

курс «Глубокое обучение»

Свёрточные нейронные сети

Александр Дьяконов
(ВМК МГУ имени М.В. Ломоносова)

2 марта 2020 года



План

Что такое изображение

Как классифицировать

Свёртка (Convolution)

Отступ (Padding) Шаг (stride)

Свёрточные нейронные сети (ConvNet, CNN)

Pooling (агрегация, субдискретизация / subsampling)

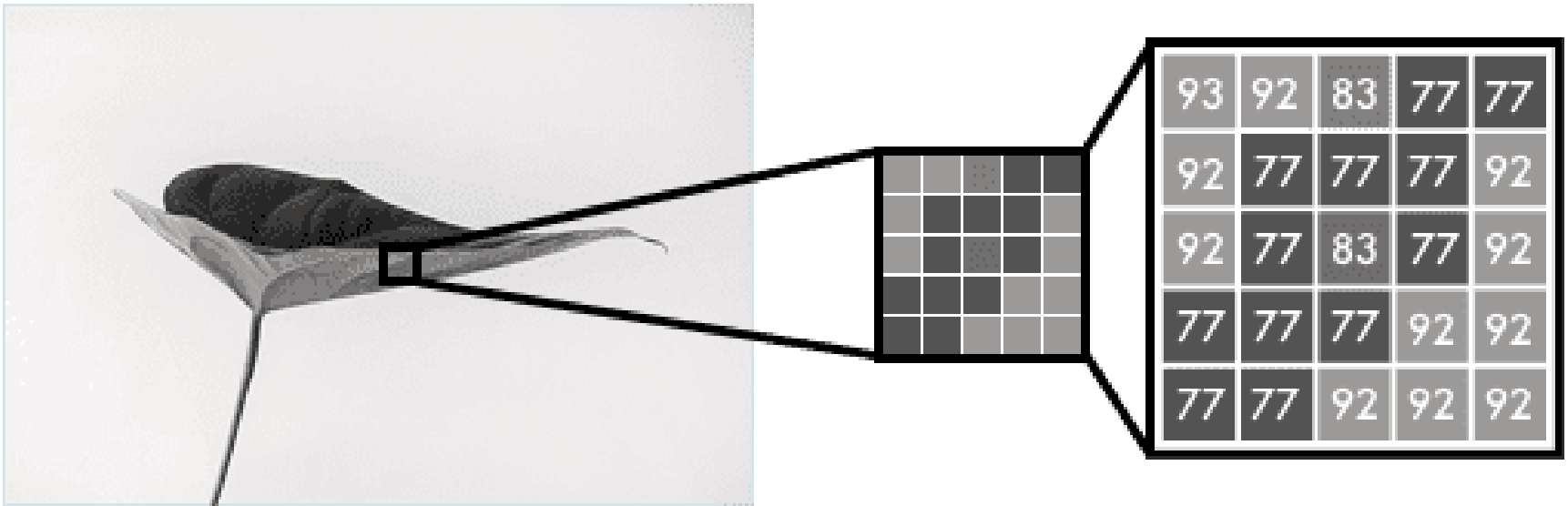
Какие бывают свёртки

Что такое изображение

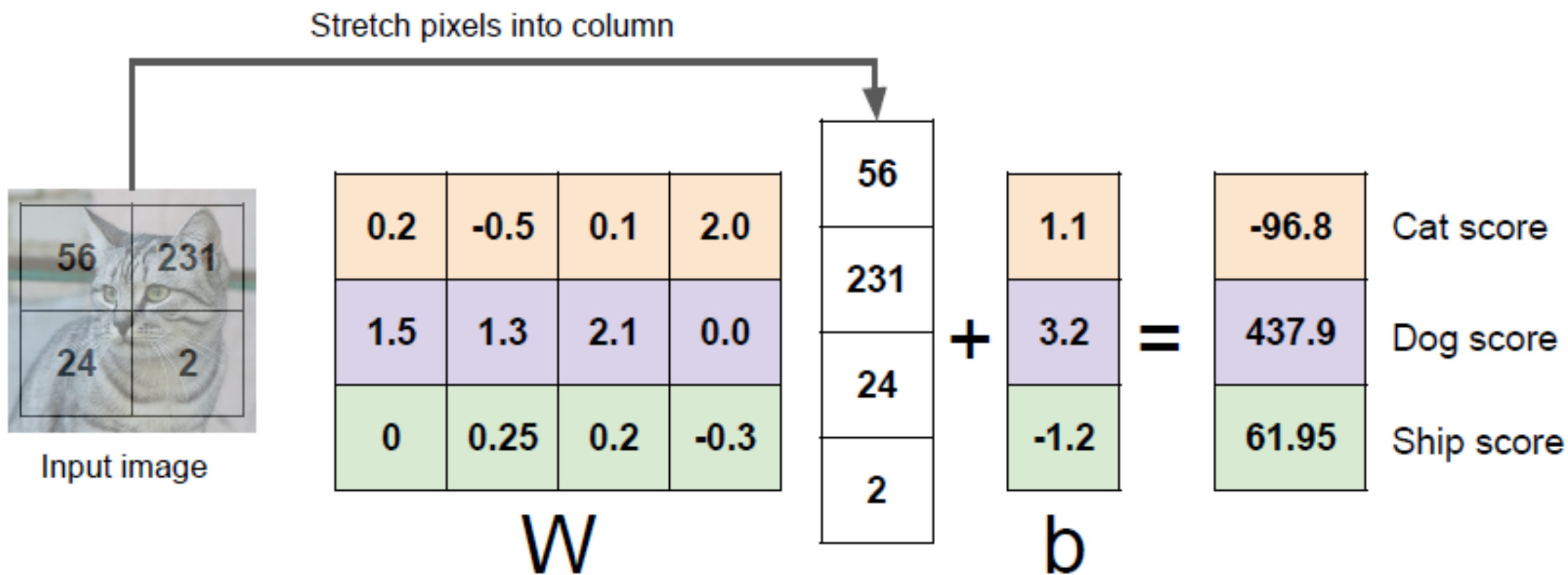


62	62	63	64	65	66	67	67	69	70	71	72	72	73	73	73	73	72	72	71	70	69	67	66	66	66	65	63	62	61	60	60	
61	62	63	64	66	66	67	68	68	69	70	71	71	72	72	73	72	72	71	71	70	69	68	66	66	66	65	65	63	62	61	60	60
61	62	63	64	66	66	68	68	69	70	70	71	72	73	73	73	72	72	71	71	69	68	67	66	66	65	65	64	63	62	61	61	
61	63	64	64	66	67	68	68	68	69	70	71	71	73	73	74	73	73	73	71	70	69	68	66	66	65	64	63	62	61	61	60	
61	63	64	65	67	68	69	69	70	70	71	71	72	55	53	69	72	72	71	71	70	69	68	67	66	65	64	63	62	60	60	60	
63	64	65	66	67	68	69	69	70	70	71	72	42	4	5	11	48	72	71	71	69	69	68	67	66	65	64	62	62	60	59	59	
63	65	66	66	68	68	69	70	71	71	71	72	18	4	4	7	8	66	71	70	69	68	68	67	66	65	64	63	61	59	59	58	
63	65	67	67	68	69	69	70	71	71	72	64	4	27	24	54	33	29	52	64	68	68	67	66	65	64	63	62	61	59	58	58	
64	65	66	66	68	69	70	71	41	24	24	12	17	24	48	60	37	43	36	52	66	68	67	66	65	64	63	61	60	59	58	57	
65	66	67	67	68	69	71	49	6	6	6	5	34	36	12	47	34	17	29	54	43	63	67	66	65	64	63	62	60	59	58	57	
64	65	66	66	68	69	38	6	6	5	5	7	16	19	4	47	44	27	24	40	67	66	66	65	65	64	63	61	60	59	58	57	
63	64	65	65	67	30	6	6	5	5	5	6	8	9	20	27	51	78	41	44	66	65	65	65	65	64	63	62	60	59	58	57	
63	64	65	65	34	5	5	5	5	5	5	5	4	19	6	7	54	64	20	59	65	65	64	64	64	63	62	61	60	59	57	56	
63	64	64	65	14	5	6	5	5	4	5	4	18	7	5	4	19	10	11	65	64	64	64	63	61	66	62	61	60	59	58	56	
63	64	64	65	53	7	4	5	6	6	7	10	6	5	5	4	21	24	18	64	64	64	63	62	64	65	62	62	60	59	58	57	
64	64	64	64	65	50	4	4	4	5	11	16	6	6	4	6	35	16	26	66	64	64	63	61	72	67	63	62	61	59	58	57	
64	64	64	64	65	46	4	4	4	5	6	9	8	5	29	10	43	56	29	57	64	64	63	61	70	67	62	64	65	59	59	57	
64	64	64	65	66	27	5	4	4	5	6	6	6	18	66	20	57	60	46	36	75	70	62	61	70	67	62	61	60	59	58	58	
49	50	62	65	57	5	5	6	5	6	6	6	6	41	59	28	60	58	44	22	63	71	72	60	69	68	61	60	58	59	59	58	
42	52	57	52	26	5	5	5	5	5	5	5	5	70	50	43	61	62	64	39	42	64	60	62	56	63	65	65	67	61	53	53	
32	32	32	33	6	5	5	5	5	5	5	6	6	11	39	21	33	51	50	45	46	18	32	36	33	23	44	70	71	51	42	27	31
50	50	51	39	5	5	5	5	6	5	6	6	42	69	28	34	42	39	43	37	26	29	40	26	29	26	35	42	35	33	18	19	
52	53	51	22	5	5	5	5	6	5	6	5	44	56	17	51	54	53	54	56	51	22	54	54	55	55	54	53	53	53	52	52	
54	54	53	8	5	5	5	5	6	5	6	13	52	42	21	51	54	51	49	49	50	22	41	45	42	42	41	40	41	44	43	42	
52	52	54	36	8	5	5	6	6	5	6	28	55	32	32	54	53	51	51	51	51	44	25	51	51	49	49	50	49	48	46	46	
54	54	52	53	30	7	5	6	6	5	6	40	54	29	52	51	53	56	55	52	52	51	38	52	52	50	49	46	46	45	46	47	
51	52	51	53	27	14	5	4	5	4	7	47	51	21	39	49	47	49	52	52	52	49	35	31	48	46	47	47	47	46	46	43	
48	50	51	53	25	14	17	8	4	4	17	46	40	18	43	47	46	49	52	54	53	53	54	18	50	49	46	47	47	47	47	45	
49	49	49	49	22	12	20	24	6	14	35	51	39	48	48	50	51	51	49	51	51	52	50	41	58	48	47	47	47	45	45	46	
51	49	50	50	22	13	19	36	13	12	42	50	40	73	50	50	50	49	48	49	49	48	49	45	51	46	44	44	44	42	45	47	
47	49	49	47	20	16	26	39	21	15	36	48	42	61	47	48	51	47	50	51	51	51	49	47	47	52	47	47	44	43	45	46	
48	50	48	52	19	13	33	36	18	18	36	49	51	54	47	47	49	46	46	49	49	49	49	47	44	53	44	48	44	46	46	45	

Что такое изображение



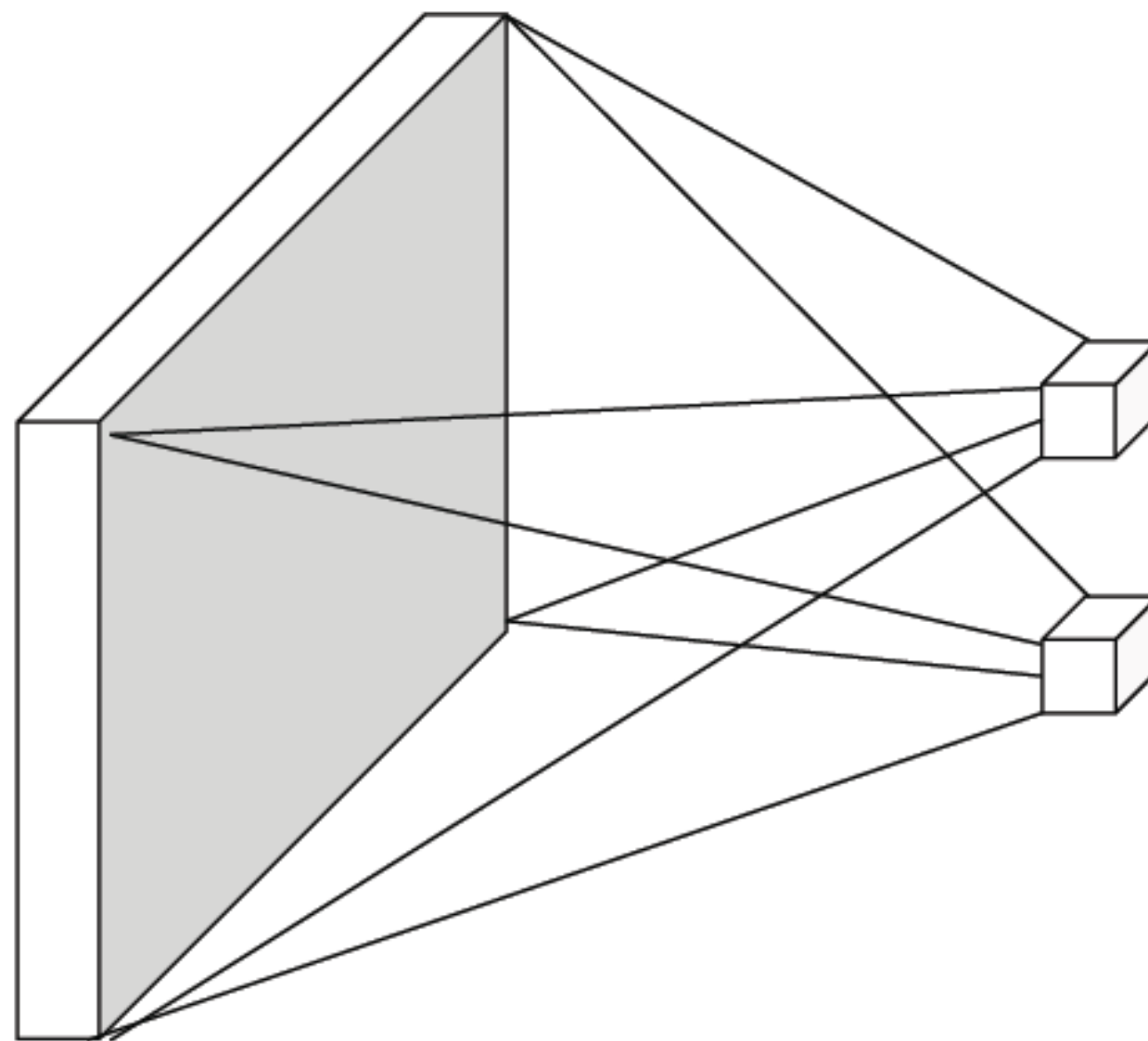
Линейный подход к классификации на несколько классов



