

The background of the slide is a photograph of the main building of Moscow State University, featuring its iconic Spasskaya Tower with a tall spire. The building is set against a blue sky with light, wispy clouds. In the foreground, there are dark, leafless trees and some lower-level urban buildings.

Глубокое обучение: Свёрточные нейронные сети

Дьяконов А.Г.

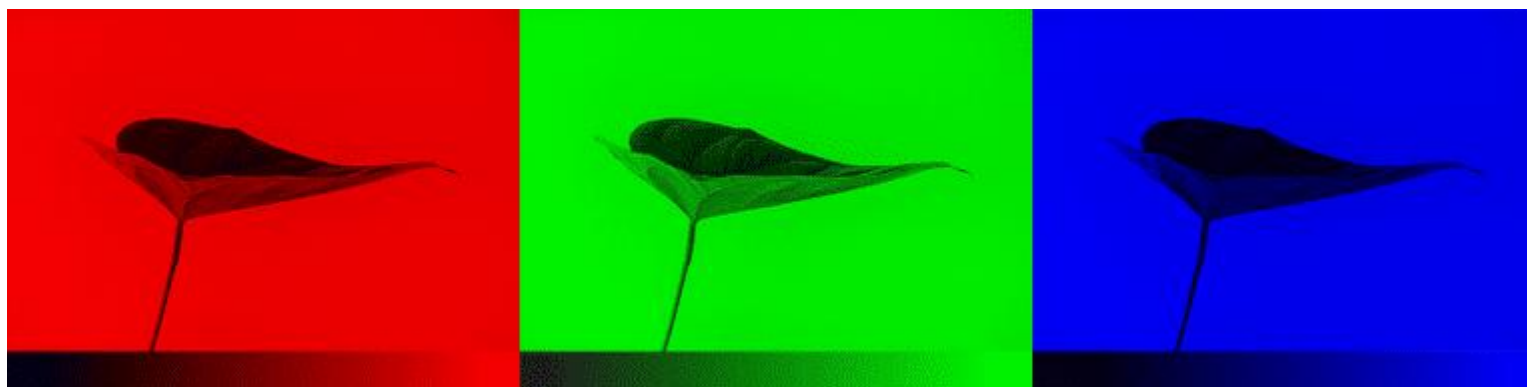
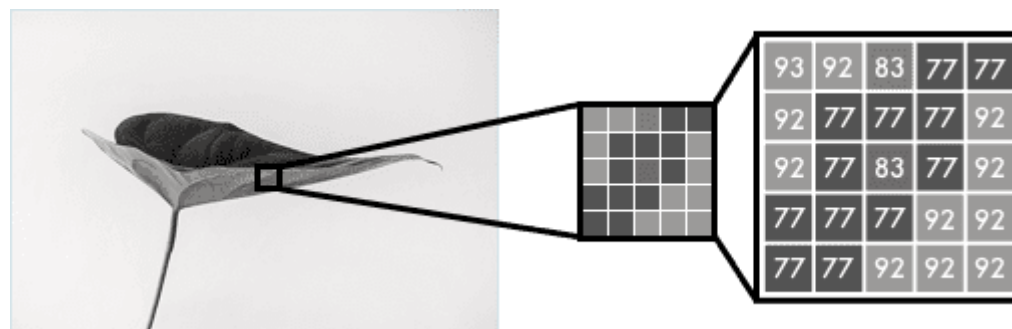
**Московский государственный университет
имени М.В. Ломоносова (Москва, Россия)**

Что такое изображение

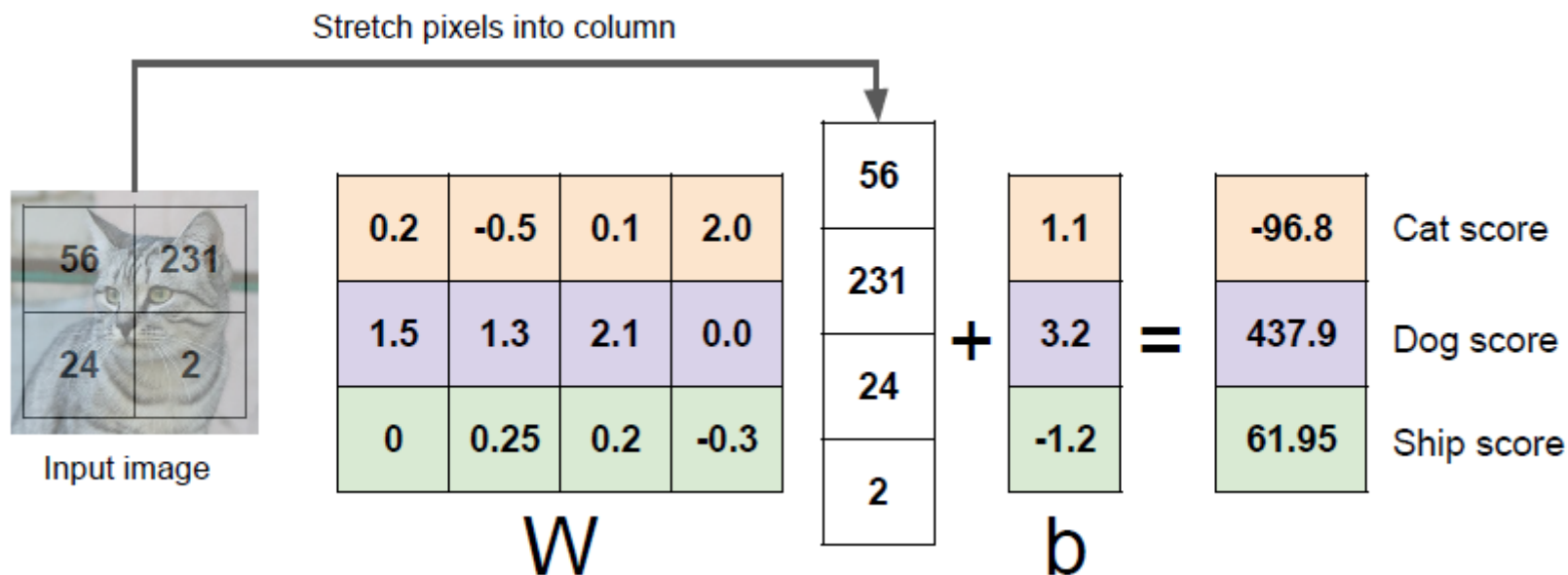


62	62	63	64	65	66	67	67	69	70	71	72	72	73	73	73	72	72	71	70	69	67	66	66	65	63	62	61	60	60	
61	62	63	64	66	66	67	68	68	69	70	71	72	72	73	72	72	71	71	70	69	68	66	66	65	65	63	62	61	60	60
61	62	63	64	66	66	68	68	69	70	71	72	73	73	73	72	71	71	69	68	67	66	66	65	65	64	63	62	61	61	60
61	63	64	64	66	67	68	68	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
63	64	65	66	67	68	69	69	70	70	71	72	42	4	5	11	58	72	71	71	69	69	68	67	66	65	64	62	62	60	59
63	65	66	66	68	68	69	70	71	71	71	72	18	4	4	7	8	66	71	70	69	68	67	66	65	64	63	61	59	59	58
63	65	67	67	68	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	30	52	66	68	67	66	65	64	63	61	60	59	58
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
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
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
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
63	64	64	65	14	5	6	5	5	4	5	4	18	7	5	4	19	10	11	65	64	64	63	61	66	62	61	60	59	58	56
63	64	64	65	53	7	4	5	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
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	58
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
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	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
32	32	32	33	6	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
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
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
54	54	53	8	5	5	5	6	5	6	5	6	13	52	42	21	51	54	51	49	49	50	22	41	45	42	41	40	41	44	43
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
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
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	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	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	46
51	49	50	50	22	13	19	36	13	12	42	50	40	73	50	50	50	49	48	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	47	52	47	47	44	43
48	50	48	52	19	13	33	38	18	18	36	49	51	54	47	47	49	46	46	49	49	49	47	44	53	44	48	44	46	46	45

