

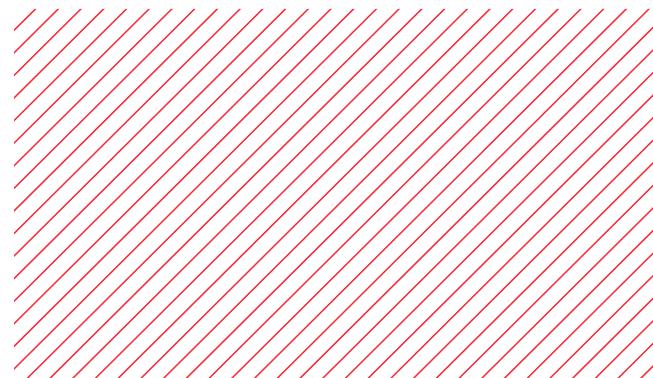
академия  
больших  
данных



# Сверточные нейронные сети (CNN)

Даниил Лысухин

Программист-исследователь в команде машинного  
зрения

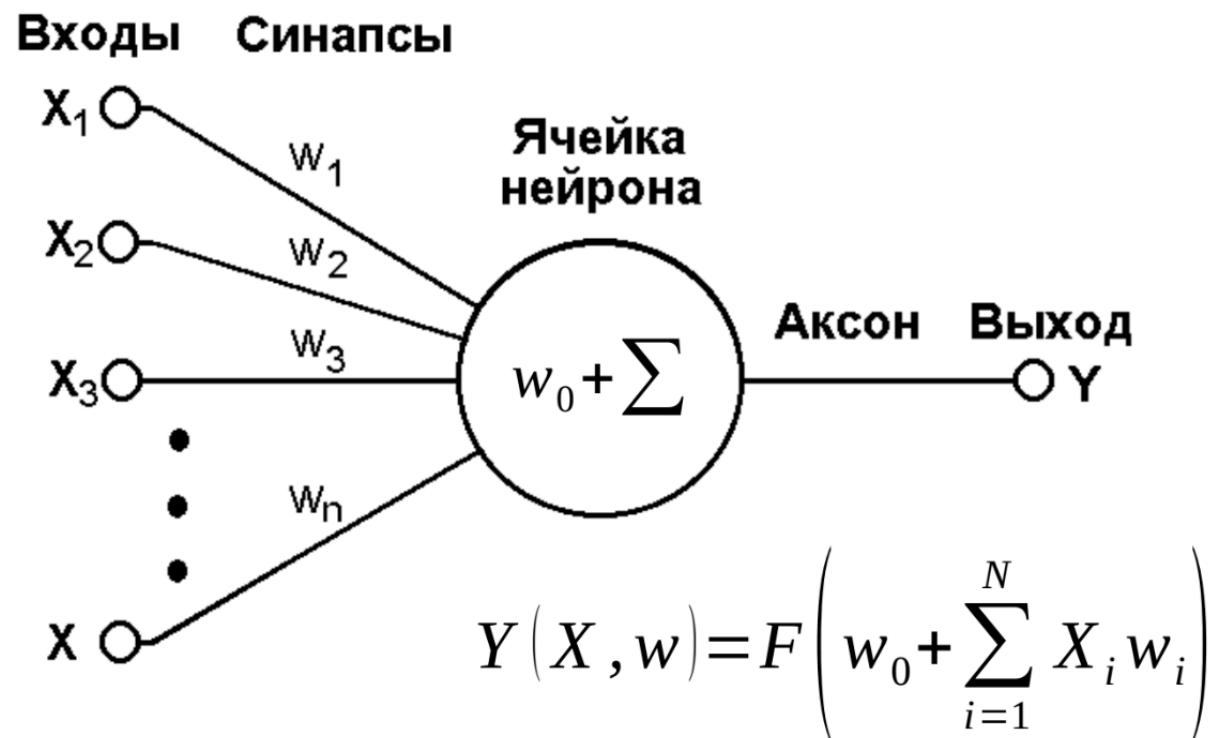




# Recap

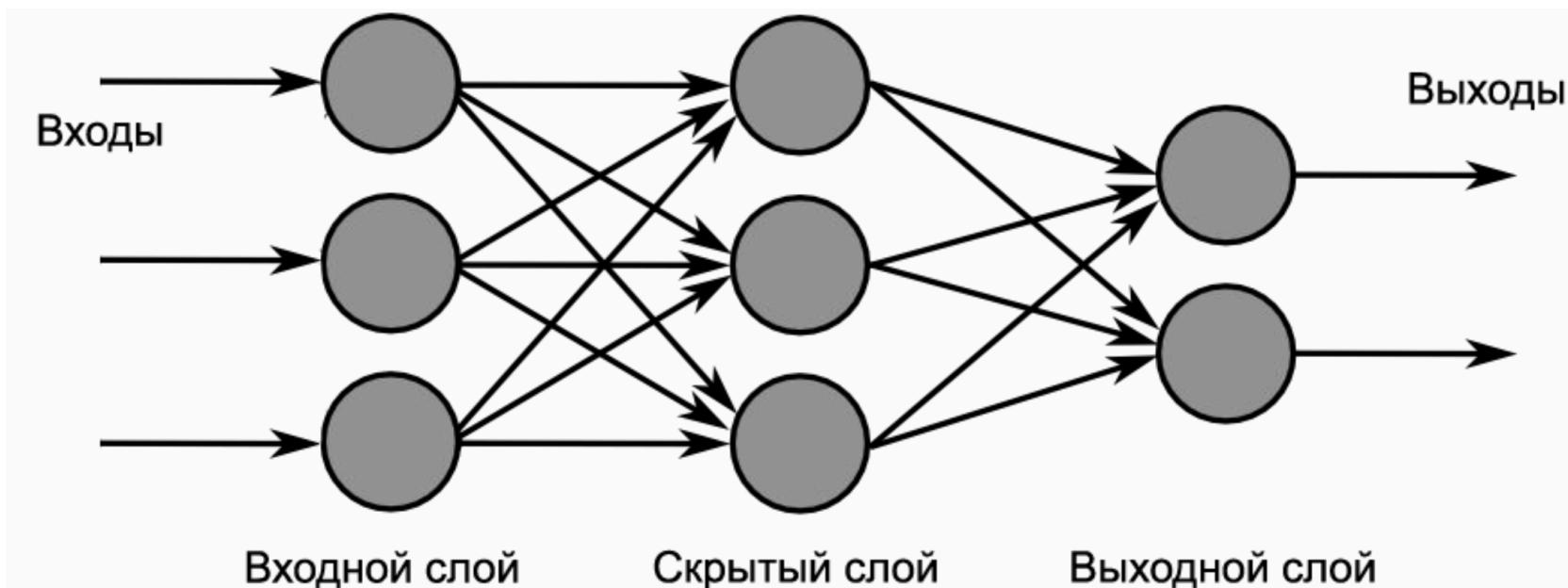
# Recap: нейрон

- Модель: линейная комбинация входов + нелинейная функция
- Как влияет нелинейность?

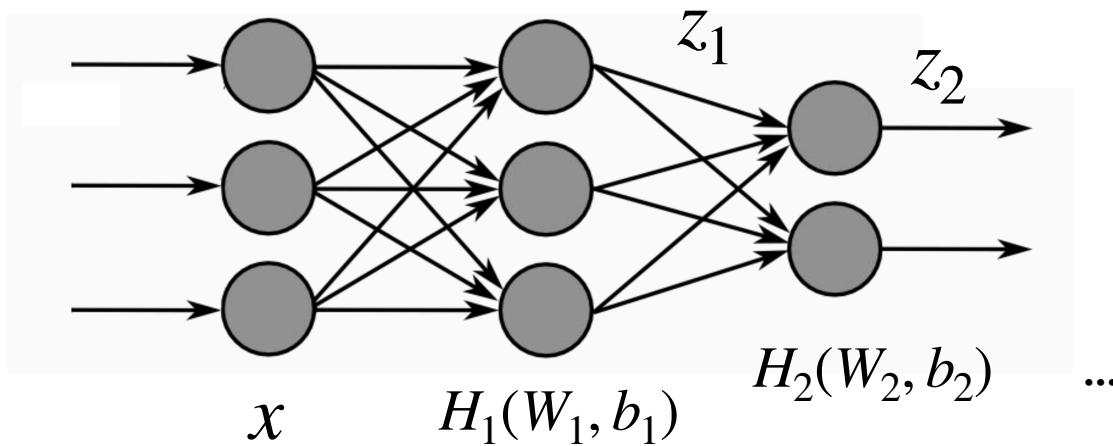


# Recap: полносвязная сеть

- Объединение отдельных нейронов в слои
- Если FC с 1 скрытым слоем - универсальный аппроксиматор, зачем что-то еще?



# Recap: пайплайн обучения с учителем



$$z_1 = Activation(H_1(x)) = Activation(W_1 \times x + b_1)$$

$$z_2 = Activation(H_2(z_1)) = Activation(W_2 \times z_1 + b_2)$$

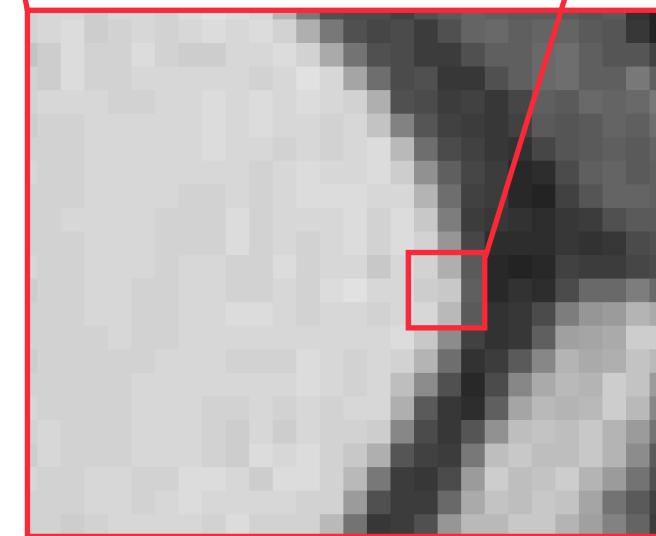
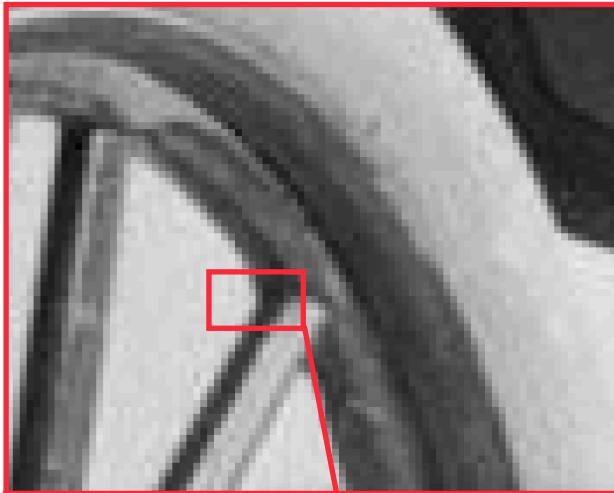
$$W^{t+1} = W^t - \eta \times \nabla Loss_W$$

- Как считается loss на данных?

# Как устроены картинки



# Как устроены картинки



[214, 176, 90]  
[207, 204, 97]  
[218, 186, 93]

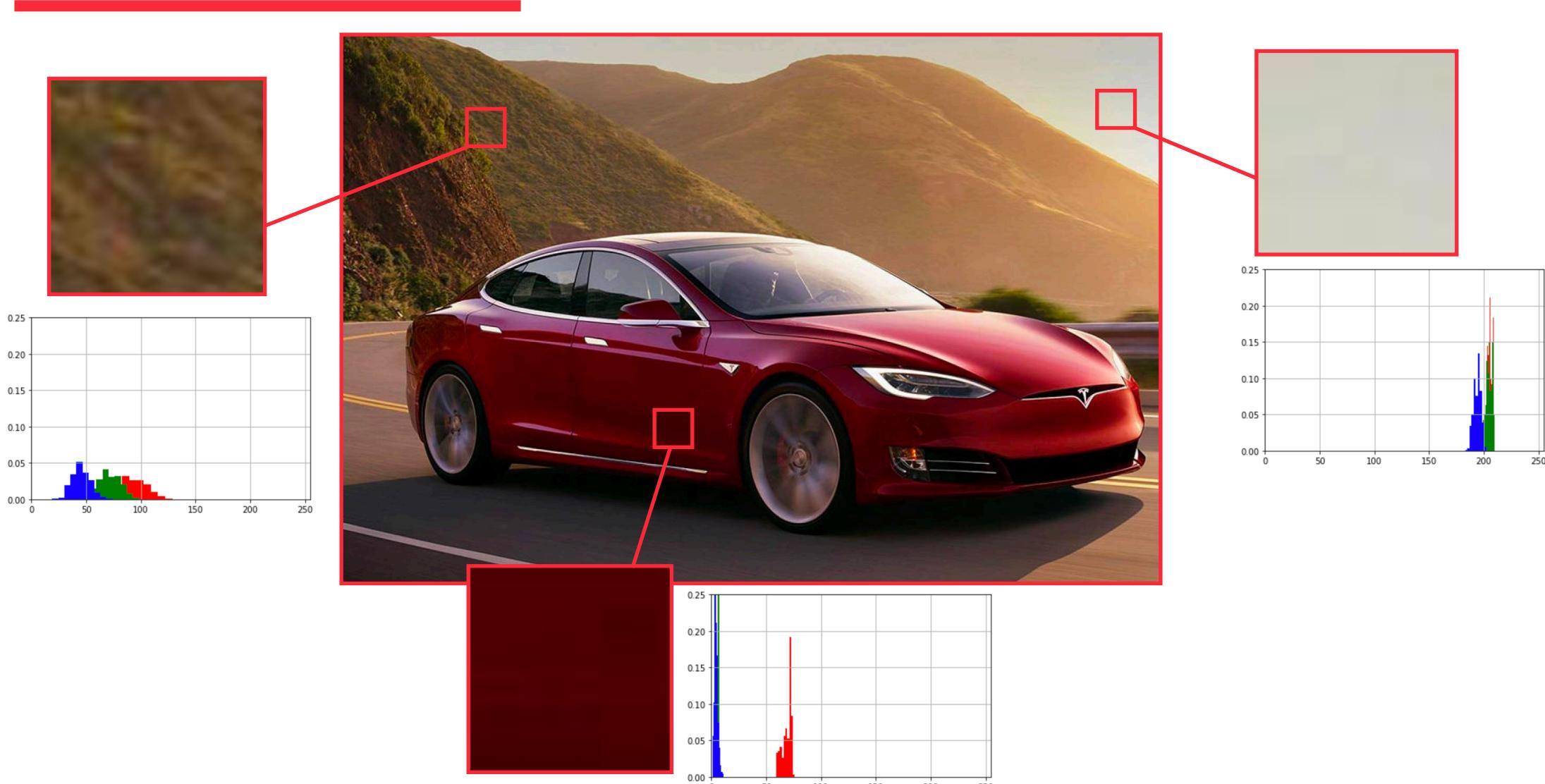
214	176	90
207	204	97
218	186	93

# Как устроены картинки

---

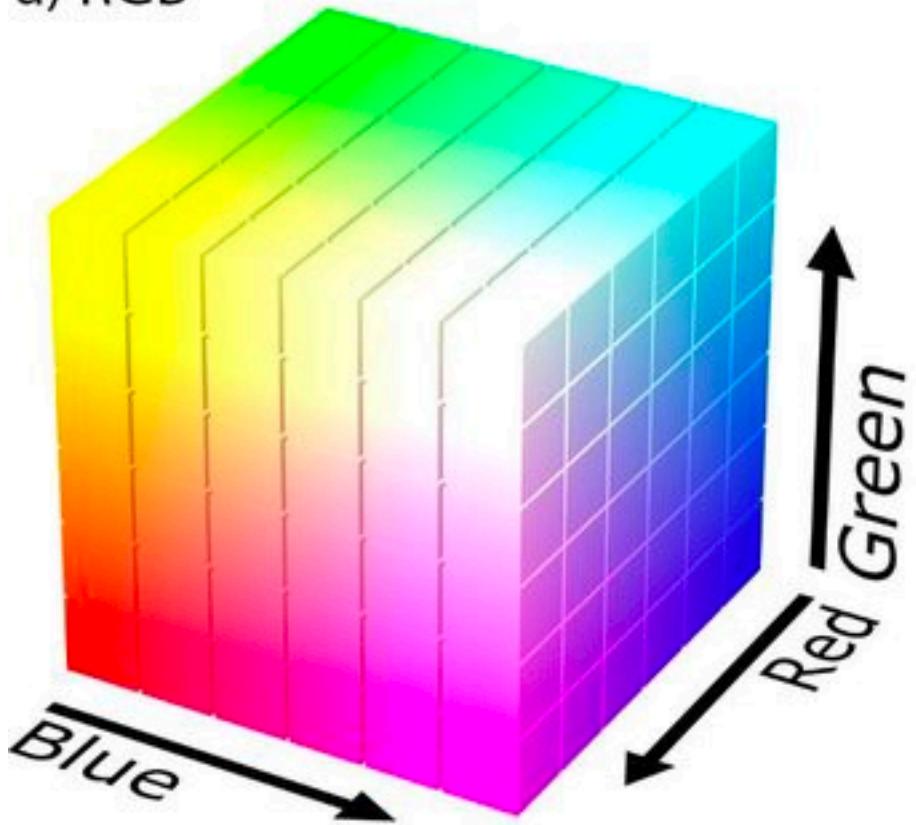


# Как устроены картинки

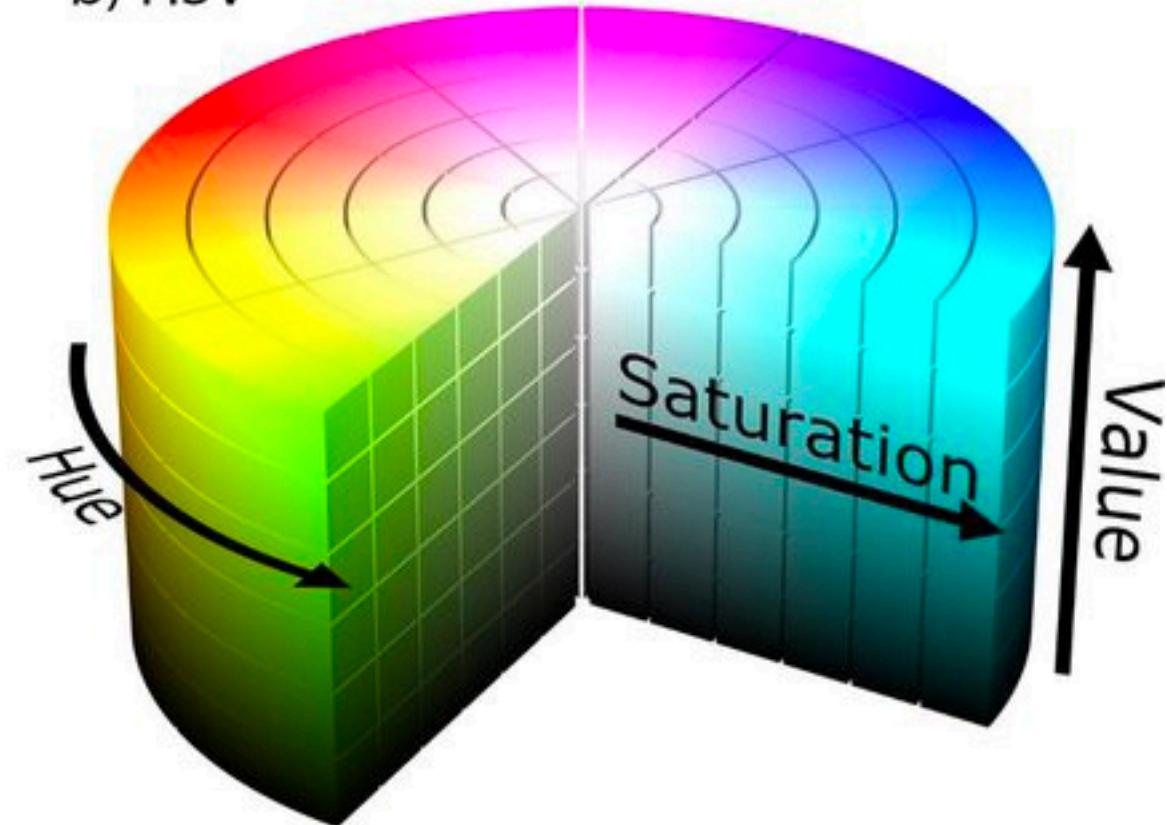


# Не только RGB

a) RGB

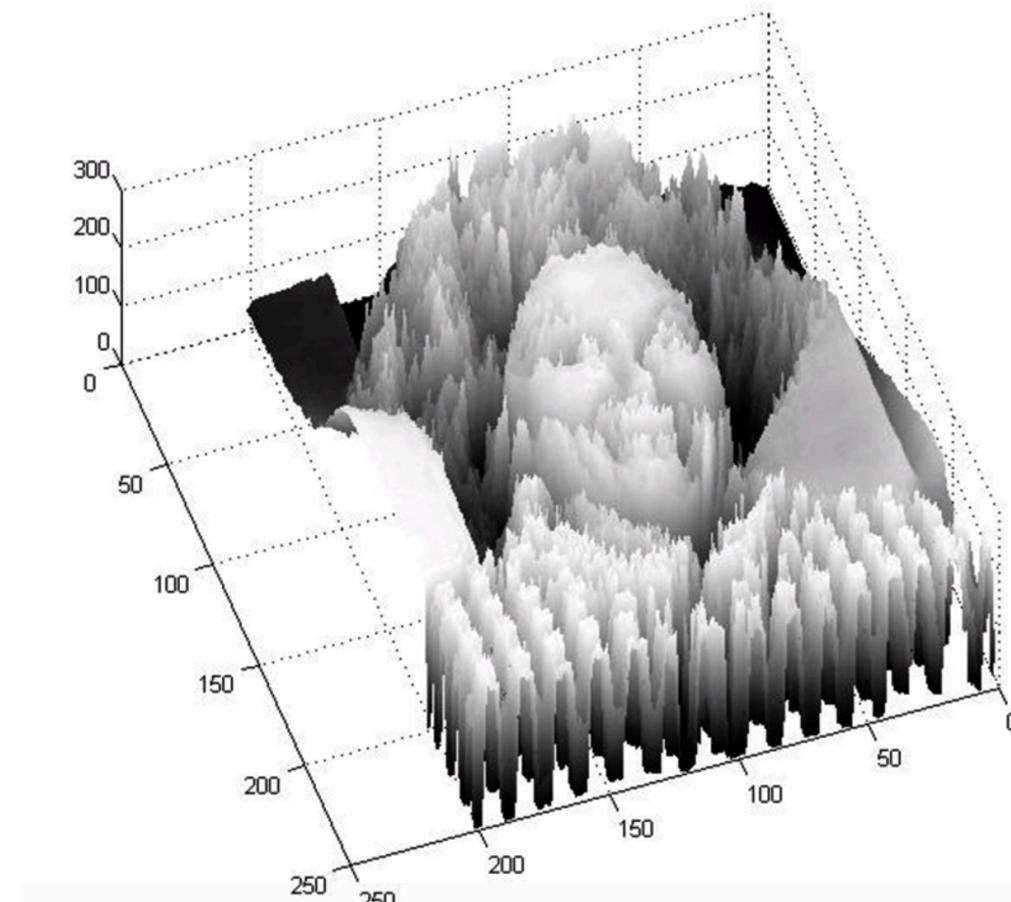


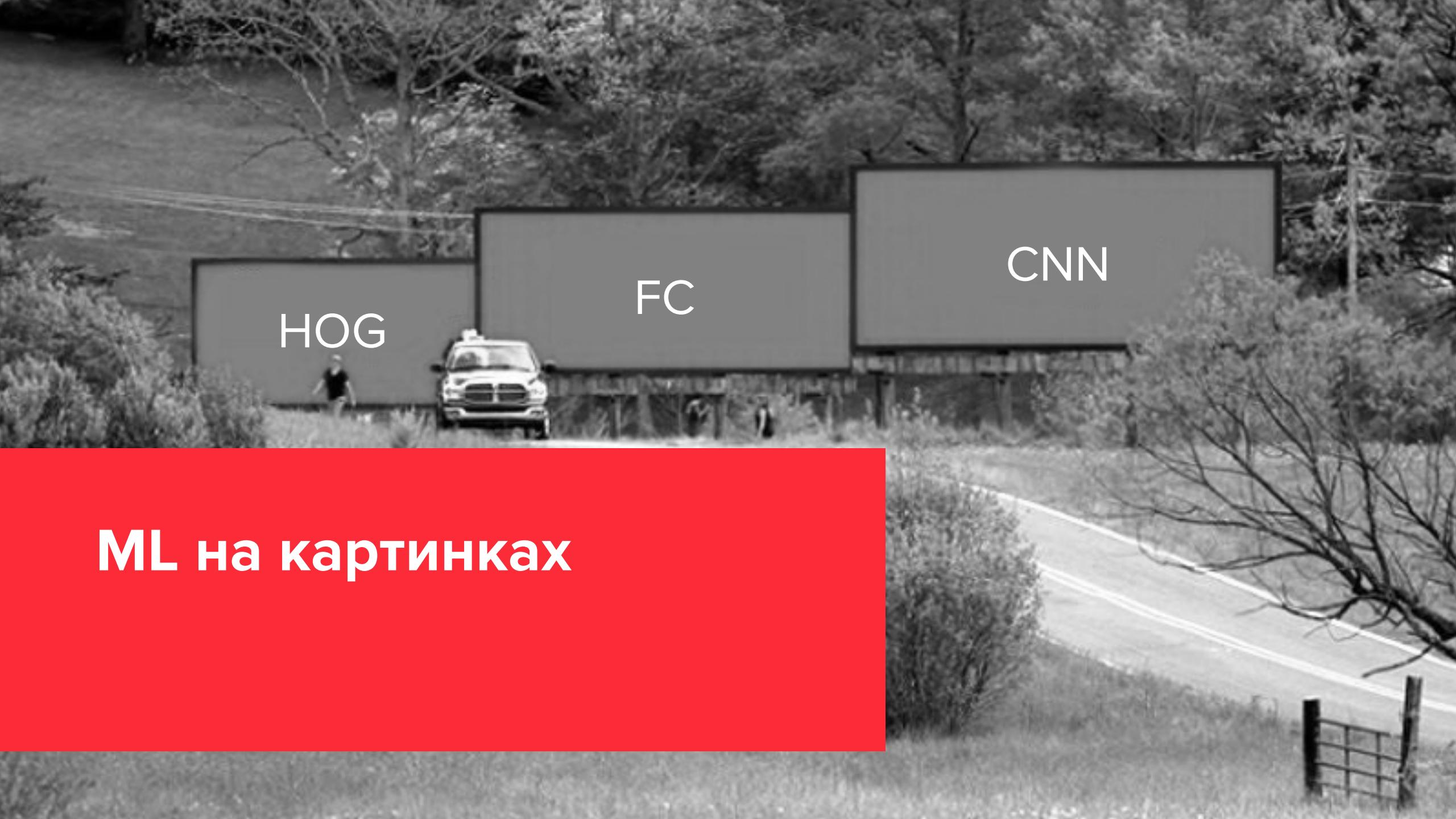
b) HSV



# Изображение как функция

---





CNN

FC

HOG

ML на картинках



# Как засунуть картинку в ML

---

- Табличные данные подходят для ML “из коробки”
- Что делать с изображениями?
  - “Придуманные” (hand-crafted) признаки (HOG, Haar, ...)
  - Значение пикселей = признаки
  - Автоматическое извлечение признаков