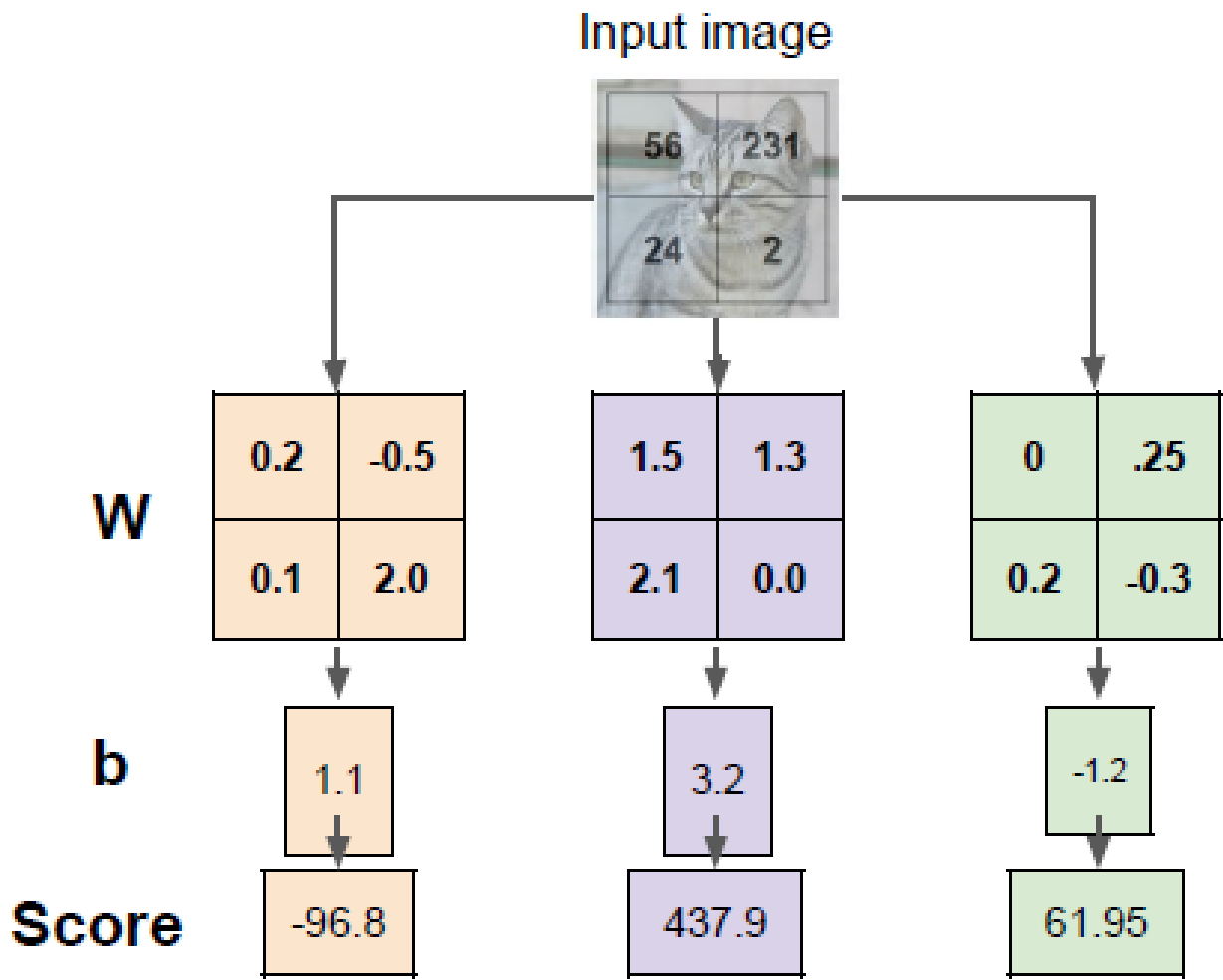
3 класса = 3 вектора признаков

Изображение = вытянуть в вектор

Линейный подход к классификации на несколько классов



Линейный подход к классификации на несколько классов



Вектора весов можно потом опять «прорешить» в изображения:



Проблема

Детектирование объекта в одном месте изображения
Примитивность модели

В обычных сетях слишком много параметров!

если изображение $256 \times 256 \times 3 \sim 200k$,
то чтобы изображение \rightarrow изображение
надо $3.9 \cdot 10^9$ параметров!

Как сравнивать изображения

-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1

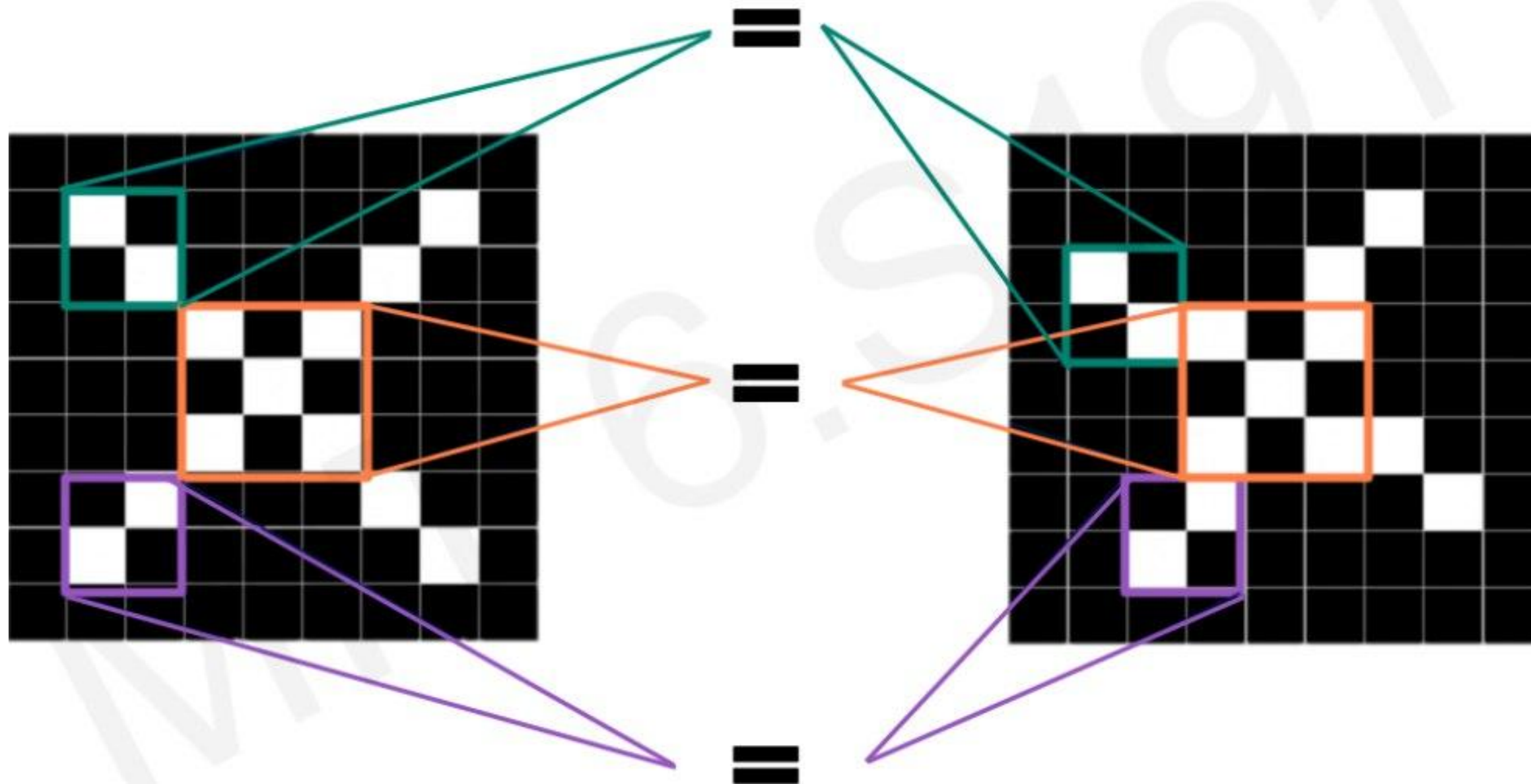


-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	1	-1	-1
-1	1	-1	-1	-1	1	-1	-1	-1
-1	-1	1	1	-1	1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1
-1	-1	-1	1	-1	1	1	-1	-1
-1	-1	-1	1	-1	-1	-1	1	-1
-1	-1	1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1

Image is represented as matrix of pixel values... and computers are literal!
We want to be able to classify an X as an X even if it's shifted, shrunk, rotated, deformed.

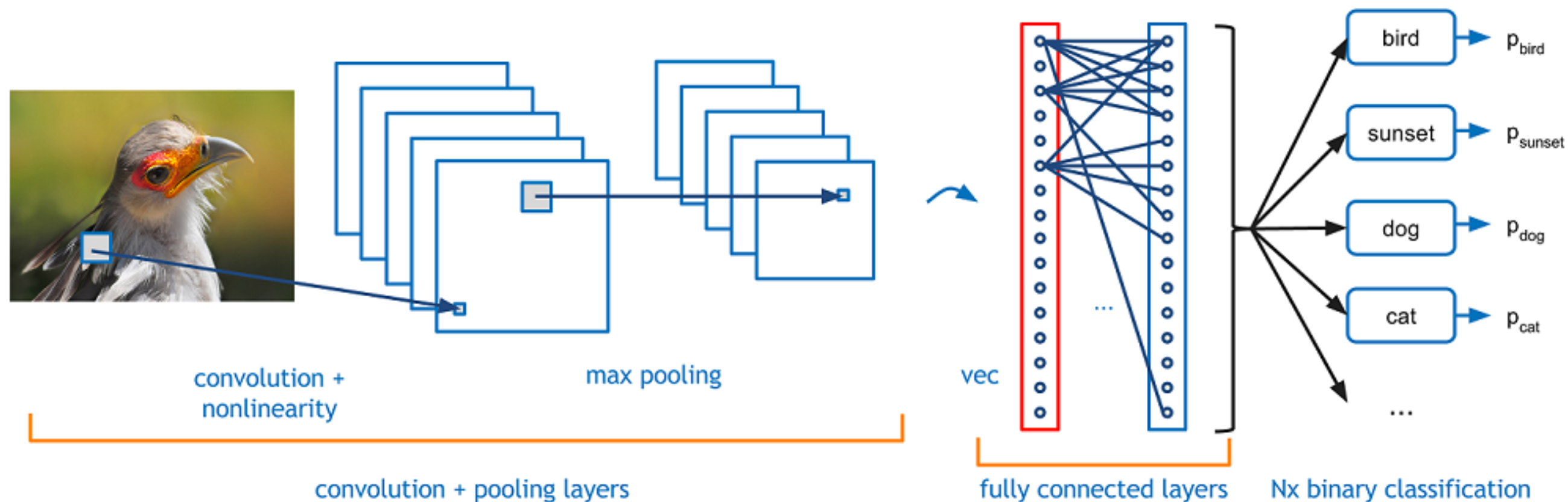
<http://introtodeeplearning.com/>

Как сравнивать изображения



<http://introtodeeplearning.com/>

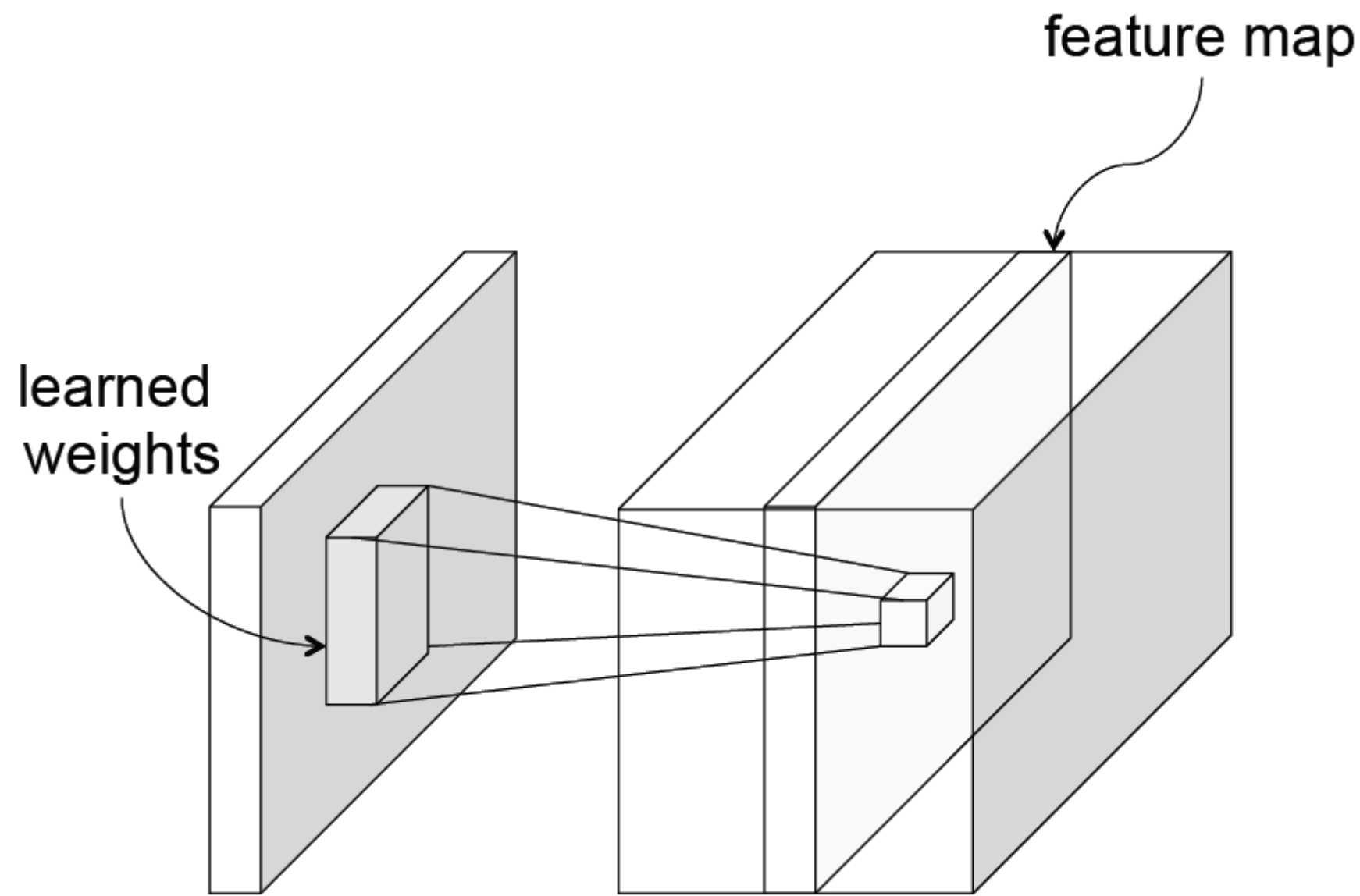
Свёрточные нейронные сети (ConvNet, CNN)