Что такое изображение



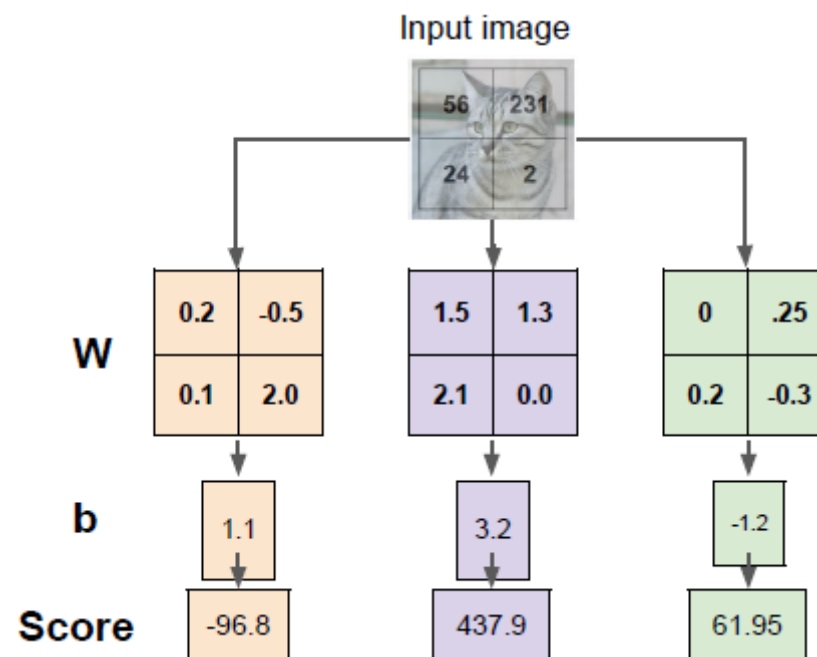
Линейный подход к классификации на несколько классов



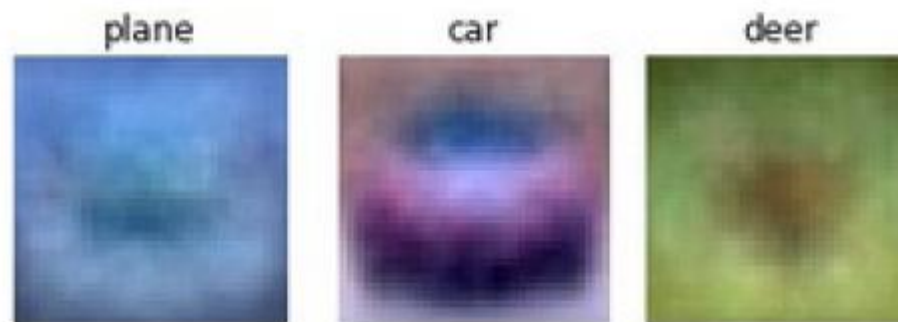
3 класса = 3 вектора признаков

Изображение = вытянуть в вектор

Линейный подход к классификации на несколько классов



Вектора весов можно потом опять «прорешить» в изображения:



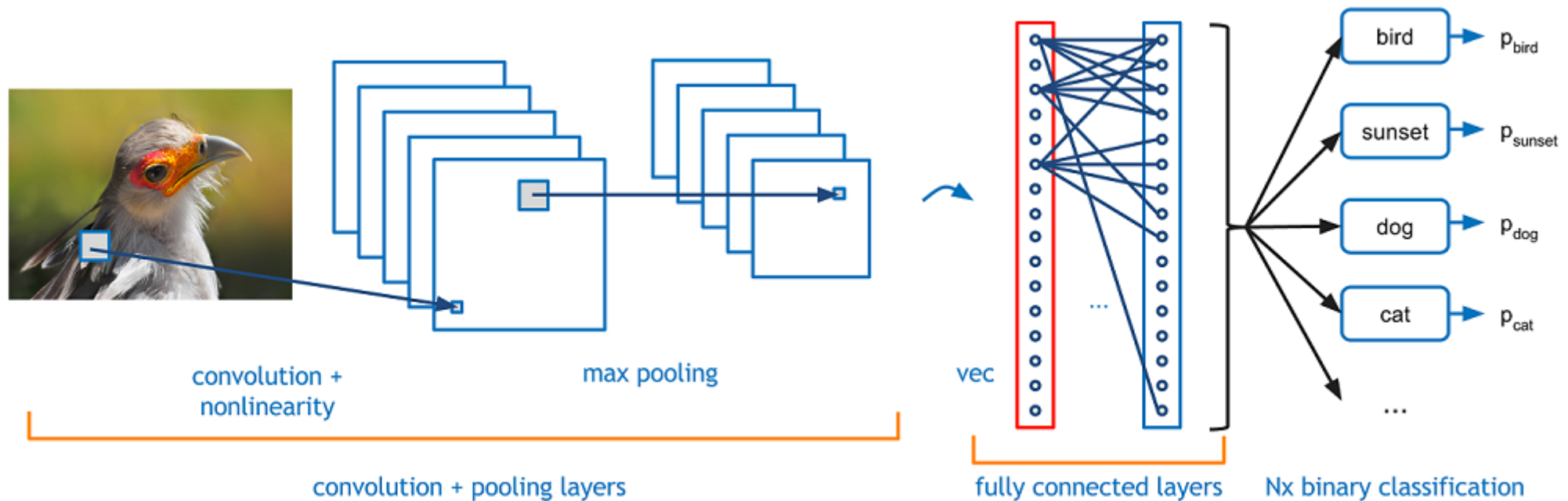
Проблема

Детектирование объекта в одном месте изображения
Примитивность модели

В обычных сетях слишком много параметров!

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

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/>

Что такое свёртка (Convolution)

сначала 1D-свёртки

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:

$$\begin{aligned} I * K = & \\ & (i_1 k_1 + \dots + i_r k_r, \\ & i_2 k_1 + \dots + i_{r+1} k_r, \\ & \dots \\ & i_{n-r+1} k_1 + \dots + i_n k_r) \\ & \in \mathbb{R}^{n-r+1} \end{aligned}$$

Свёртка (Convolution)

-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		

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>

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

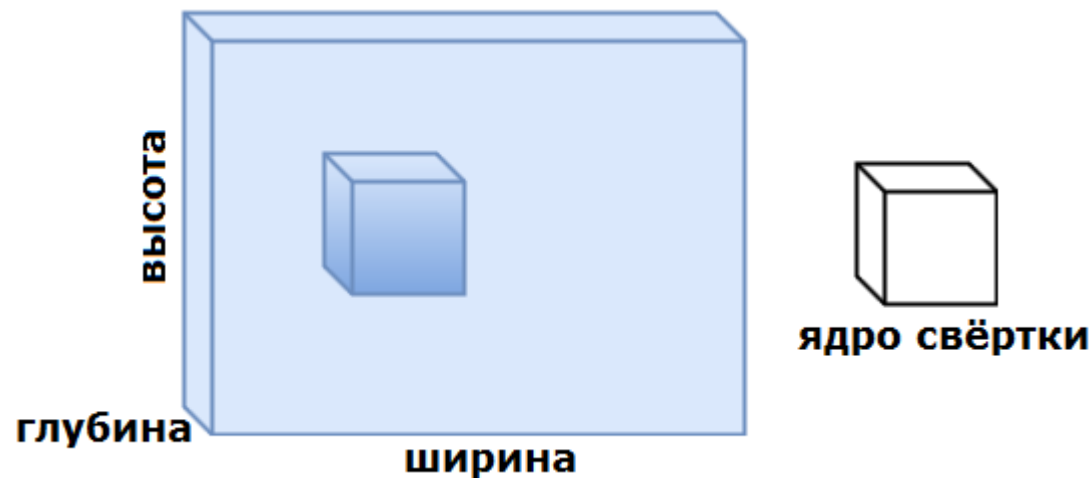
3	3	2	1	0
0	0	1	3	1
3	1 ₀	2 ₁	2 ₂	3
2	0 ₂	0 ₂	2 ₀	2
2	0 ₀	0 ₁	0 ₂	1