# Hand-crafted features (HOG)

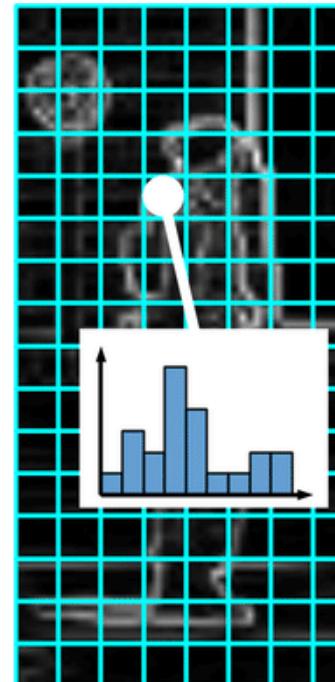
---



detection window  
slides over an  
image



at each location where  
the window is applied,  
gradients are  
computed



window is evenly  
partitioned into cells  
and each pixel of the  
cell contributes to cell  
gradient orientation  
histogram



orientation histograms  
for overlapping 2x2  
blocks of cells are  
normalized and  
collected to form the  
final descriptor

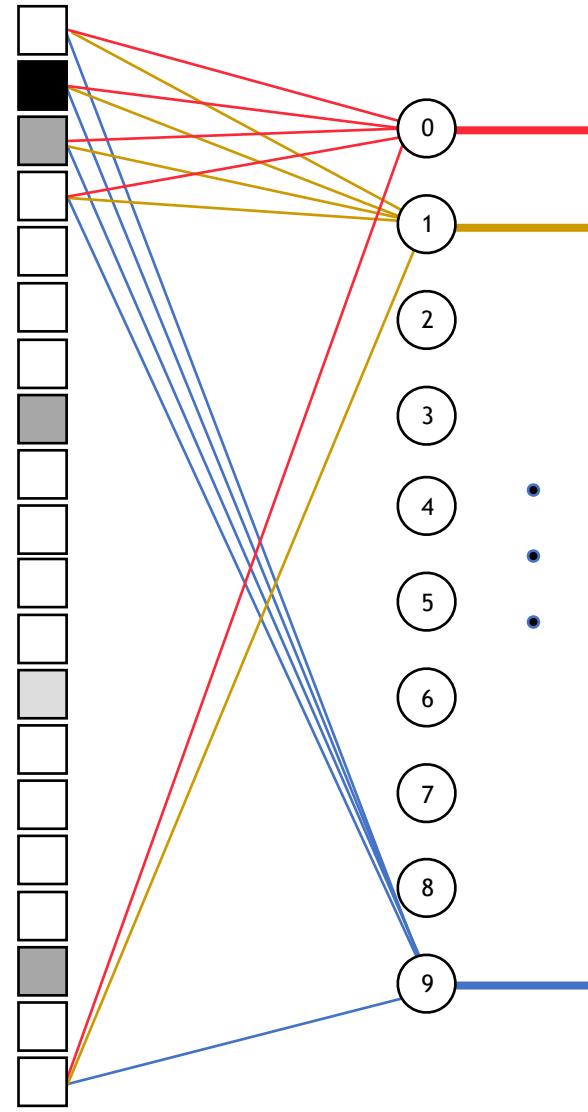
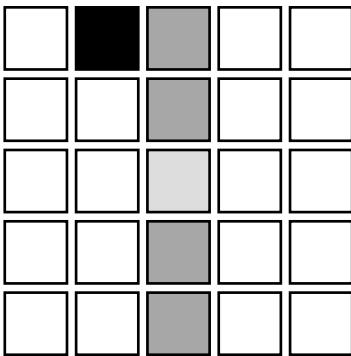


# Картина vs полносвязная сеть

---

- Рассматриваем яркость каждого пикселя как признак
  - Grayscale-картина  $5 \text{ px} \times 5 \text{ px} = 25$  признаков
  - RGB-картина -  $5 \times 5 \times 3 = 75$  признаков
- Модель - полносвязная сеть с 1 слоем

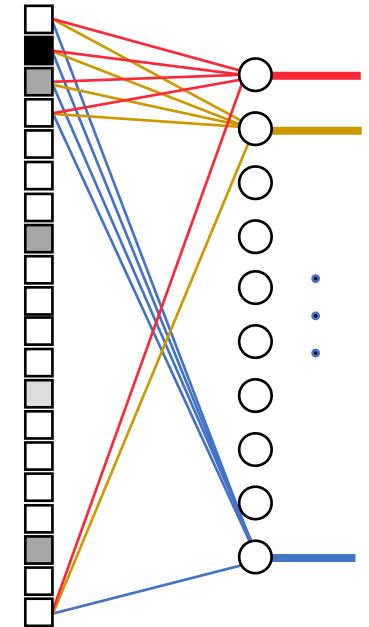
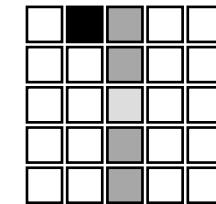
# Картина vs полносвязная сеть



# Картина vs полносвязная сеть

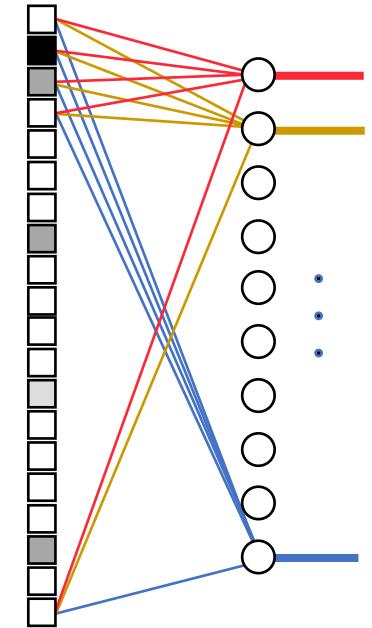
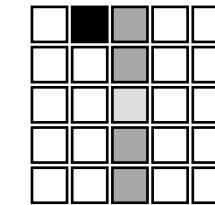
---

- Проблема #1: число параметров
  - ImageNet: картинки RGB,  $224 \times 224$ , 1000 классов
  - Число весов в сети с 1 слоем:
    - $3 \times 224 \times 224 \times 1000 = 150\ 528\ 000$

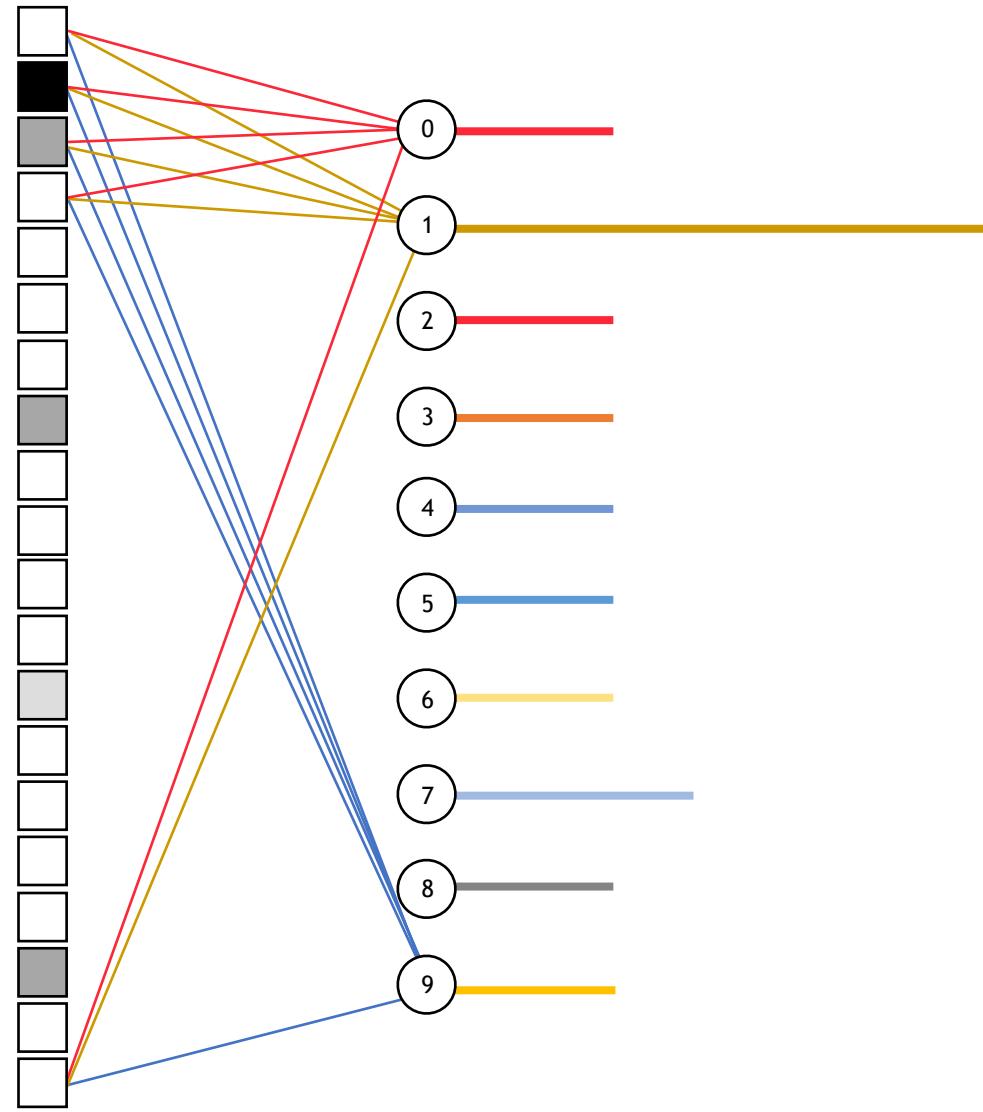
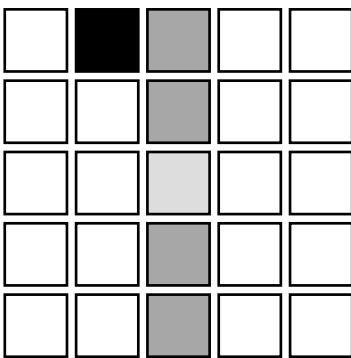


# Картина vs полносвязная сеть

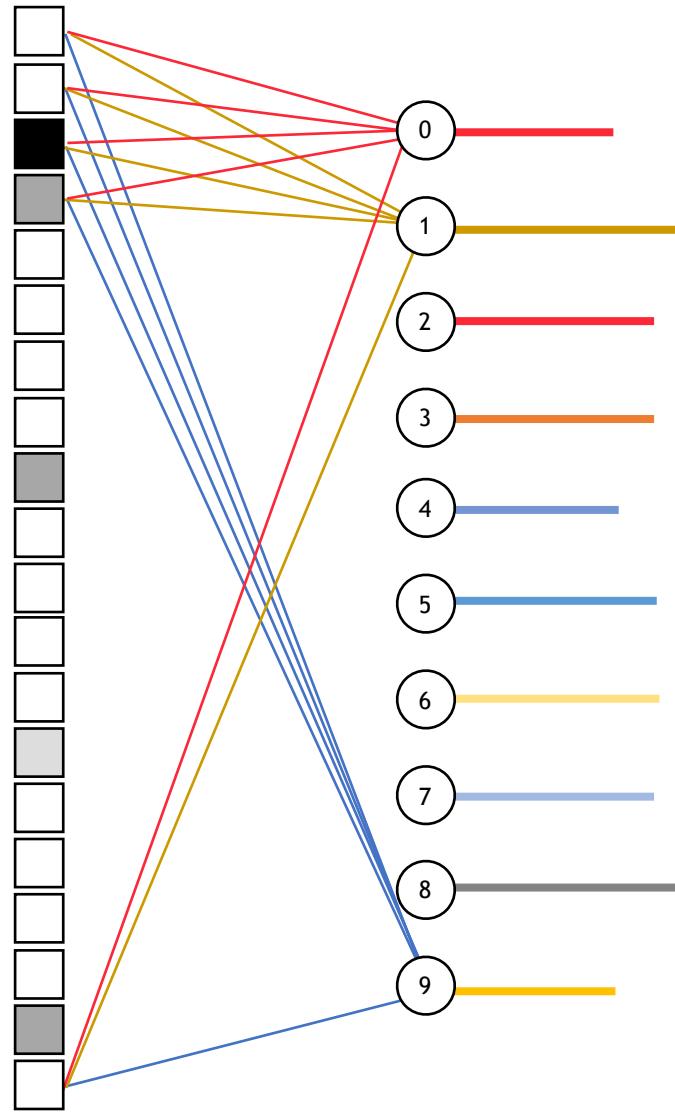
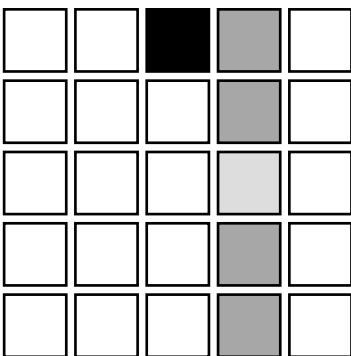
- Проблема #2: нет устойчивости к изменениям на входе
  - Подвинули картинку на 1 пиксель - что будет?



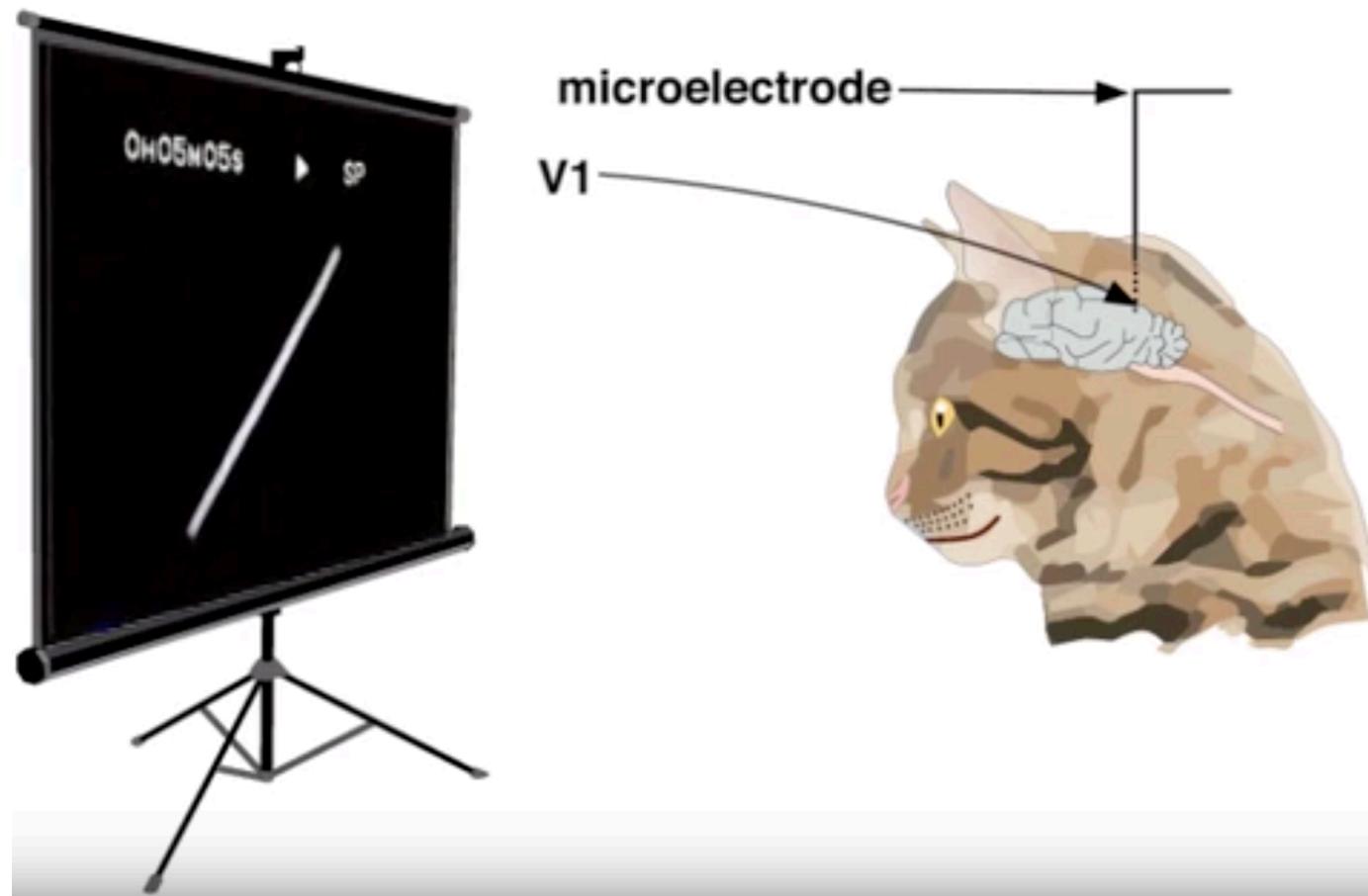
# Картина vs полносвязная сеть



# Картина vs полносвязная сеть

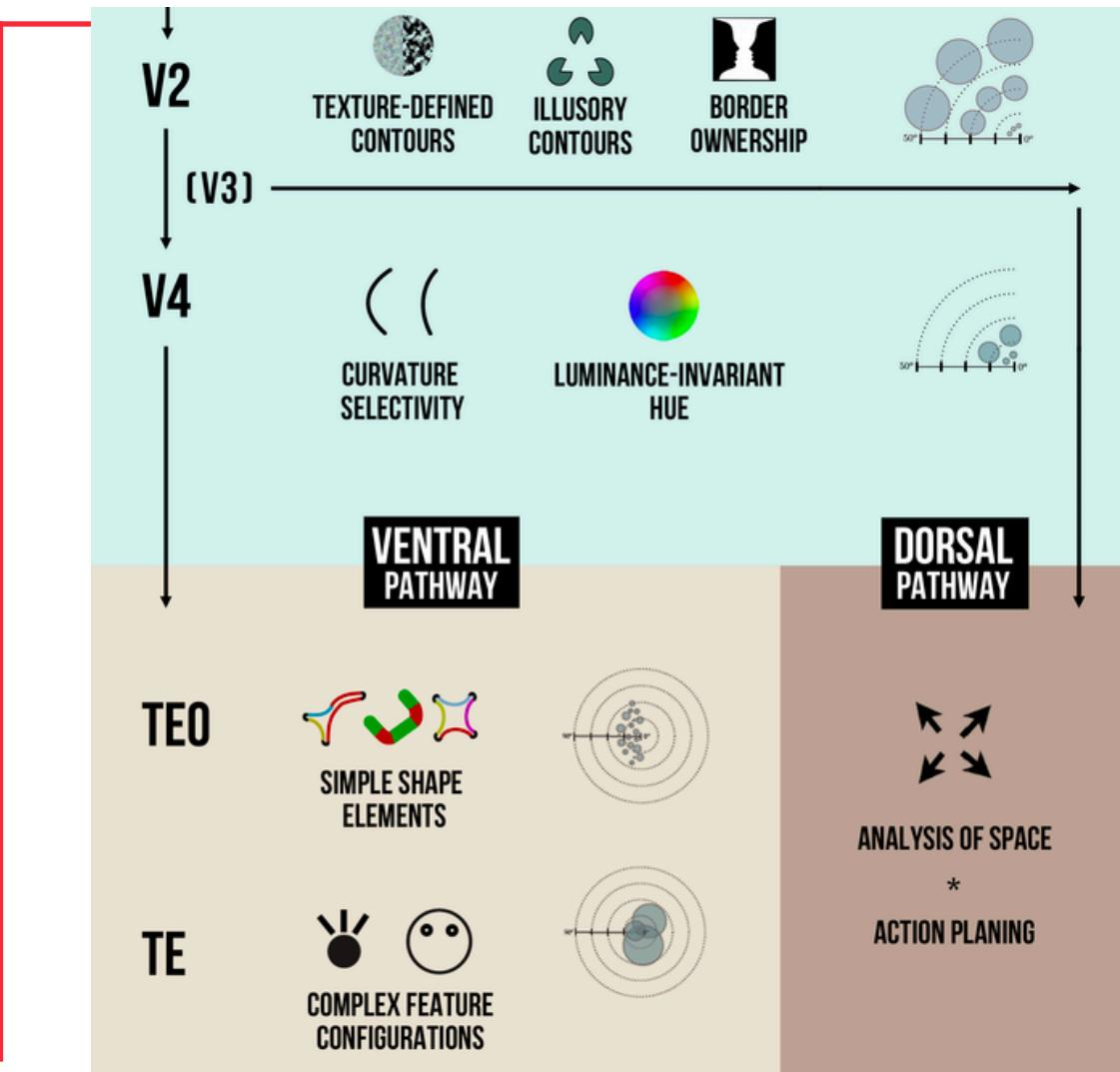
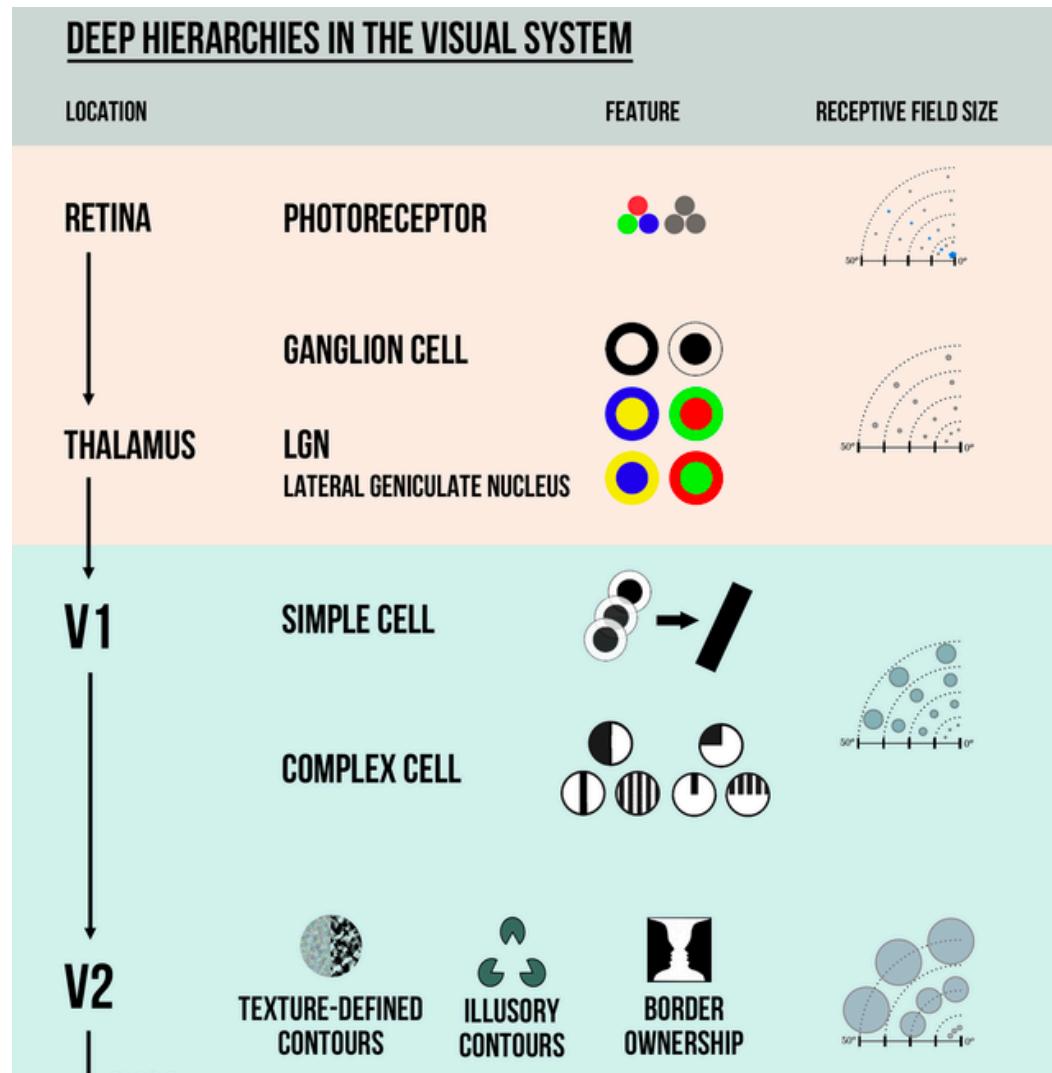


# Восприятие у животных



- Huber & Wiesel, 1959
- Определили в зрительной коре группы нейронов, реагирующих на **границы** под определенным **углом** (simple cells)
- Также обнаружили клетки, **“обобщдающие”** информацию из первичных клеток (complex cells)

# Восприятие у животных

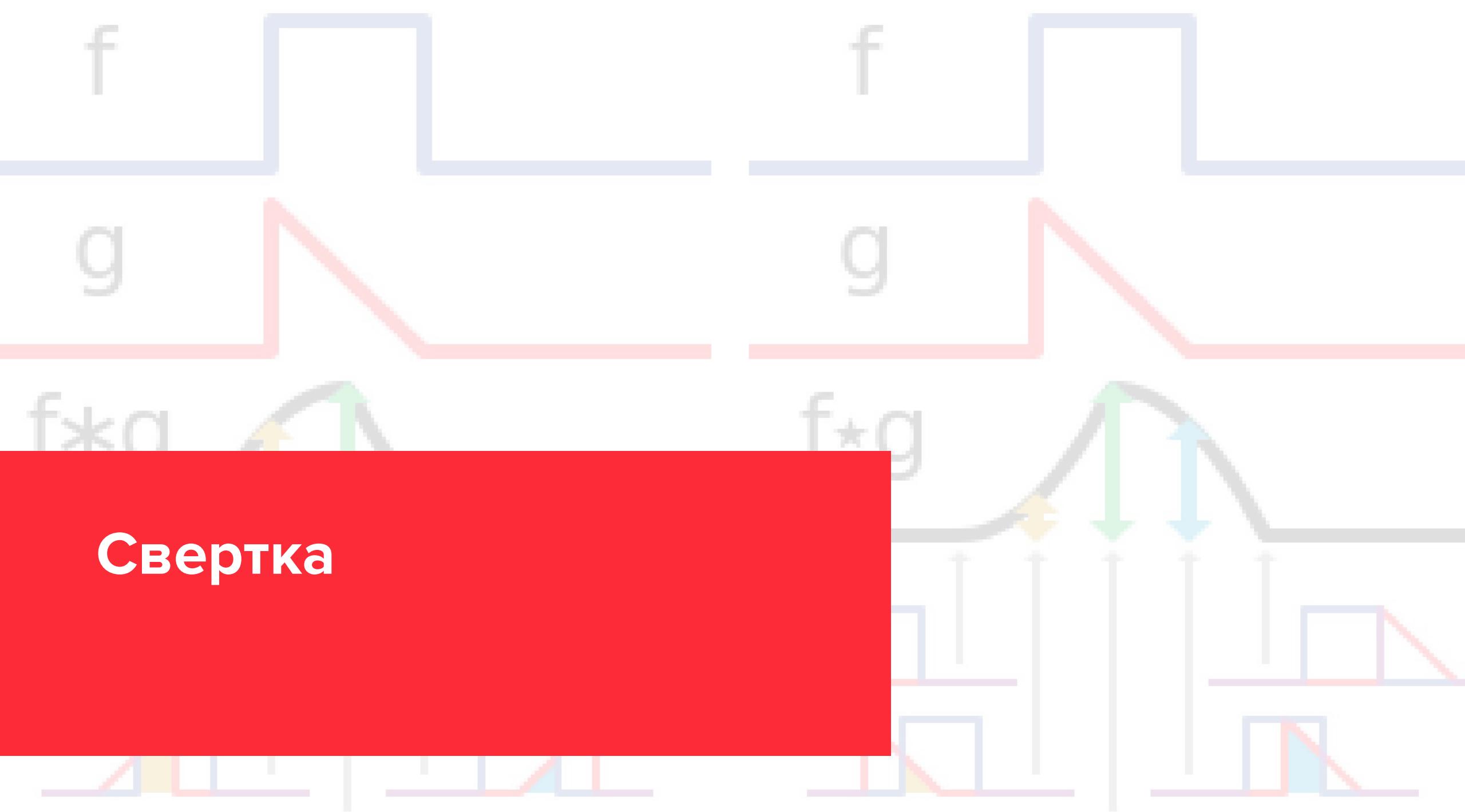




# Восприятие у животных

---

- Иерархичное
- Каждый этап извлекает информацию нового уровня абстракции
  - Начальные уровни - “сырая” информация (цвета, границы, линии, ...)
  - Следующие уровни - группы границ, паттерны, сложная геометрия, ...





