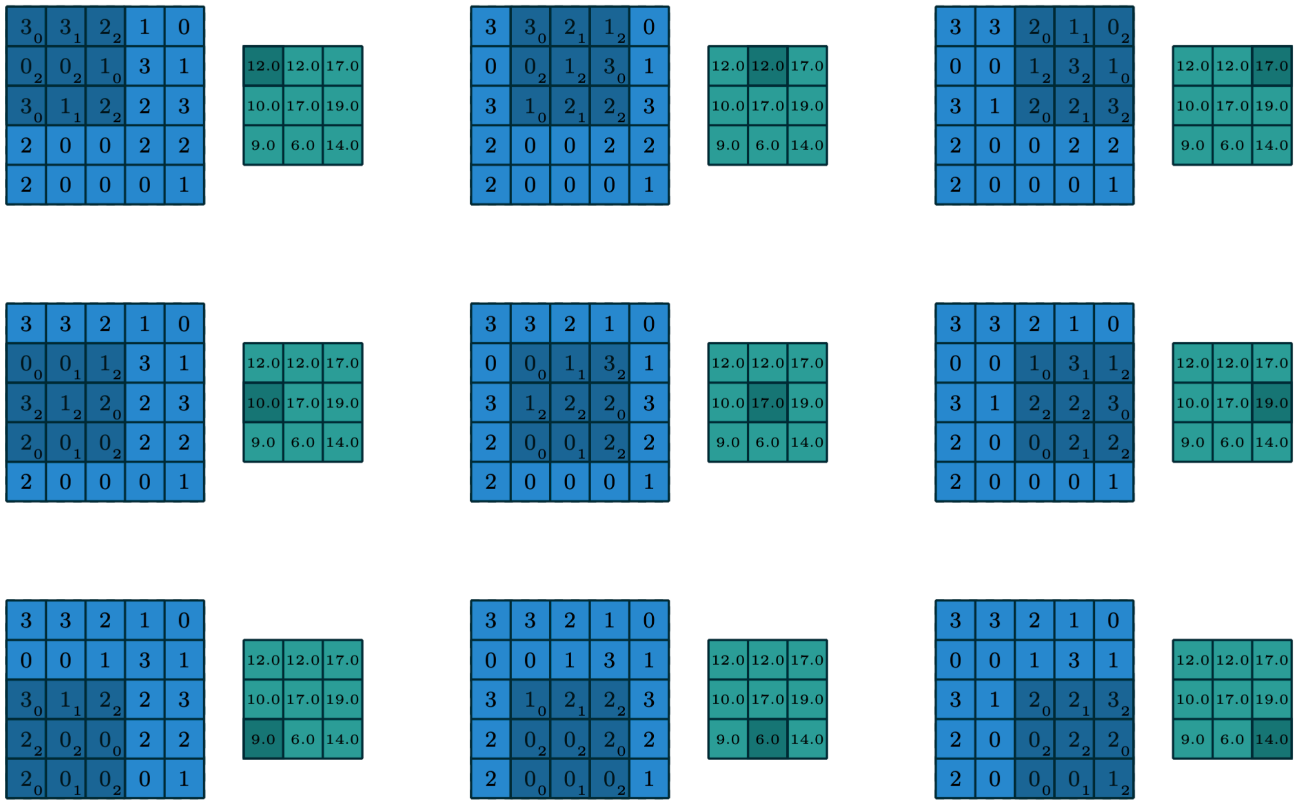
- Операция свёртки выявляет наличие на изображении паттерна, который задаётся фильтром
- Чем сильнее на участке изображения представлен паттерн, тем больше будет значение свёртки