**– специальный вид нейронных сетей,
для обработки равномерных сигналов**

<https://adeshpande3.github.io/adeshpande3.github.io/A-Beginner's-Guide-To-Understanding-Convolutional-Neural-Networks/>

Свёрточные нейронные сети (ConvNet, CNN)



<http://slazebni.cs.illinois.edu/spring17/>

Что такое 1-D свёртка (Convolution)

пусть $I = (i_1, \dots, i_n) \in \mathbb{R}^n$ – сигнал / массив, $K = (k_1, \dots, k_r) \in \mathbb{R}^r$ – ядро свёртки,
тогда свёртка:

$$I * K = (i_1k_1 + \dots + i_rk_r, i_2k_1 + \dots + i_{r+1}k_r, \dots, i_{n-r+1}k_1 + \dots + i_nk_r) \in \mathbb{R}^{n-r+1}$$

-1	0	1							
1	2	3	4	5	5	4	3	2	1
	3 – 1								

	-1	0	1						
1	2	3	4	5	5	4	3	2	1
	2	4 – 2							

							-1	0	1
1	2	3	4	5	5	4	3	2	1
	2	2	2	1	-1	-2	-2	1 – 3	

Отступ (Padding)

Нулевой

-1	0	2	0	-1				
0	0	1	2	3	4	5	0	0
		-1	0	0	6	7		

Константный

-1	0	2	0	-1				
1	1	1	2	3	4	5	5	5
		-2	-1	0	1	2		

Зеркальный

-1	0	2	0	-1				
2	1	1	2	3	4	5	5	4
		-3	-1	0	1	3		

Циклический

-1	0	2	0	-1				
4	5	1	2	3	4	5	1	2
		-5	-5	0	5	5		

2-D свёртка (Convolution)

$$(I * K)_{xy} = \sum_{i=1}^h \sum_{j=1}^r K_{ij} I_{x+i-1, y+j-1}$$

$$\begin{bmatrix} 1 & 2 & 3 \\ 3 & 4 & 5 \\ 1 & 0 & 2 \end{bmatrix} * \begin{bmatrix} 0 & 1 \\ 1 & -1 \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} * \begin{bmatrix} 0 & 1 \\ 1 & -1 \end{bmatrix} & \begin{bmatrix} 2 & 3 \\ 4 & 5 \end{bmatrix} * \begin{bmatrix} 0 & 1 \\ 1 & -1 \end{bmatrix} \\ \begin{bmatrix} 3 & 4 \\ 1 & 0 \end{bmatrix} * \begin{bmatrix} 0 & 1 \\ 1 & -1 \end{bmatrix} & \begin{bmatrix} 4 & 5 \\ 0 & 2 \end{bmatrix} * \begin{bmatrix} 0 & 1 \\ 1 & -1 \end{bmatrix} \end{bmatrix} =$$

$$= \begin{bmatrix} 1 & 2 \\ 5 & 3 \end{bmatrix}$$

может быть немного другая индексация

хорошее объяснение:

Vincent Dumoulin, Francesco Visin - A guide to convolution arithmetic for deep learning 2018

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

Свёртка (Convolution)

3 ₀	3 ₁	2 ₂	1	0
0 ₂	0 ₂	1 ₀	3	1
3 ₀	1 ₁	2 ₂	2	3
2	0	0	2	2
2	0	0	0	1

12.0	12.0	17.0
10.0	17.0	19.0
9.0	6.0	14.0

3	3 ₀	2 ₁	1 ₂	0
0	0 ₂	1 ₂	3 ₀	1
3	1 ₀	2 ₁	2 ₂	3
2	0	0	2	2
2	0	0	0	1

12.0	12.0	17.0
10.0	17.0	19.0
9.0	6.0	14.0

3	3	2 ₀	1 ₁	0 ₂
0	0	1 ₂	3 ₂	1 ₀
3	1	2 ₀	2 ₁	3 ₂
2	0	0	2	2
2	0	0	0	1

12.0	12.0	17.0
10.0	17.0	19.0
9.0	6.0	14.0

3	3	2	1	0
0 ₀	0 ₁	1 ₂	3	1
3 ₂	1 ₂	2 ₀	2	3
2 ₀	0 ₁	0 ₂	2	2
2	0	0	0	1

12.0	12.0	17.0
10.0	17.0	19.0
9.0	6.0	14.0

3	3	2	1	0
0	0 ₀	1 ₁	3 ₂	1
3	1 ₂	2 ₂	2 ₀	3
2	0 ₀	0 ₁	2 ₂	2
2	0	0	0	1

12.0	12.0	17.0
10.0	17.0	19.0
9.0	6.0	14.0

3	3	2	1	0
0	0	1 ₀	3 ₁	1 ₂
3	1	2 ₂	2 ₂	3 ₀
2	0	0 ₀	2 ₁	2 ₂
2	0	0	0	1

12.0	12.0	17.0
10.0	17.0	19.0
9.0	6.0	14.0

3	3	2	1	0
0	0	1	3	1
3 ₀	1 ₁	2 ₂	2	3
2 ₂	0 ₂	0 ₀	2	2
2 ₀	0 ₁	0 ₂	0	1

12.0	12.0	17.0
10.0	17.0	19.0
9.0	6.0	14.0

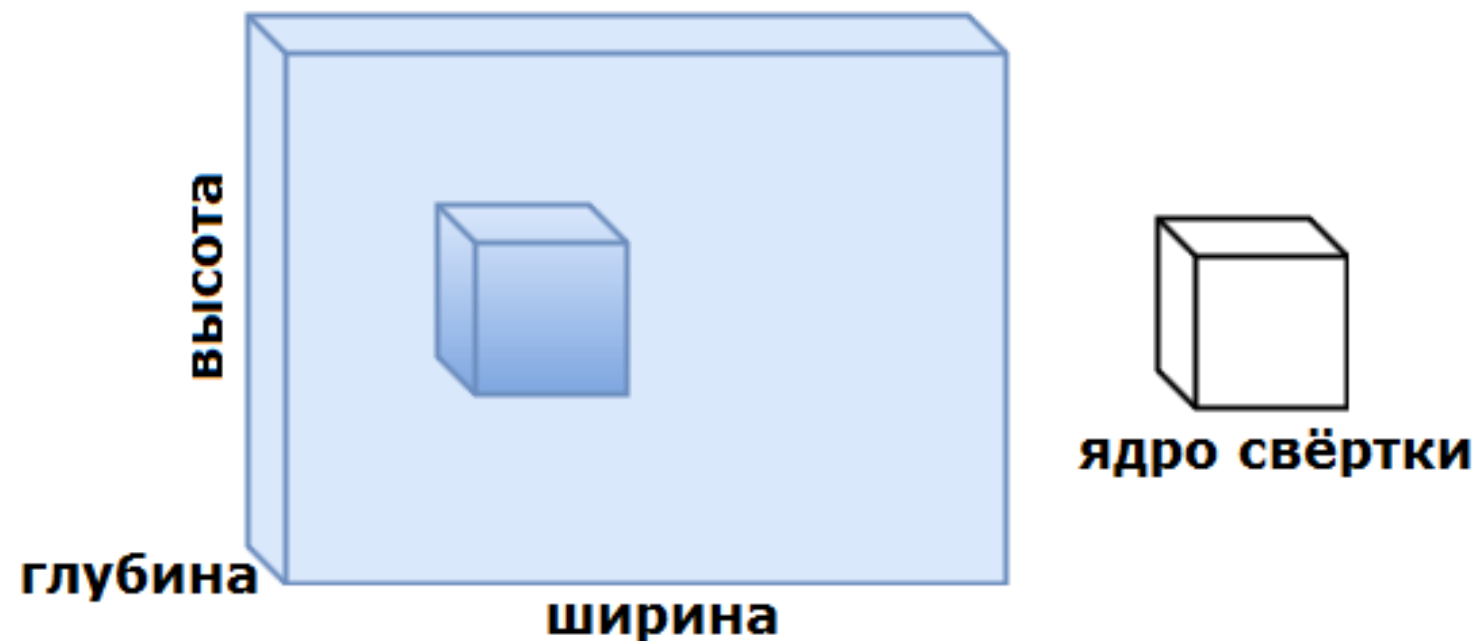
3	3	2	1	0
0	0	1	3	1
3	1 ₀	2 ₁	2 ₂	3
2	0 ₂	0 ₂	2 ₀	2
2	0 ₀	0 ₁	0 ₂	1

12.0	12.0	17.0
10.0	17.0	19.0
9.0	6.0	14.0

3	3	2	1	0
0	0	1	3	1
3	1	2 ₀	2 ₁	3 ₂
2	0	0 ₂	2 ₂	2 ₀
2	0	0 ₀	0 ₁	1 ₂

12.0	12.0	17.0
10.0	17.0	19.0
9.0	6.0	14.0

Свёртка (Convolution)



Глубина (depth) / число каналов

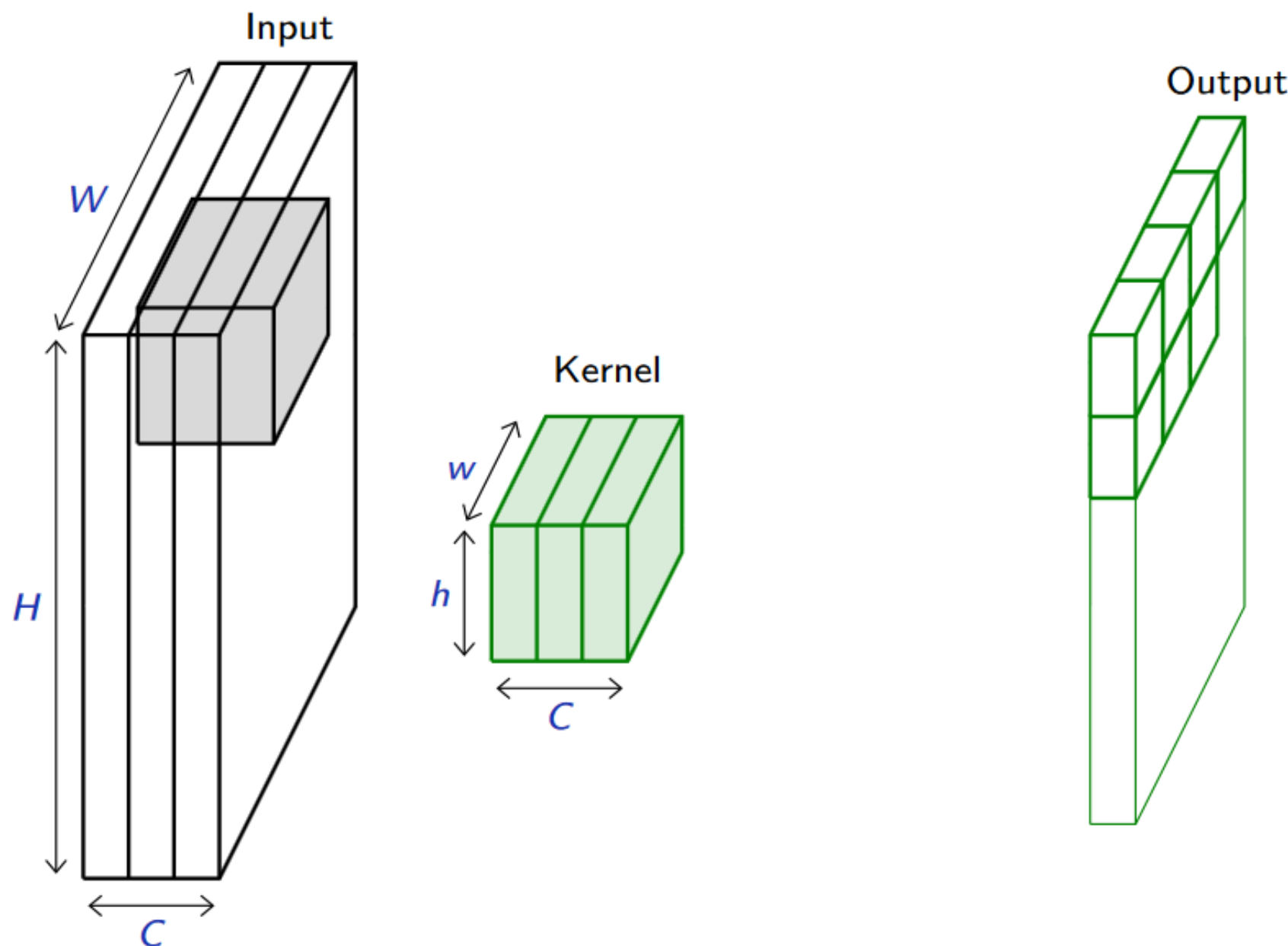
Высота (height) и ширина (width) каждого ядра

Шаг (stride) — на сколько смещается ядро при вычислении свёрток (чем больше, тем меньше размер итогового изображения)

Отступ (padding) – для дополнения изображения нулями по краям

Ядро (kernel) или фильтр (filter) – размерность как у предыдущего тензора; в 3D длина и ширина меньше (глубина совпадает)

Свёртка (Convolution)



Что делает свёртка?



<https://algotravelling.com/ru/%d0%bc%d0%b0%d1%88%d0%b8%d0%bd%d0%bd%d0%be%d0%b5-%d0%be%d0%b1%d1%83%d1%87%d0%b5%d0%bd%d0%b8%d0%b5/>

Что делает свёртка?



Что делает свёртка?



```
import matplotlib.pyplot as plt
import numpy as np
from skimage import data, color

image = color.rgb2gray(data.chelsea())
```

```
gx = np.empty(image.shape, dtype=np.double)
gx[:, 0] = 0
gx[:, -1] = 0
gx[:, 1:-1] = image[:, :-2] - image[:, 2:]
```

```
gy = np.empty(image.shape, dtype=np.double)
gy[0, :] = 0
gy[-1, :] = 0
gy[1:-1, :] = image[:-2, :] - image[2:, :]
```

```
fig, (ax1, ax2, ax3) = plt.subplots(1, 3, figsize=(7, 4),
sharex=True, sharey=True)
ax1.axis('off')
ax1.imshow(image, cmap=plt.cm.gray)
# ax1.set_title('Original image')
```

```
ax2.axis('off')
ax2.imshow(gx, cmap=plt.cm.gray)
# ax2.set_title('Horizontal gradients')
```

```
ax3.axis('off')
ax3.imshow(gy, cmap=plt.cm.gray)
plt.tight_layout(pad=0.0, h_pad=0, w_pad=0)
# ax3.set_title('Vertical gradients')
```

Свёртка (Convolution): мотивация

Раньше: обработка изображений – специально построенные свёртки
edges, corners, colors, shapes...