# Фильтрация в СV

---

- Выделение границ на изображении можно реализовать применением **сверточной фильтрации**
- Типичные применения фильтрации: удаление шума, размытие, повышение резкости, выделение границ, ...



# Свертка

---

- **Свертка** - операция, при которой
  - Ядро свертки ( $N$ -мерный массив) “накладывается” поочередно ко всем позициям в  $N$ -мерном сигнале
  - В каждой позиции считается скалярное произведение между ядром свертки и попавшими “под него” значениями сигнала
  - Новое значение сигнала в позиции = значение скалярного произведения

# Свертка (1D)



x0	x1	x2	x3	x4
----	----	----	----	----

Входной сигнал  $x_i$

Ядро свертки  $w$

Размер **3**

Веса  $w_0, w_1, w_2$

w0	w1	w2
----	----	----

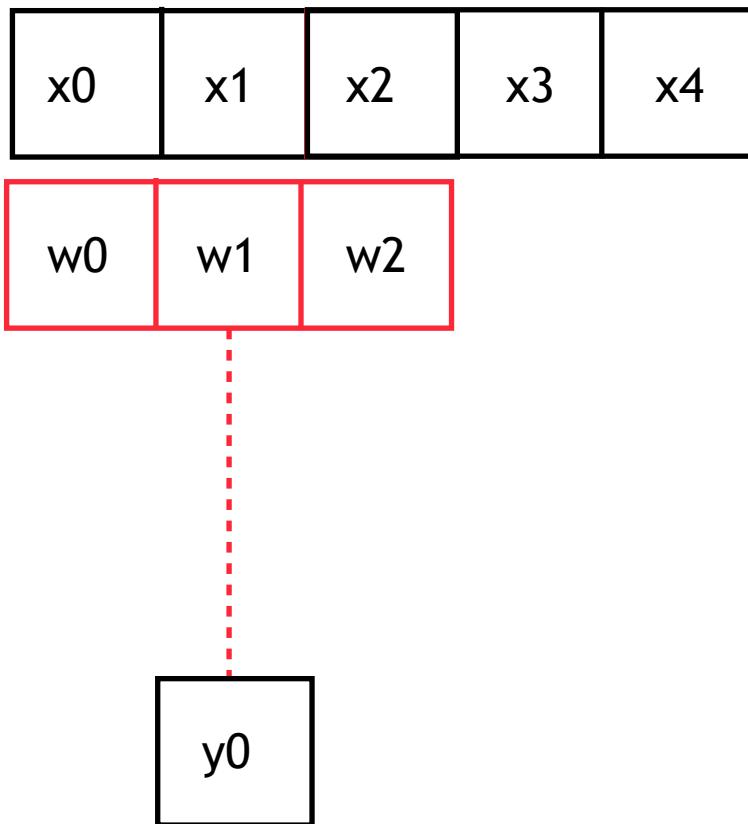
$$y_i = (x * w)_i = \sum_{k=0}^{K-1} x_{i+k} w_k$$

Результат свертки  $y_i$

# Свертка (1D)

Ядро свертки **w**  
Размер **3**  
Веса  $w_0, w_1, w_2$

$w_0$	$w_1$	$w_2$
-------	-------	-------



Входной сигнал  $x_i$

$$y_i = (x * w)_i = \sum_{k=0}^{K-1} x_{i+k} w_k$$

$$y_0 = x_0 \times w_0 + x_1 \times w_1 + x_2 \times w_2$$

Результат свертки  $y_i$

# Свертка (1D)

Ядро свертки **w**  
Размер **3**

Веса  $w_0, w_1, w_2$

$w_0$	$w_1$	$w_2$
-------	-------	-------

$x_0$	$x_1$	$x_2$	$x_3$	$x_4$
-------	-------	-------	-------	-------

Входной сигнал  $x_i$

$w_0$	$w_1$	$w_2$
-------	-------	-------

$$y_i = (x * w)_i = \sum_{k=0}^{K-1} x_{i+k} w_k$$

$$y_0 = x_0 \times w_0 + x_1 \times w_1 + x_2 \times w_2$$

$$y_1 = x_1 \times w_0 + x_2 \times w_1 + x_3 \times w_2$$

$y_0$	$y_1$
-------	-------

Результат свертки  $y_i$

# Свертка (1D)

Ядро свертки **w**

Размер **3**

Веса  $w_0, w_1, w_2$

$w_0$	$w_1$	$w_2$
-------	-------	-------

$x_0$	$x_1$	$x_2$	$x_3$	$x_4$
-------	-------	-------	-------	-------

Входной сигнал  **$x_i$**

$$y_i = (x * w)_i = \sum_{k=0}^{K-1} x_{i+k} w_k$$

$$y_0 = x_0 \times w_0 + x_1 \times w_1 + x_2 \times w_2$$

$$y_1 = x_1 \times w_0 + x_2 \times w_1 + x_3 \times w_2$$

$$y_2 = x_2 \times w_0 + x_3 \times w_1 + x_4 \times w_2$$

$y_0$	$y_1$	$y_2$
-------	-------	-------

Результат свертки  **$y_i$**

# Свертка (1D)

x0	x1	x2	x3	x4
----	----	----	----	----

Входной сигнал  $x_i$

Ядро свертки  $w$

Размер **3**

Веса  $w_0, w_1, w_2$

w0	w1	w2
----	----	----

$$y_i = (x * w)_i = \sum_{k=0}^{K-1} x_{i+k} w_k$$

$$y_0 = x_0 \times w_0 + x_1 \times w_1 + x_2 \times w_2$$

$$y_1 = x_1 \times w_0 + x_2 \times w_1 + x_3 \times w_2$$

$$y_2 = x_2 \times w_0 + x_3 \times w_1 + x_4 \times w_2$$

y0	y1	y2
----	----	----

Результат свертки  $y_i$

# Свертка (2D)

Входной сигнал  $X_{ij}$

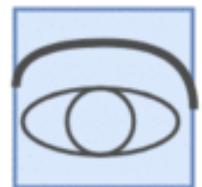
X00	X01	X02	X03
X10	X11	X12	X13
X20	X21	X22	X23
X30	X31	X32	X33

$$Y_{ij} = (X * W)_{ij} = \sum_{u=0}^{K-1} \sum_{v=0}^{K-1} X_{i+u, j+v} W_{uv}$$

Результат свертки  $Y_{ij}$

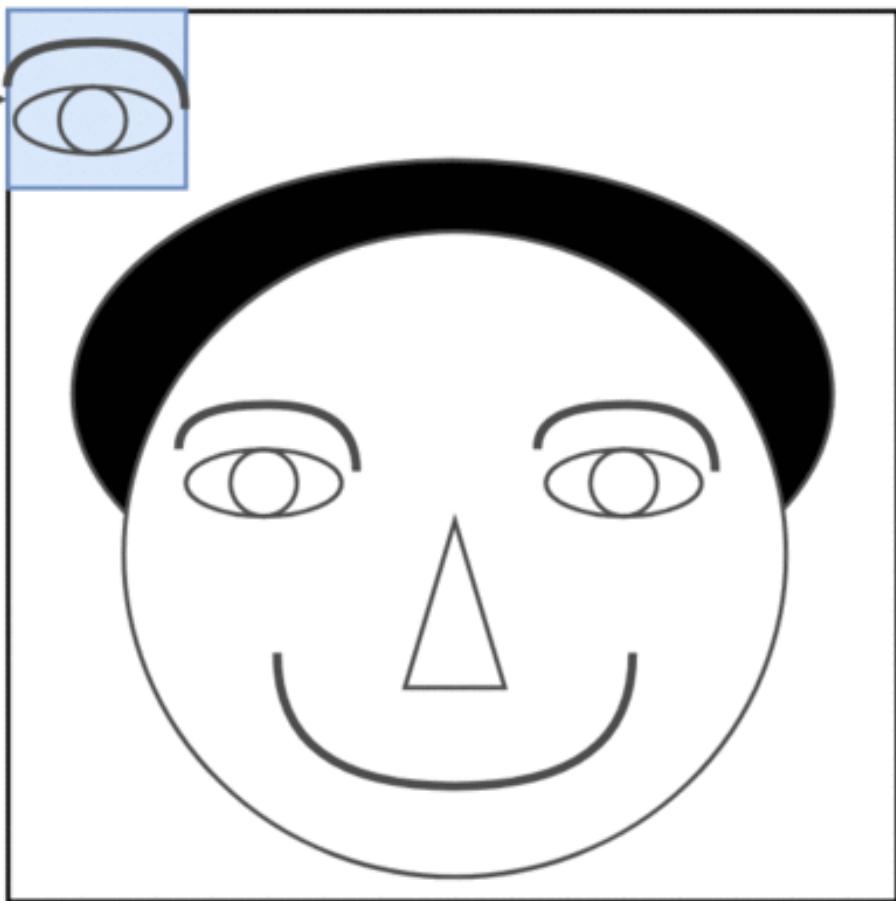
Y00	Y01
Y10	Y11

Ядро свертки  $W$   
Размер **3x3**  
Веса  $W_{ij}$



Convolution  
Kernel

\*

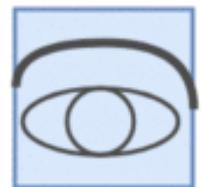


Image

=

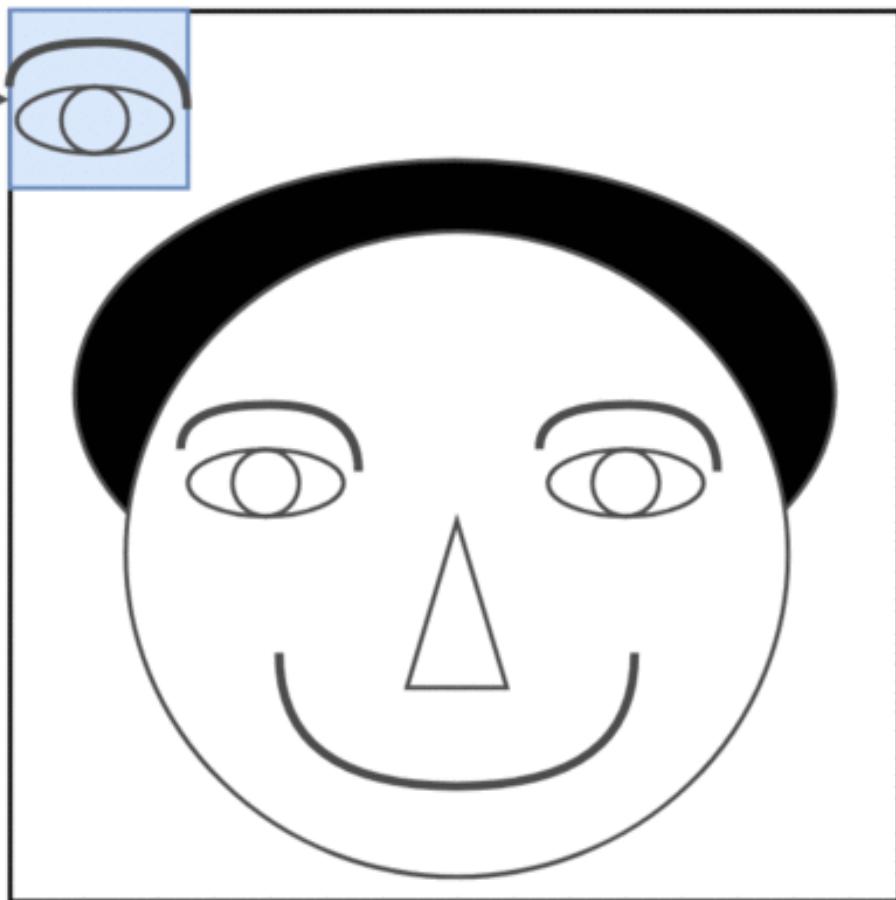
0			

Convolution Output



Convolution  
Kernel

\*



Image

=

0			

Convolution Output

# Свертка

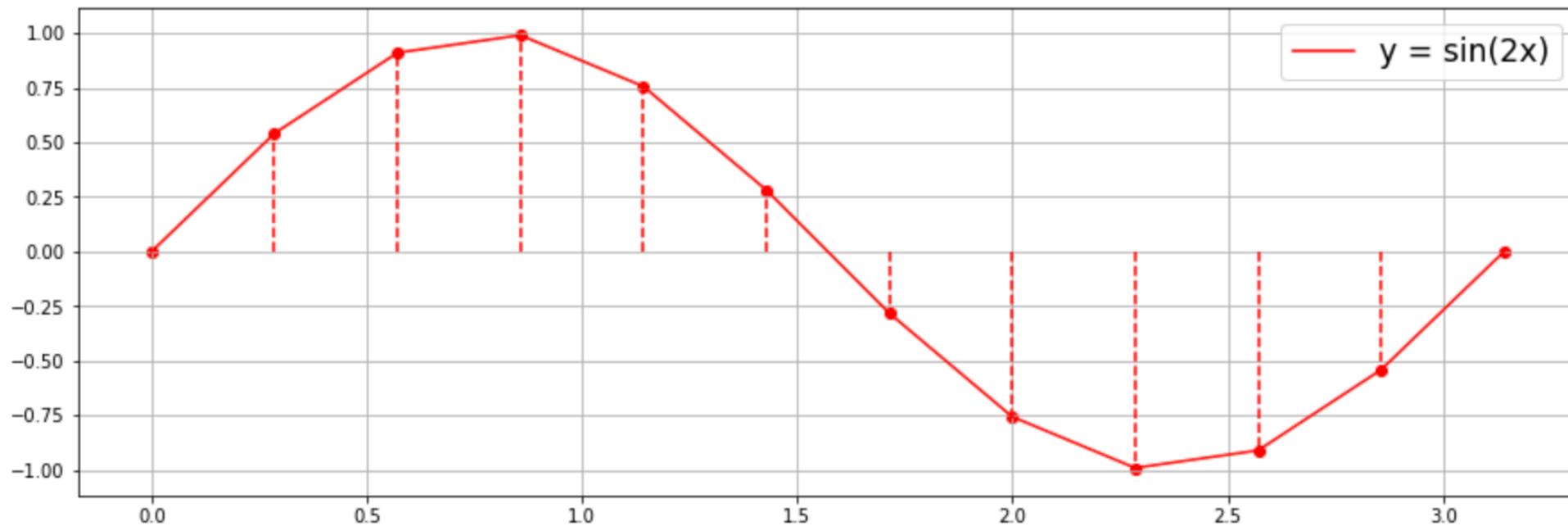
---

- В математике **свертка** - это операция над функциями **f** и **g**, в результате которой получается третья функция **h**, показывающая взаимную корреляцию функцию **f(x)** и **g(-x)**.

$$(f * g)(x) \stackrel{\text{def}}{=} \int_{\mathbb{R}^n} f(y) g(x - y) dy = \int_{\mathbb{R}^n} f(x - y) g(y) dy.$$

# Сглаживание сверткой (1D)

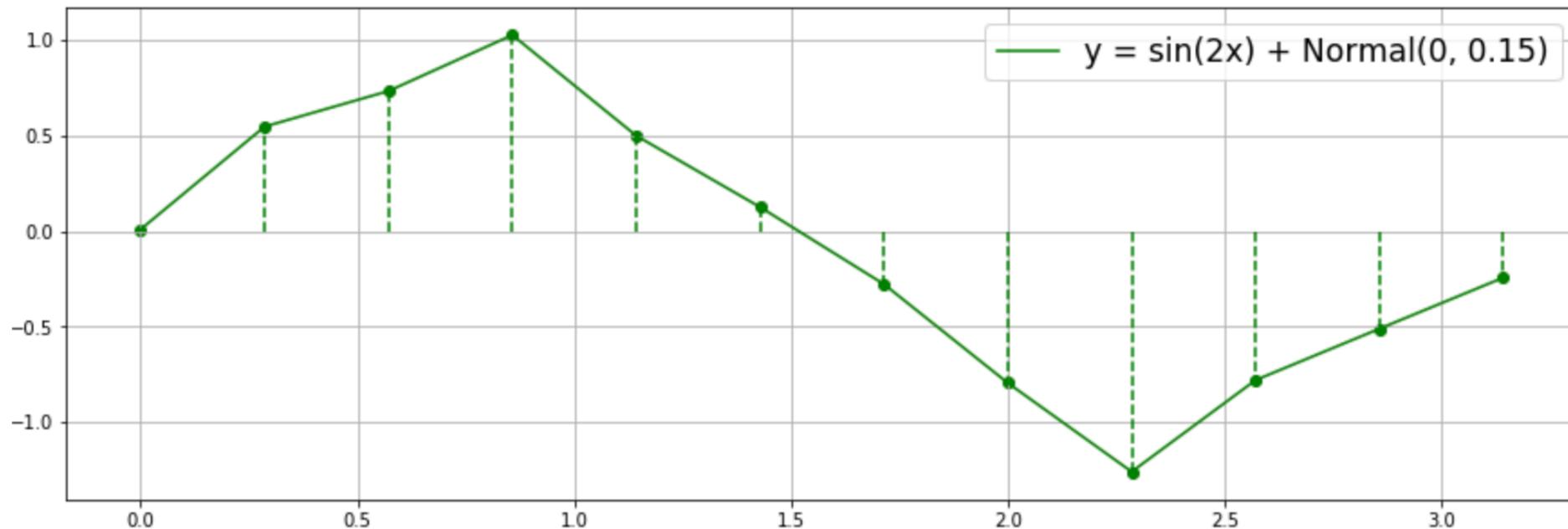
“Чистый” сигнал  $y = \sin(2x)$



0.00 0.54 0.91 0.99 0.76 0.28 -0.28 -0.76 -0.99 -0.91 -0.54 0.00

# Сглаживание сверткой (1D)

“Шумный” сигнал  $y = \sin(2x) + \text{Normal}(0, 0.15)$



0.01 0.55 0.73 1.03 0.50 0.13 -0.28 -0.79 -1.26 -0.78 -0.51 -0.24

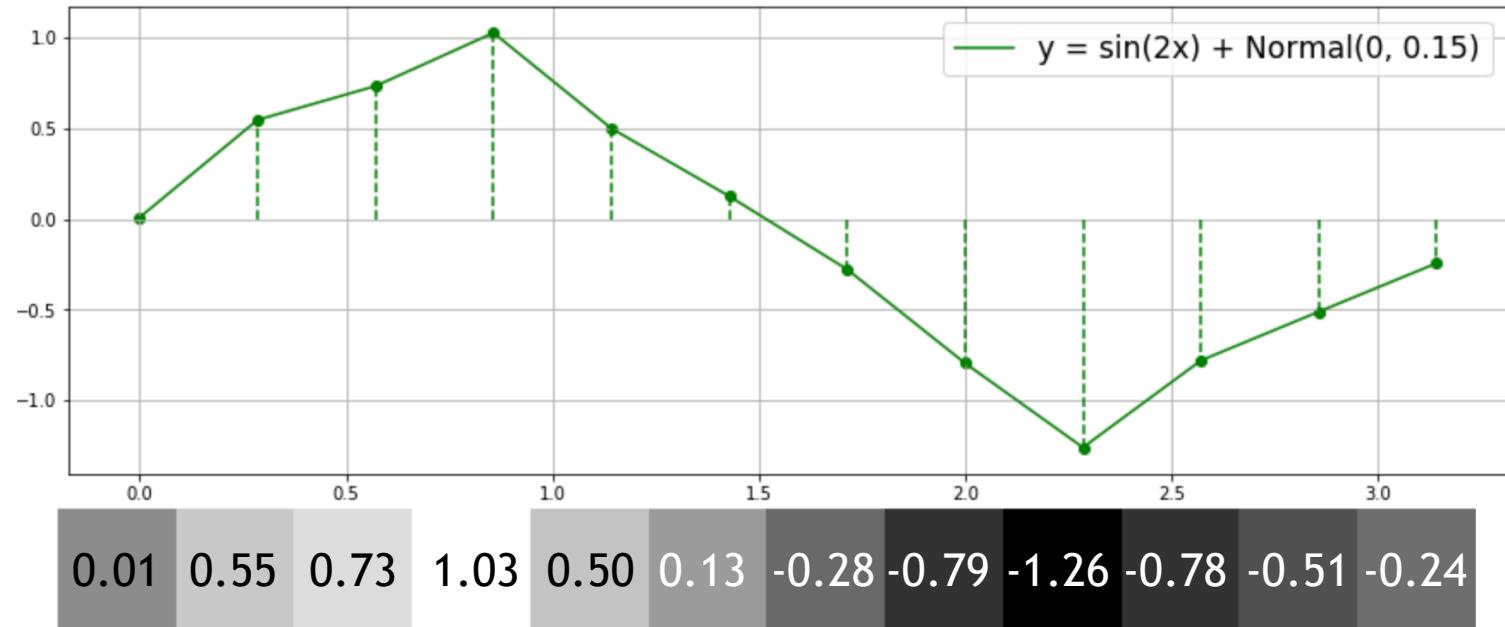


# Сглаживание сверткой (1D)

---

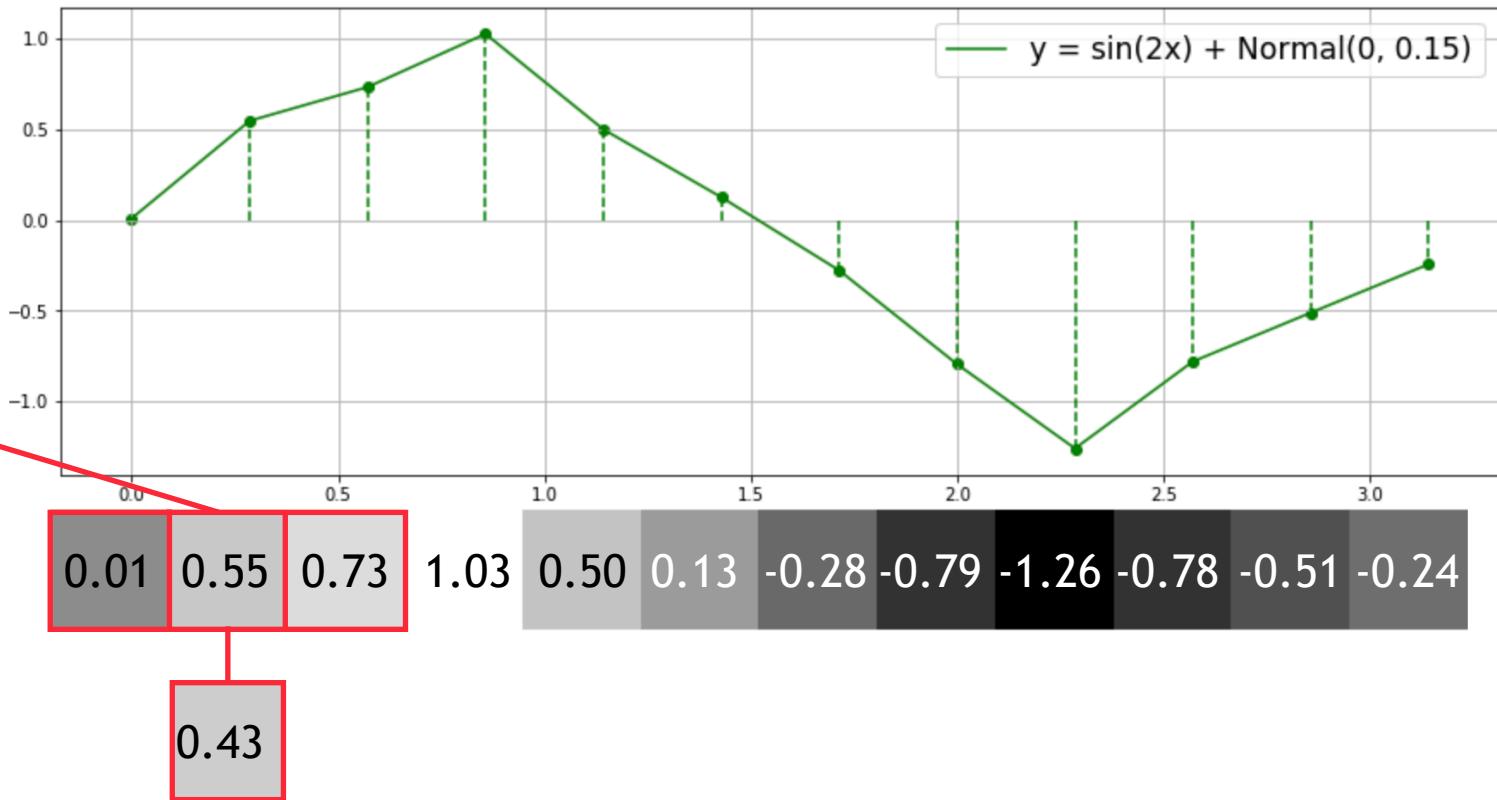
- Сделаем шумный одномерный сигнал более “гладким”
- Будем усреднять значения сигнала по трем точкам ( $i - 1, i, i + 1$ )
  - Размер свертки 3
    - Ядро свертки =  $\left[ \frac{1}{3}; \frac{1}{3}; \frac{1}{3} \right]$
    - Сворачивание с ядром из одинаковых элементов называется **box filtering**

$$\begin{array}{|c|c|c|} \hline \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \hline \end{array}$$



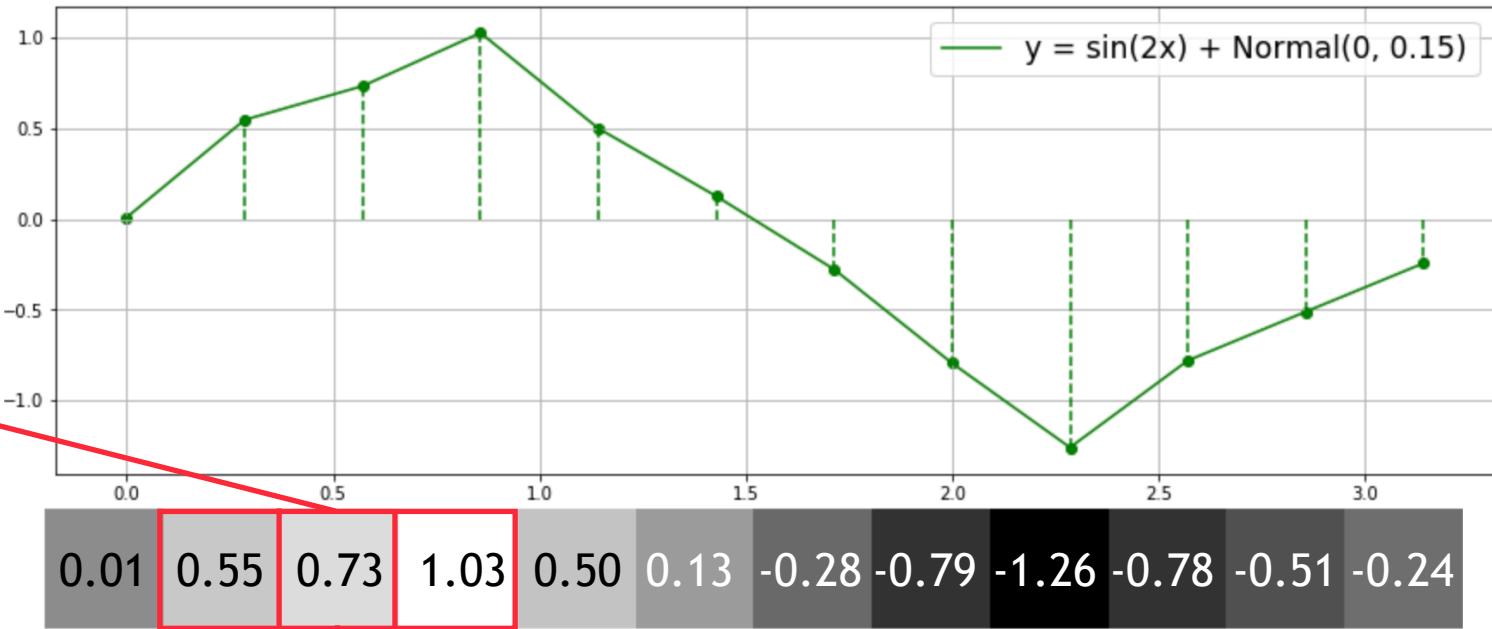
$$\begin{array}{|c|c|c|} \hline \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \hline \end{array}$$

$$\frac{1}{3} \times 0.01 + \frac{1}{3} \times 0.55 + \frac{1}{3} \times 0.73 = 0.43$$



$$\begin{array}{|c|c|c|} \hline \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \hline \end{array}$$