Свёртка

- Результат свёртки изображения с фильтром — новое изображение

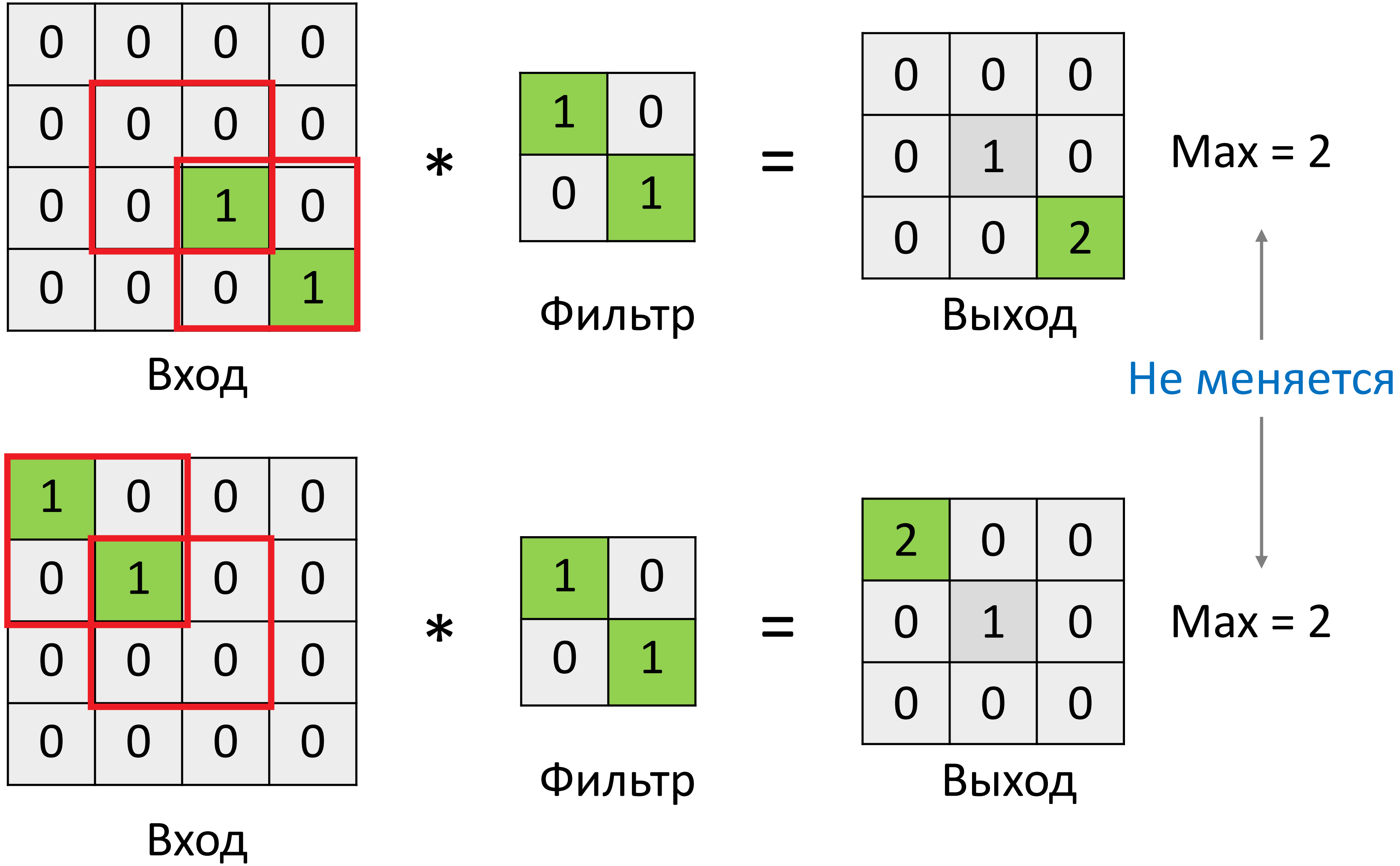


Свёртка



https://arxiv.org/pdf/1603.07285.pdf

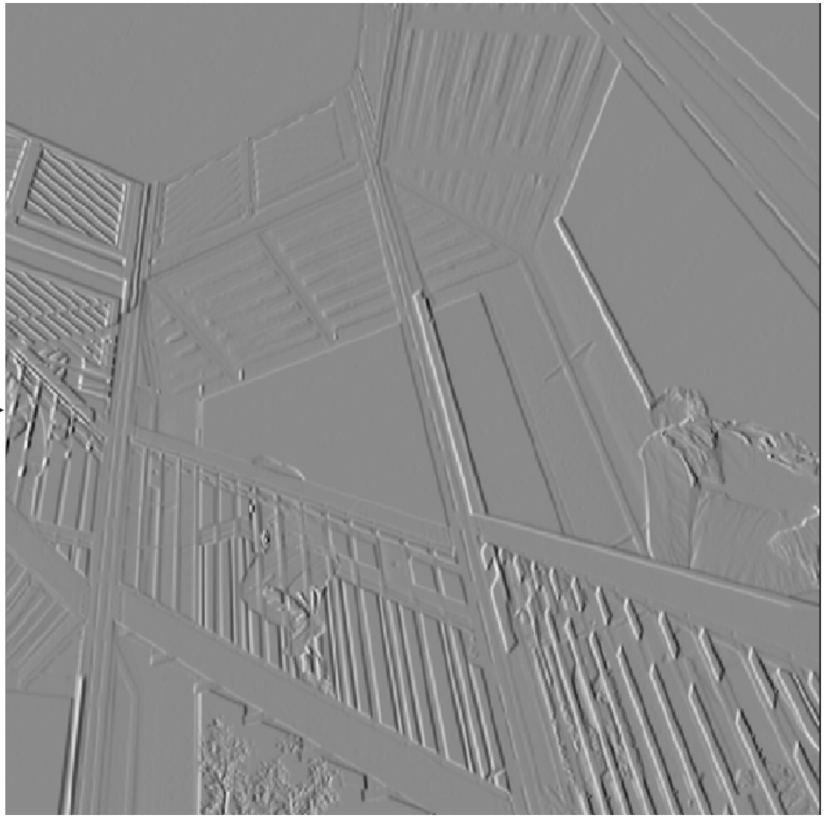
Свёртка инвариантна к сдвигам



Свёртки в компьютерном зрении

$$\begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix}$$

Horizontal Sobel kernel



https://towardsdatascience.com/intuitively-understanding-convolutions-for-deep-learning-1f6f42faee1