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

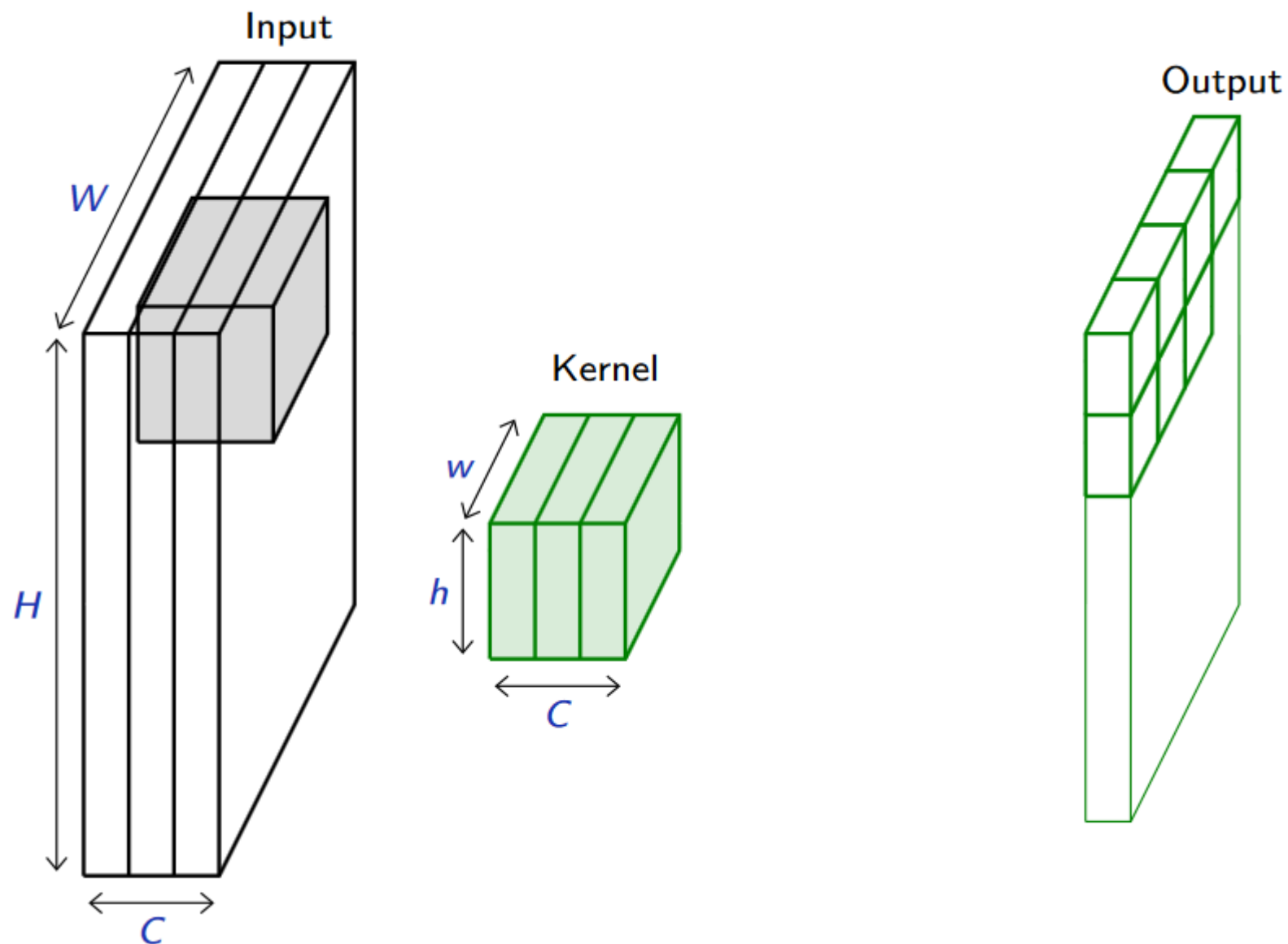
Высота (height) и ширина (width) каждого ядра

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')

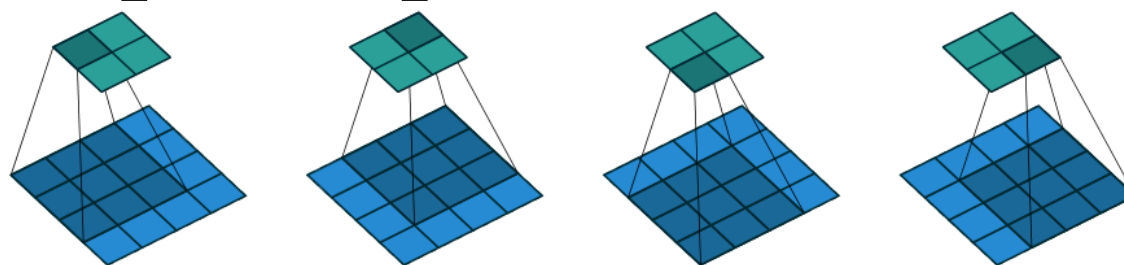
ax2.axis('off')
ax2.imshow(gx, cmap=plt.cm.gray)
# ax2.set_title('Horizontal gradients')