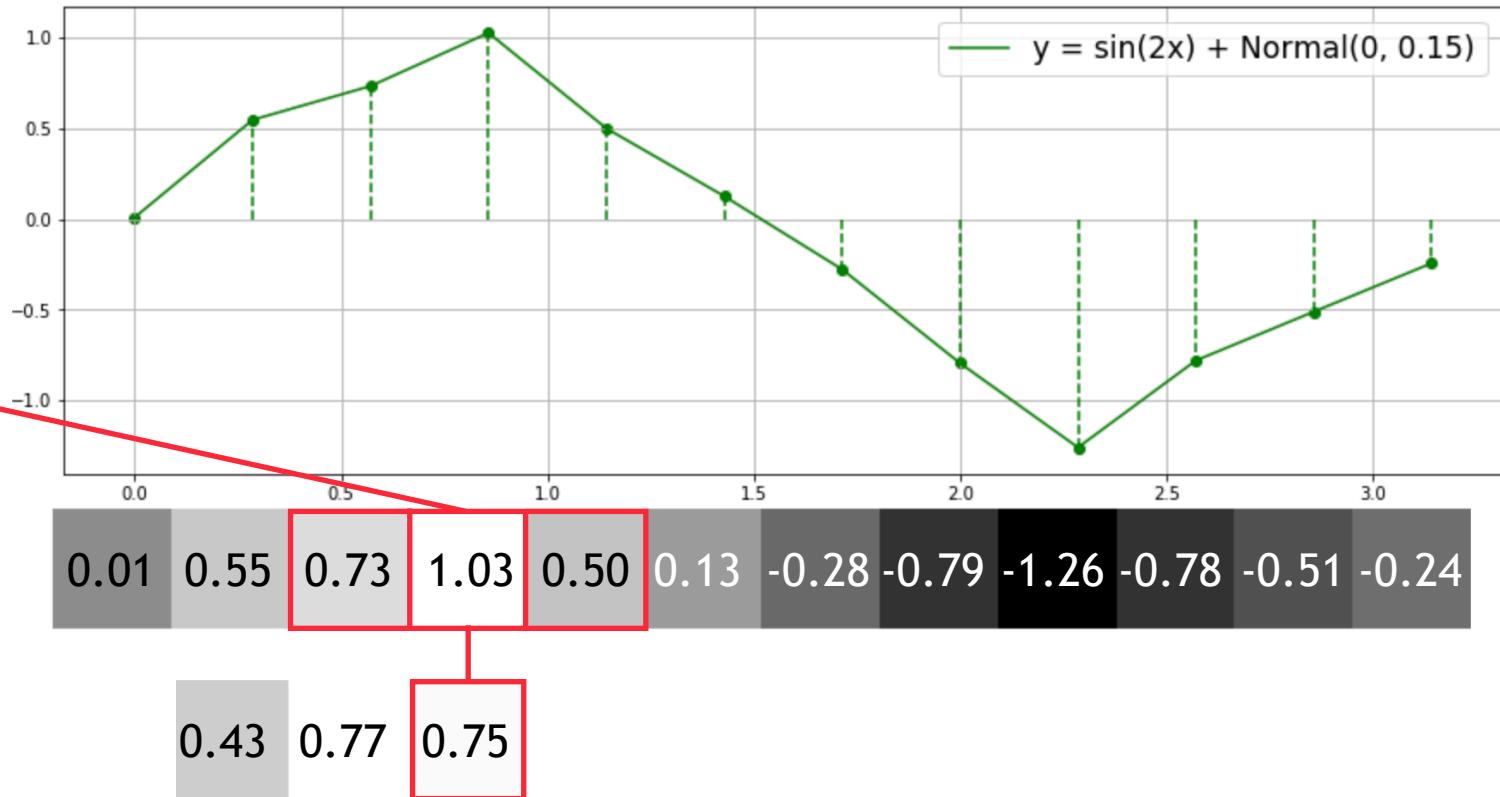
$$\frac{1}{3} \times 0.55 + \frac{1}{3} \times 0.73 + \frac{1}{3} \times 1.03 = 0.77$$



$$\begin{array}{|c|c|} \hline 0.43 & 0.77 \\ \hline \end{array}$$

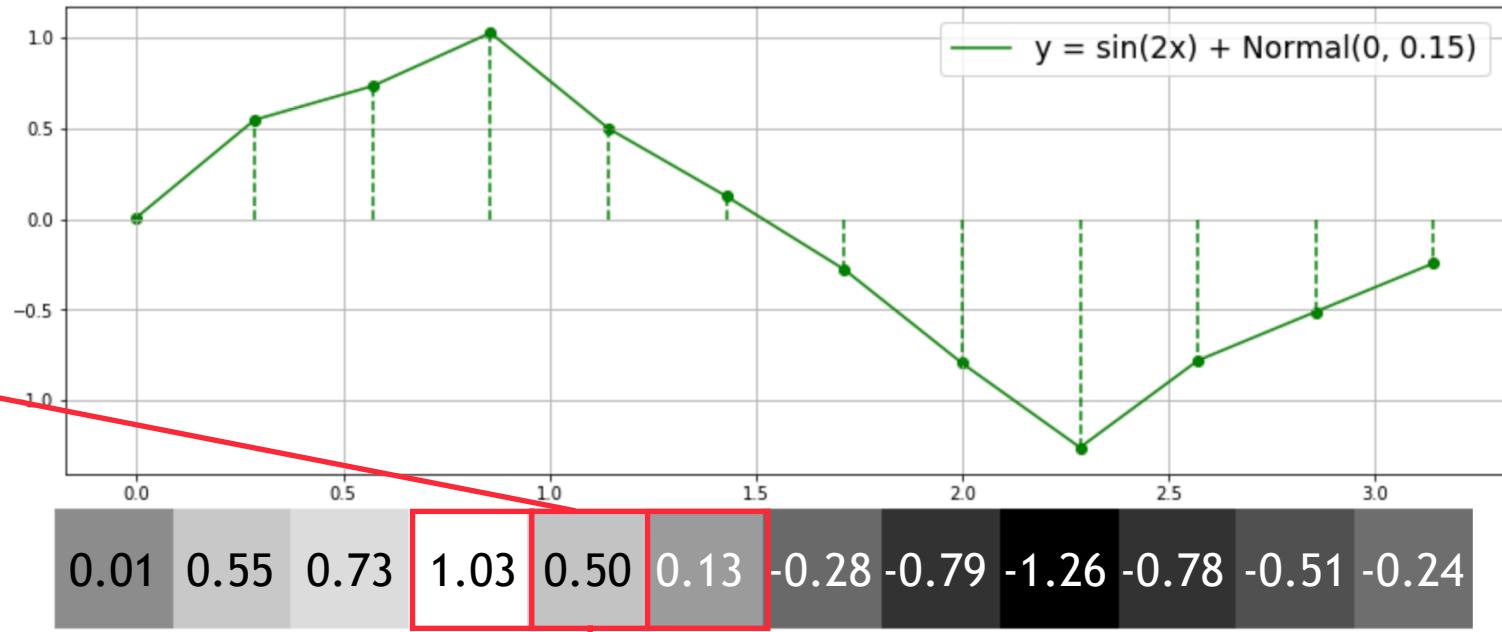
$$\begin{array}{|c|c|c|} \hline \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \hline \end{array}$$

$$\frac{1}{3} \times 0.73 + \frac{1}{3} \times 1.03 + \frac{1}{3} \times 0.50 = 0.75$$



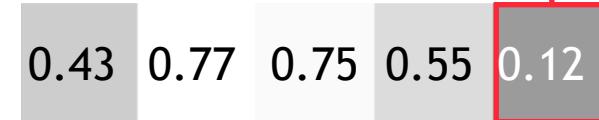
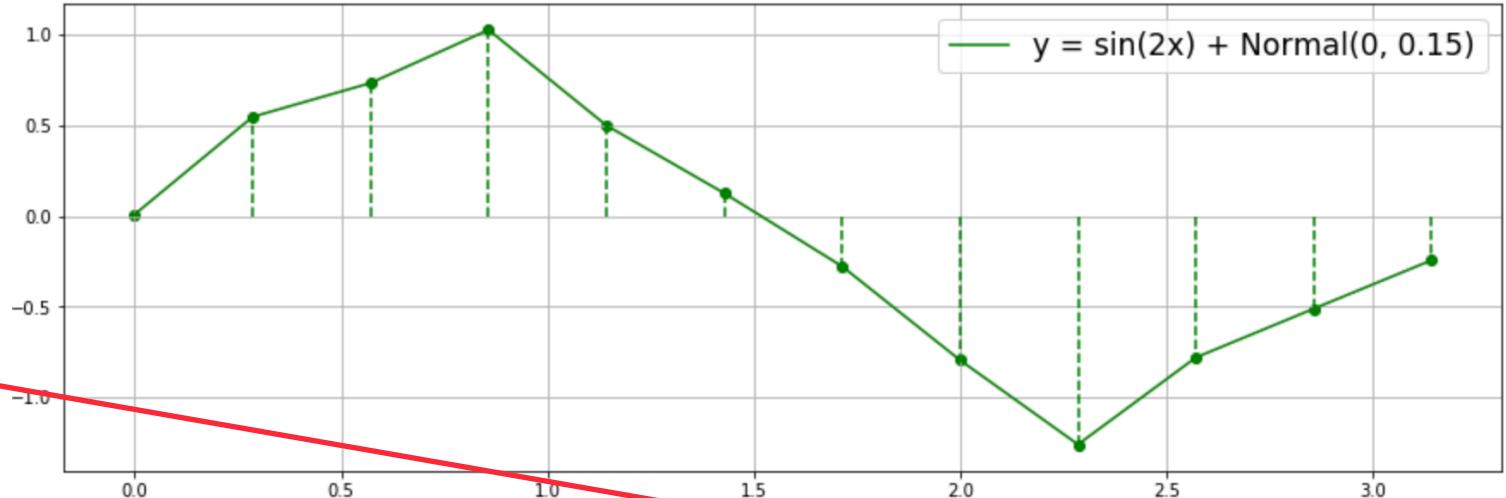
$$\begin{array}{|c|c|c|} \hline \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \hline \end{array}$$

$$\frac{1}{3} \times 1.03 + \frac{1}{3} \times 0.50 + \frac{1}{3} \times 0.13 = 0.55$$



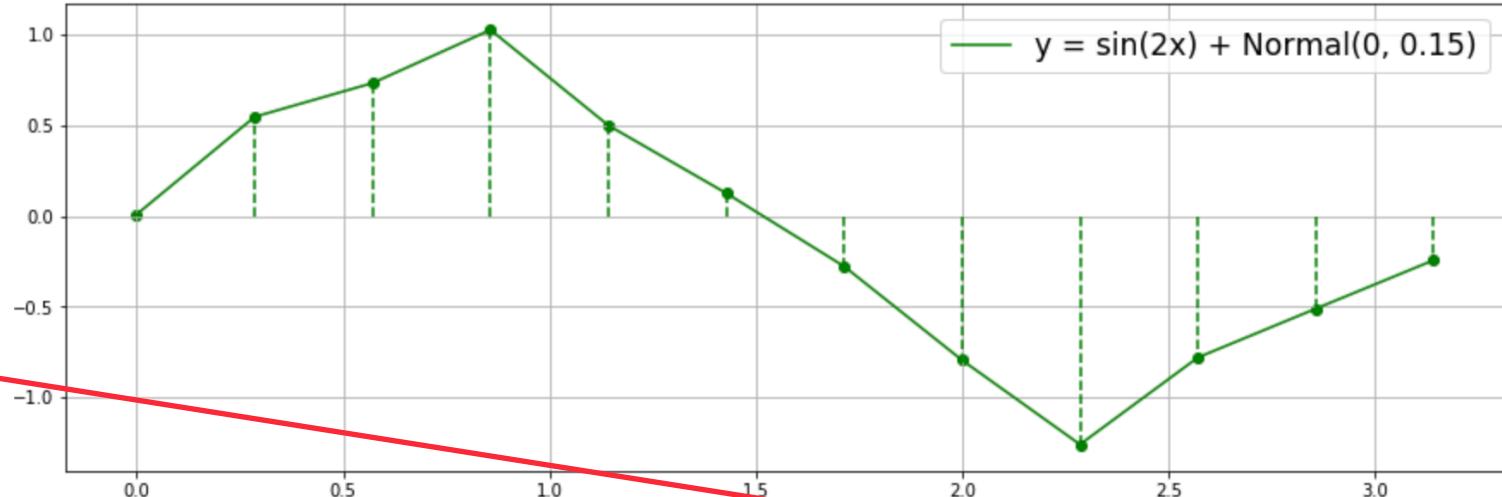
0.43	0.77	0.75	0.55
------	------	------	------

$$\begin{array}{|c|c|c|} \hline \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \hline \end{array}$$

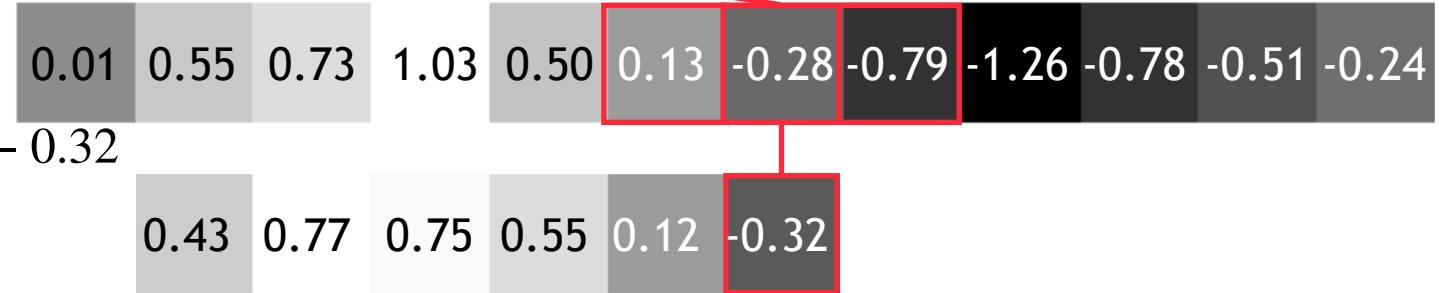


$$\frac{1}{3} \times 0.50 + \frac{1}{3} \times 0.13 + \frac{1}{3} \times (-0.28) = 0.12$$

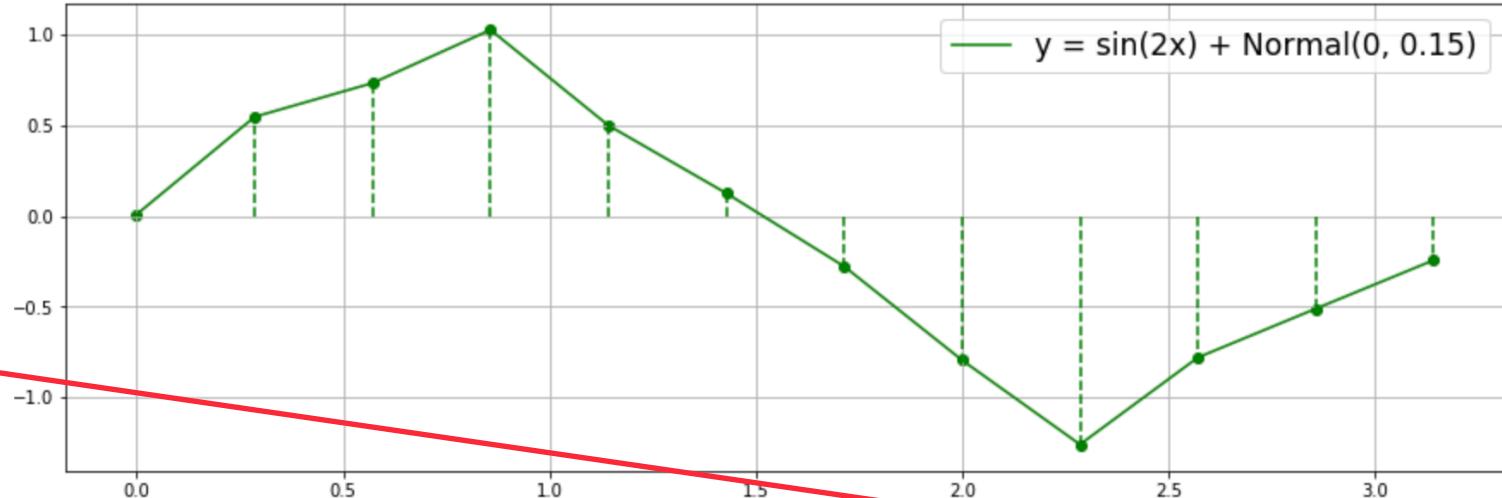
$$\begin{array}{|c|c|c|} \hline \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \hline \end{array}$$



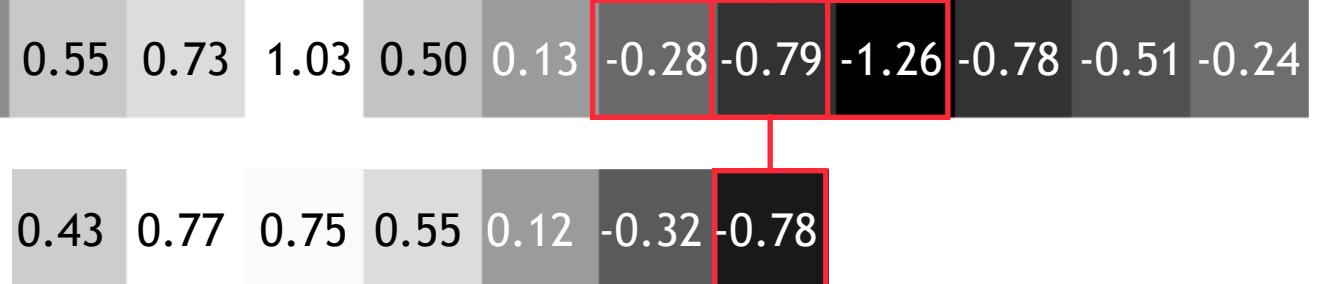
$$\frac{1}{3} \times (-0.13) + \frac{1}{3} \times (-0.28) + \frac{1}{3} \times (-0.79) = -0.32$$



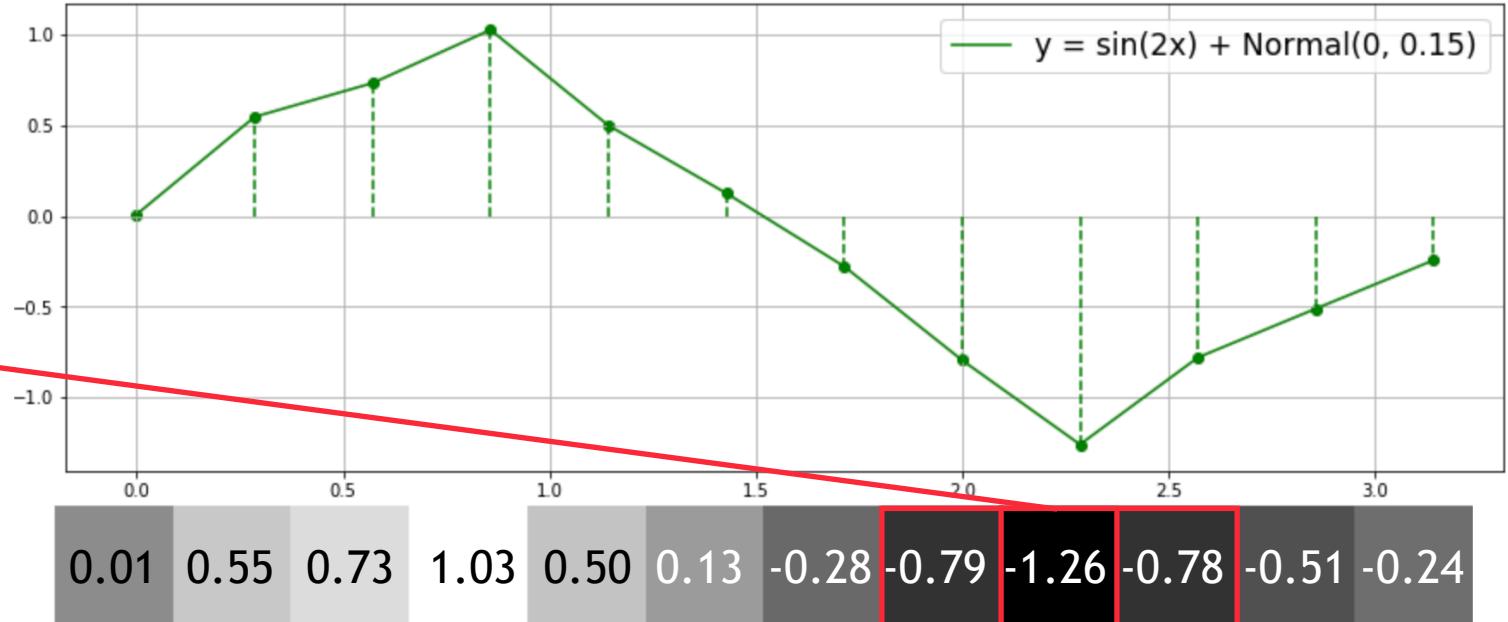
$$\begin{array}{|c|c|c|} \hline \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \hline \end{array}$$



$$\frac{1}{3} \times (-0.28) + \frac{1}{3} \times (-0.79) + \frac{1}{3} \times (-1.26) = -0.78$$



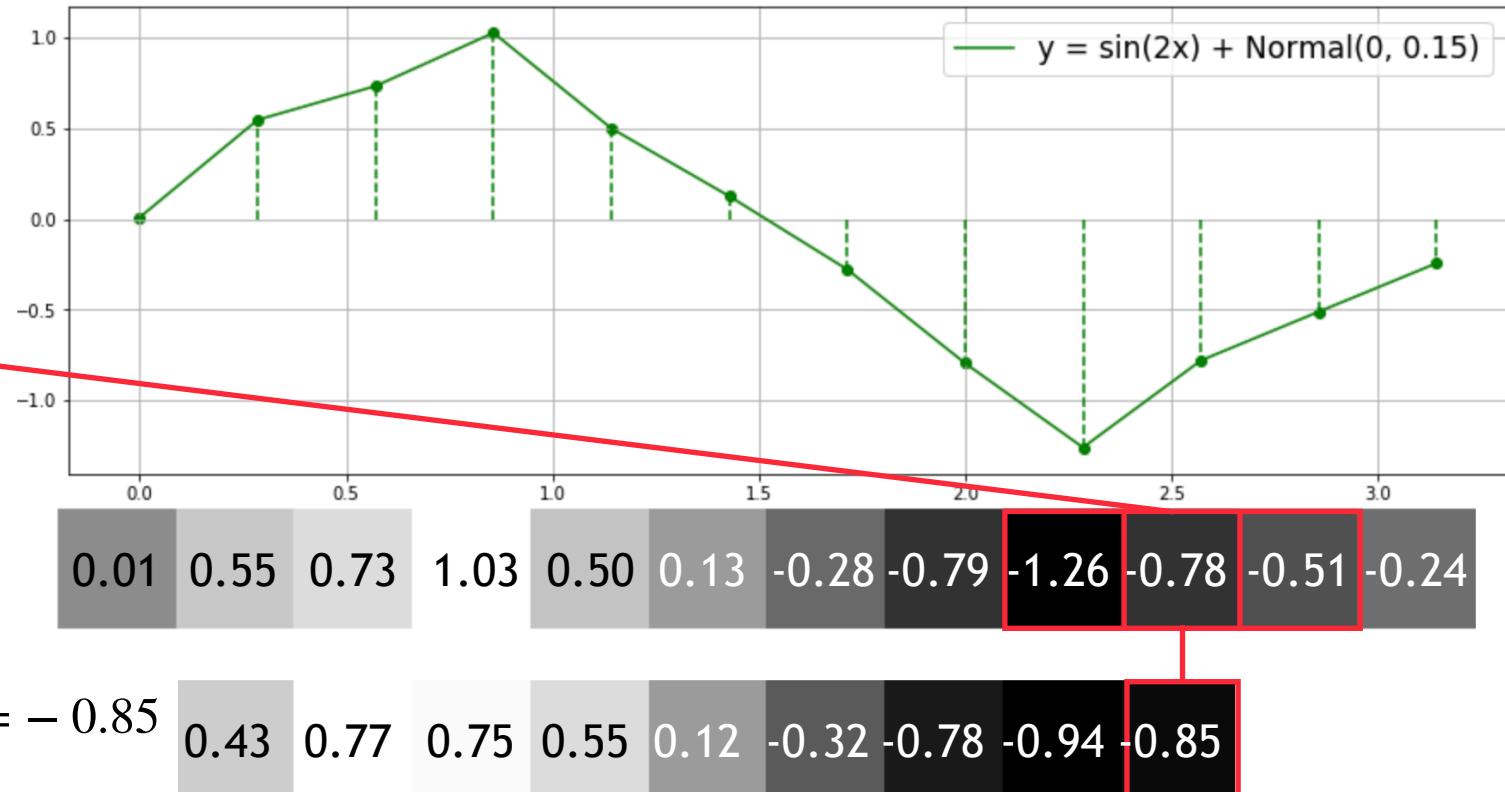
$$\begin{bmatrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \end{bmatrix}$$



$$\frac{1}{3} \times (-0.79) + \frac{1}{3} \times (-1.26) + \frac{1}{3} \times (-0.78) = -0.94$$

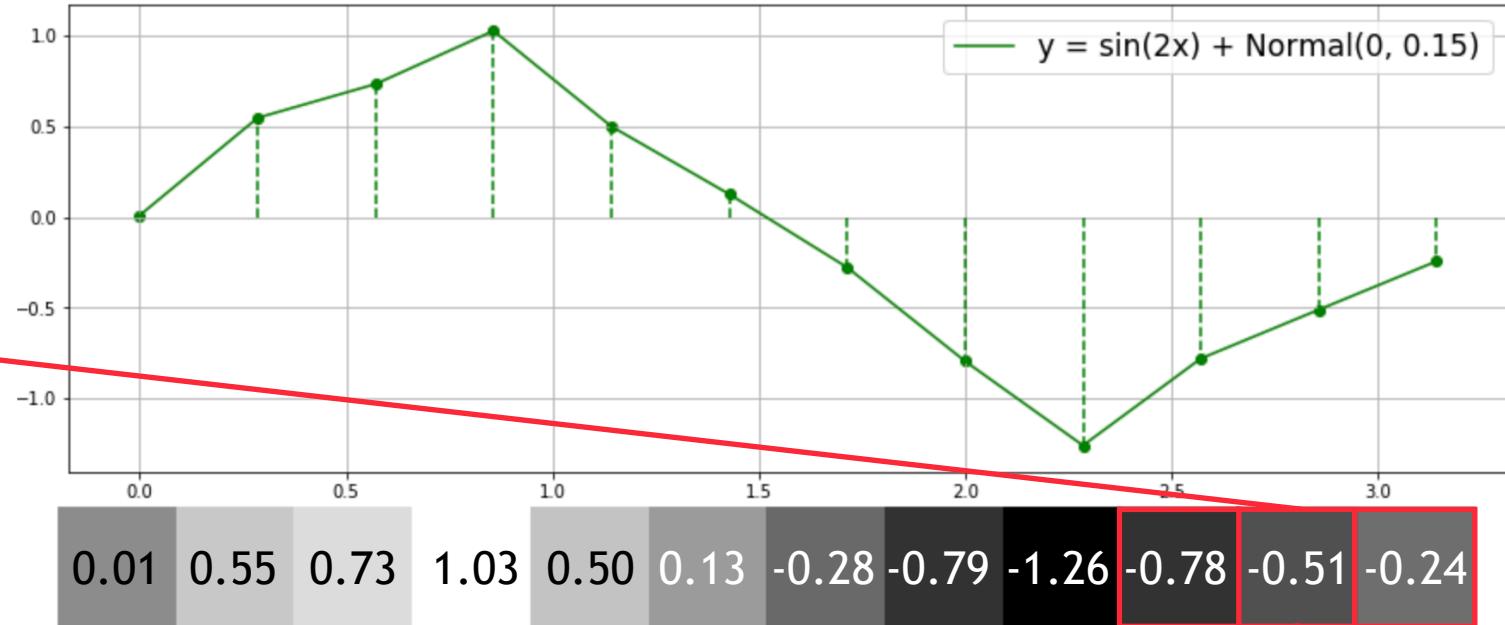
0.43	0.77	0.75	0.55	0.12	-0.32	-0.78	<b>-0.94</b>
------	------	------	------	------	-------	-------	--------------

$$\begin{array}{|c|c|c|} \hline \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \hline \end{array}$$



$$\frac{1}{3} \times (-1.26) + \frac{1}{3} \times (-0.78) + \frac{1}{3} \times (-0.51) = -0.85$$

$$\begin{array}{|c|c|c|} \hline \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \hline \end{array}$$