Свёртки в компьютерном зрении

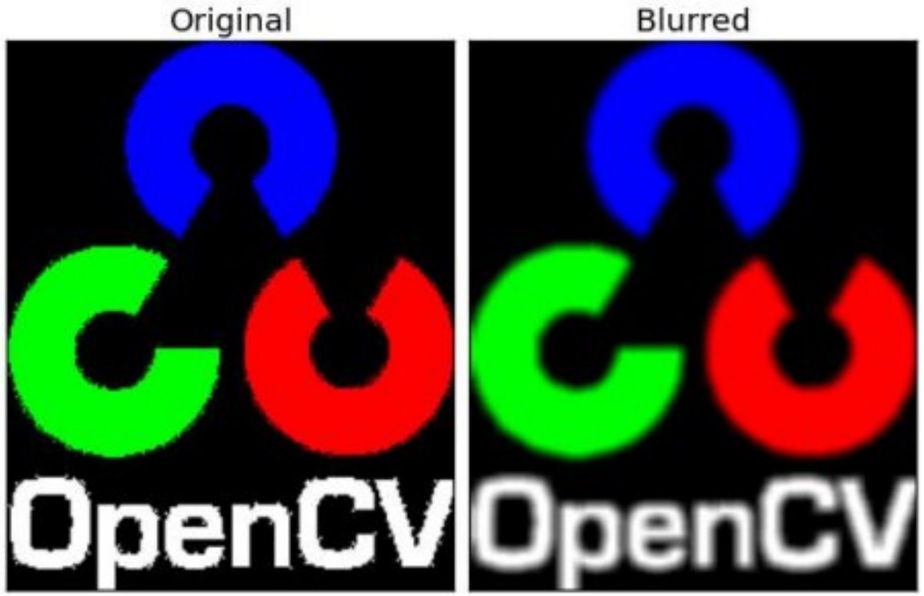
+ a

=

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

https://ai.stanford.edu/~syyeung/cvweb/tutorial1.html

Свёртки в компьютерном зрении



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

https://docs.opencv.org/master/d4/d13/tutorial_py_filtering.html

Свёртка

$$\mathrm{Im}^{out}(x, y) = \sum_{i=-d}^{d} \sum_{j=-d}^{d} K(i, j)\, \mathrm{Im}^{in}(x + i, y + j)$$