Сейчас: не будем специально строить свёртки – их параметры настроятся сами!

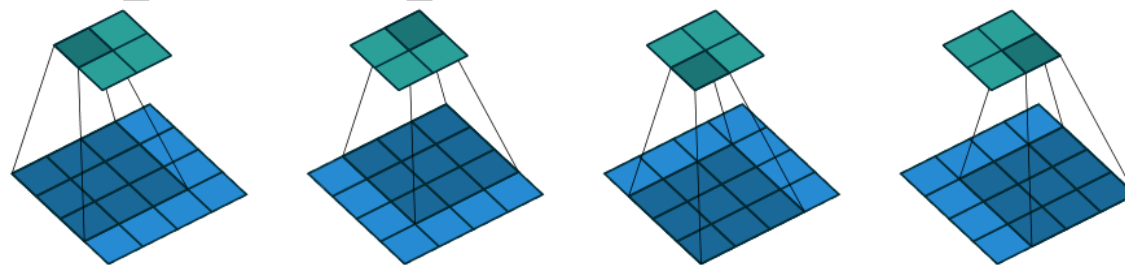
Важно: свёртку можно применять к изображениям любых размеров!

Нет ограничений на размеры входа...

Отступ (padding) – чтобы сохранялись размеры изображения

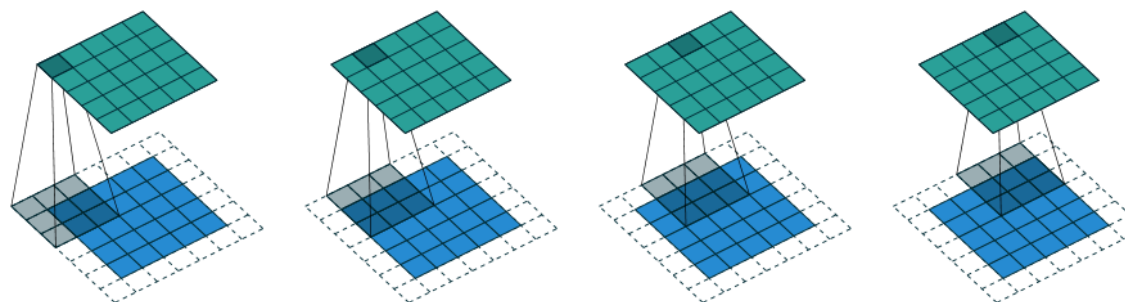
padding = VALID (в TF)

$$\begin{bmatrix} 1 & 2 & 3 \\ 3 & 4 & 5 \\ 1 & 0 & 2 \end{bmatrix} * \begin{bmatrix} 0 & 1 \\ 1 & -1 \end{bmatrix} = \begin{bmatrix} 1 & 2 \\ 5 & 3 \end{bmatrix}$$



padding = SAME (в TF)

$$\begin{bmatrix} 1 & 2 & 3 \\ 3 & 4 & 5 \\ 1 & 0 & 2 \end{bmatrix} * \begin{bmatrix} 0 & 1 \\ 1 & -1 \end{bmatrix} \equiv \begin{bmatrix} 1 & 2 & 3 & 0 \\ 3 & 4 & 5 & 0 \\ 1 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 0 & 1 \\ 1 & -1 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 5 \\ 5 & 3 & 4 \\ 0 & 2 & 0 \end{bmatrix}$$



Шаг (stride)

сдвигаемся при вычислении свёртки (можно в каждой её размерности)



с шагом 2

$$\begin{bmatrix} 1 & 2 & 3 & 1 & 0 \\ 3 & 4 & 5 & 2 & 1 \\ 1 & 0 & 2 & 3 & 4 \\ 0 & 1 & 3 & 1 & 2 \\ 1 & 2 & 4 & 2 & 3 \end{bmatrix} * \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & -1 \\ 0 & -1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 2 \\ -5 & 2 \end{bmatrix}$$

0 ₀	0 ₁	0 ₂	0	0	0	0
0 ₂	3 ₂	3 ₀	2	1	0	0
0 ₀	0 ₁	0 ₂	1	3	1	0
0	3	1	2	2	3	0
0	2	0	0	2	2	0
0	2	0	0	0	1	0
0	0	0	0	0	0	0

6.0	17.0	3.0
8.0	17.0	13.0
6.0	4.0	4.0

0	0	0 ₀	0 ₁	0 ₂	0	0
0	3	3 ₂	2 ₂	1 ₀	0	0
0	0	0 ₀	1 ₁	3 ₂	1	0
0	3	1	2	2	3	0
0	2	0	0	2	2	0
0	2	0	0	0	1	0
0	0	0	0	0	0	0

6.0	17.0	3.0
8.0	17.0	13.0
6.0	4.0	4.0

0	0	0	0	0 ₀	0 ₁	0 ₂
0	3	3	2	1 ₂	0 ₂	0 ₀
0	0	0	1	3 ₀	1 ₁	0 ₂
0	3	1	2	2	3	0
0	2	0	0	2	2	0
0	2	0	0	0	1	0
0	0	0	0	0	0	0

6.0	17.0	3.0
8.0	17.0	13.0
6.0	4.0	4.0

0	0	0	0	0	0	0
0	3	3	2	1	0	0
0 ₀	0 ₁	0 ₂	1	3	1	0
0 ₂	3 ₂	1 ₀	2	2	3	0
0 ₀	2 ₁	0 ₂	0	2	2	0
0	2	0	0	0	1	0
0	0	0	0	0	0	0

6.0	17.0	3.0
8.0	17.0	13.0
6.0	4.0	4.0

0	0	0	0	0	0	0
0	3	3	2	1	0	0
0	0	0 ₀	1 ₁	3 ₂	1	0
0	3	1 ₂	2 ₂	2 ₀	3	0
0	2	0 ₀	0 ₁	2 ₂	2	0
0	2	0	0	0	1	0
0	0	0	0	0	0	0

6.0	17.0	3.0
8.0	17.0	13.0
6.0	4.0	4.0

0	0	0	0	0	0	0
0	3	3	2	1	0	0
0	0	0	1	3 ₀	1 ₁	0 ₂
0	3	1	2	2 ₂	3 ₂	0 ₀
0	2	0	0	2 ₀	2 ₁	0 ₂
0	2	0	0	0	1	0
0	0	0	0	0	0	0

6.0	17.0	3.0
8.0	17.0	13.0
6.0	4.0	4.0

0	0	0	0	0	0	0
0	3	3	2	1	0	0
0	0	0	1	3	1	0
0	3	1	2	2	3	0
0 ₀	2 ₁	0 ₂	0	2	2	0
0 ₂	2 ₂	0 ₀	0	0	1	0
0 ₀	0 ₁	0 ₂	0	0	0	0

6.0	17.0	3.0
8.0	17.0	13.0
6.0	4.0	4.0

0	0	0	0	0	0	0
0	3	3	2	1	0	0
0	0	0	1	3	1	0
0	3	1	2	2	3	0
0	2	0 ₀	0 ₁	2 ₂	2	0
0	2	0 ₂	0 ₂	0 ₀	1	0
0	0	0 ₀	0 ₁	0 ₂	0	0

6.0	17.0	3.0
8.0	17.0	13.0
6.0	4.0	4.0

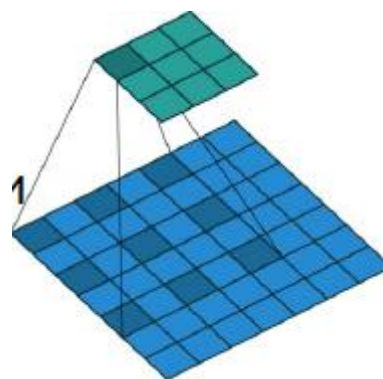
0	0	0	0	0	0	0
0	3	3	2	1	0	0
0	0	0	1	3	1	0
0	3	1	2	2	3	0
0	2	0	0	2 ₀	2 ₁	0 ₂
0	2	0	0	0 ₂	1 ₂	0 ₀
0	0	0	0	0 ₀	0 ₁	0 ₂

6.0	17.0	3.0
8.0	17.0	13.0
6.0	4.0	4.0

Свёртка (Convolution)

Параметры (torch.nn.Conv2d):

- Количество каналов на входе и выходе
- Размеры ядра
- Смещение (stride) – можно понижать разрешение
- Padding
- Dilation – увеличить область зависимости
- Размер выхода сети



Реализация свёртки

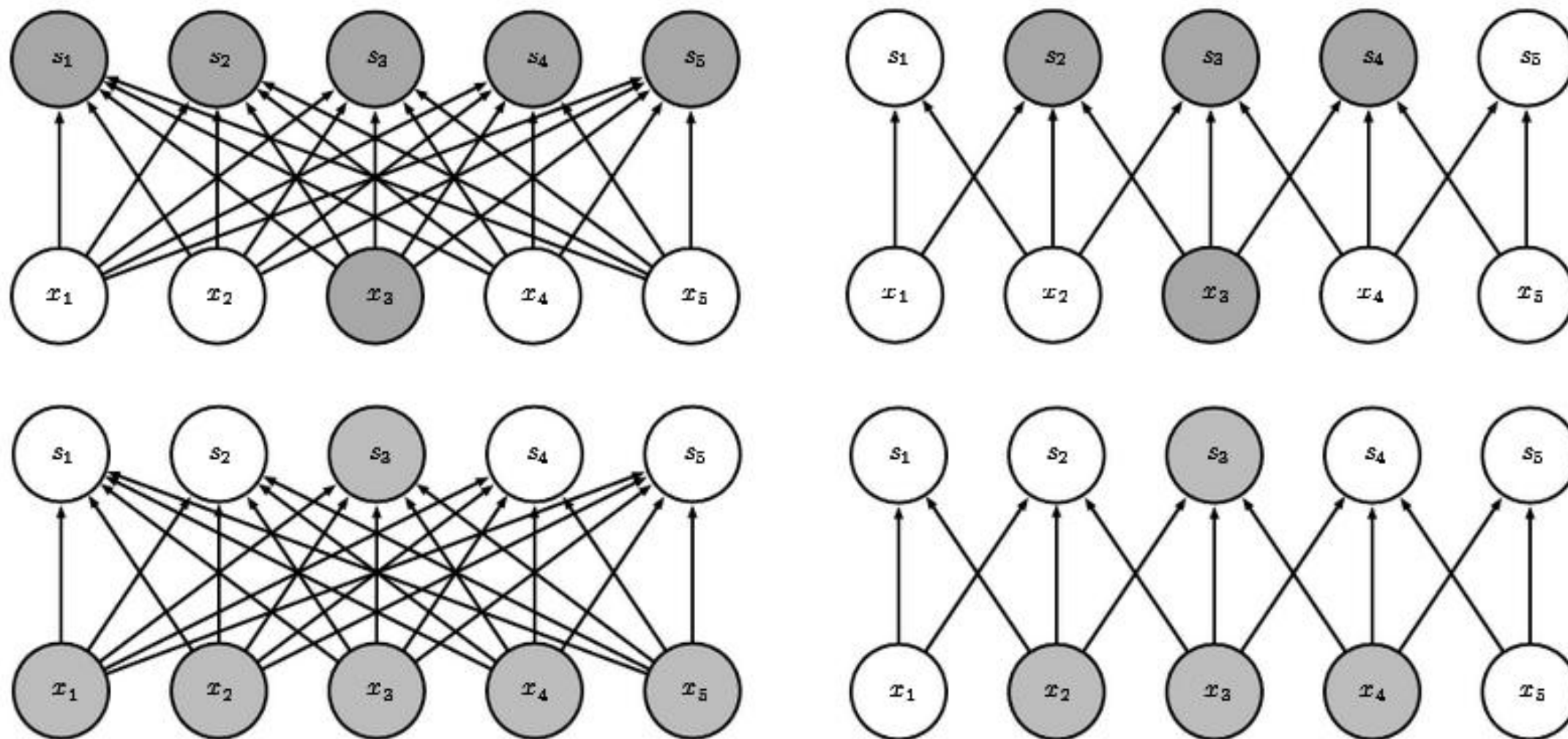
Это линейная операция!

$$\begin{pmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{pmatrix} * \begin{pmatrix} k_{11} & k_{12} \\ k_{21} & k_{22} \end{pmatrix} = \begin{pmatrix} k_{11} & k_{12} & 0 & k_{21} & k_{22} & 0 & 0 & 0 & 0 \\ 0 & k_{11} & k_{12} & 0 & k_{21} & k_{22} & 0 & 0 & 0 \\ 0 & 0 & 0 & k_{11} & k_{12} & 0 & k_{21} & k_{22} & 0 \\ 0 & 0 & 0 & 0 & k_{11} & k_{12} & 0 & k_{21} & k_{22} \end{pmatrix} \cdot \begin{pmatrix} x_{11} \\ x_{21} \\ x_{31} \\ x_{21} \\ x_{22} \\ x_{23} \\ x_{31} \\ x_{32} \\ x_{33} \end{pmatrix} =$$

$$= \begin{pmatrix} k_{11}x_{11} + k_{12}x_{21} + k_{21}x_{21} + k_{22}x_{22} \\ k_{11}x_{21} + k_{12}x_{31} + k_{21}x_{22} + k_{22}x_{23} \\ k_{11}x_{31} + k_{12}x_{21} + k_{21}x_{31} + k_{22}x_{32} \\ k_{11}x_{22} + k_{12}x_{23} + k_{21}x_{32} + k_{22}x_{33} \end{pmatrix}$$

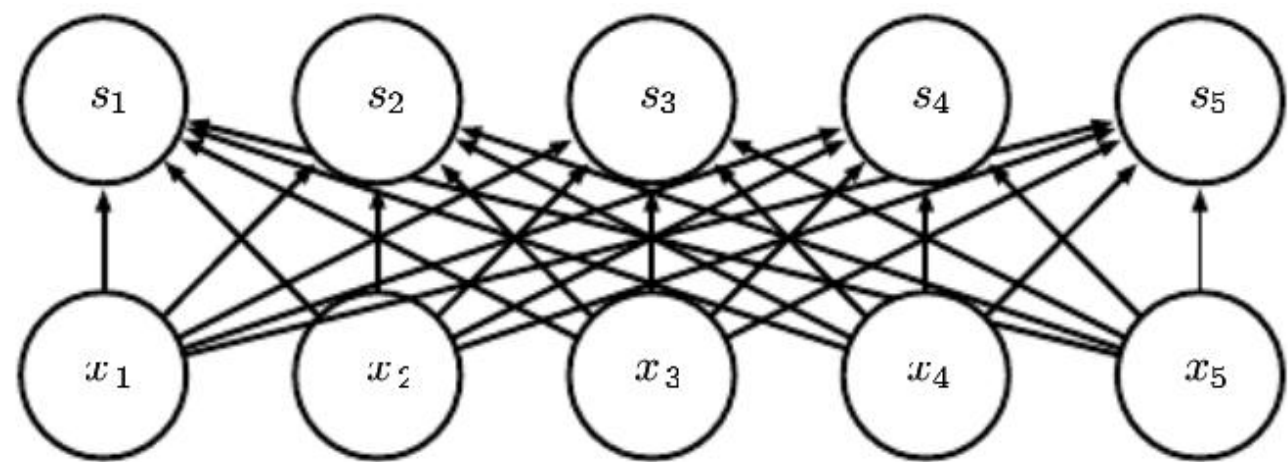
Поэтому надо быстро делать матричные перемножения...