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')
```

Отступ (padding) – чтобы сохранялись размеры изображения

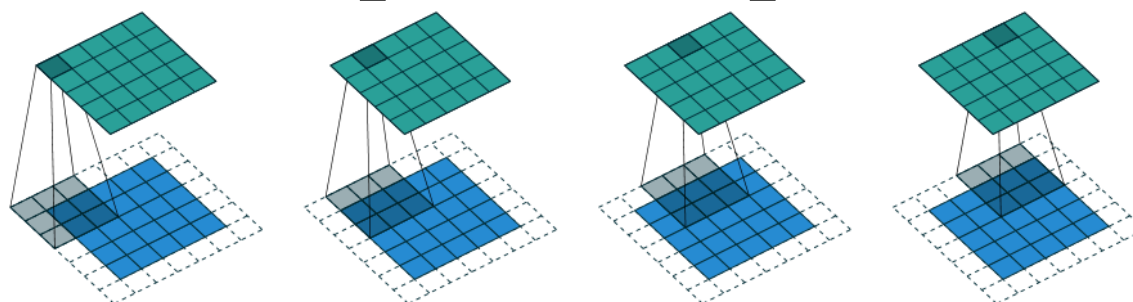
padding = VALID (в TF)

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



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

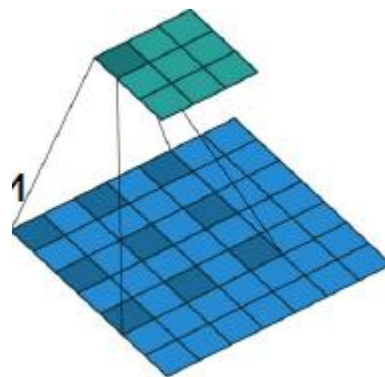
0	0	0	0	0	0	0
0	3	3	2	1	0	0
0	0	0	1	3	1	0
0	3	1	2	2	3	0
0	2	0	0	2 ₀	2 ₁	0 ₂
0	2	0	0	0 ₂	1 ₂	0 ₀
0	0	0	0	0 ₀	0 ₁	0 ₂

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!