Свёртка

$$\mathrm{Im}^{out}(x, y) = \sum_{i=-d}^{d} \sum_{j=-d}^{d} K(i, j)\, \mathrm{Im}^{in}(x + i, y + j)$$

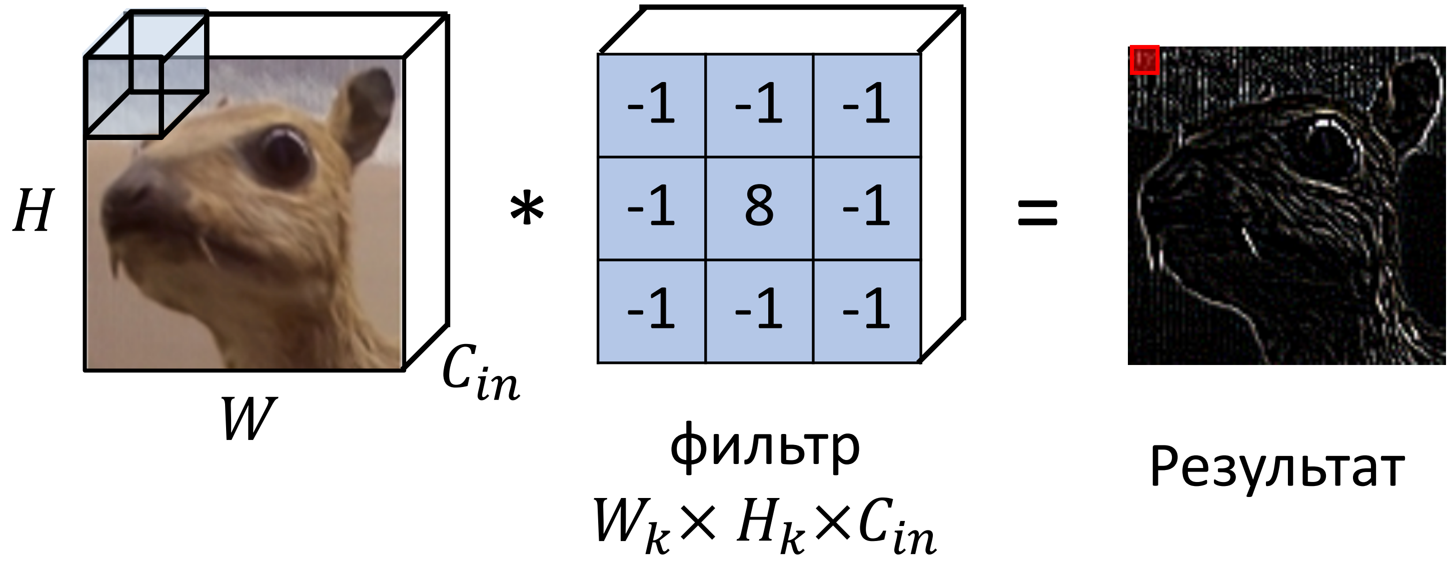
- Пиксель в результирующем изображении зависит только от небольшого участка исходного изображения (local connectivity)
- Веса одни и те же для всех пикселей результирующего изображения (shared weights)

Свёртка

- Обычно исходное изображение цветное!
- Это означает, что в нём несколько каналов (R, G, B)
- Учтём в формуле:

$$\mathrm{Im}^{out}(x, y) = \sum_{i=-d}^{d} \sum_{j=-d}^{d} \sum_{c=1}^{C} K(i, j, c)\, \mathrm{Im}^{in}(x + i, y + j, c)$$