Разреженные взаимодействия (sparse interactions)

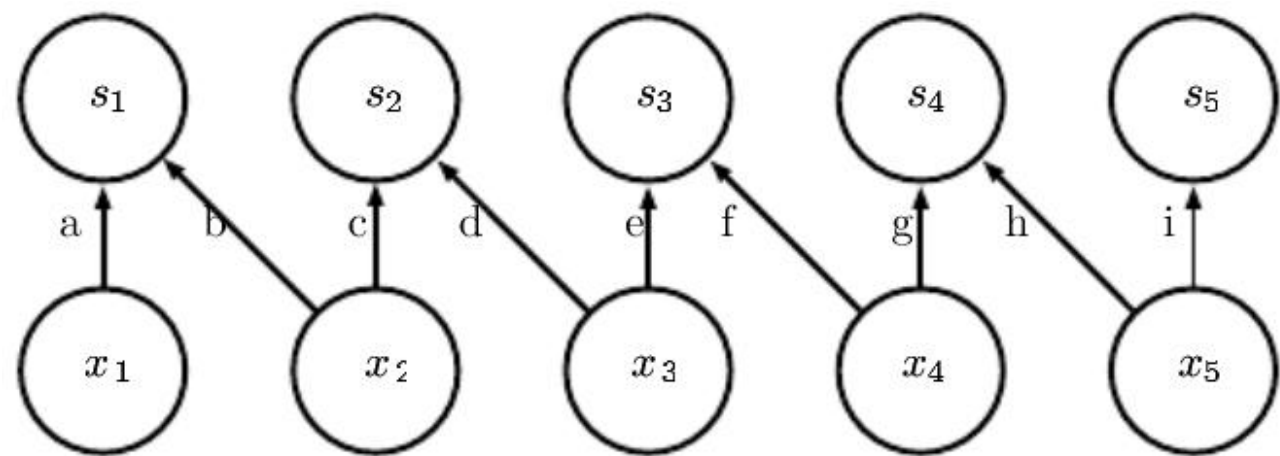


<http://www.deeplearningbook.org/contents/convnets.html>

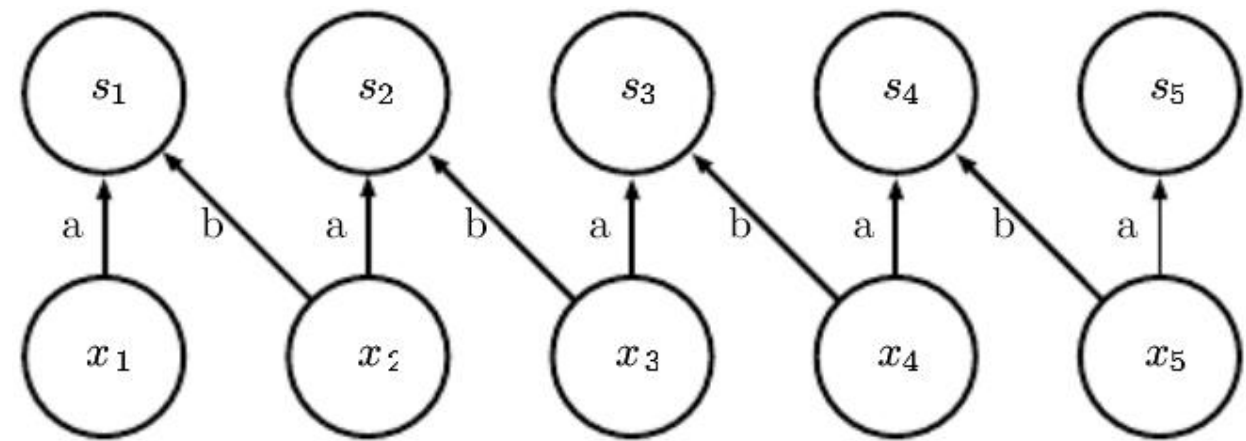
Полная связность (full connections)



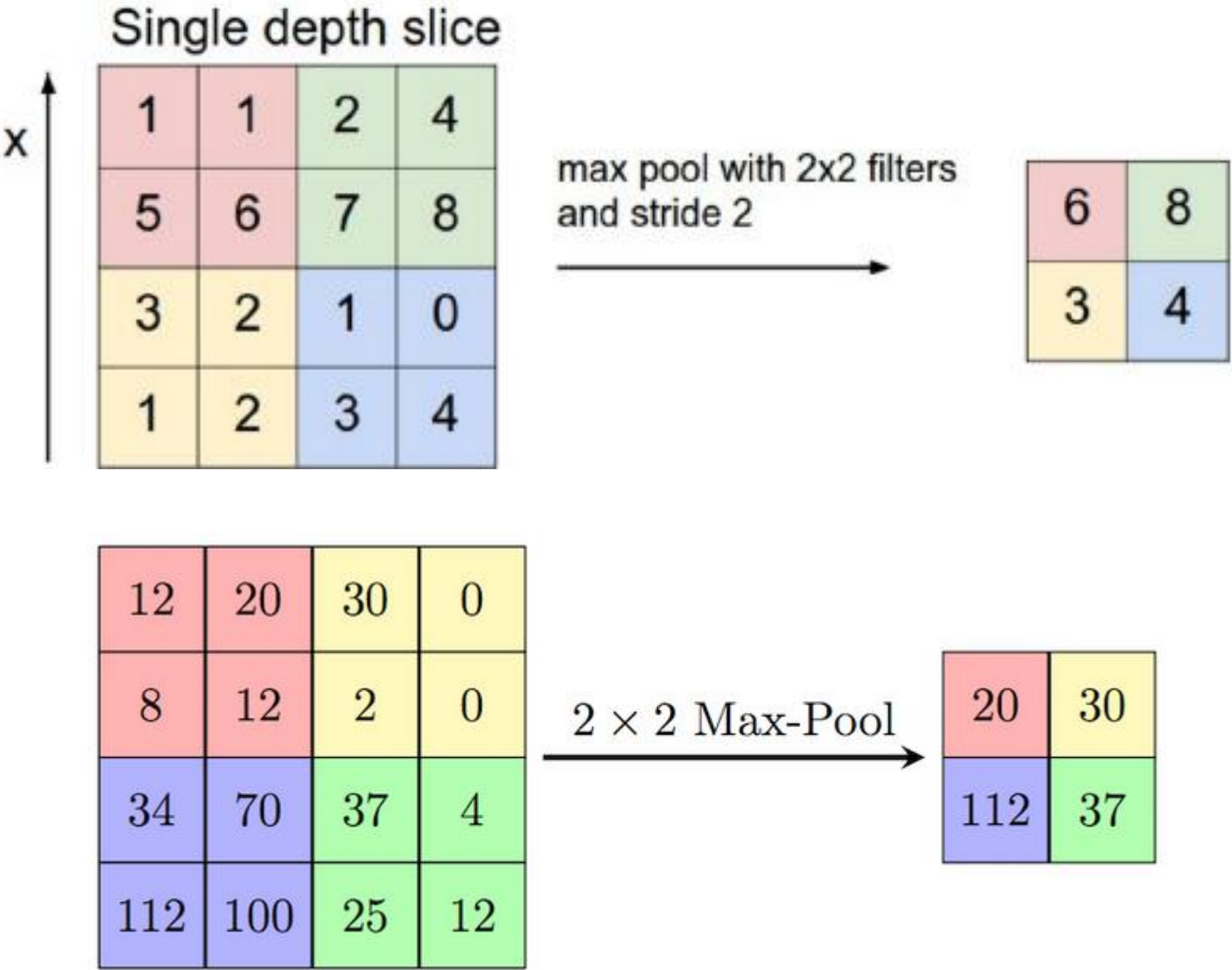
Локальная связность (local connections)
неразделяемая свёртка (unshared convolution)



Свёртка (convolution)



Pooling (агрегация, субдискретизация / subsampling)



Агрегация (Pooling) усреднением

3	3	2	1	0
0	0	1	3	1
3	1	2	2	3
2	0	0	2	2
2	0	0	0	1

1.7	1.7	1.7
1.0	1.2	1.8
1.1	0.8	1.3

3	3	2	1	0
0	0	1	3	1
3	1	2	2	3
2	0	0	2	2
2	0	0	0	1

1.7	1.7	1.7
1.0	1.2	1.8
1.1	0.8	1.3

3	3	2	1	0
0	0	1	3	1
3	1	2	2	3
2	0	0	2	2
2	0	0	0	1

1.7	1.7	1.7
1.0	1.2	1.8
1.1	0.8	1.3

3	3	2	1	0
0	0	1	3	1
3	1	2	2	3
2	0	0	2	2
2	0	0	0	1

1.7	1.7	1.7
1.0	1.2	1.8
1.1	0.8	1.3

3	3	2	1	0
0	0	1	3	1
3	1	2	2	3
2	0	0	2	2
2	0	0	0	1

1.7	1.7	1.7
1.0	1.2	1.8
1.1	0.8	1.3

3	3	2	1	0
0	0	1	3	1
3	1	2	2	3
2	0	0	2	2
2	0	0	0	1

1.7	1.7	1.7
1.0	1.2	1.8
1.1	0.8	1.3

3	3	2	1	0
0	0	1	3	1
3	1	2	2	3
2	0	0	2	2
2	0	0	0	1

1.7	1.7	1.7
1.0	1.2	1.8
1.1	0.8	1.3

3	3	2	1	0
0	0	1	3	1
3	1	2	2	3
2	0	0	2	2
2	0	0	0	1

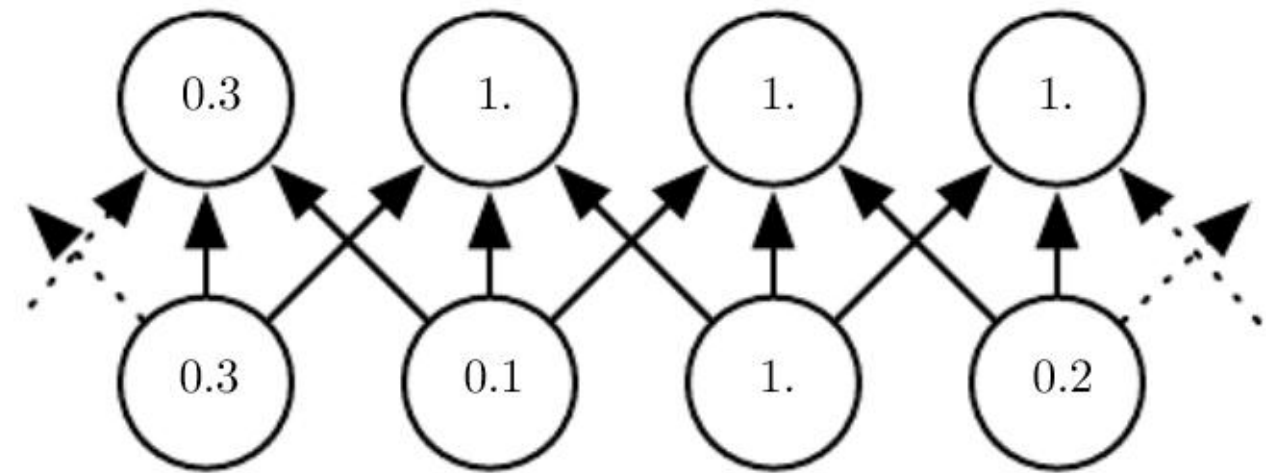
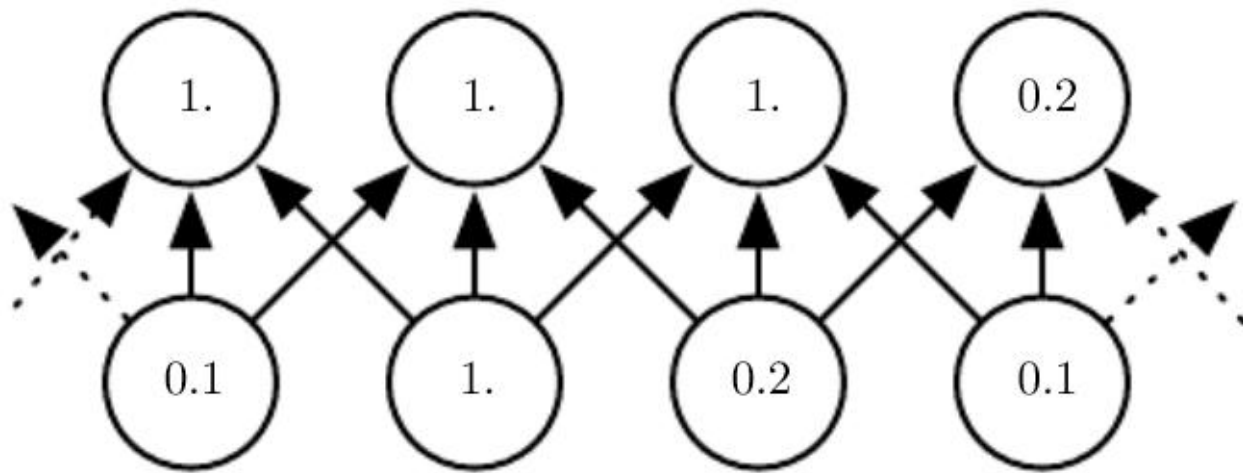
1.7	1.7	1.7
1.0	1.2	1.8
1.1	0.8	1.3

3	3	2	1	0
0	0	1	3	1
3	1	2	2	3
2	0	0	2	2
2	0	0	0	1

1.7	1.7	1.7
1.0	1.2	1.8
1.1	0.8	1.3

Агрегация (Pooling)

Max-pooling – инвариантность к небольшим сдвигам



Аналог голосования...