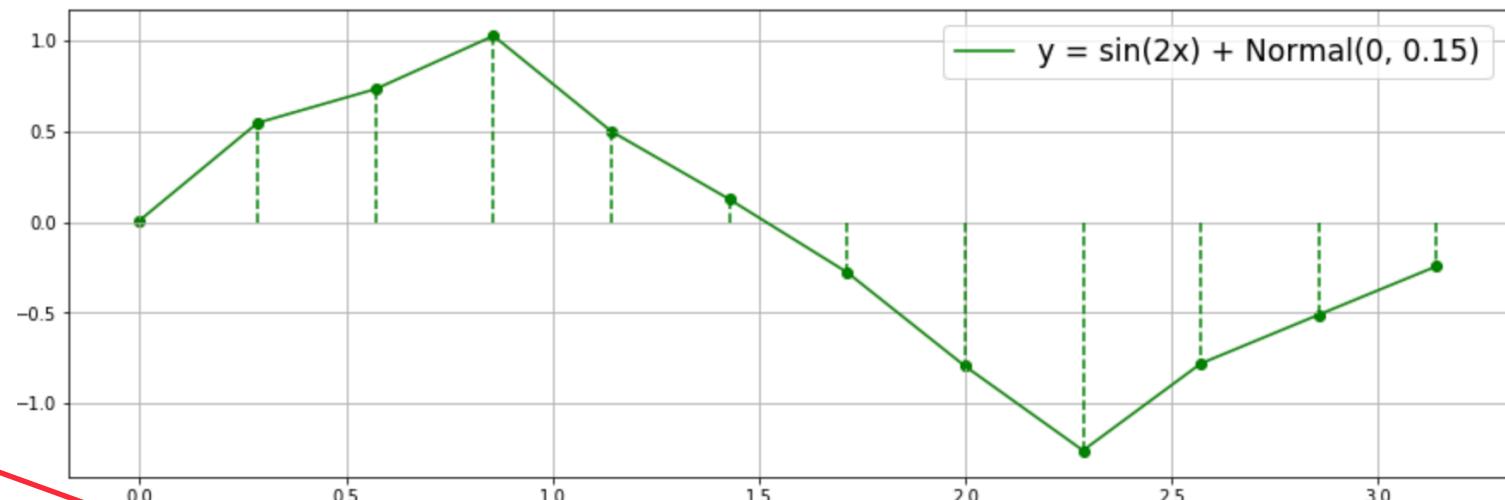
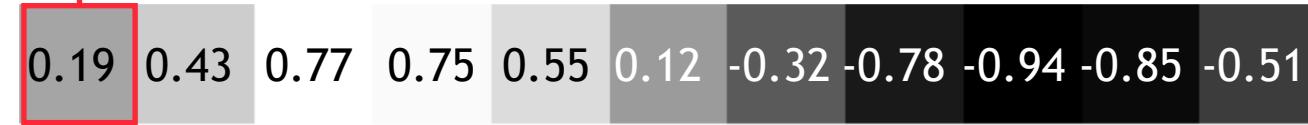


$$\frac{1}{3} \times (-0.78) + \frac{1}{3} \times (-0.51) + \frac{1}{3} \times (-0.24) = -0.51$$

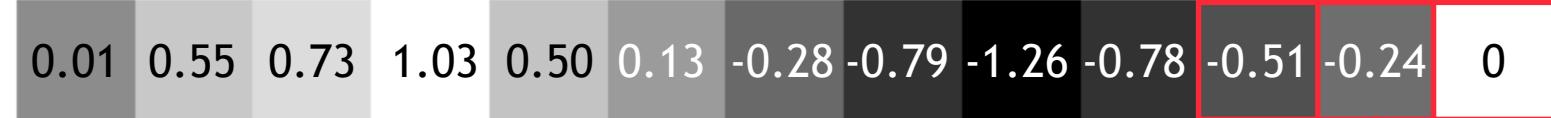
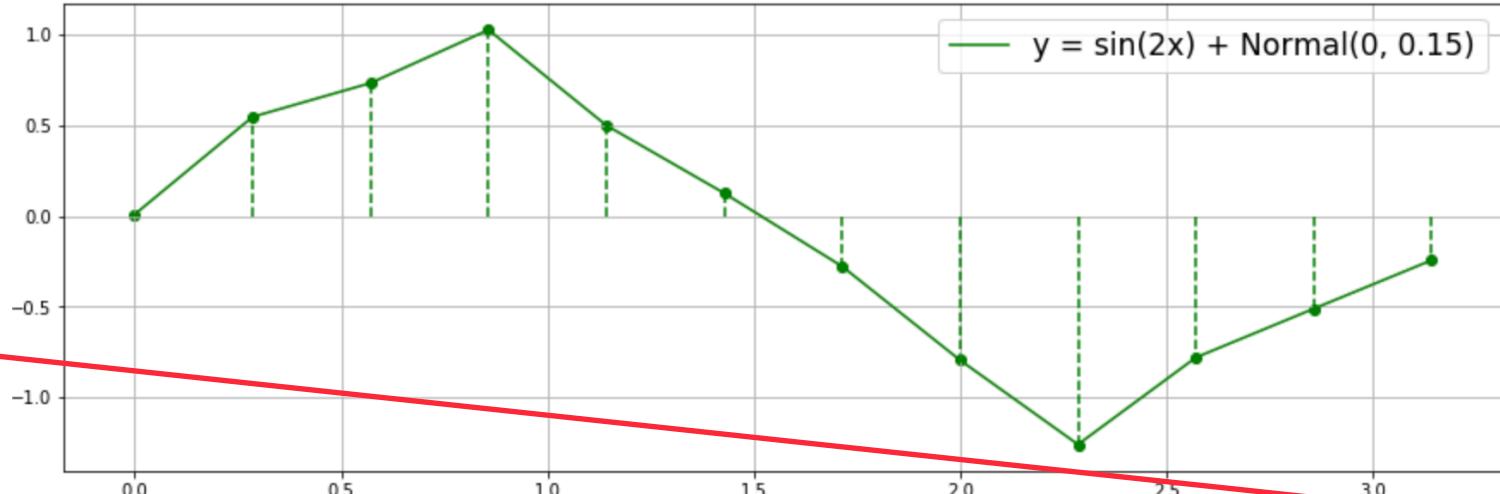
0.43	0.77	0.75	0.55	0.12	-0.32	-0.78	-0.94	-0.85	<b>-0.51</b>		
------	------	------	------	------	-------	-------	-------	-------	--------------	--	--

$$\begin{array}{|c|c|c|} \hline \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \hline \end{array}$$

$$\frac{1}{3} \times 0 + \frac{1}{3} \times 0.01 + \frac{1}{3} \times 0.55 = 0.19$$

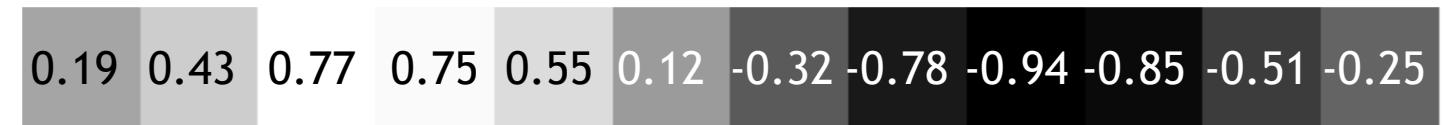
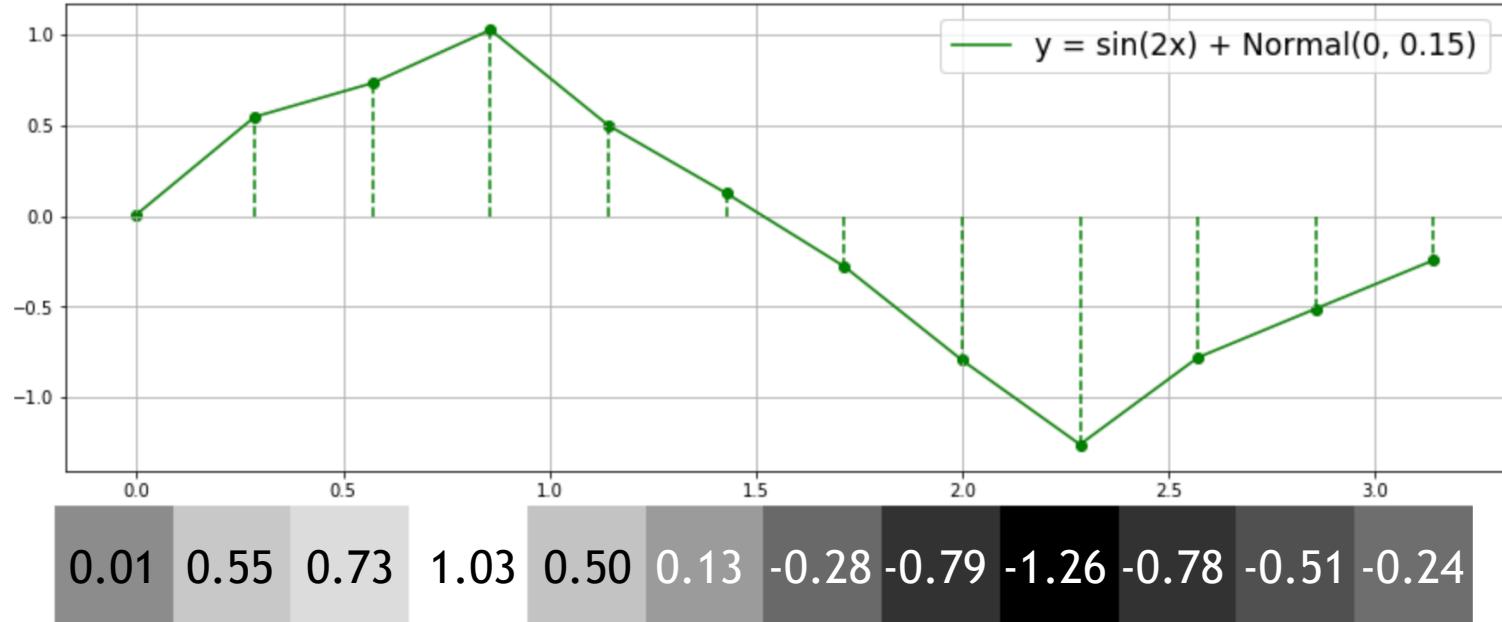


$$\begin{array}{|c|c|c|} \hline \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \hline \end{array}$$

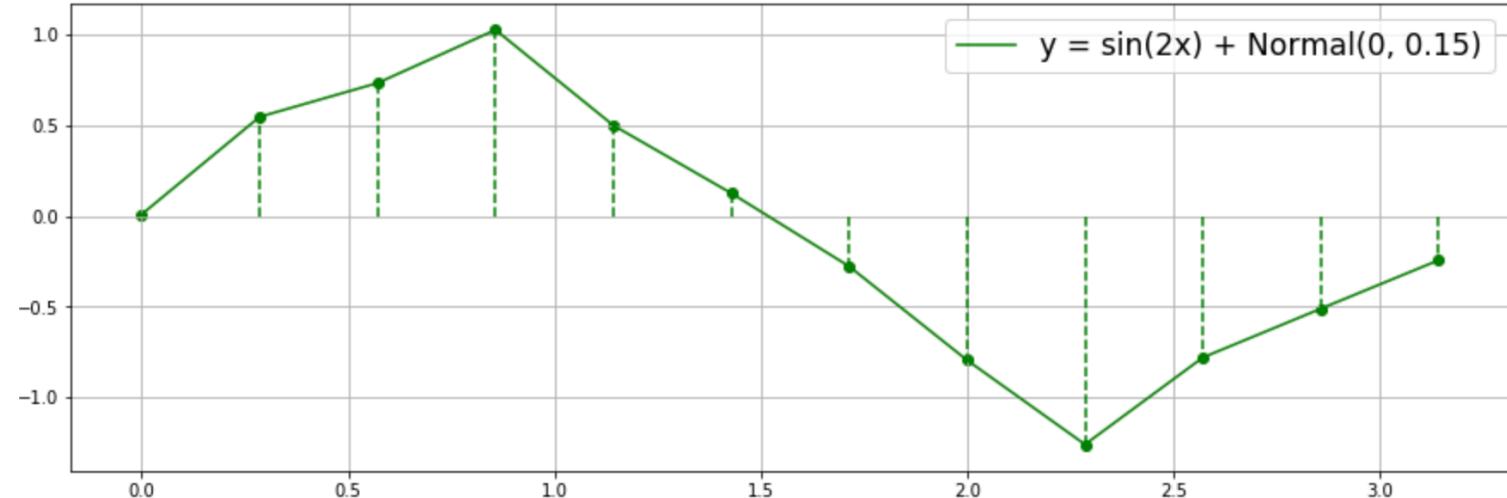


$$\frac{1}{3} \times (-0.51) + \frac{1}{3} \times (-0.24) + \frac{1}{3} \times 0 = -0.25$$

$$\begin{array}{|c|c|c|} \hline \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \hline \end{array}$$

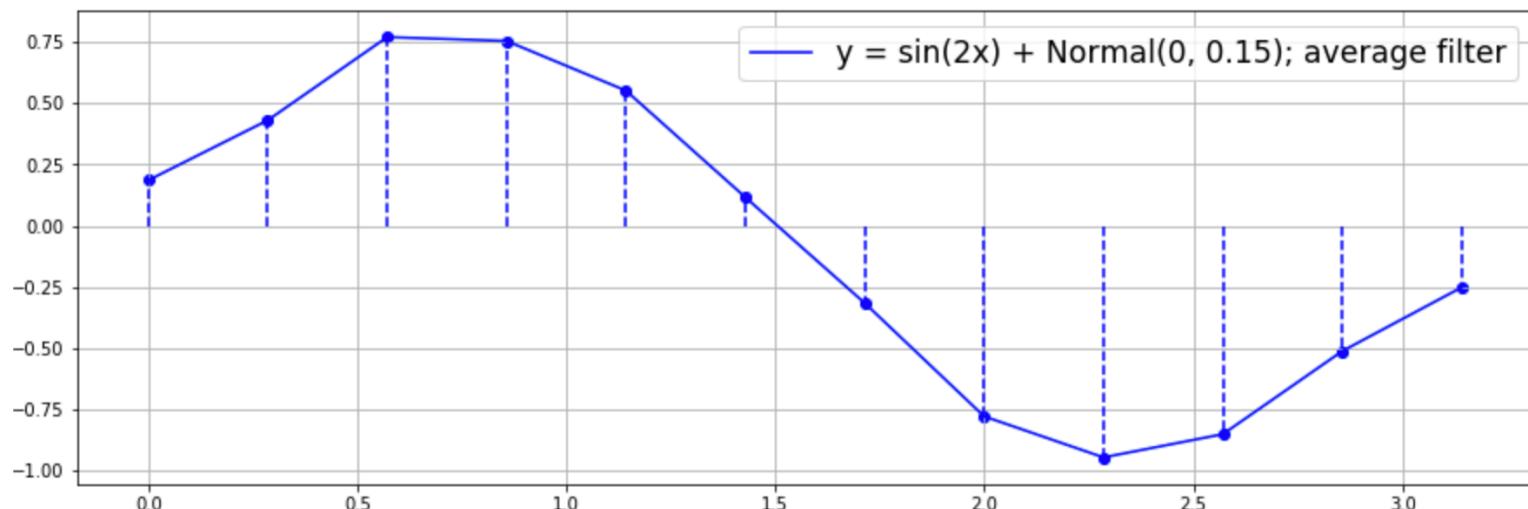


$$\begin{bmatrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \end{bmatrix}$$

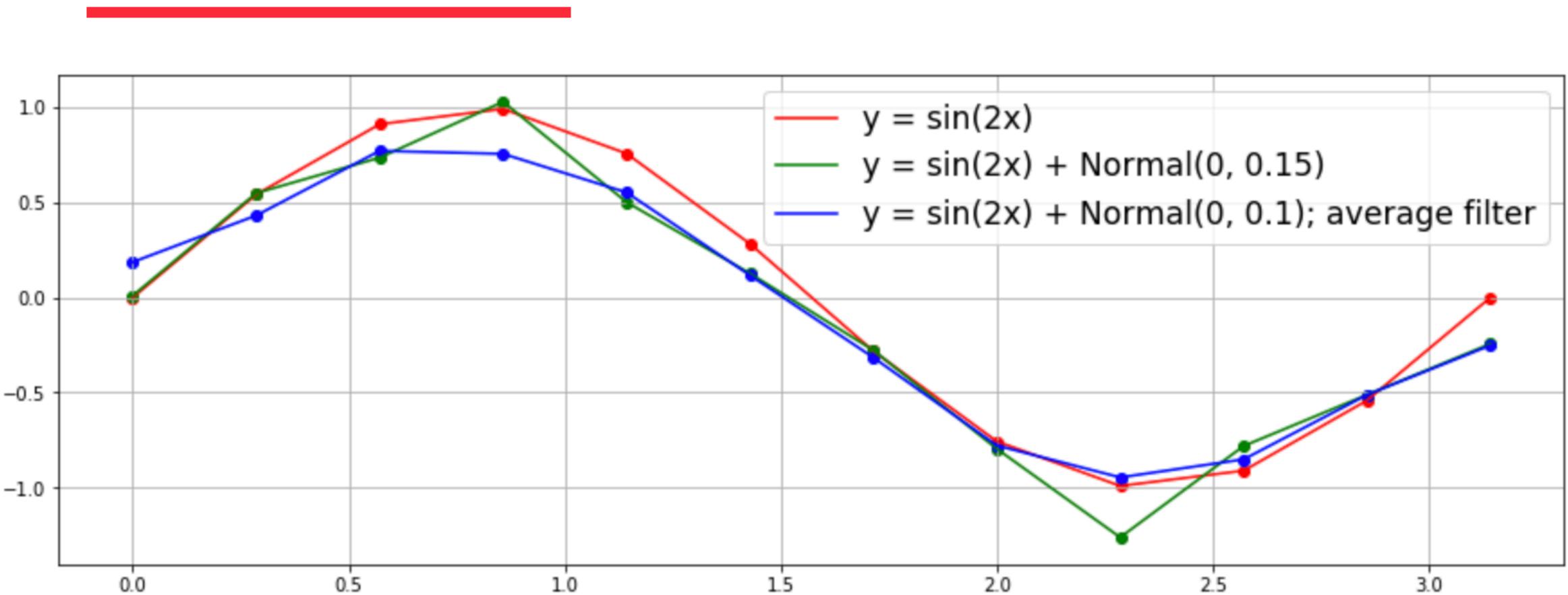


0.19	0.43	0.77	0.75	0.55	0.12	-0.32	-0.78	-0.94	-0.85	-0.51	-0.25
------	------	------	------	------	------	-------	-------	-------	-------	-------	-------

0.19	0.43	0.77	0.75	0.55	0.12	-0.32	-0.78	-0.94	-0.85	-0.51	-0.25
------	------	------	------	------	------	-------	-------	-------	-------	-------	-------



# Сглаживание сверткой (1D)



# Box Filter (2D)

---



1080px

$$\frac{1}{9} \times$$

1	1	1
1	1	1
1	1	1



# Box Filter (2D)

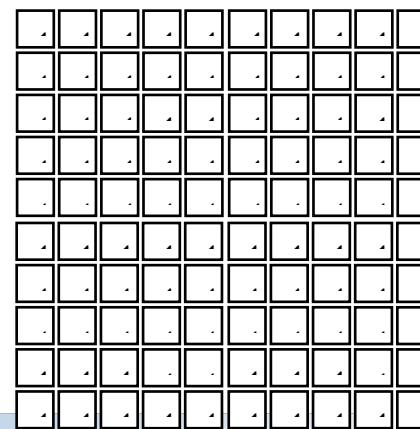
---



1080px



$$\frac{1}{11^2} \times$$



# Box Filter (2D)

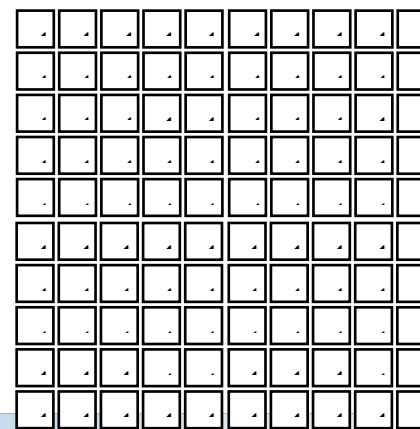
---



1080px



$$\frac{1}{51^2} \times$$



# Sobel, градиент, проекция на Ox (2D)

---



1080px



-1	0	1
-2	0	2
-1	0	1

# Sobel, градиент, проекция на Oy (2D)

---



1080px



-1	-2	-1
0	0	0
1	2	1

# Sobel, градиент, норма (2D)

---



1080px



-1	-2	-1
0	0	0
1	2	1



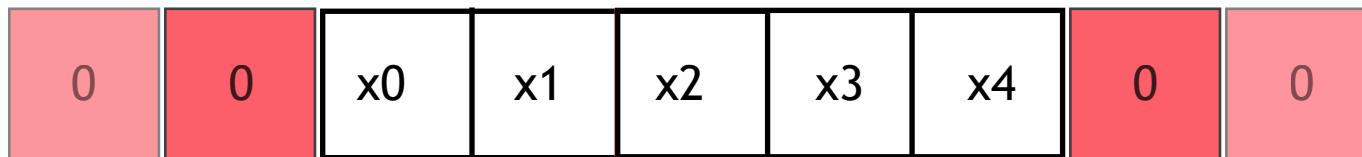
# Свойства свертки

---

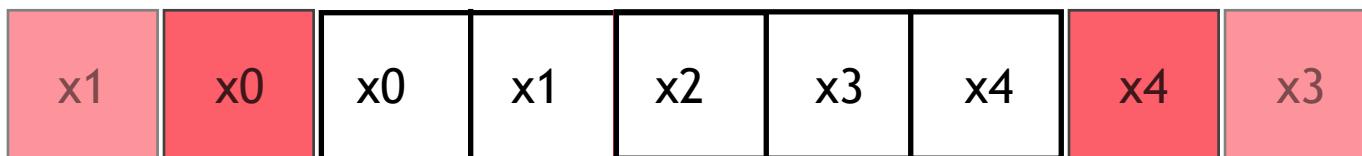
- Ассоциативность:  $a * (b * c) = (a * b) * c$
- Коммутативность:  $a * b = b * a$
- Линейность:
  - $(a + b) * c = a * c + b * c$
  - $(\alpha a) * b = \alpha (a * b)$

# Добавление отступа (Padding mode)

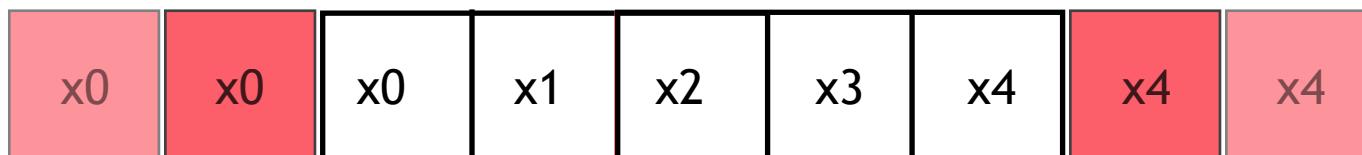
- Zero padding



- Reflect padding



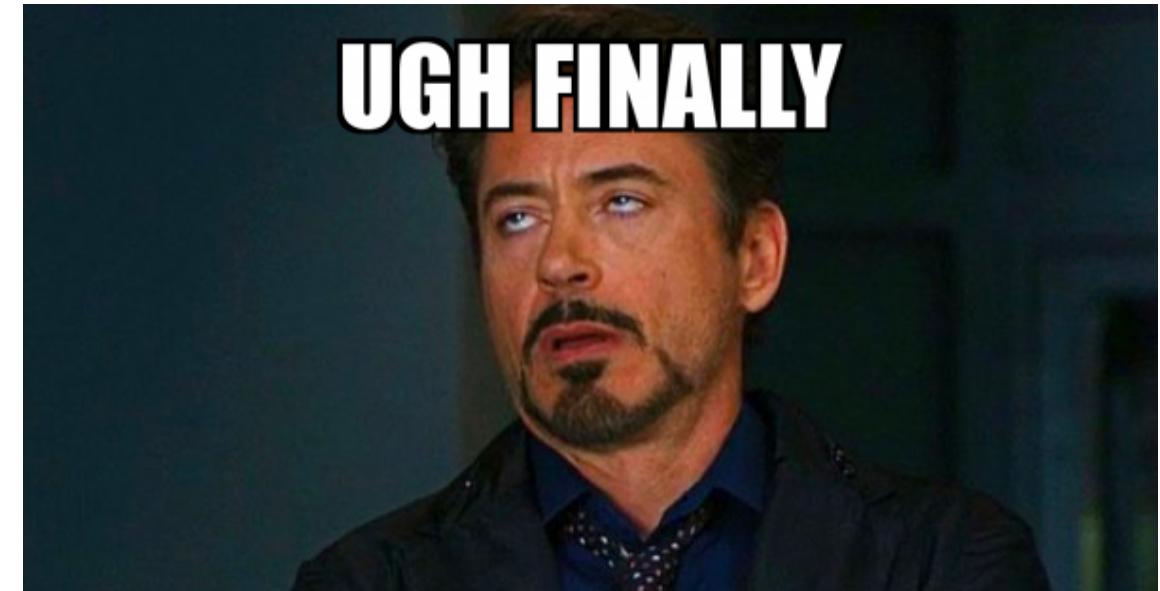
- Replicate padding

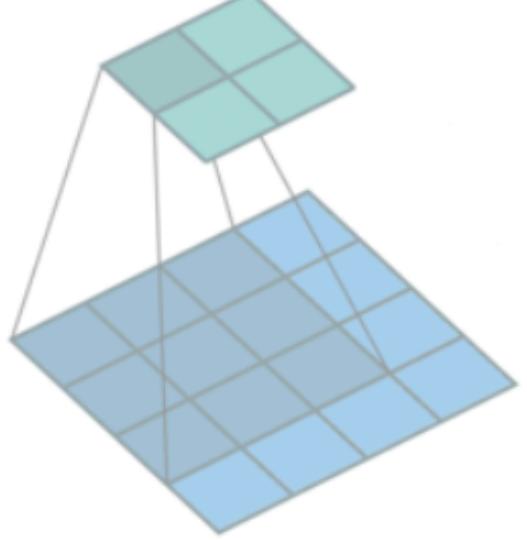


# Сверточный слой

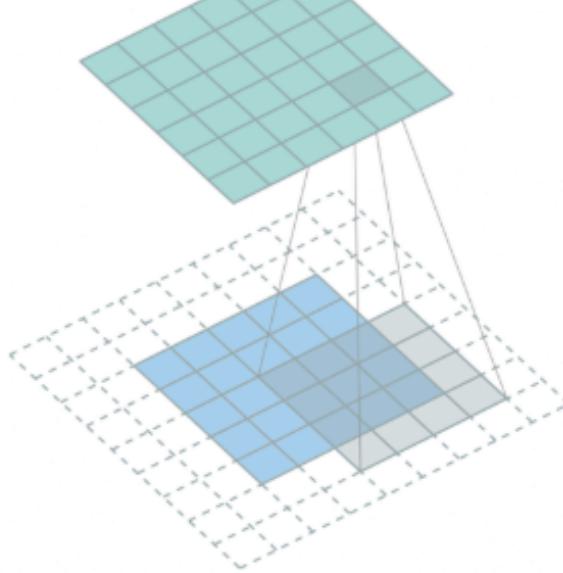
---

- В “классической” свертке ядро фиксировано (веса заданы)
- Что, если сделать веса ядра свертки обучаемыми?
  - Получится сверточный слой

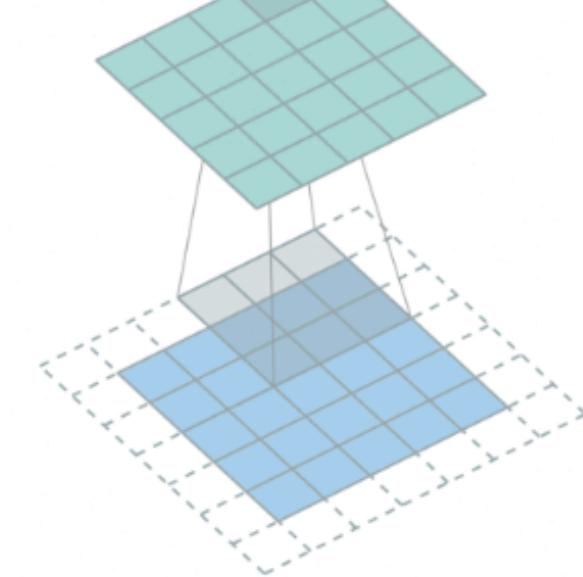




No padding, no strides



Arbitrary padding, no strides



Half padding, no strides

# Сверточный слой

# Сверточный слой

---

- Сверточный слой хранит тензор весов  $W$  размера  $D_{in} \times D_{out} \times size \times size$ :
  - $D_{in}$  - число каналов предыдущего слоя
  - $D_{out}$  - число каналов текущего слоя (= число различных сверток)
  - $size$  - размер ядра свертки (1, 3, 5, ...)
- У слоя может быть смещение (bias)  $b$  (размера  $D_{out}$ )
- Еще параметры слоя:
  - Padding (zero / reflect / replicate / ...)
  - Stride, dilation, ...