Свёртка



Свёртка

- Одна свёртка выделяет конкретный паттерн на изображении
- Нам интересно искать много паттернов
- Сделаем результат трёхмерным:

$$\mathrm{Im}^{out}(x, y, t) = \sum_{i=-d}^{d} \sum_{j=-d}^{d} \sum_{c=1}^{C} K_t(i, j, c)\, \mathrm{Im}^{in}(x + i, y + j, c)$$

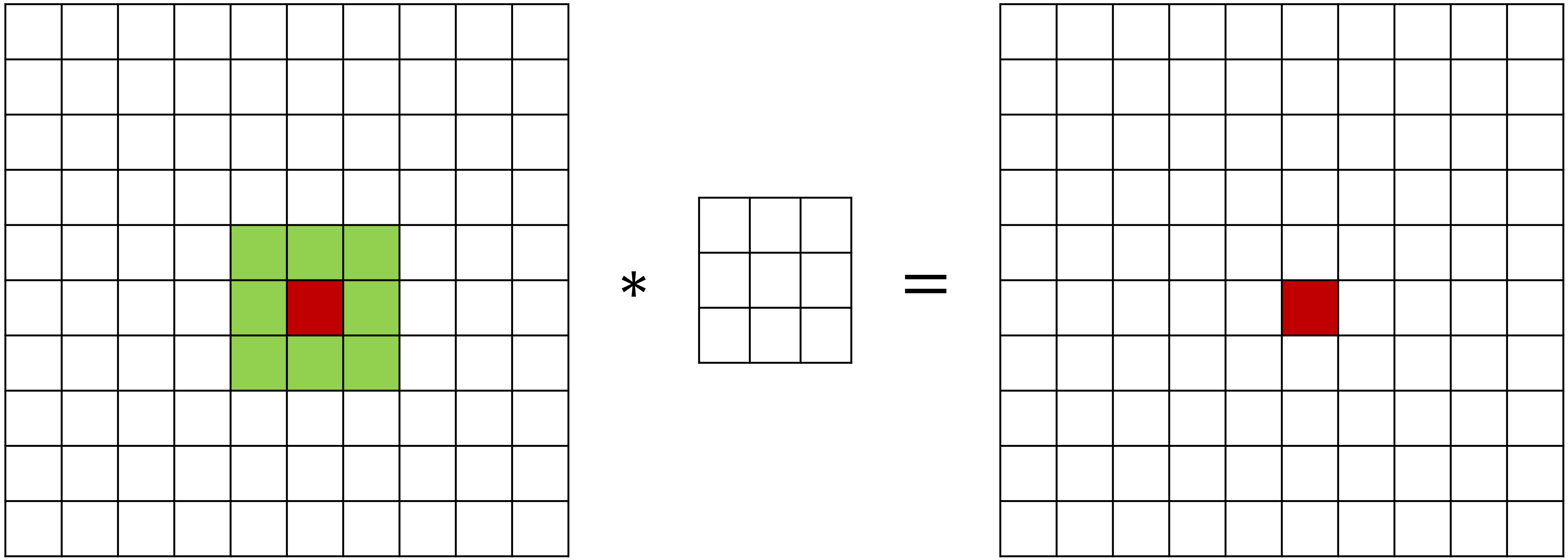
Число параметров

$$\mathrm{Im}^{out}(x,y,t) = \sum_{i=-d}^{d}\sum_{j=-d}^{d}\sum_{c=1}^{C} K_t(i,j,c)\,\mathrm{Im}^{in}(x+i,y+j,c)$$