Если надо найти кошку, то в определённой окрестности
⇒ опросить соседей, есть ли кошка

Агрегация (Pooling): виды

Есть разные виды пулинга:

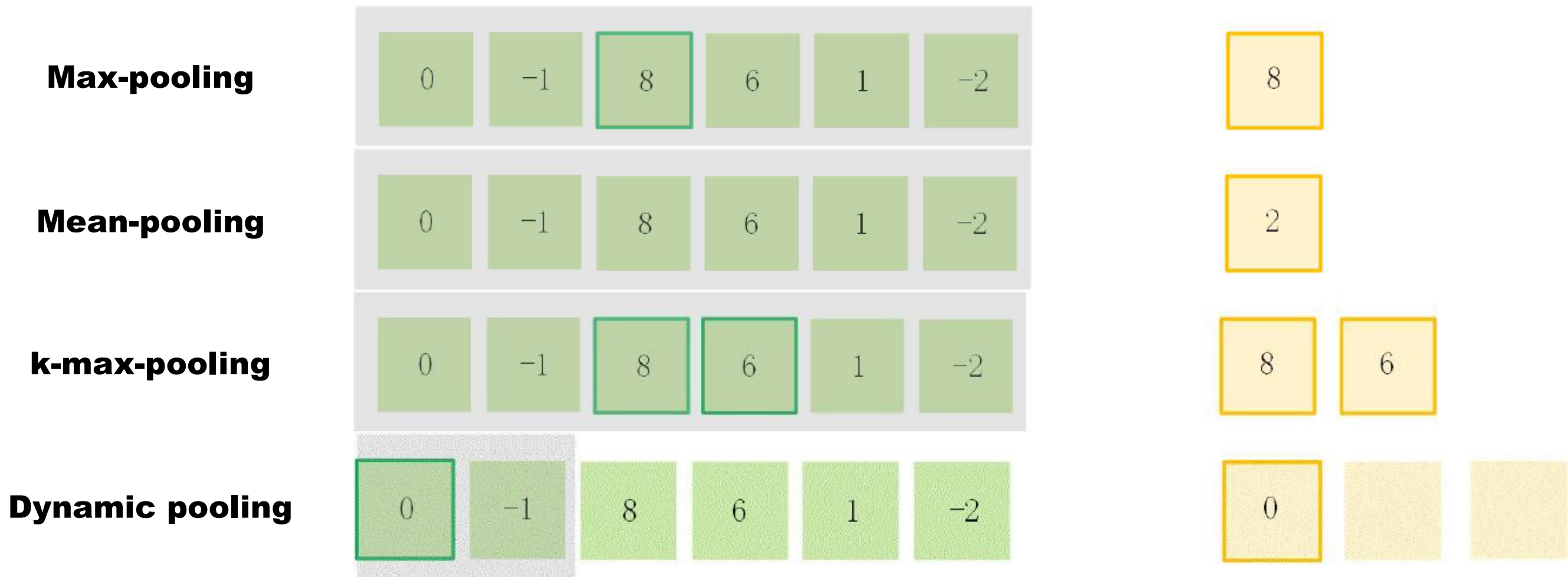
- усреднение
- усреднение с весами
- L2-норма
- **Stochastic Pooling**

выдаём значение с вероятностью ~ значение

При дифференцировании возвращают градиент в позициях максимумов

С помощью пулинга можно приводить изображение к нужному размеру!
(его можно делать с шагом)

Агрегация (Pooling): виды



<http://www.phontron.com/class/nn4nlp2020/schedule.html>

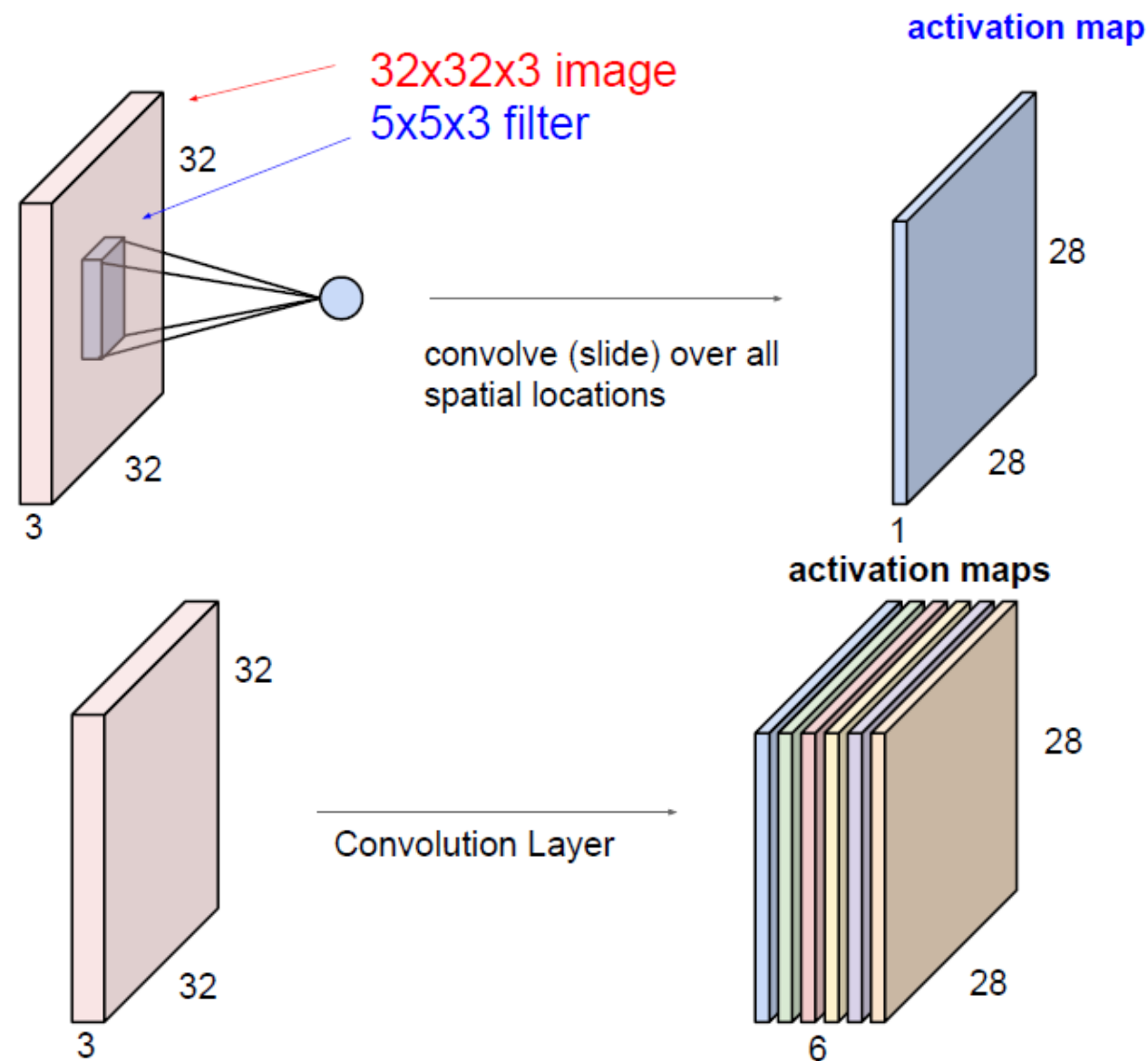
Устройство слоя свёрточной НС:

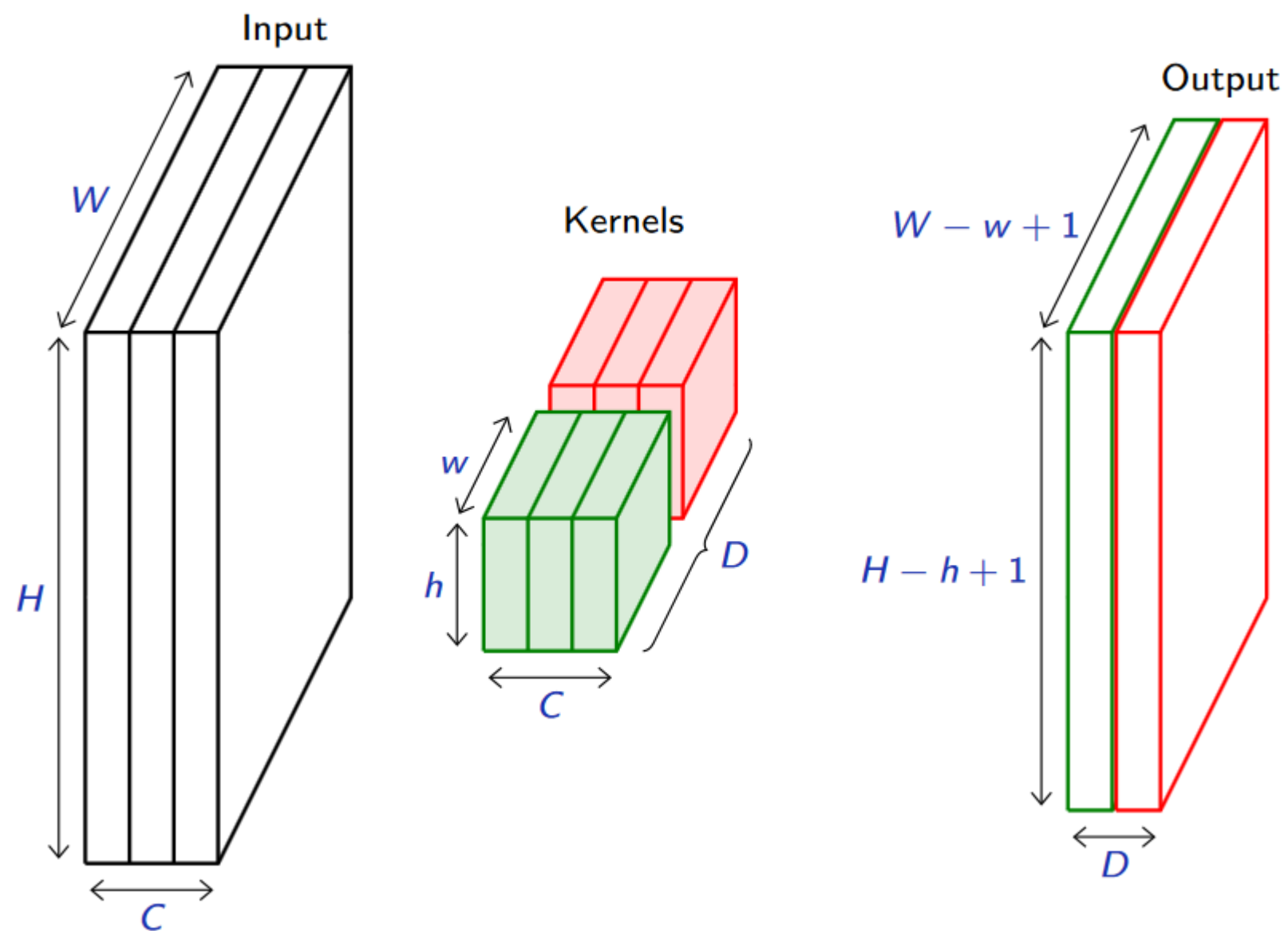
свёртка → нелинейность → пулинг

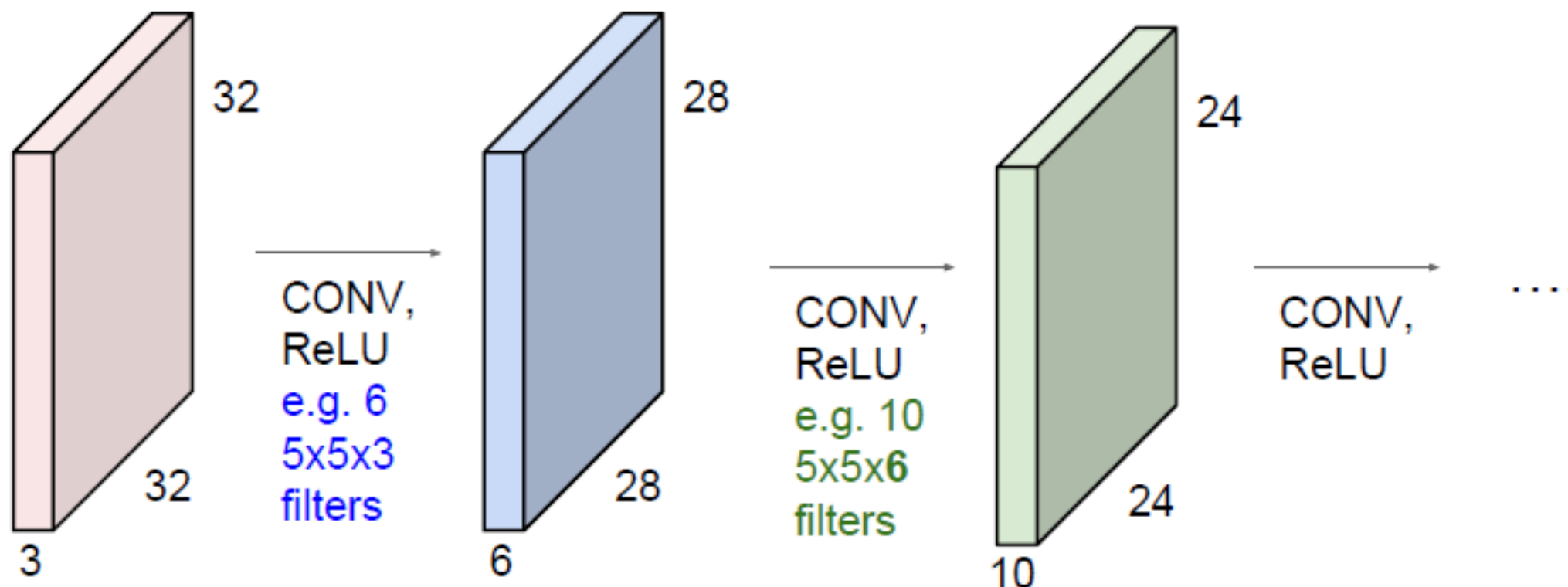
Мотивация:

- **разреженные взаимодействия (sparse interactions)**
нет связи нейронов «каждый с каждым»
У свёрточных НС мало весов!!!
- **разделение параметров (parameter sharing)**
одна свёртка используется «по всему изображению»
⇒ мало параметров
- **инвариантные преобразования (equivariant representations)**
инвариантность относительно сдвига

<http://www.deeplearningbook.org/contents/convnets.html>

Свёрточная НС: тензор \rightarrow тензор **$32 \times 32 \times 3 \rightarrow 28 \times 28 \times 6$ (карта признаков)**

Свёрточная НС: каждая свёртка – 1 «лист»