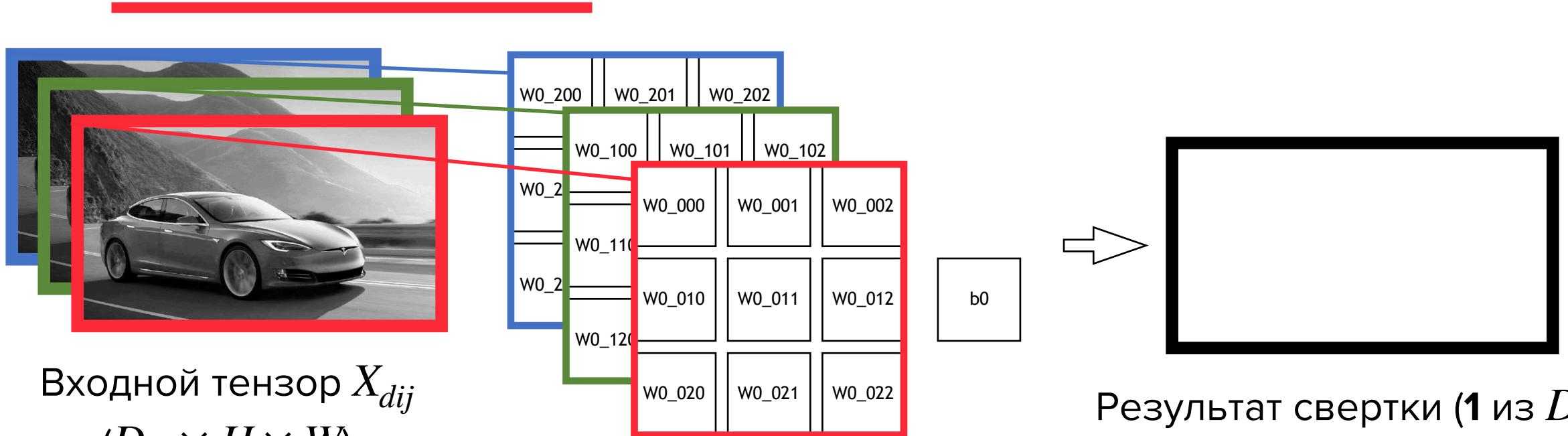
# Сверточный слой



Входной тензор  $X_{dij}$

$(D_{in} \times H \times W)$

# Сверточный слой



Входной тензор  $X_{dij}$

$(D_{in} \times H \times W)$

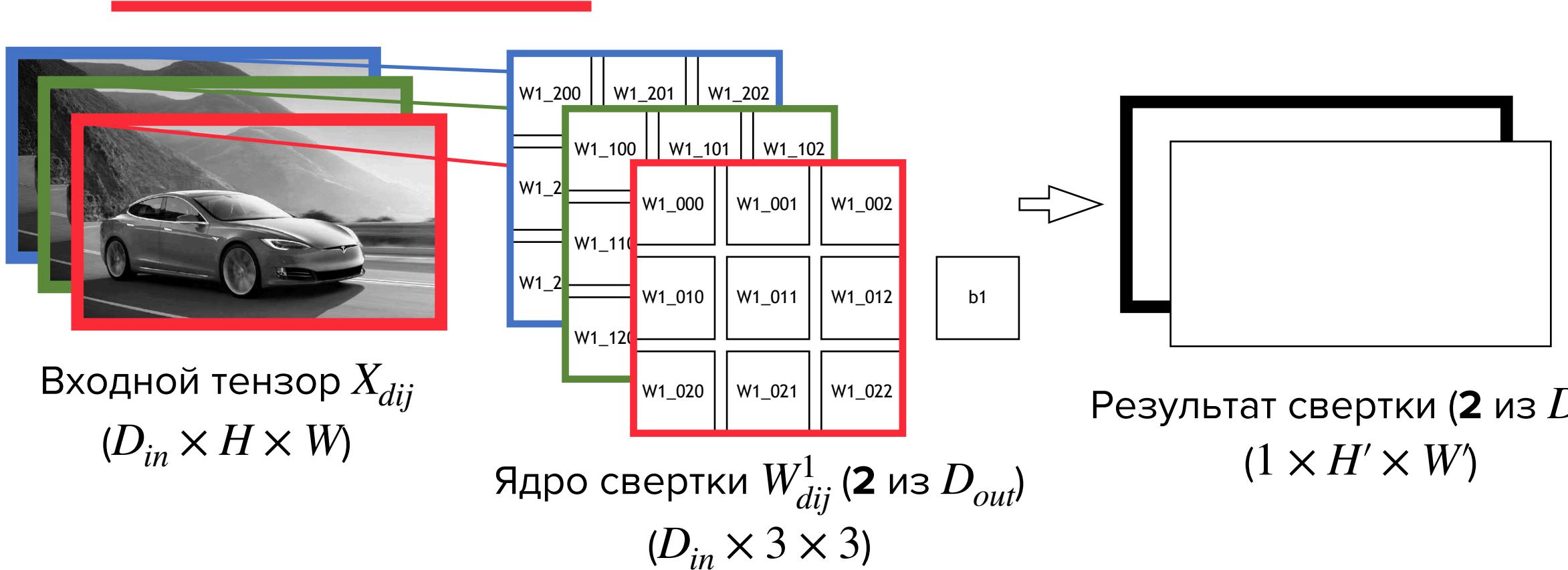
Ядро свертки  $W_{dij}^0$  (**1** из  $D_{out}$ )

$(D_{in} \times 3 \times 3)$

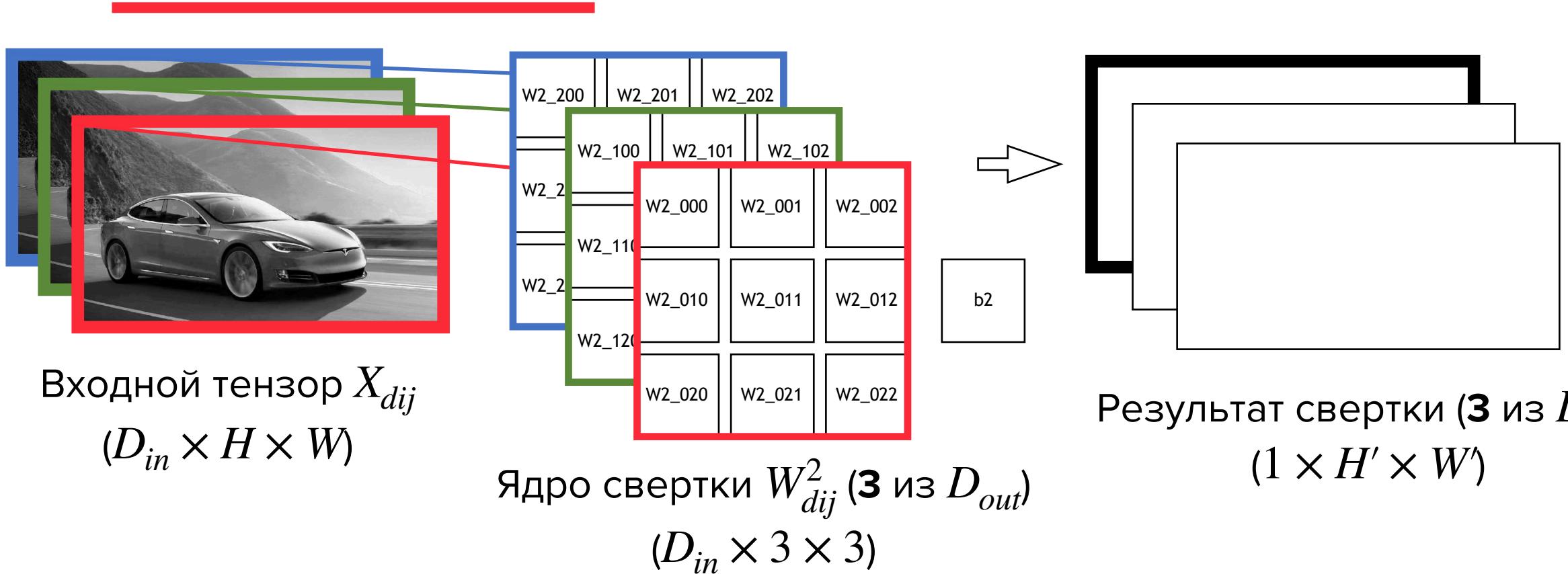
Результат свертки (**1** из  $D_{out}$ )

$(1 \times H' \times W')$

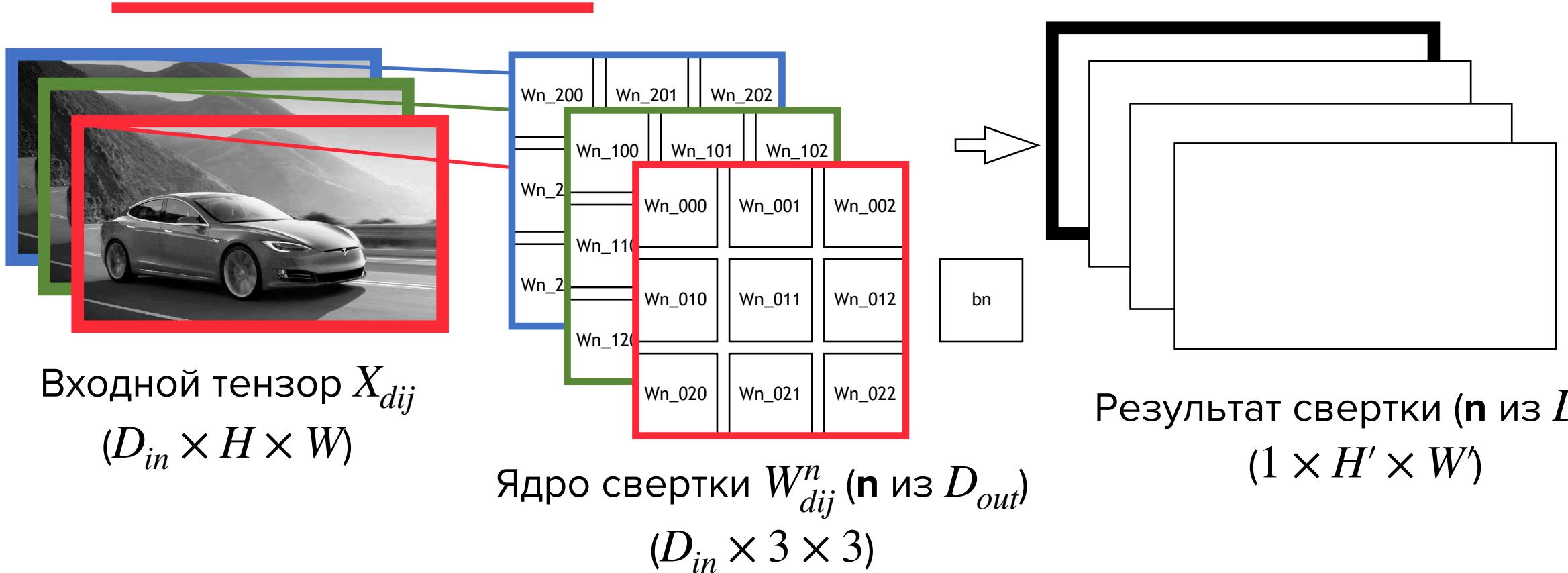
# Сверточный слой



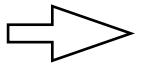
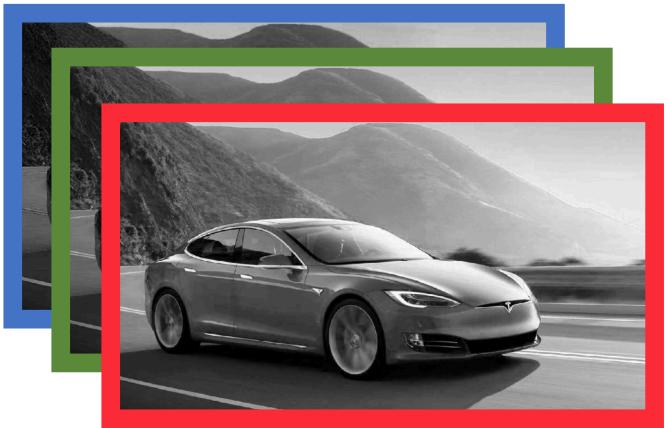
# Сверточный слой



# Сверточный слой

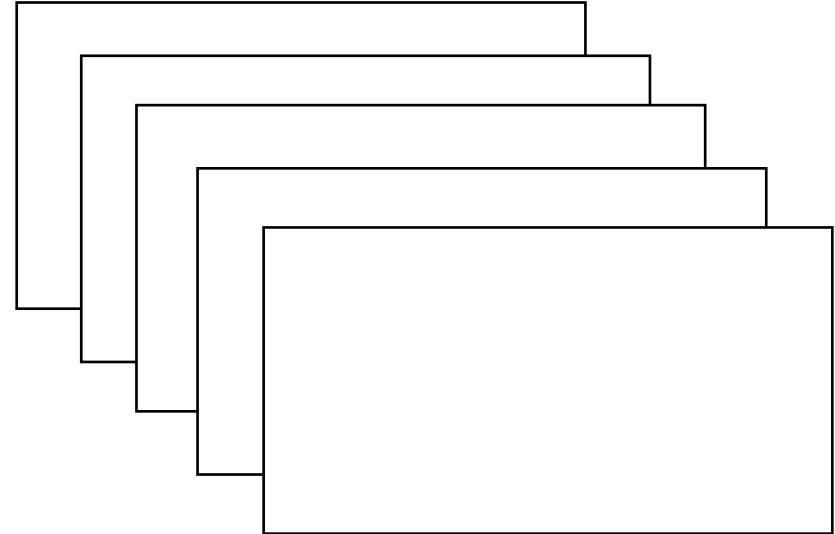


# Сверточный слой



Входной тензор  $X_{dij}$

$(D_{in} \times H \times W)$



Выходной тензор

$(D_{out} \times H' \times W')$



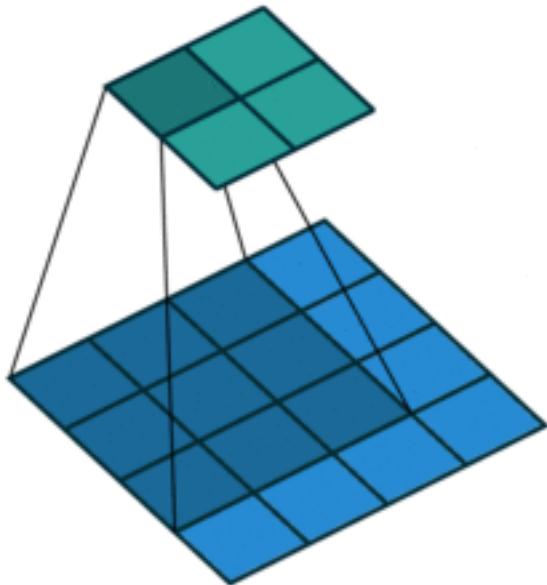
# Сверточный слой: padding

---

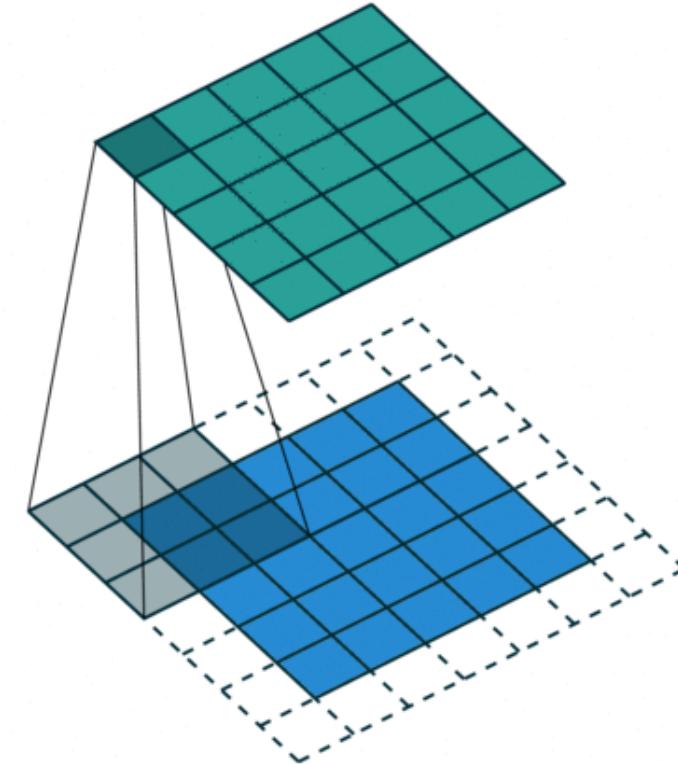
- Для сохранения размеров ( $h, w$ ) можно добавлять к тензору элементы на границе (**отступ, padding**)
- Величина padding влияет на пространственный размер выходного тензора

# Сверточный слой: padding

---



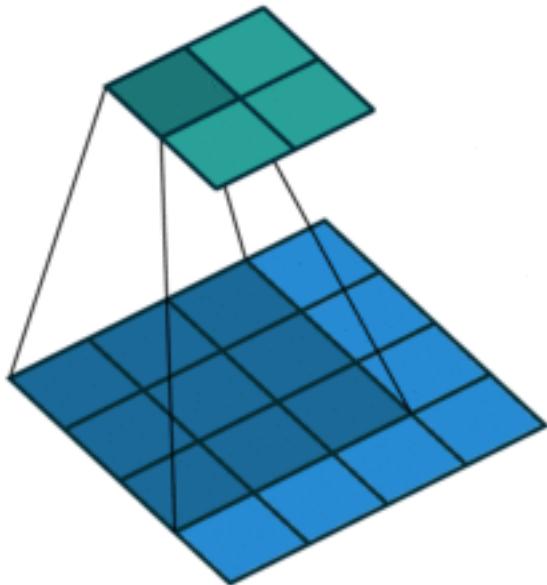
Padding = 0



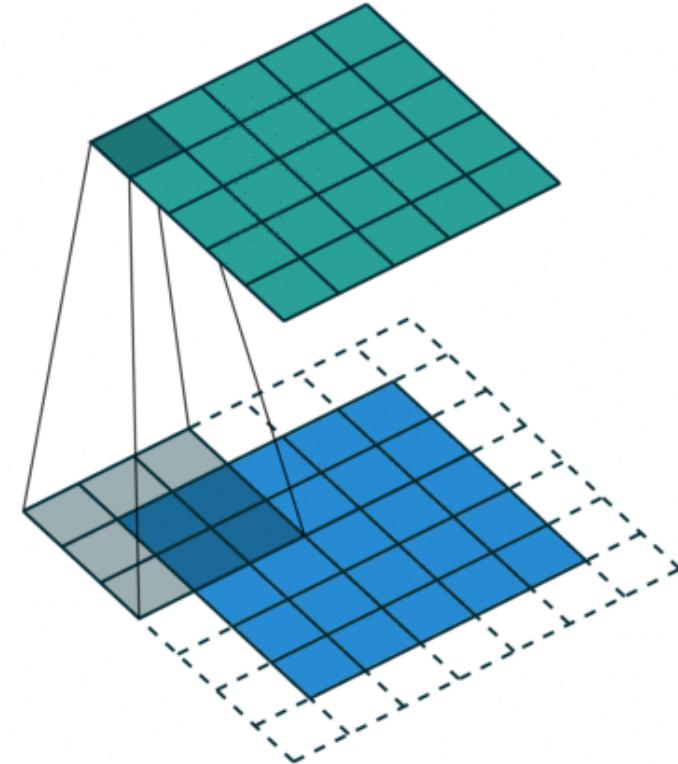
Padding = 1

# Сверточный слой: padding

---



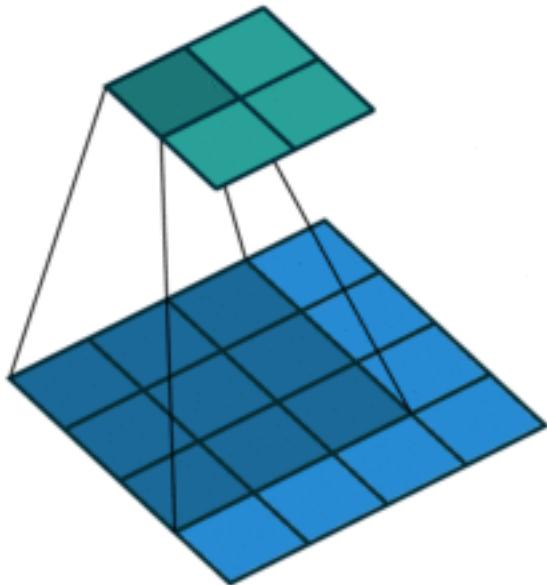
Padding = 0



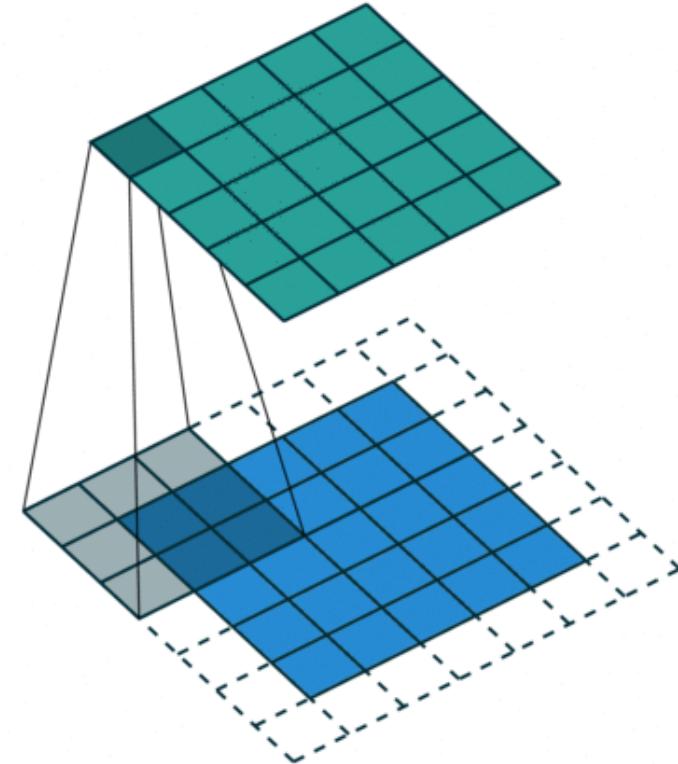
Padding = 1

# Сверточный слой: padding

---



Padding = 0



Padding = 1



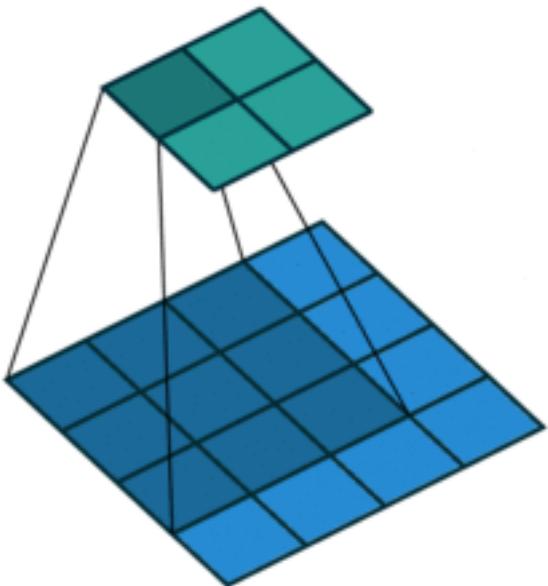
# Сверточный слой: шаг окна (stride)

---

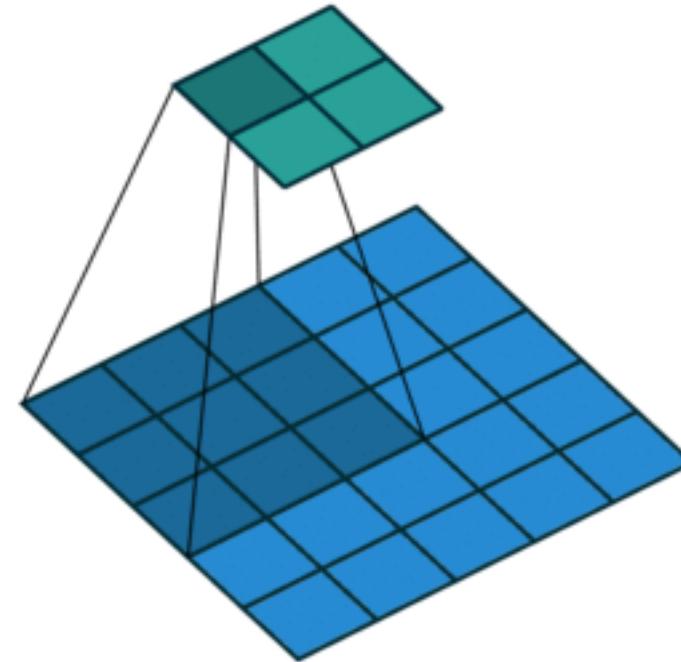
- “Шаг” ядра свертки вдоль осей x-y не обязан быть = 1
- Этот шаг называется **stride**
- Stride влияет на пространственный размер выходного тензора