Поэтому надо быстро делать матричные перемножения...

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

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

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

Свёртка (Convolution): мотивация

Раньше: обработка изображений – специально построенные свёртки

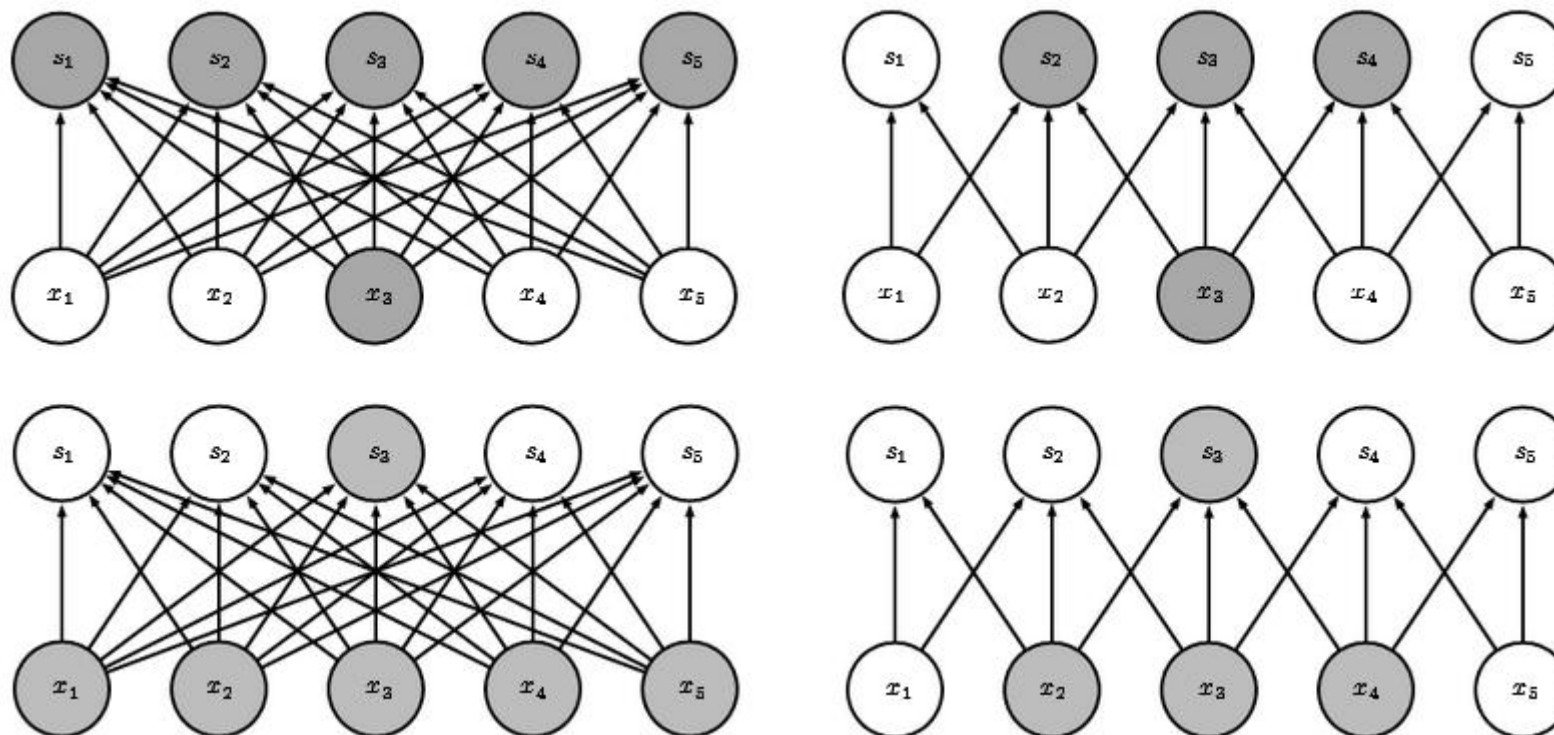


Сейчас: не будем специально строить свёртки – их параметры настроятся сами!

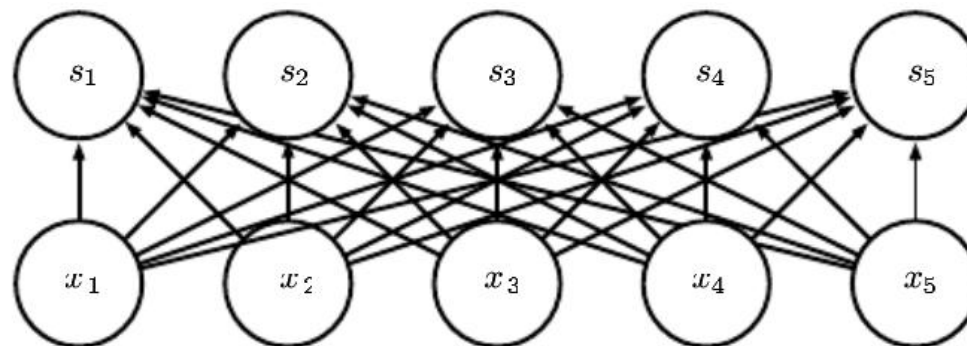