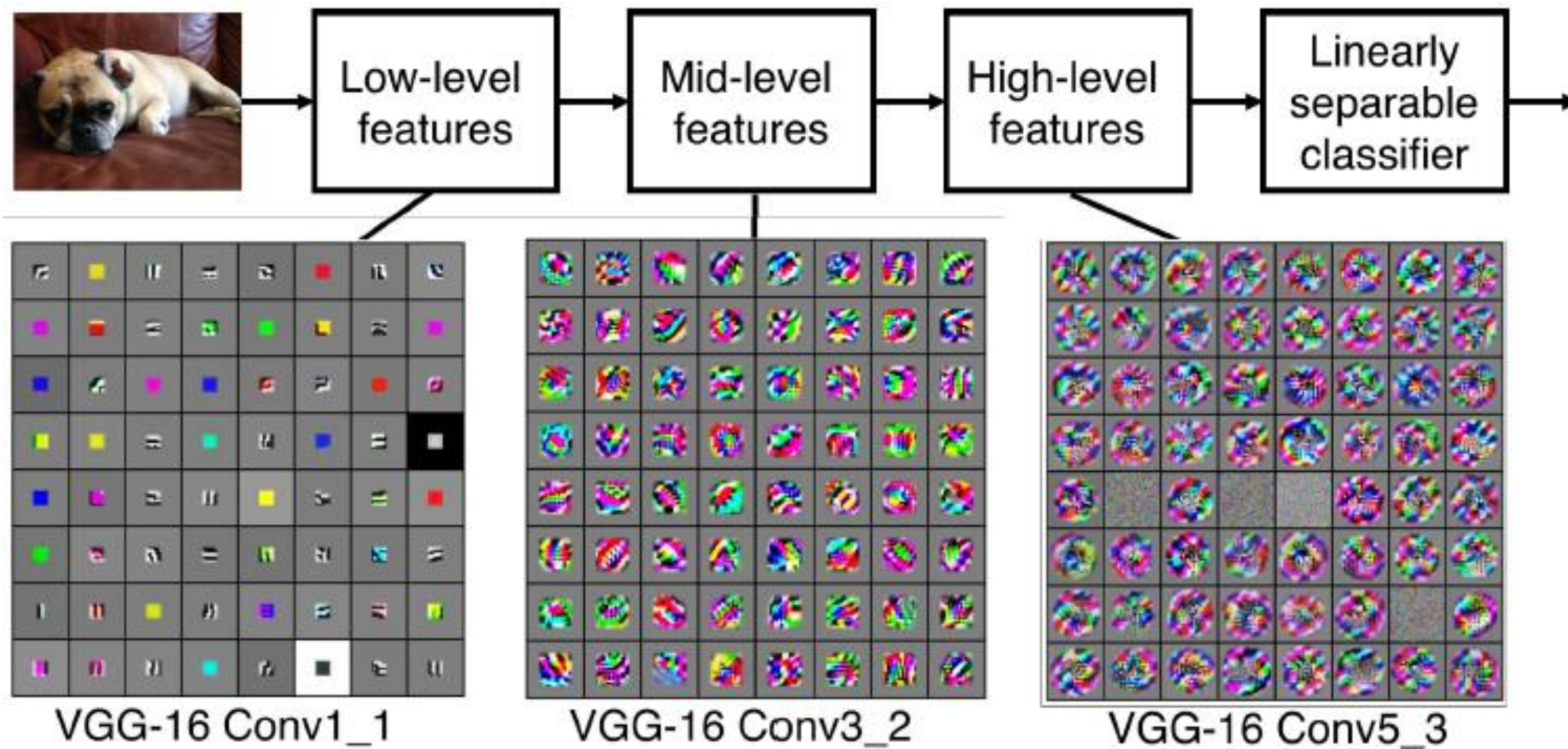
Свёрточная НС: тензор → тензор

Каждый тензор:
ширина × высота × # признаков / каналов (глубина)

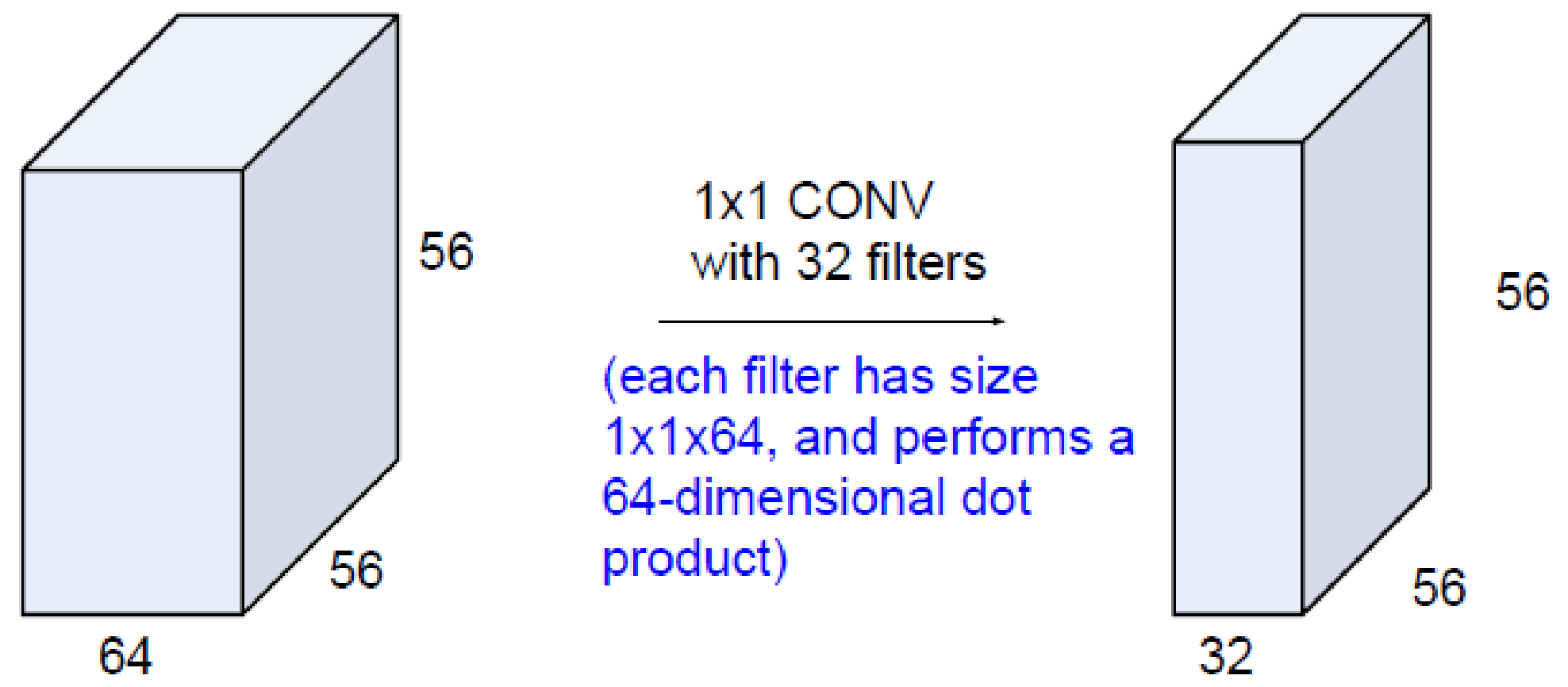
Важно:

Свёрточный слой: тензор → тензор (м.б. другой размерности)

Визуализация признаков

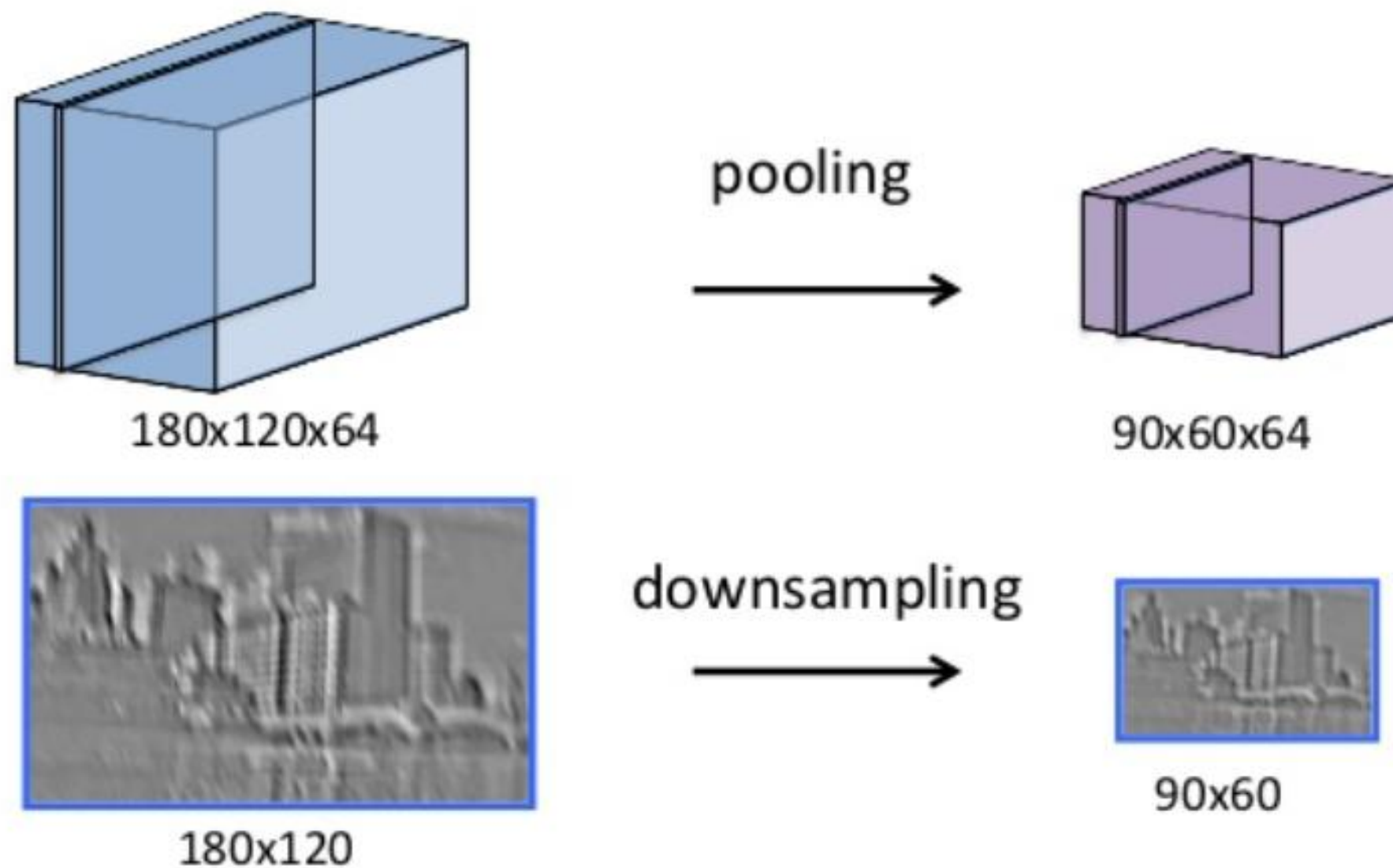


Смысл свёрток 1×1



Преобразование признаков!