# Сверточный слой: шаг окна (stride)

---



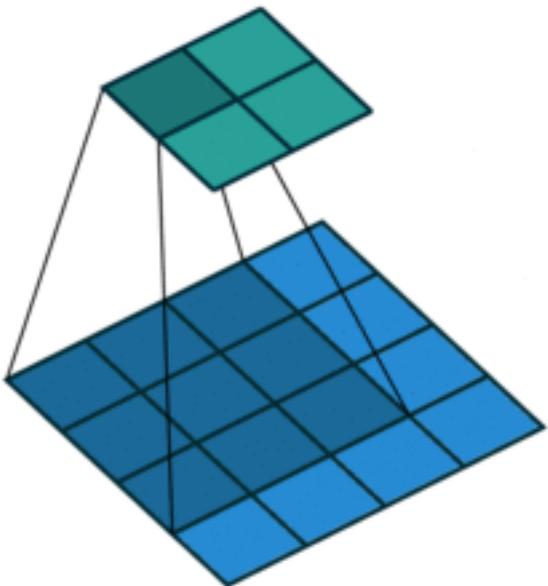
Stride = 1



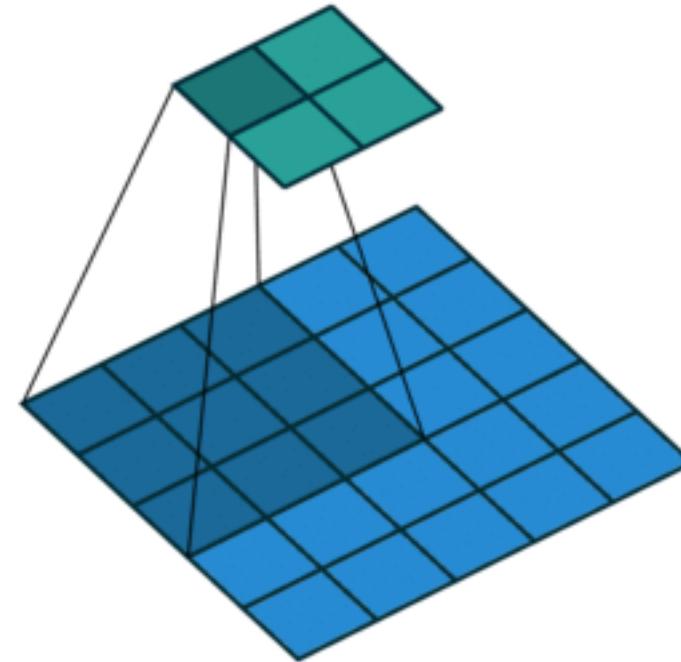
Stride = 2

# Сверточный слой: шаг окна (stride)

---



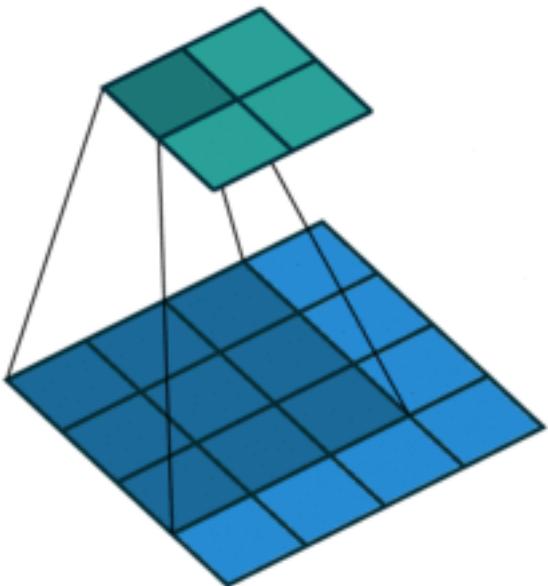
Stride = 1



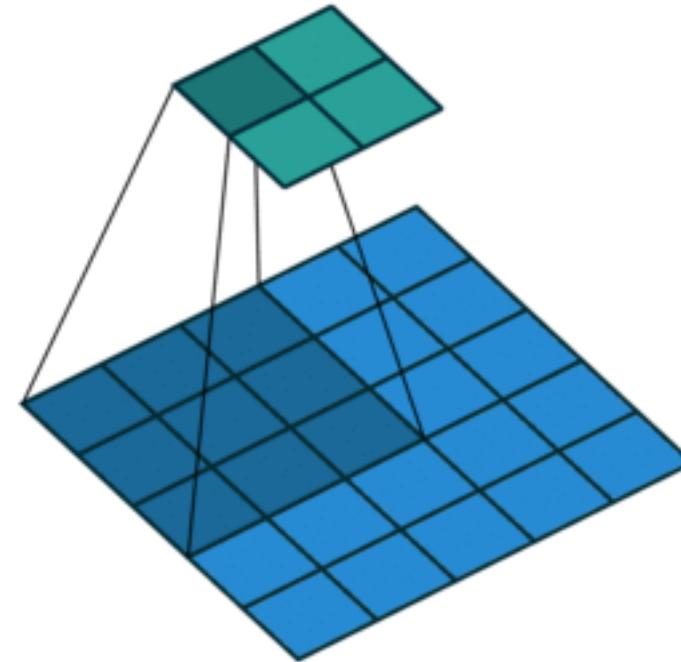
Stride = 2

# Сверточный слой: шаг окна (stride)

---



Stride = 1

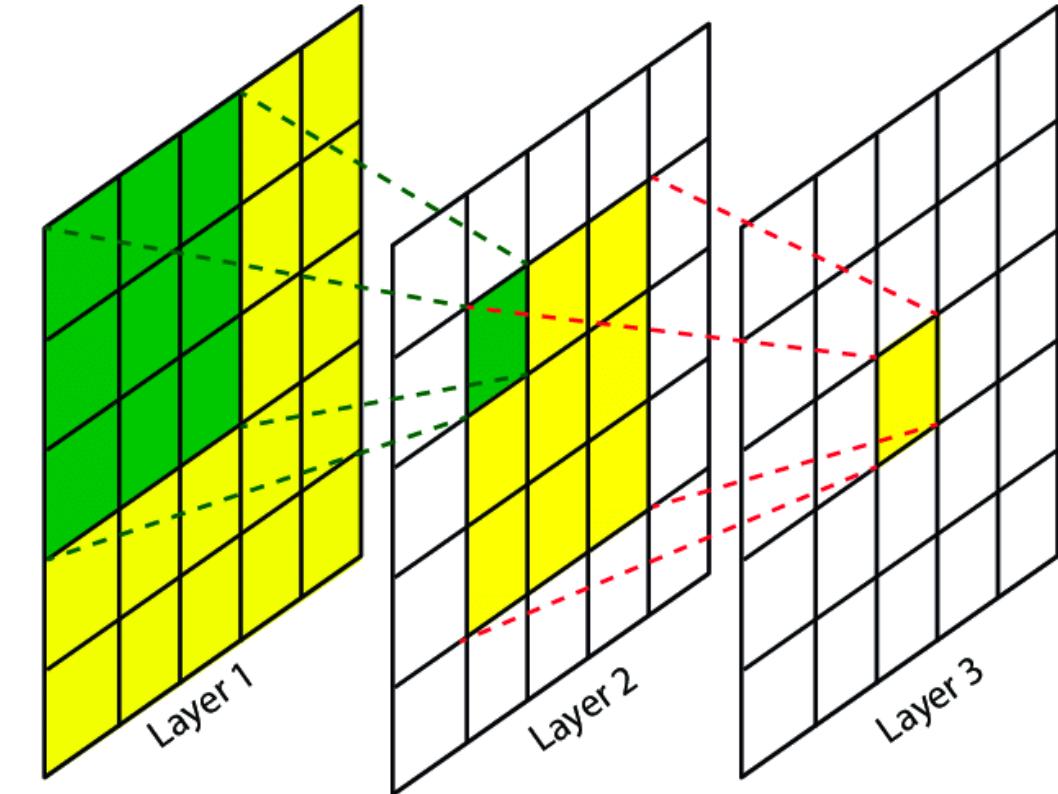


Stride = 2

# Сверточный слой: receptive field

- **Рецептивное поле (receptive field)**

нейрона = размер области на исходном изображении, пиксели которой дают вклад в активацию данного нейрона



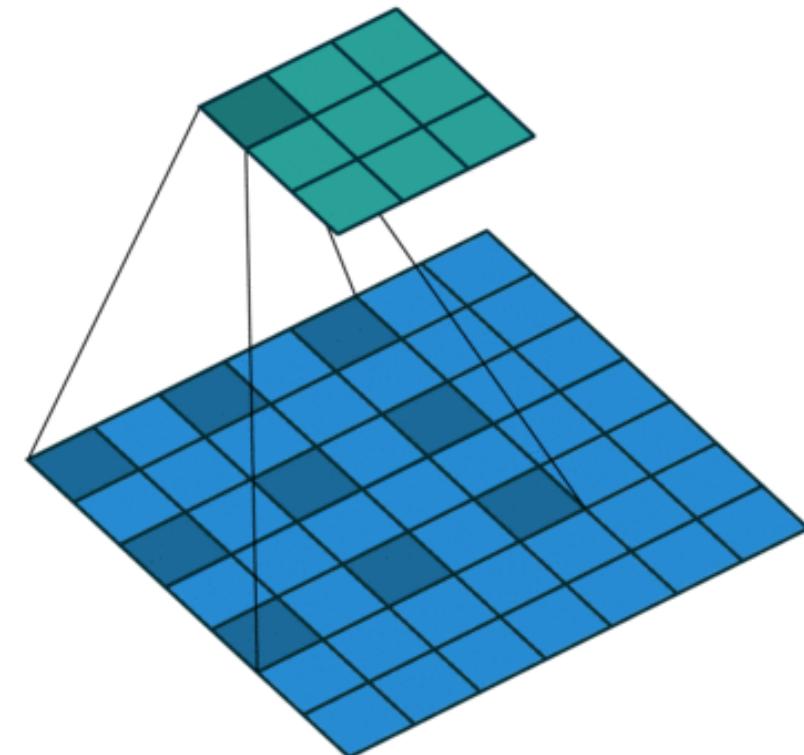
Рецептивное поле нейрона на Layer 2: 3x3

Рецептивное поле нейрона на Layer 3: 5x5

# Сверточный слой: dilation

---

- Быстро увеличить рецептивное поле нейронов можно с помощью dilated-сверток

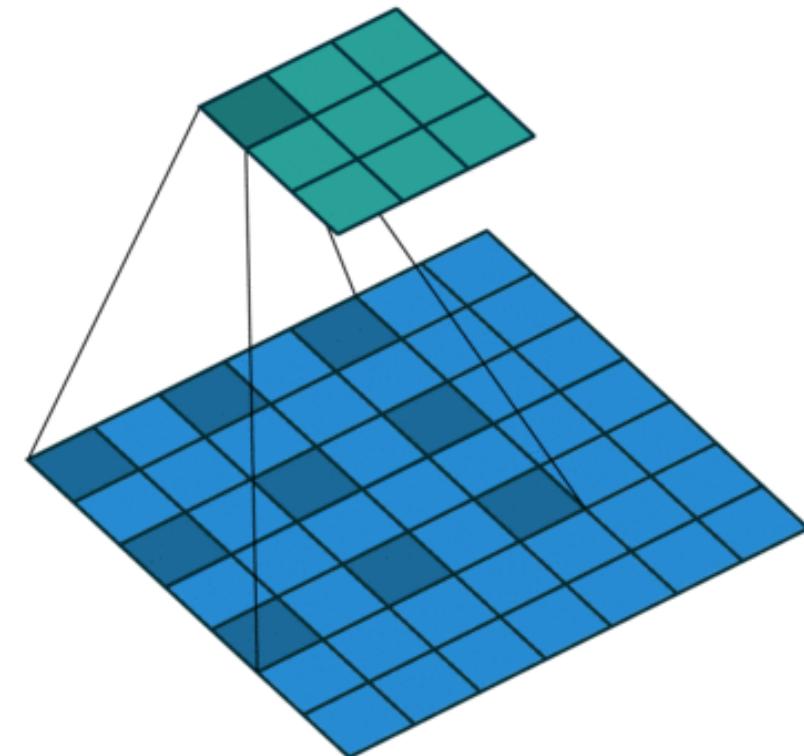


[https://github.com/vdumoulin/conv\\_arithmetic](https://github.com/vdumoulin/conv_arithmetic)

# Сверточный слой: dilation

---

- Быстро увеличить рецептивное поле нейронов можно с помощью dilated-сверток



[https://github.com/vdumoulin/conv\\_arithmetic](https://github.com/vdumoulin/conv_arithmetic)

# Сверточный слой: размер выхода

---

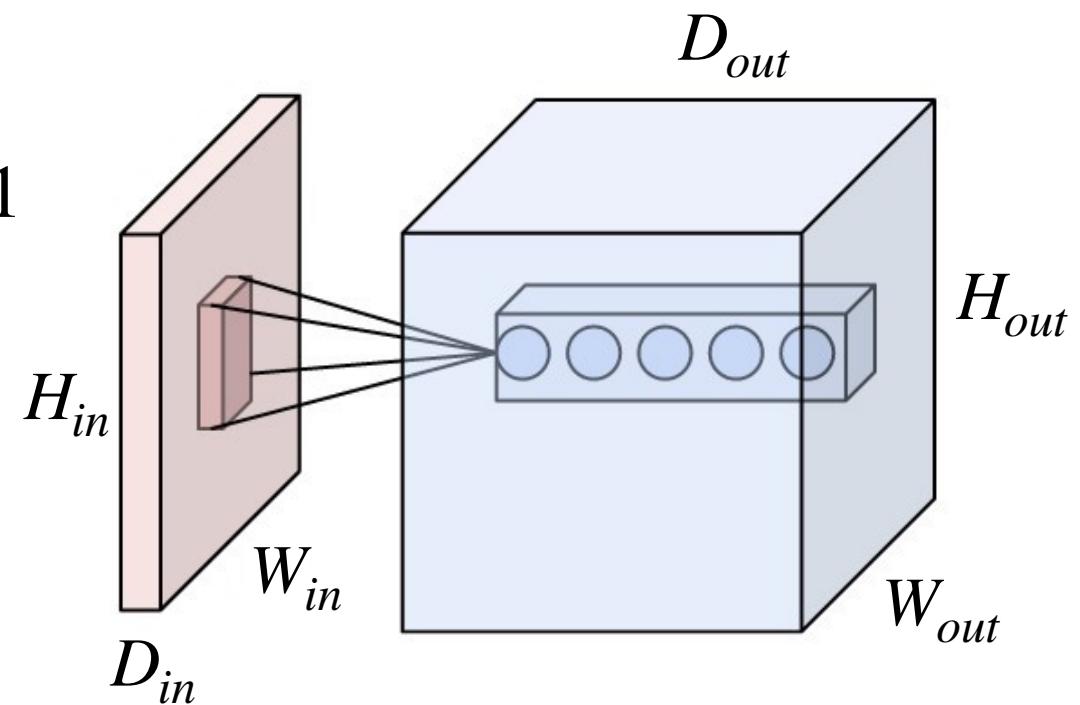
- Размер входного тензора:  $D_{in} \times H_{in} \times W_{in}$
- Параметры слоя:
  - $D_{in}$  - число каналов на входе (глубина входного тензора)
  - $D_{out}$  - число каналов на выходе (= число различных сверток)
  - Параметры сверток:
    - Размер ядра (*size*)
    - Размер отступа (*padding*)
    - Шаг (*stride*)

# Сверточный слой: размер выхода

- Размер выходного тензора:  $D_{out} \times H_{out} \times W_{out}$ :

- $$H_{out} = \frac{(H_{in} - size + 2 \times padding)}{stride} + 1$$

- $$W_{out} = \frac{(W_{in} - size + 2 \times padding)}{stride} + 1$$

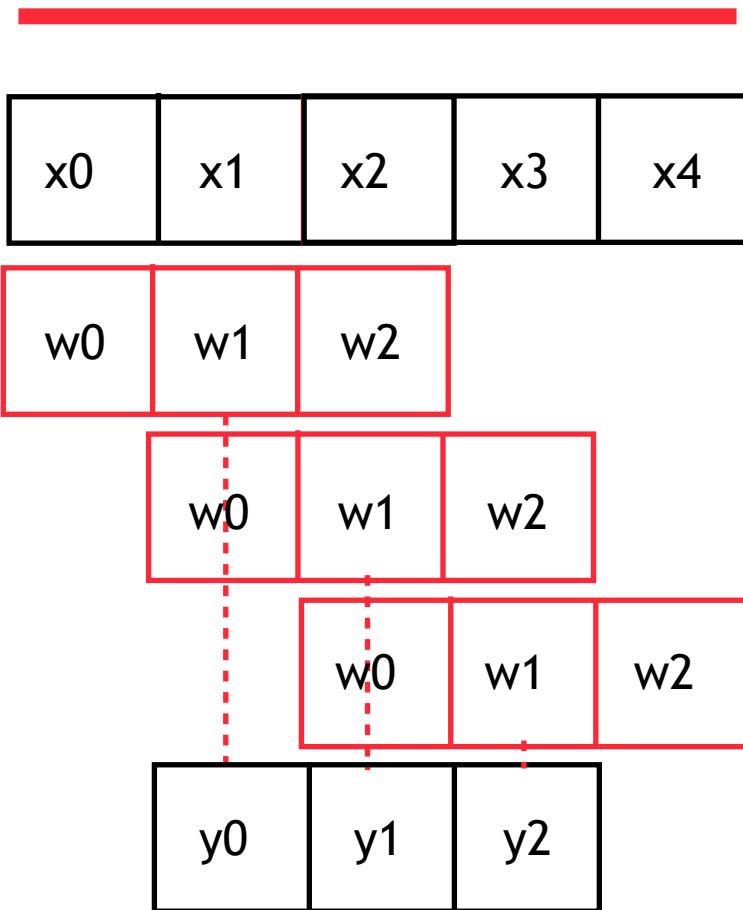


# Сверточный слой

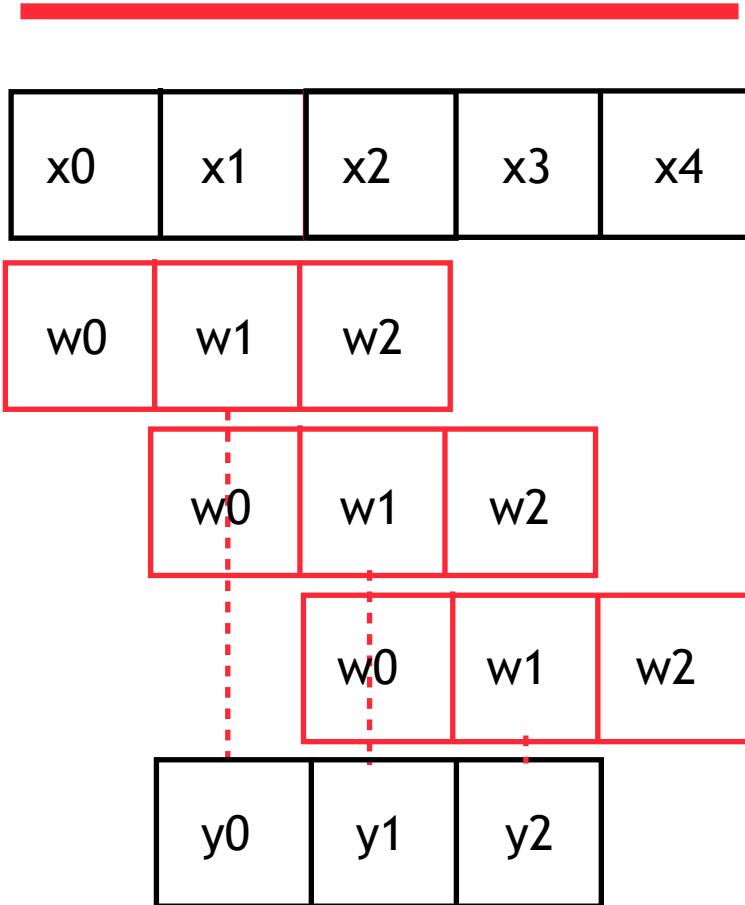
```
import torch  
  
conv_layer = torch.nn.Conv2d()
```

Init signature:  
torch.nn.Conv2d(  
 in\_channels,  
 out\_channels,  
 kernel\_size,  
 stride=1,  
 padding=0,  
 dilation=1,  
 groups=1,  
 bias=True,  
 padding\_mode='zeros',  
)

# Сверточный слой: backprop (1d)

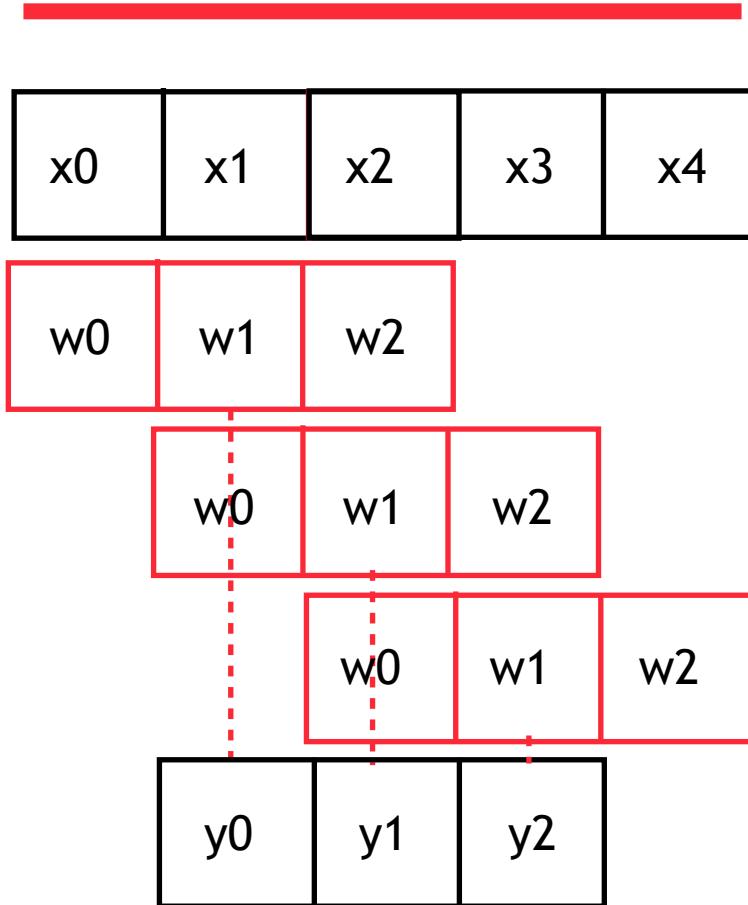


# Сверточный слой: backprop (1d)



$$\frac{\partial L}{\partial w_j} = \sum_i \frac{\partial L}{\partial y_i} \times \boxed{\frac{\partial y_i}{\partial w_j}}$$

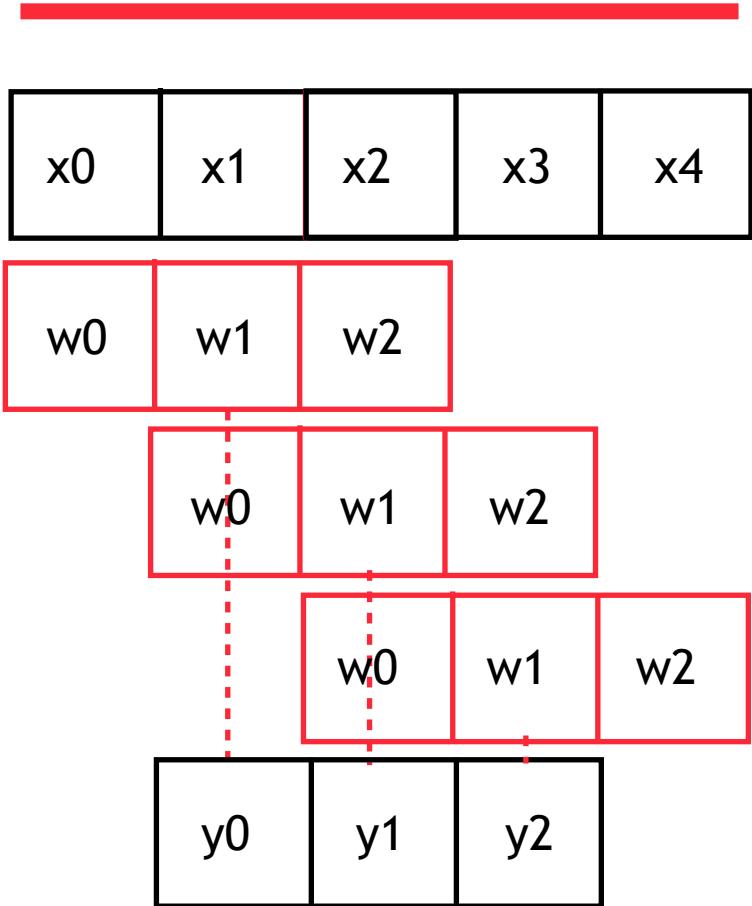
# Сверточный слой: backprop (1d)



$$\frac{\partial L}{\partial w_j} = \sum_i \frac{\partial L}{\partial y_i} \times \frac{\partial y_i}{\partial w_j}$$

$$y_i = (x * w)_i = \sum_{k=0}^{K-1} x_{i+k} w_k$$

# Сверточный слой: backprop (1d)



$$\frac{\partial L}{\partial w_j} = \sum_i \frac{\partial L}{\partial y_i} \times \frac{\partial y_i}{\partial w_j}$$