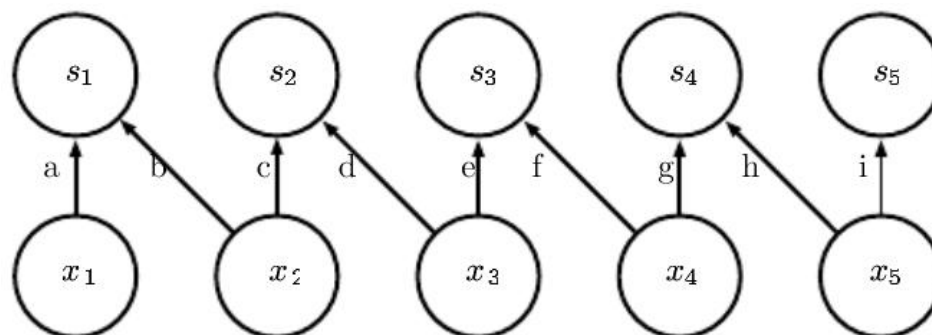
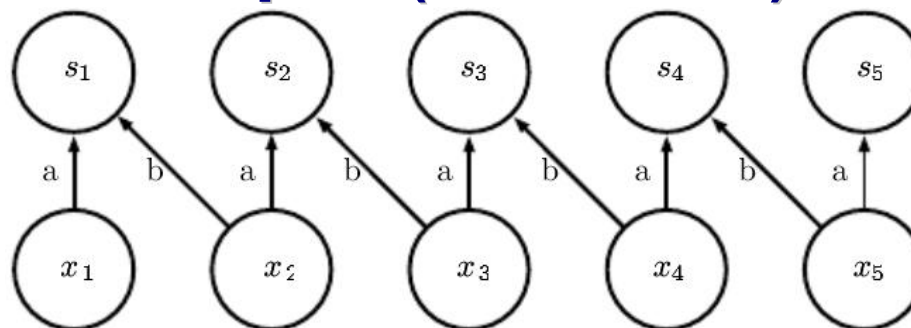
Важно: свёртку можно применять к изображениям любых размеров!

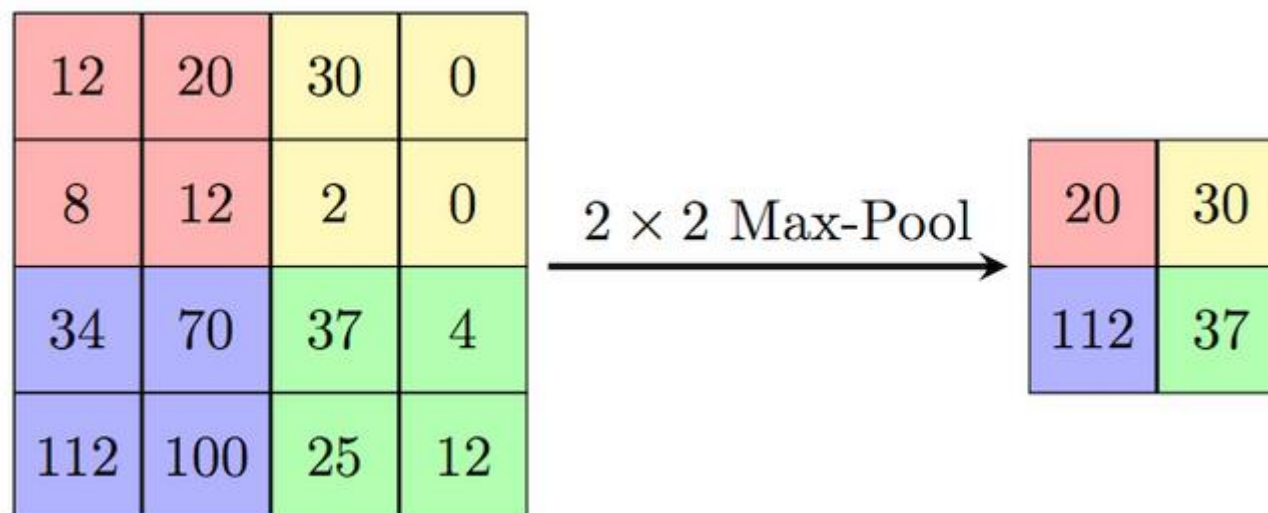
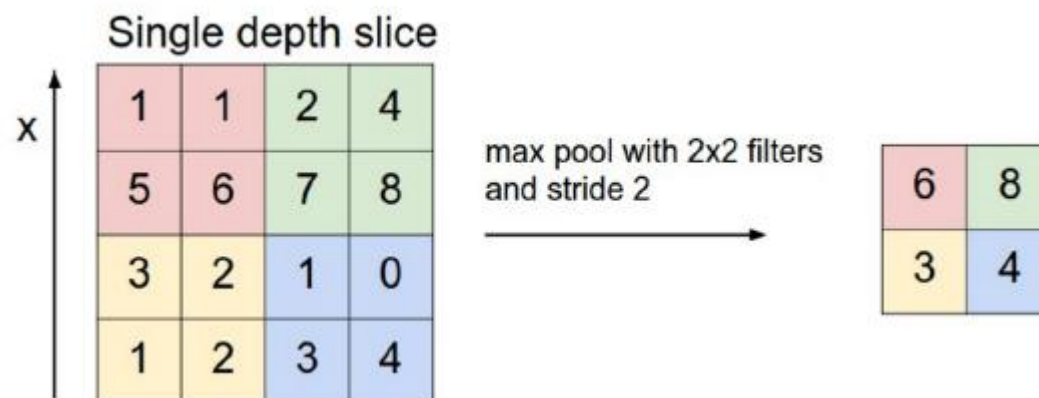
Нет ограничений на размеры входа...

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

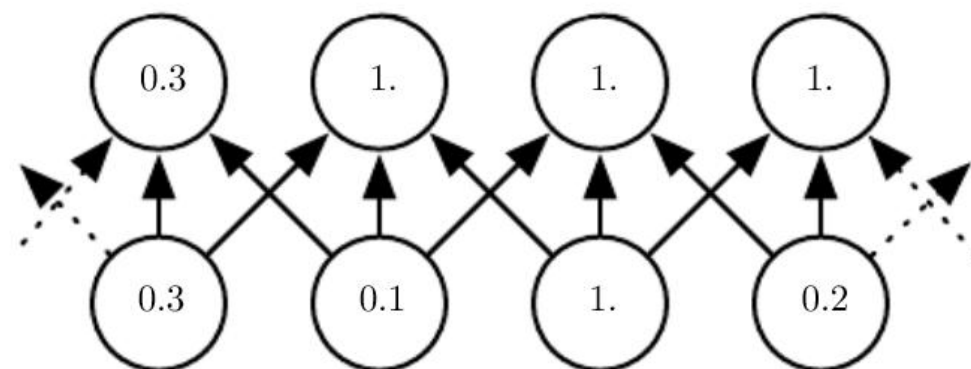
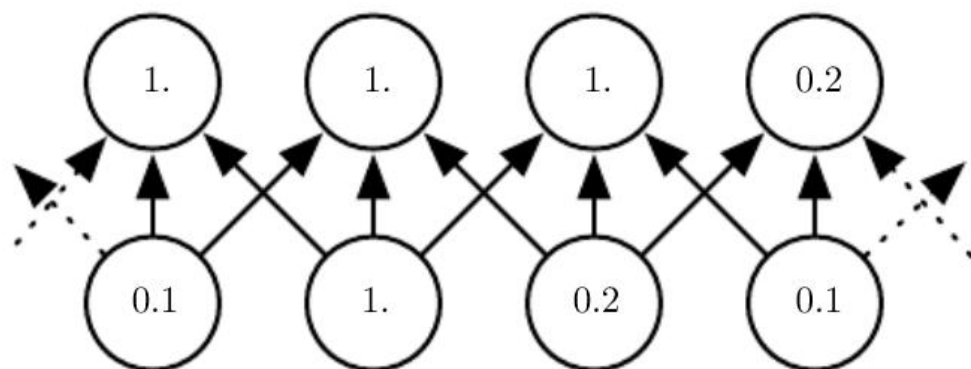
1.7	1.7	1.7
1.0	1.2	1.8
1.1	0.8	1.3

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)