- Обучается только фильтр
- $(2d+1) * C * T$ параметров

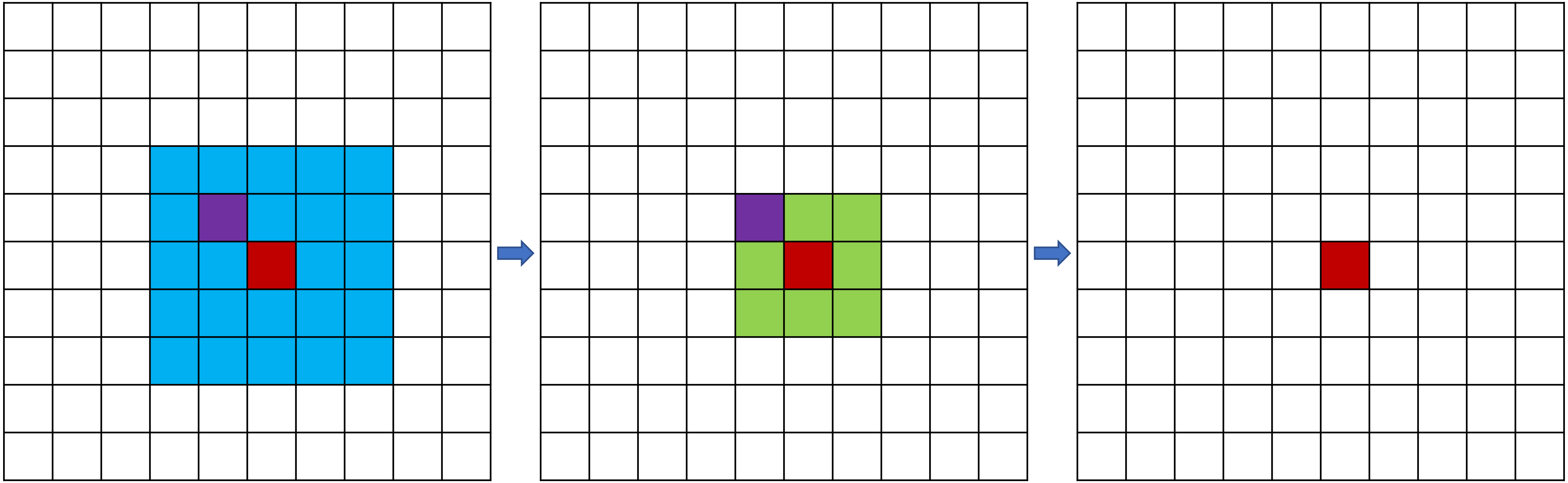
Receptive field

Receptive field



Поле восприятия: 3 x 3

Receptive field



Поле восприятия: 5 x 5

Receptive field

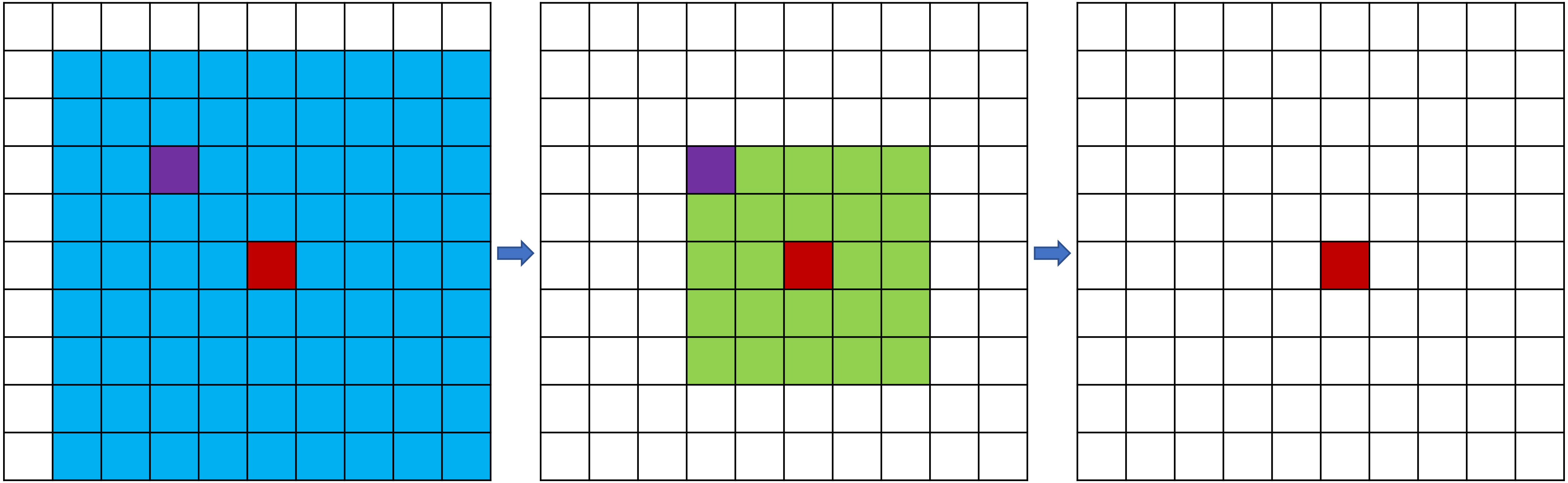
Поле восприятия для свёртки 3 x 3:

- После 1 свёрточного слоя: 3 x 3
- После 2 свёрточных слоев: 5 x 5
- После 3 свёрточных слоёв: 7 x 7

Receptive field

Поле восприятия для свёртки 5 x 5:

Receptive field



Поле восприятия: 5 x 5

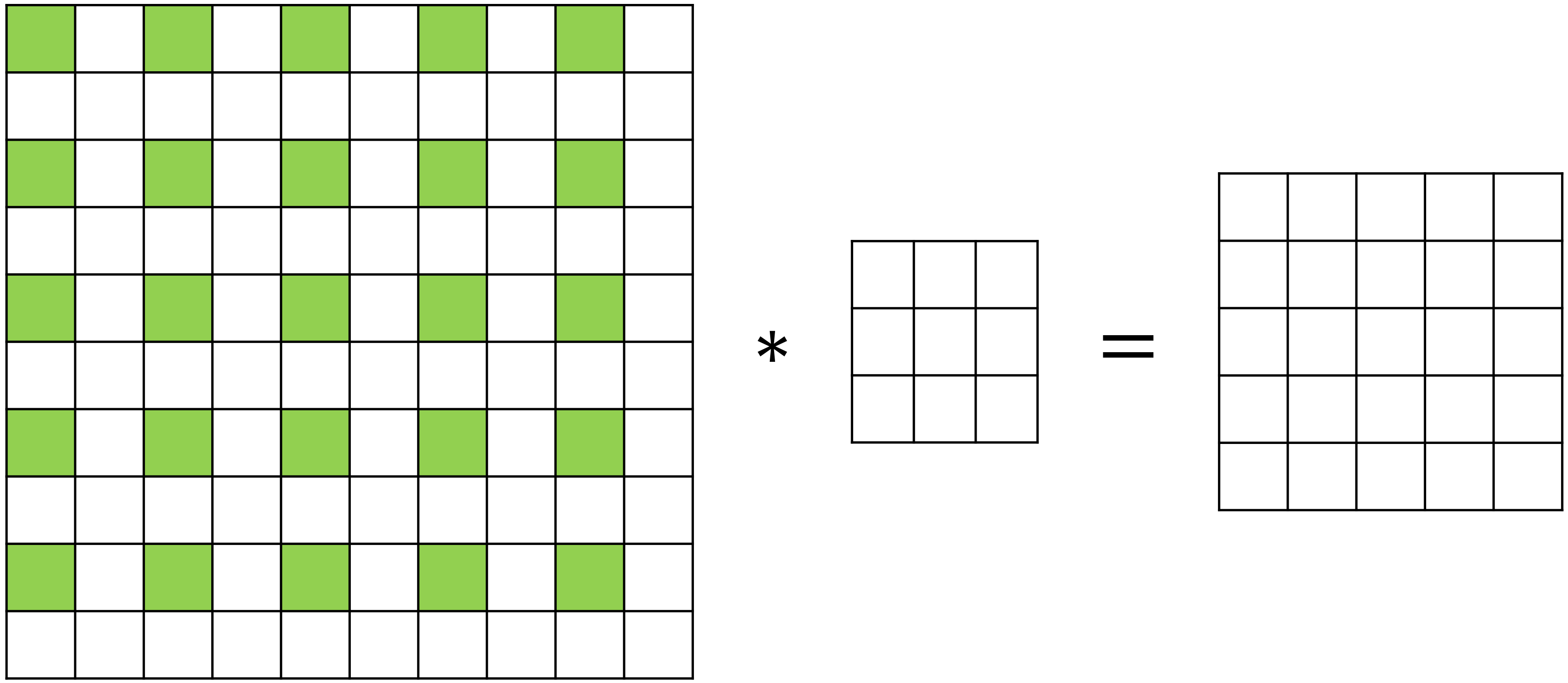
Receptive field

Поле восприятия для свёртки 5 x 5:

- После 1 свёрточного слоя: 5 x 5
- После 2 свёрточных слоев: 9 x 9
- После 3 свёрточных слоёв: 13 x 13

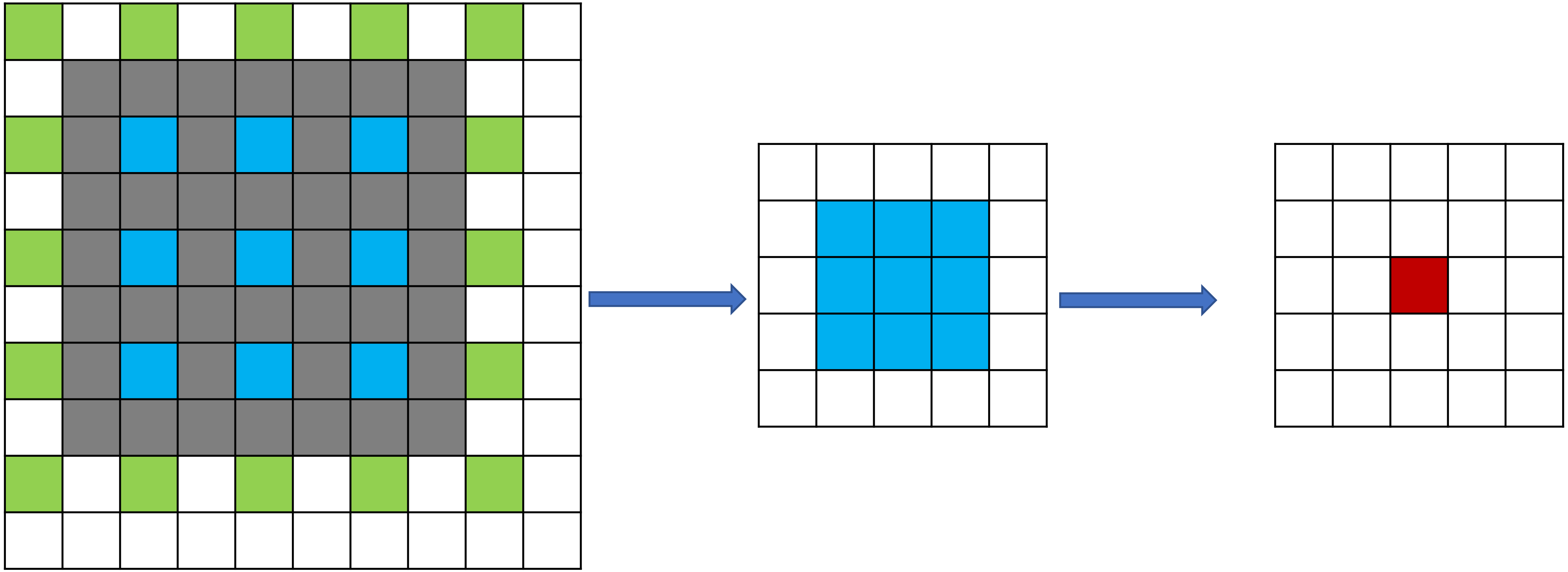
Нужно очень много слоёв, если изображение размера 512 x 512

Свёртки с пропусками (strides)



$s = 2$

Свёртки с пропусками (strides)



Поле восприятия: 7 x 7

Свёртки с пропусками (strides)

Подробности про подсчёт размера поля:
https://distill.pub/2019/computing-receptive-fields/

Резюме

- Свёрточные слои позволяют эффективно искать паттерны на изображениях
- В свёрточных сетях мало параметров, меньше риск переобучения
- Важно следить за тем, чтобы последние свёрточные слои имели размер поля восприятия, сравнимый со всей картинкой