Смысл пулинга

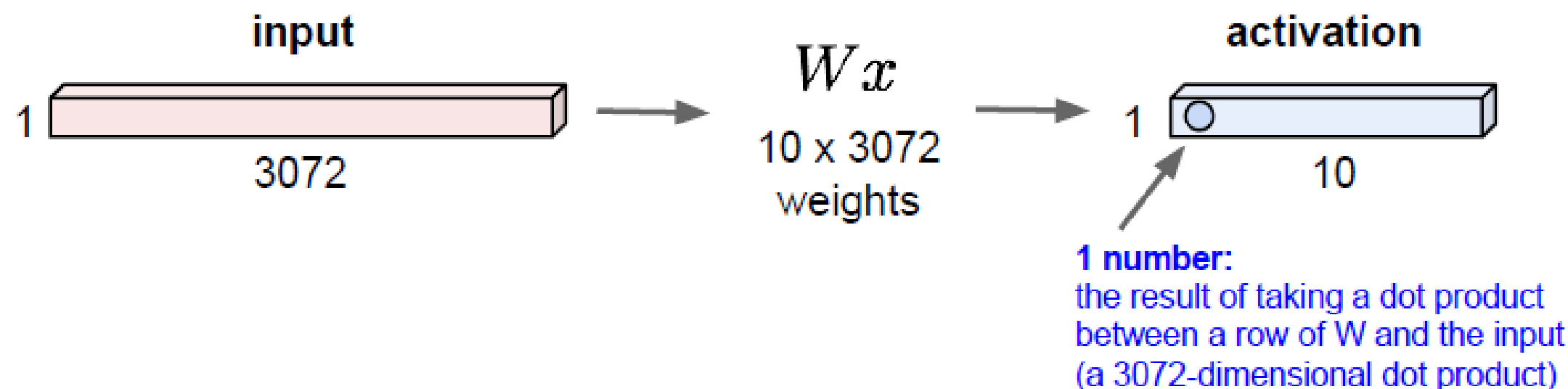


- уменьшение размеров
- на каждой карте действует независимо

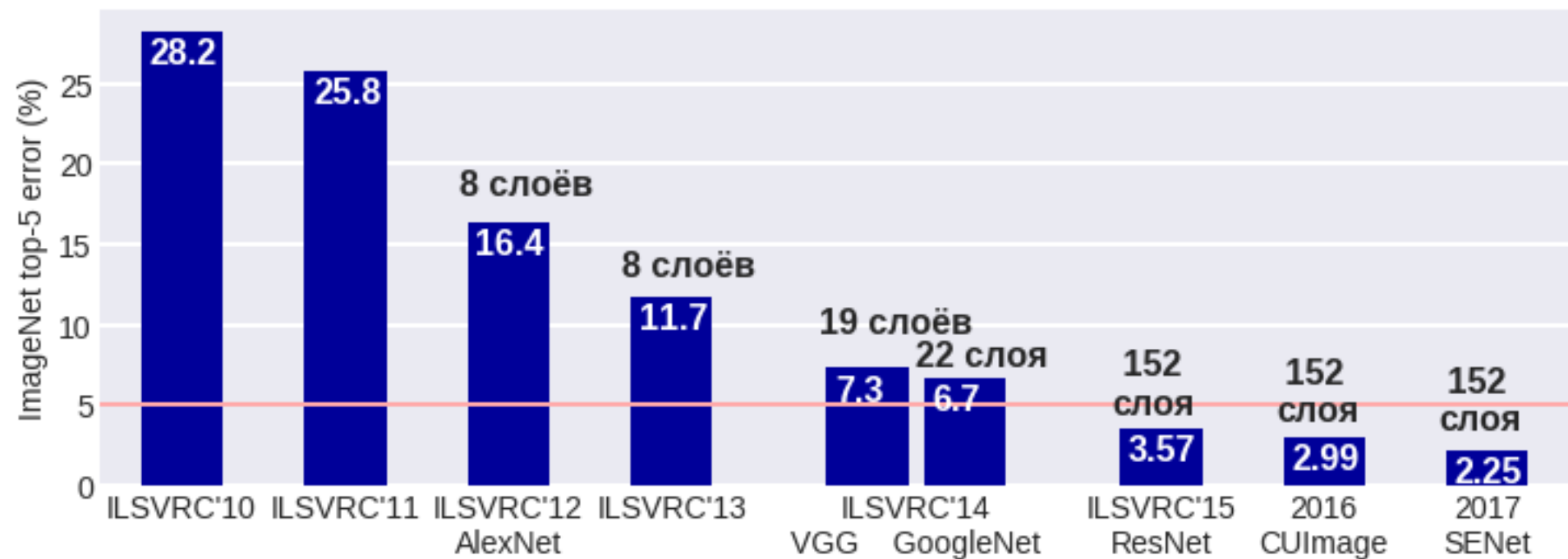
Полносвязный слой

32x32x3 image -> stretch to 3072 x 1

Each neuron
looks at the full
input volume



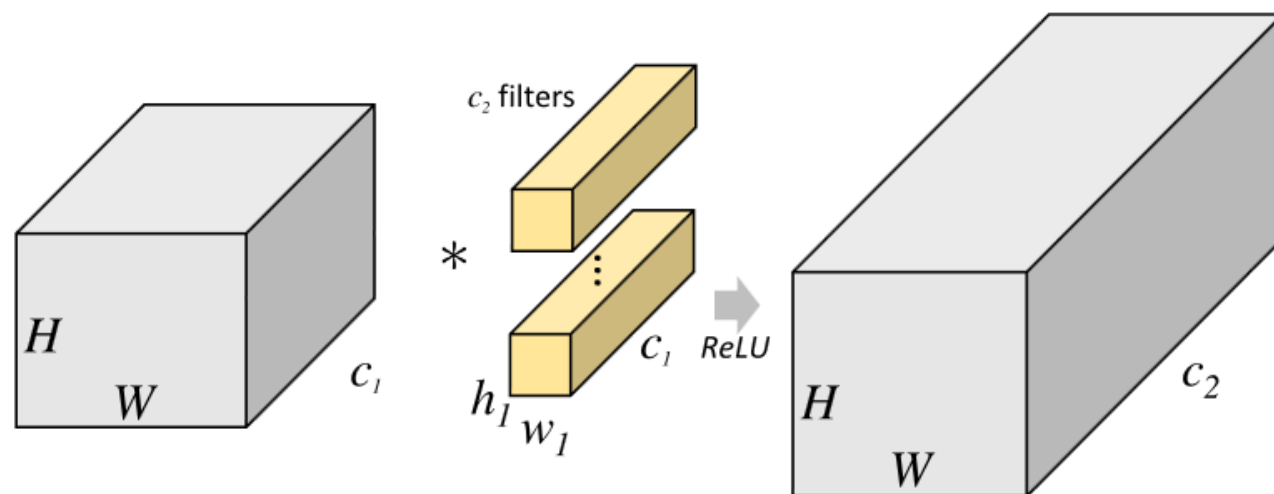
Революция в машинном обучении



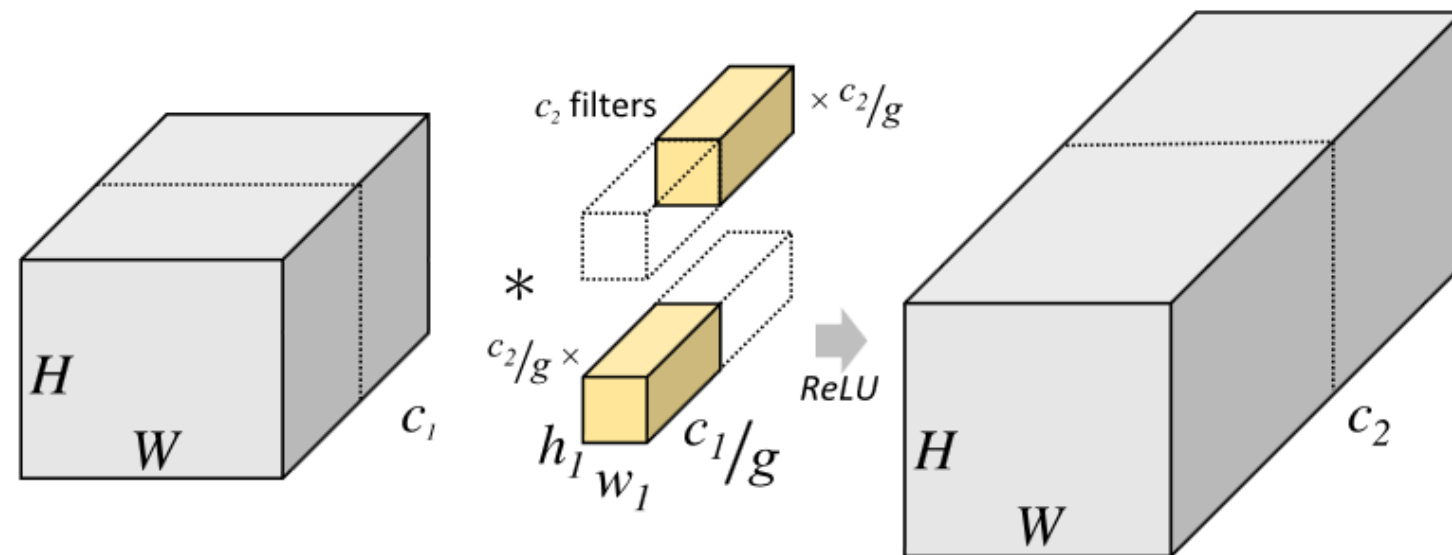
ошибка человека – 5.1

Какие бывают свёртки: Group Convolutions

Более частая ситуация



с разбиением на две группы

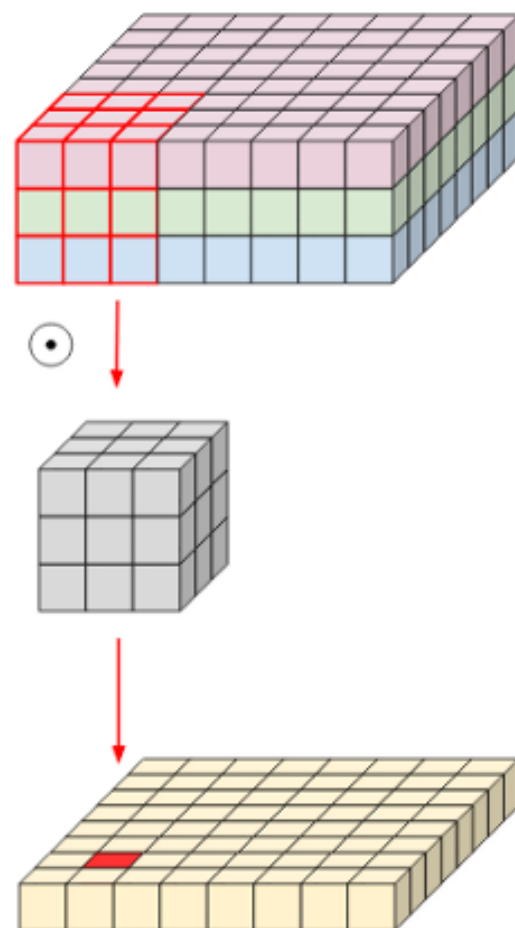


**идея из AlexNet, где были ограничения по памяти
могут быть лучшие (разреженные) признаковые представления
но выходные каналы зависят от узкой группы входных**

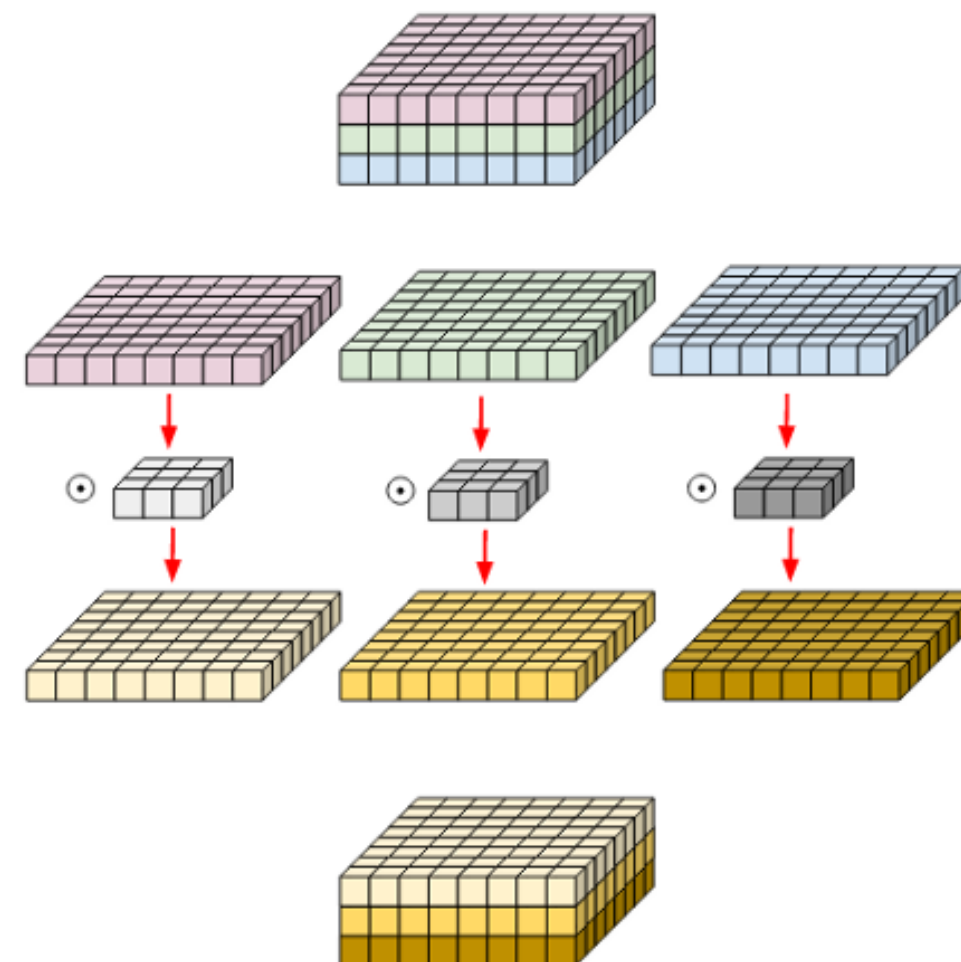
<https://blog.yani.io/filter-group-tutorial/>

Какие бывают свёртки

convolution



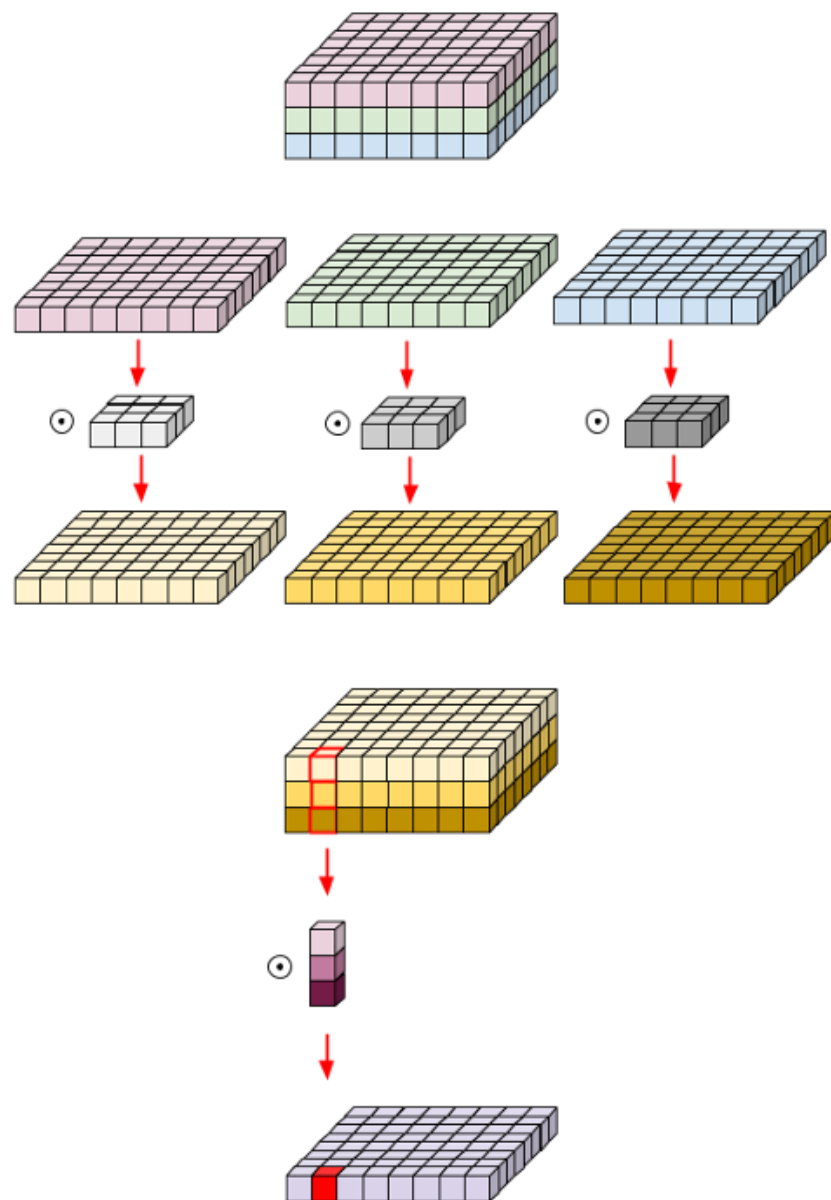
depth-wise convolution



каждый канал «сворачивается» отдельно

<https://eli.thegreenplace.net/2018/depthwise-separable-convolutions-for-machine-learning/>

Какие бывают свёртки: Depth-wise separable convolution



теперь результат зависит от всех каналов

$S=128, F=3, inC=3, outC=16$

Regular convolution:

Parameters:

$$3*3*3*16 = 432$$

Computation cost:

$$3*3*3*128*128*16 = \sim 7e6$$

Depthwise separable convolution:

Parameters:

$$3*3*3+3*16 = 75$$

Computation cost:

$$3*3*3*128*128+128*128*3*16 \\ = \sim 1.2e6$$

Итог

В изображениях свёртки – естественная операция
Естественное устройство CNN: свёртка, пулинг, нелинейность, полносвязность

В отличие от классического CV не придумываем фильтры
Они обучаются сами!

Свёртка – первый пример разделения весов.
Есть способы экономии параметров – и ими пользуются!

Свёртки продолжают совершенствоваться
(более разумные представления, экономия параметров)