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)

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)**

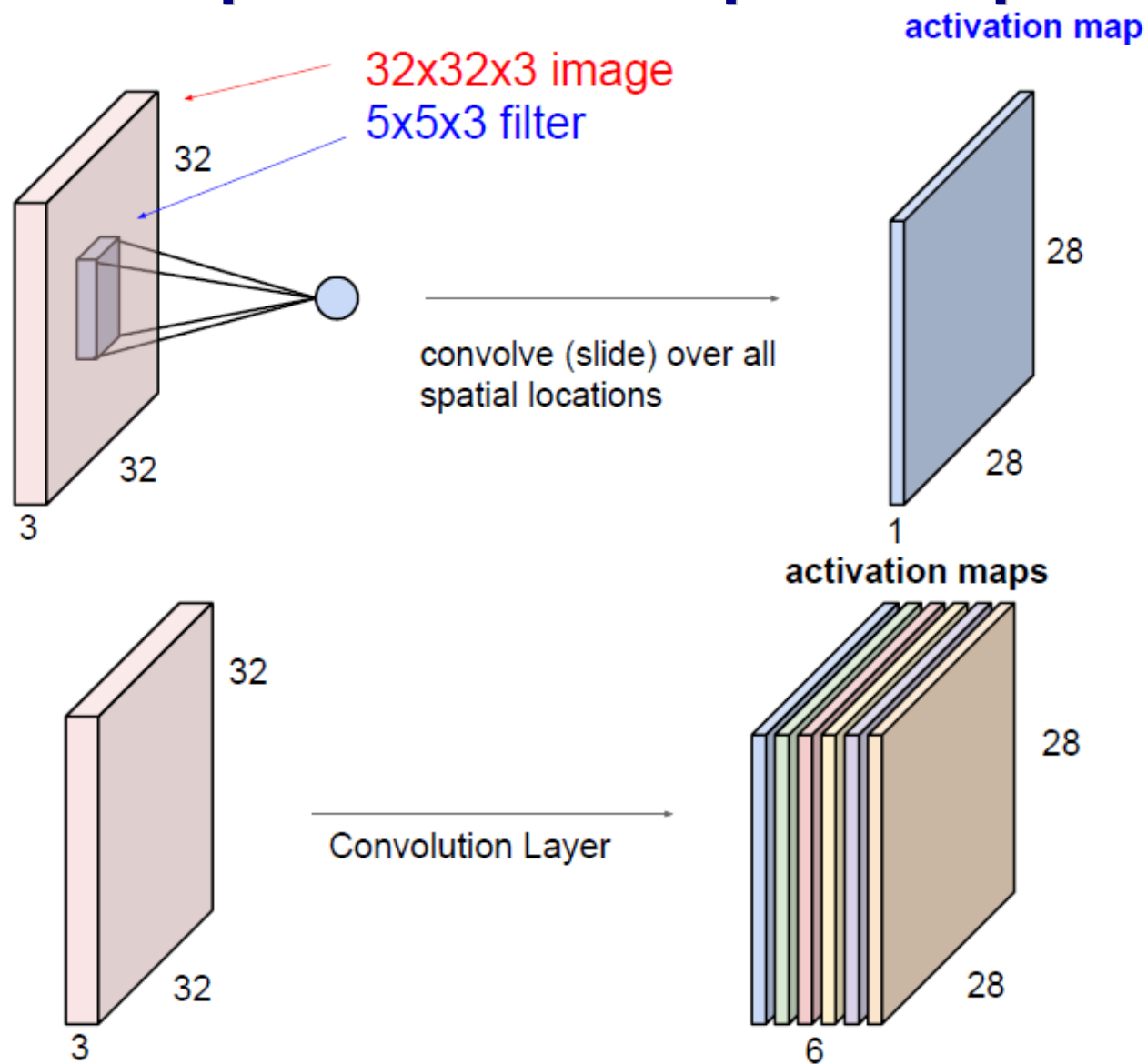
одна свёртка используется «по всему изображению»

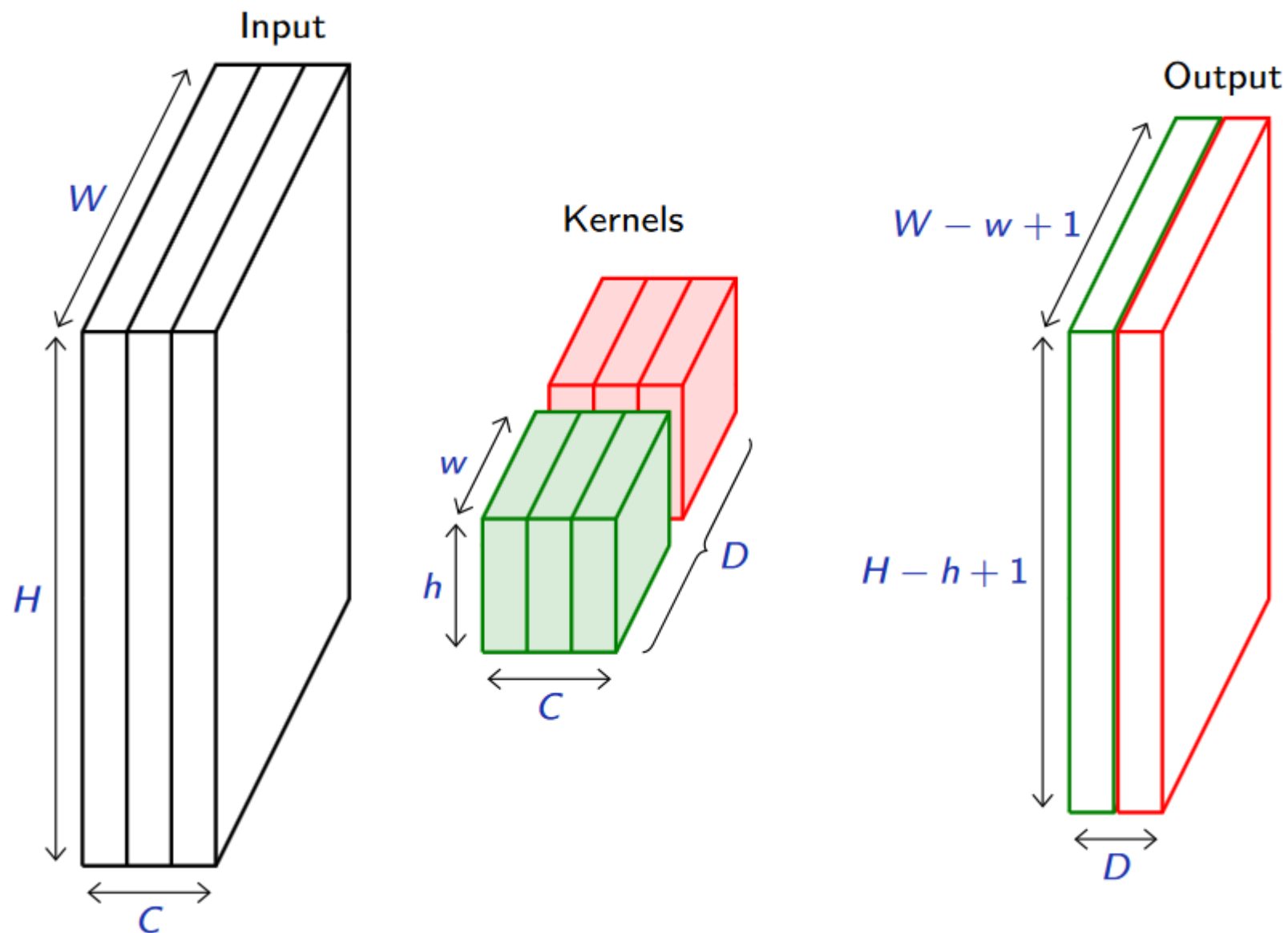
⇒ мало параметров

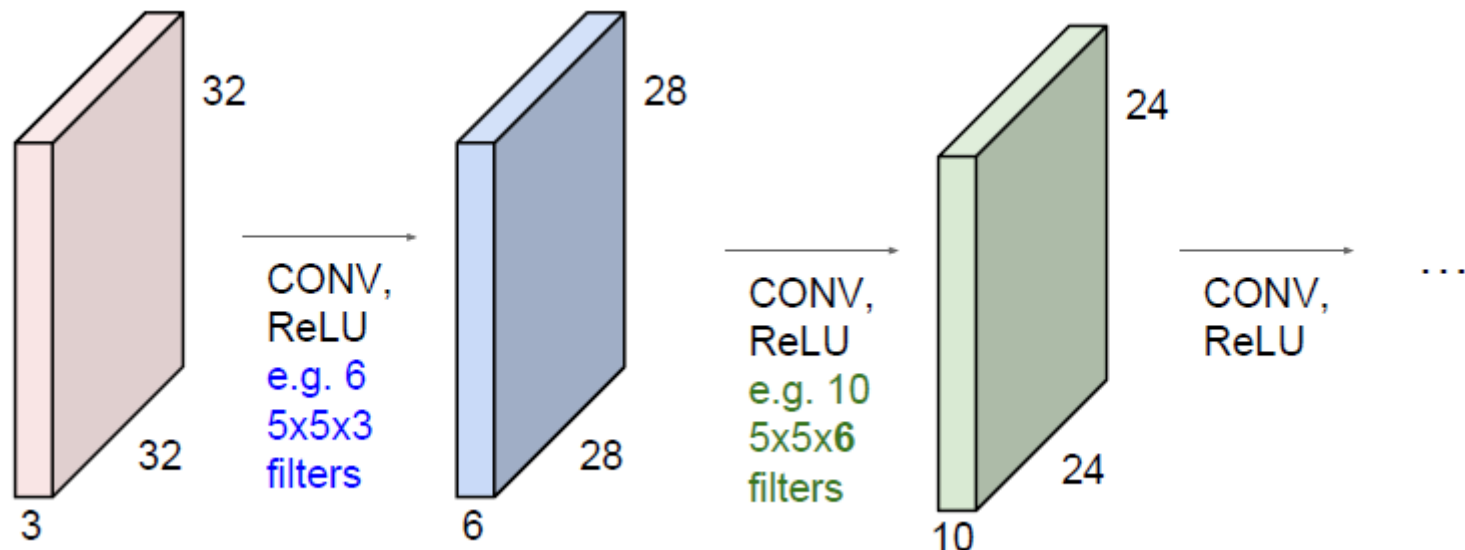
- **инвариантные преобразования (equivariant representations)**

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)**

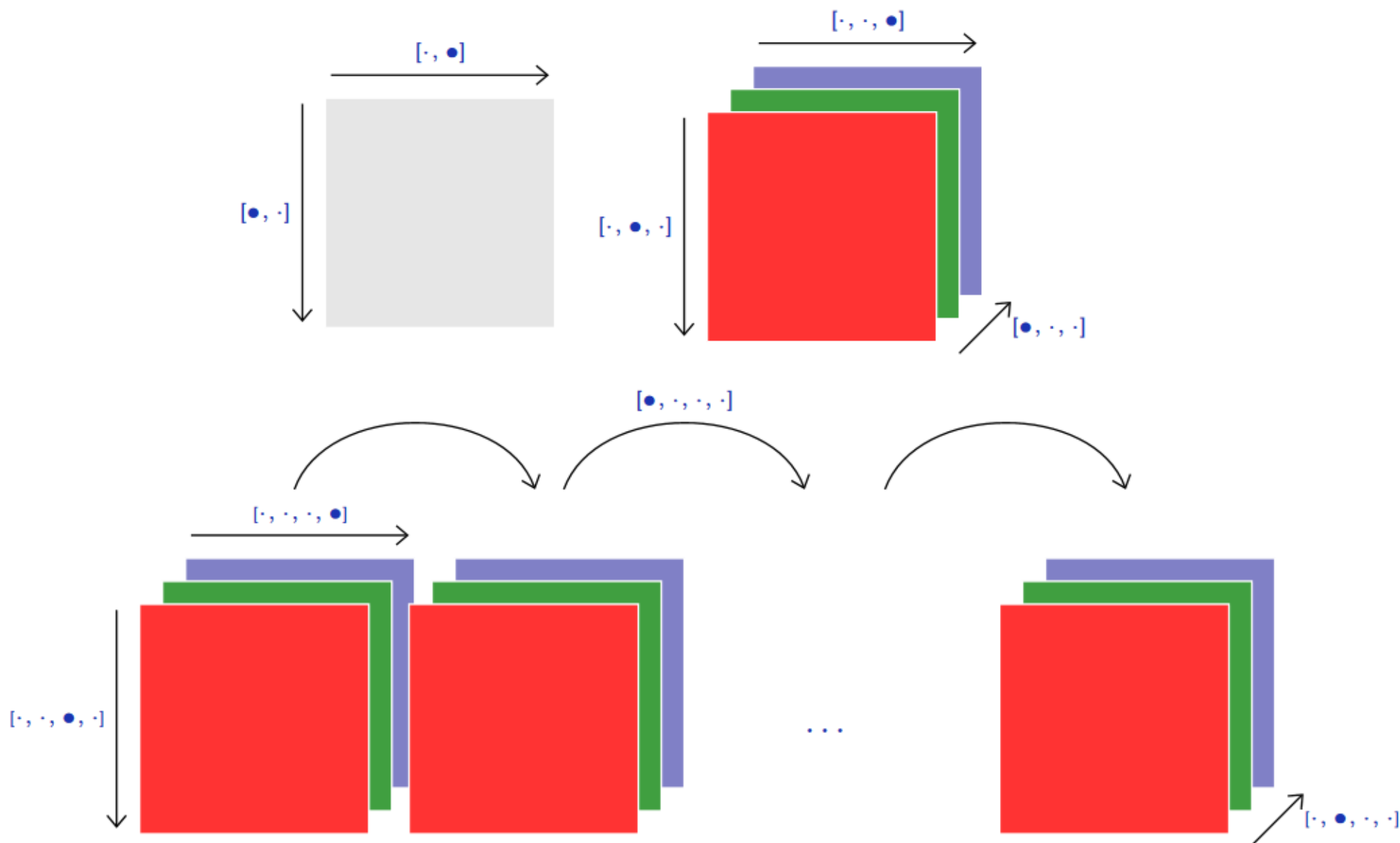
Свёрточная НС: каждая свёртка – 1 «лист»