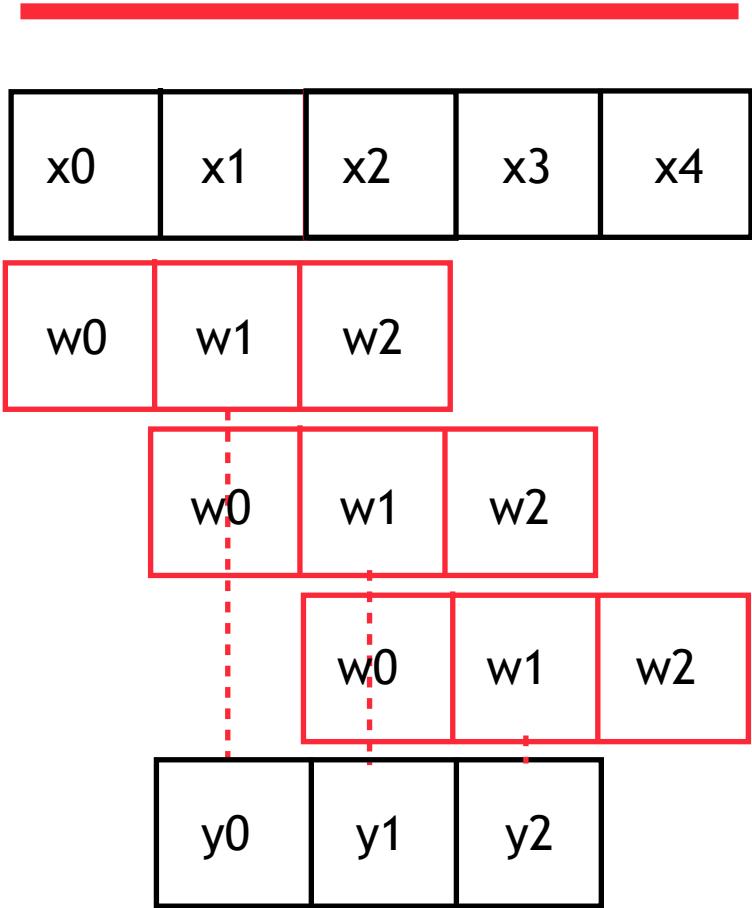
$$y_i = (x * w)_i = \sum_{k=0}^{K-1} x_{i+k} w_k$$

$$y_0 = x_0 \times w_0 + x_1 \times w_1 + x_2 \times w_2$$

$$y_1 = x_1 \times w_0 + x_2 \times w_1 + x_3 \times w_2$$

$$y_2 = x_2 \times w_0 + x_3 \times w_1 + x_4 \times w_2$$

# Сверточный слой: backprop (1d)



$$\frac{\partial L}{\partial w_j} = \sum_i \frac{\partial L}{\partial y_i} \times \frac{\partial y_i}{\partial w_j}$$

$$y_i = (x * w)_i = \sum_{k=0}^{K-1} x_{i+k} w_k$$

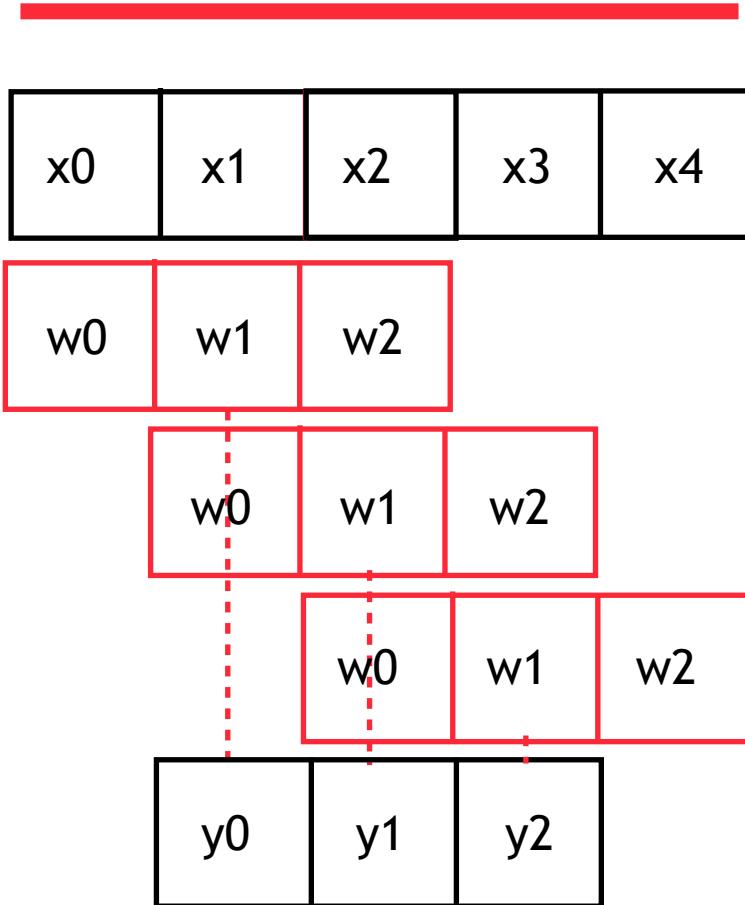
$$y_0 = x_0 \times w_0 + x_1 \times w_1 + x_2 \times w_2$$

$$y_1 = x_1 \times w_0 + x_2 \times w_1 + x_3 \times w_2$$

$$y_2 = x_2 \times w_0 + x_3 \times w_1 + x_4 \times w_2$$

$$\frac{\partial y_i}{\partial w_j} = x_{i+j}$$

# Сверточный слой: backprop (1d)

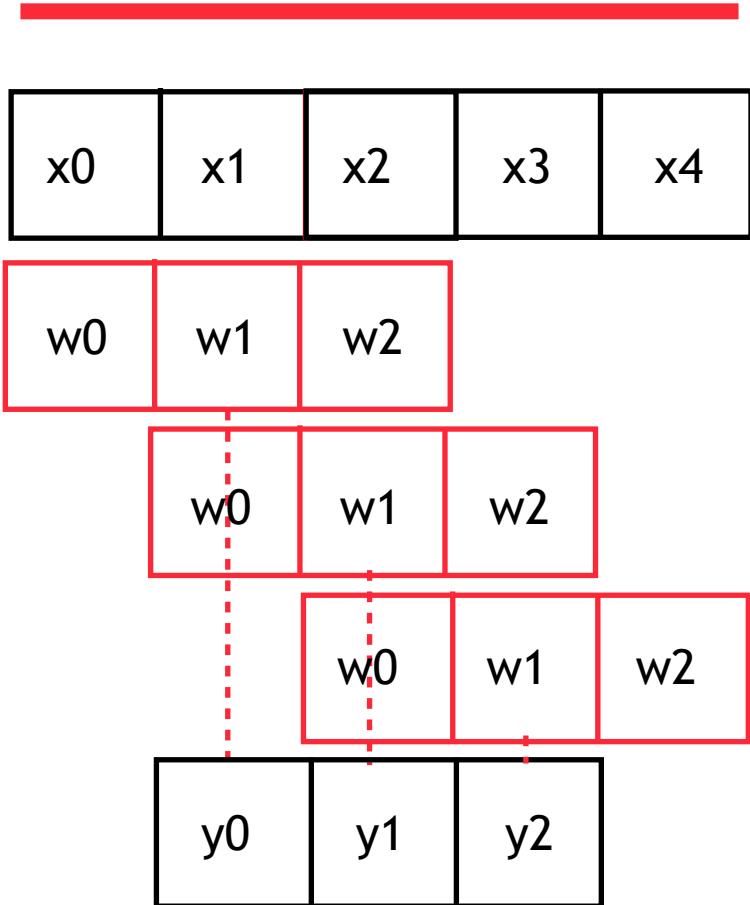


$$\frac{\partial L}{\partial w_j} = \sum_i \frac{\partial L}{\partial y_i} \times \frac{\partial y_i}{\partial w_j}$$

$$y_i = (x * w)_i = \sum_{k=0}^{K-1} x_{i+k} w_k$$

$$\frac{\partial y_i}{\partial w_j} = x_{i+j}$$

# Сверточный слой: backprop (1d)



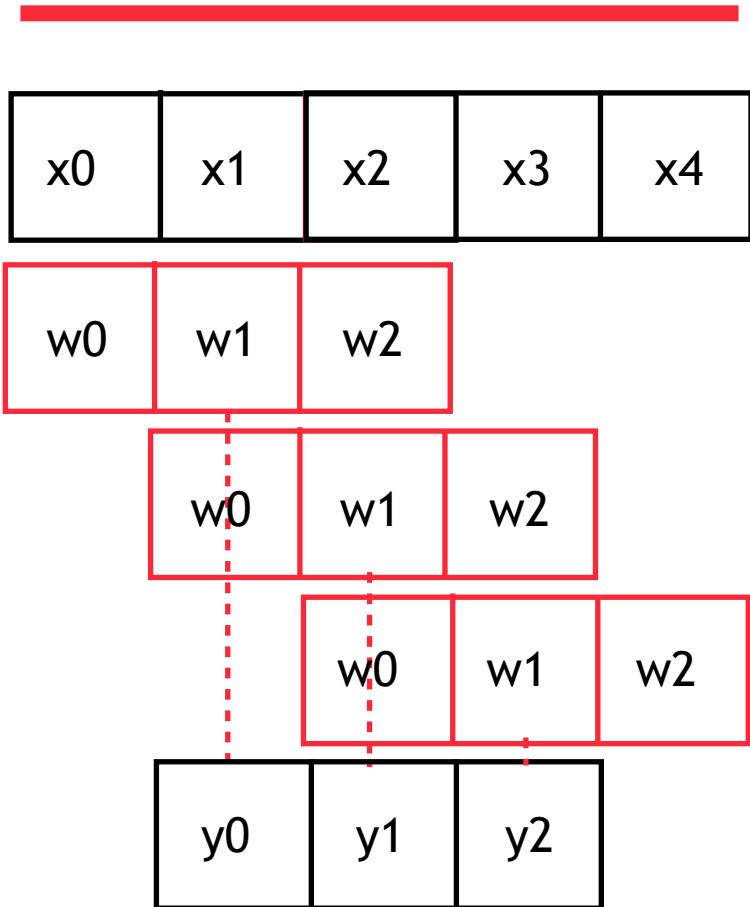
$$\frac{\partial L}{\partial w_j} = \sum_i \frac{\partial L}{\partial y_i} \times \frac{\partial y_i}{\partial w_j}$$

$$y_i = (x * w)_i = \sum_{k=0}^{K-1} x_{i+k} w_k$$

$$\frac{\partial y_i}{\partial w_j} = x_{i+j}$$

$$\frac{\partial L}{\partial w_j} = \sum_i \frac{\partial L}{\partial y_i} \times x_{i+j}$$

# Сверточный слой: backprop (1d)



$$y_i = (x * w)_i = \sum_{k=0}^{K-1} x_{i+k} w_k$$

$$\frac{\partial L}{\partial w_j} = \sum_i \frac{\partial L}{\partial y_i} \times x_{i+j}$$

# Сверточный слой: backprop (1d)

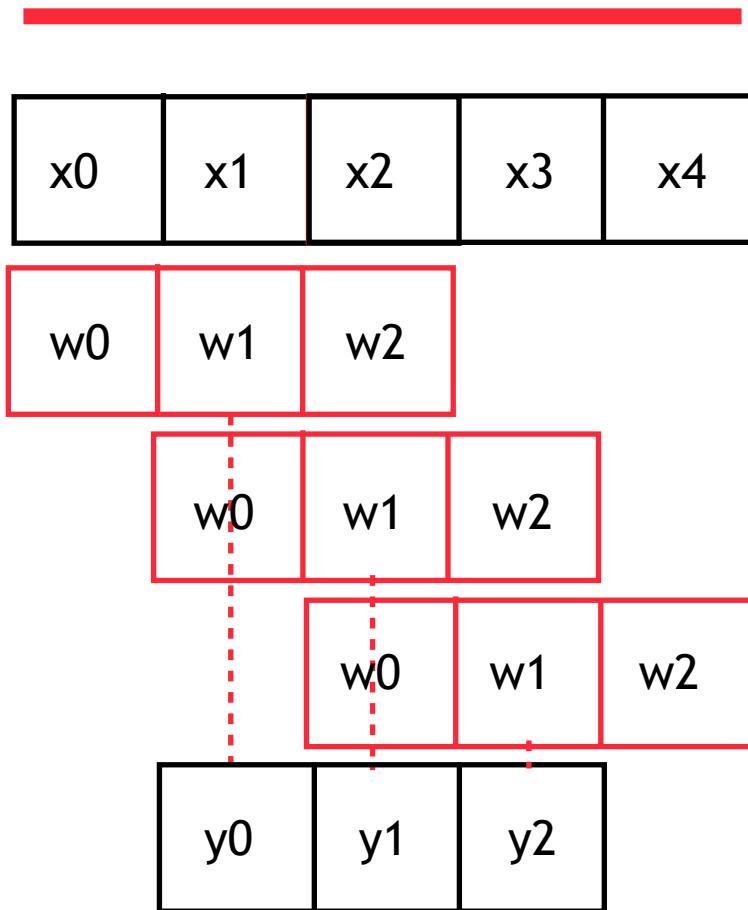
---



$$y_i = (x * w)_i = \sum_{k=0}^{K-1} x_{i+k} w_k$$

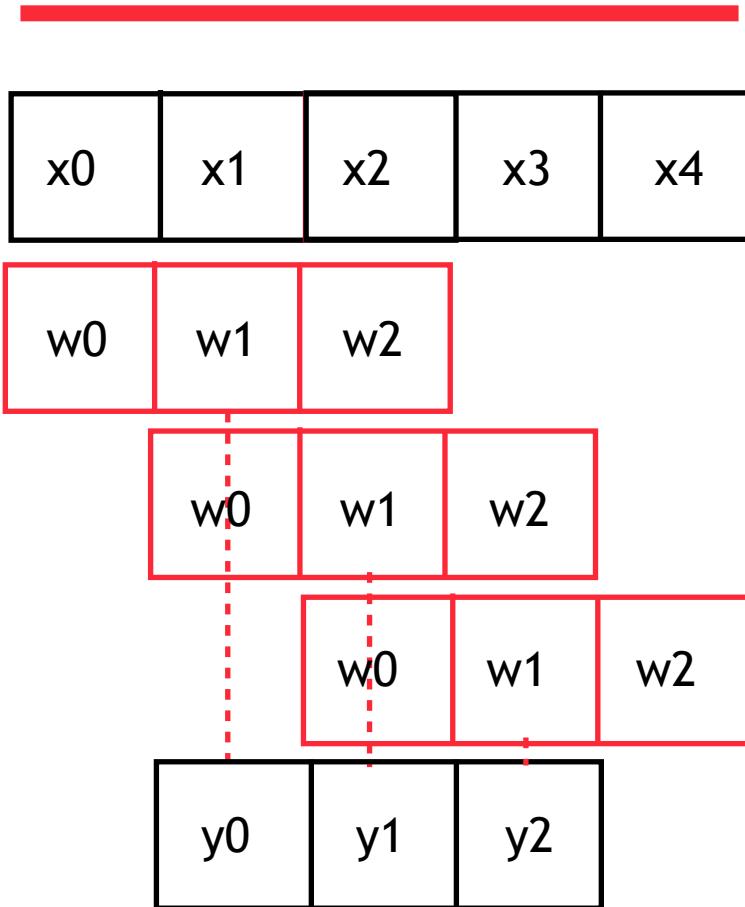
$$\frac{\partial L}{\partial w_j} = \sum_i \frac{\partial L}{\partial y_i} \times x_{i+j}$$

# Сверточный слой: backprop (1d)



$$\frac{\partial L}{\partial w_j} = (x * \frac{\partial L}{\partial y})_j$$

# Сверточный слой: backprop (1d)



$$\frac{\partial L}{\partial w_j} = (x * \frac{\partial L}{\partial y})_j$$

- Градиент по весам ядра свертки = свертка входного вектора с градиентом по выходному вектору
- А в случае 2d?



# Сверточный слой

---

- Обычно в сетях больше одного сверточного слоя, в глубоких сетях - десятки и сотни
- Чем глубже расположен слой, тем более “абстрактные” признаки он извлекает
- Эффективно наращивать глубину сетей помогает слой Pooling

**Пулинг**

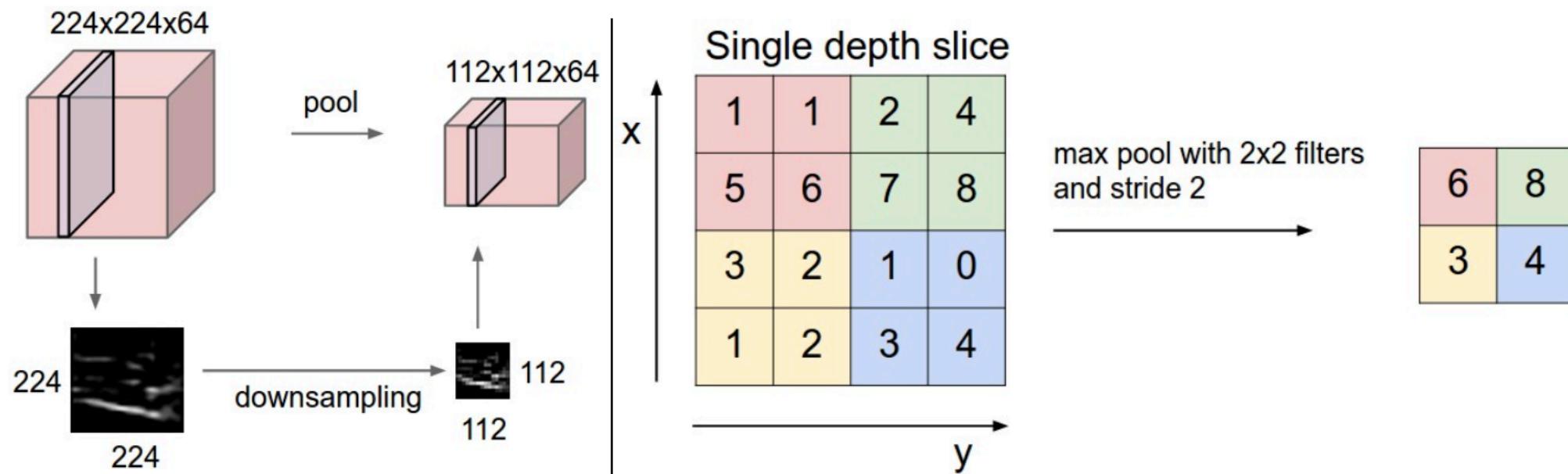


# Pooling

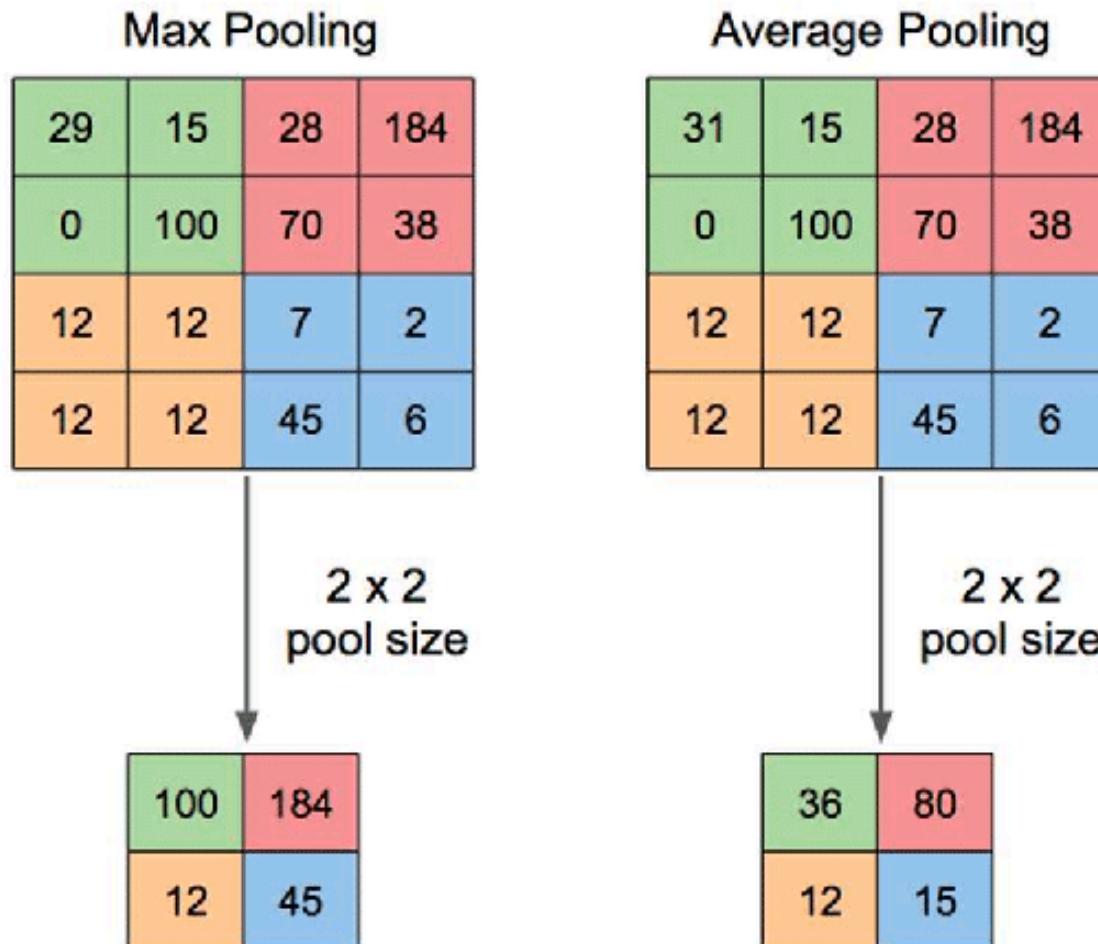
---

- Слой пулинга:
  - Разбивает тензор на части по ширине и высоте
  - В каждой из частей считает статистику (max - MaxPool, average - AveragePool, ...)
  - Полученные статистики склеивает обратно в тензор
  - На выходе получается тензор меньшего размера (но той же глубины)

# Pooling (2x2)



# Pooling: Max & Average



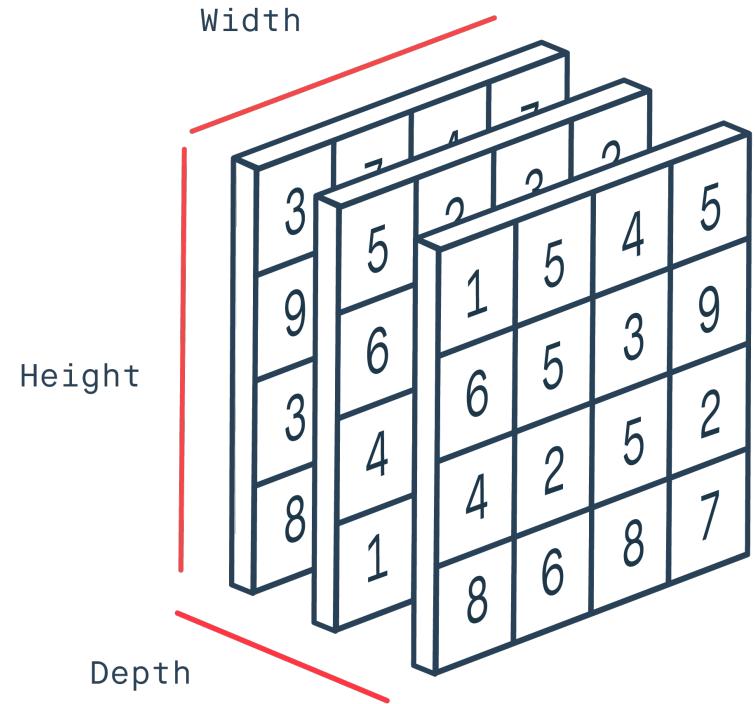


# Pooling

---

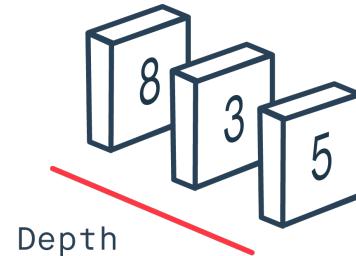
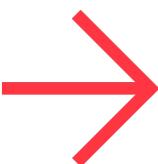
- При пулинге увеличивается рецептивное поле нейронов
- Пулинг позволяет делать сверточные более глубокими ( $\Rightarrow$  извлекать больше признаков)
- Иногда вместо пулинга используют сверточный слой с увеличенным шагом (stride)

# Global Pooling



Height x Width x Depth

- Если размер ядра пулинга = размеру тензора, то это **Global Pooling**
- После GP получается вектор длины = глубине тензора



1 x 1 x Depth

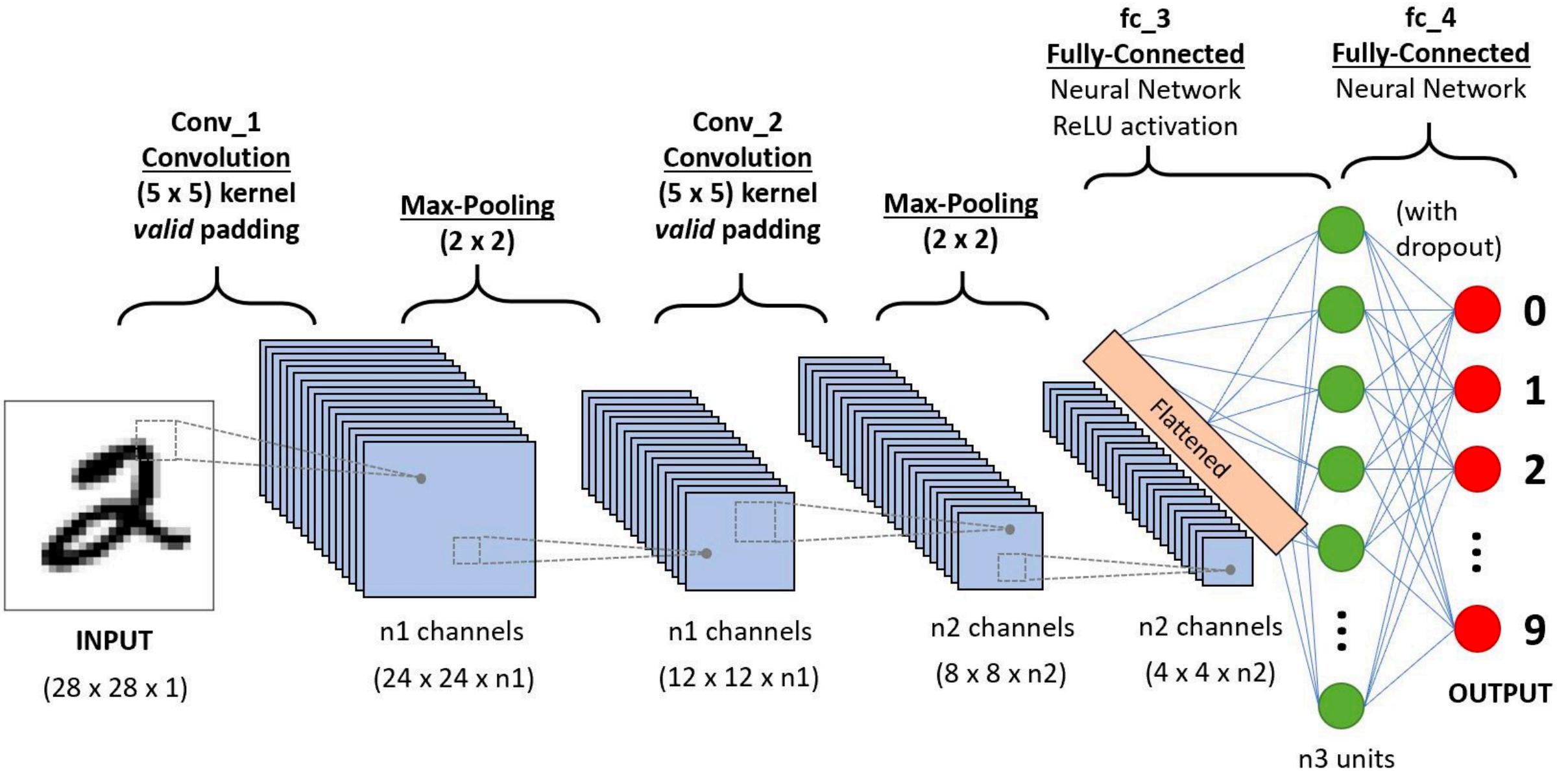


# Pooling: backprop

---

- MaxPool:
  - При прямом проходе запоминаются позиции, в которых стояли максимальные элементы
  - При обратном проходе градиент протекает только через них
- AveragePool:
  - Градиент для оператора усреднения?

**CNN**



# CNN: фильтры первого слоя AlexNet



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

# CNN

---

- Простейшая архитектура на свертках = чередование сверточных слоев и пулинга + полно связанный слой в конце

```
my_amazing_cnn = nn.Sequential(nn.Conv2d(in_channels=3, out_channels=32, kernel_size=(3, 3), stride=1, padding=1),
                                nn.ReLU(),
                                nn.MaxPool2d(kernel_size=(2, 2)),

                                nn.Conv2d(in_channels=32, out_channels=64, kernel_size=(3, 3), stride=1, padding=1),
                                nn.ReLU(),
                                nn.MaxPool2d(kernel_size=(2, 2)),

                                nn.Conv2d(in_channels=64, out_channels=128, kernel_size=(3, 3), stride=1, padding=1),
                                nn.ReLU(),
                                nn.MaxPool2d(kernel_size=(2, 2)),

                                nn.Flatten(),
                                nn.Linear(2048, 10))
```

# Резюме



# Сверточный слой

---

- Сверточный слой - эффективная замена полносвязному при работе с изображениями за счет:
  - Учета локальной связности пикселей
  - Инвариантности к переносу
  - Экономии весов



# Сверточная сеть

---

- Простейшая сверточная сеть:
  - (Сверточный слой -> Активация -> Пулинг)  $\times N \rightarrow FC$
- Сверточные слои извлекают признаки
- Пулинг помогает балансировать пространственный размер и глубину карт активаций
- Полносвязный слой делает классификацию на извлеченных признаках



# В следующей лекции

---

- Архитектуры современных CNN