Свёрточная НС: тензор \rightarrow тензор

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

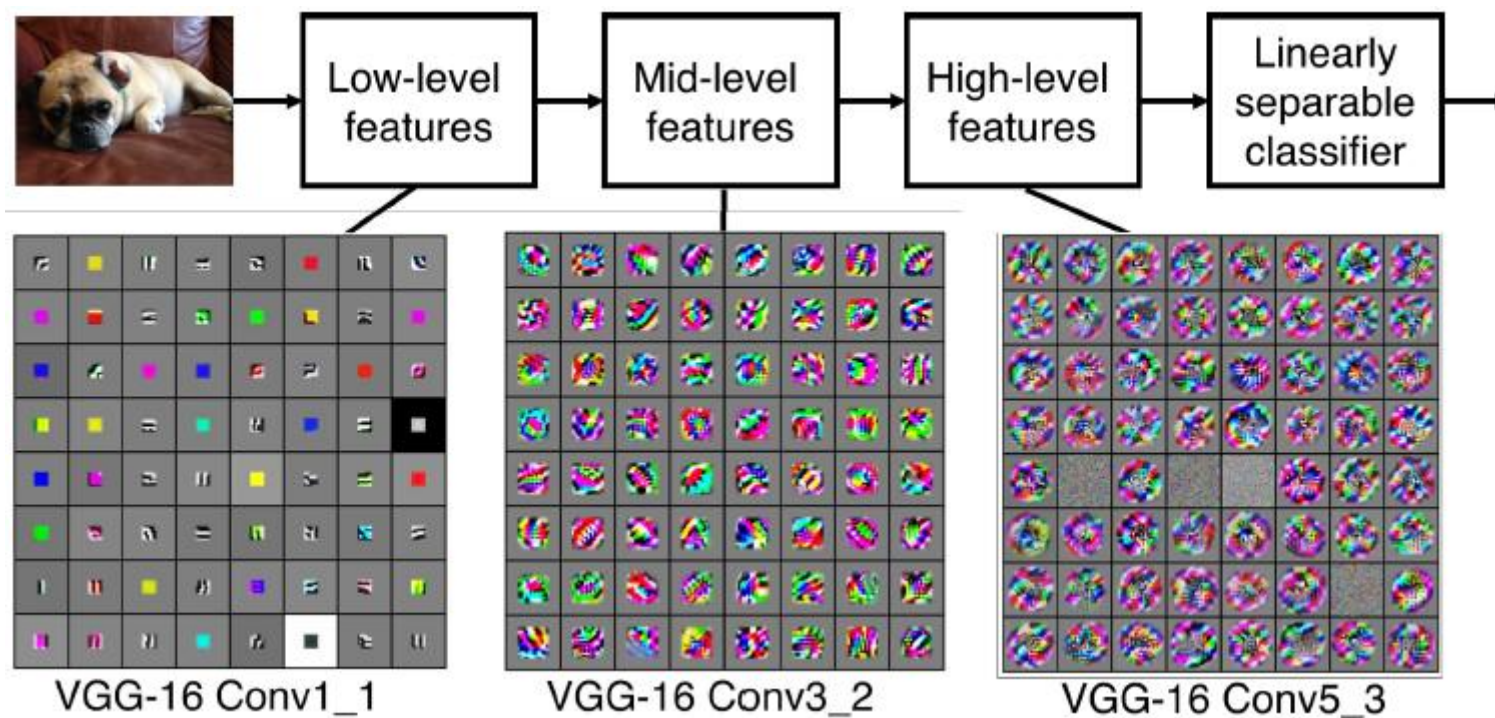
Важно:
Свёрточный слой:
тензор \rightarrow тензор (м.б. другой размерности)

Тензор – у нас «многомерная матрица»

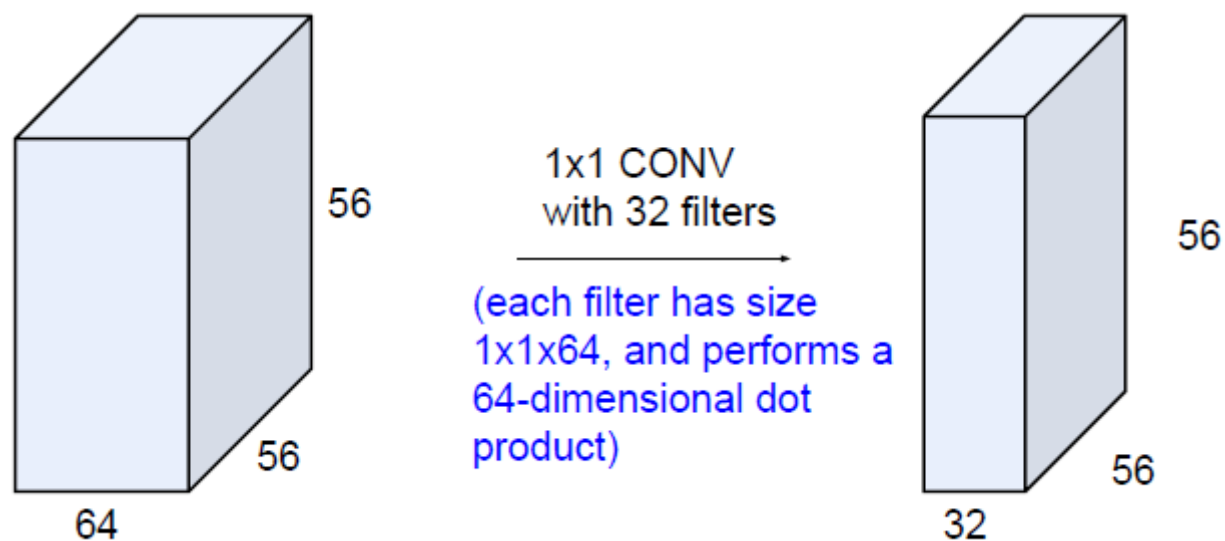


<https://fleuret.org/ee559/ee559-slides-1-5-high-dimension-tensors.pdf>

Визуализация признаков

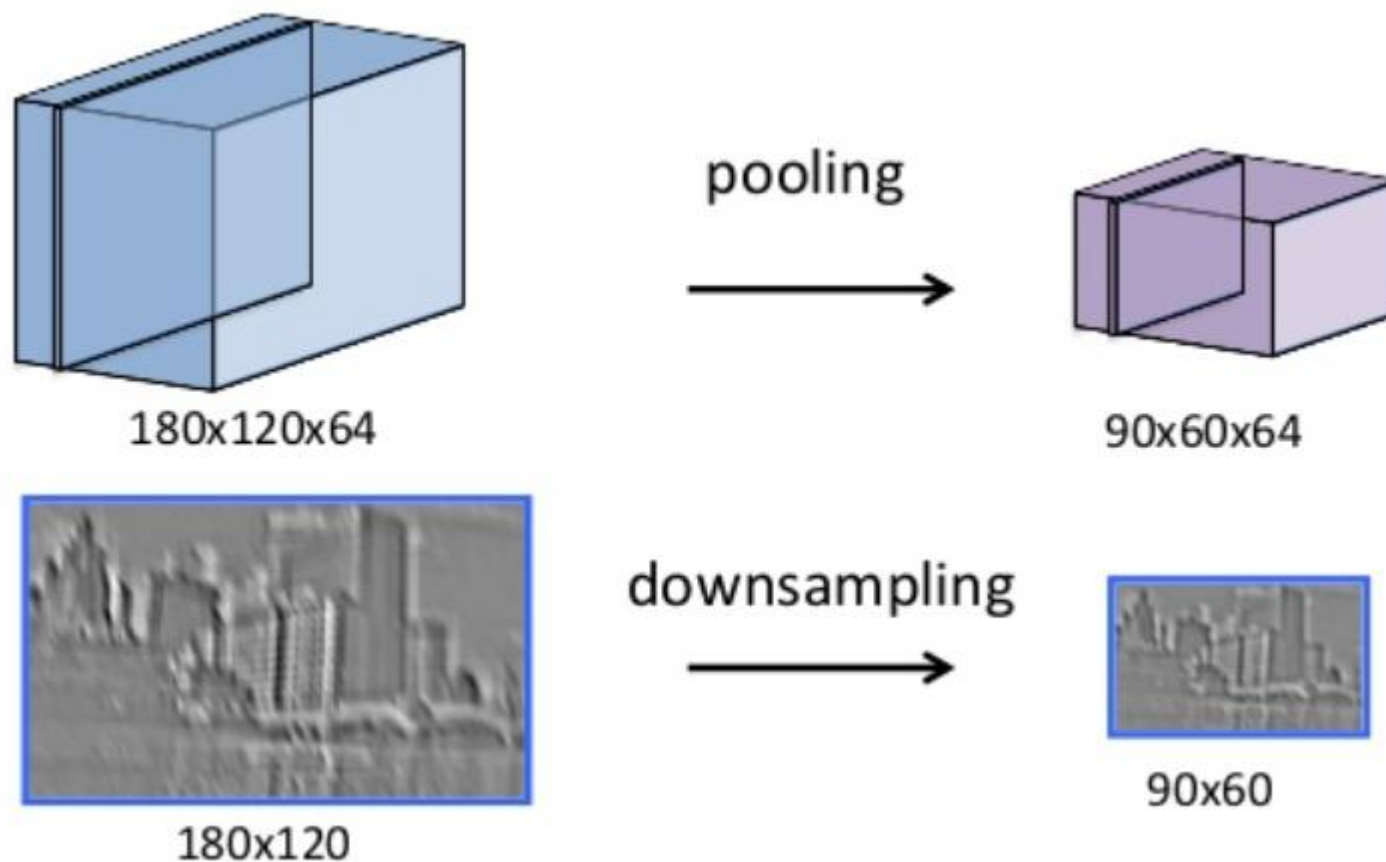


Смысл свёрток 1×1



Преобразование признаков!

Смысл пулинга

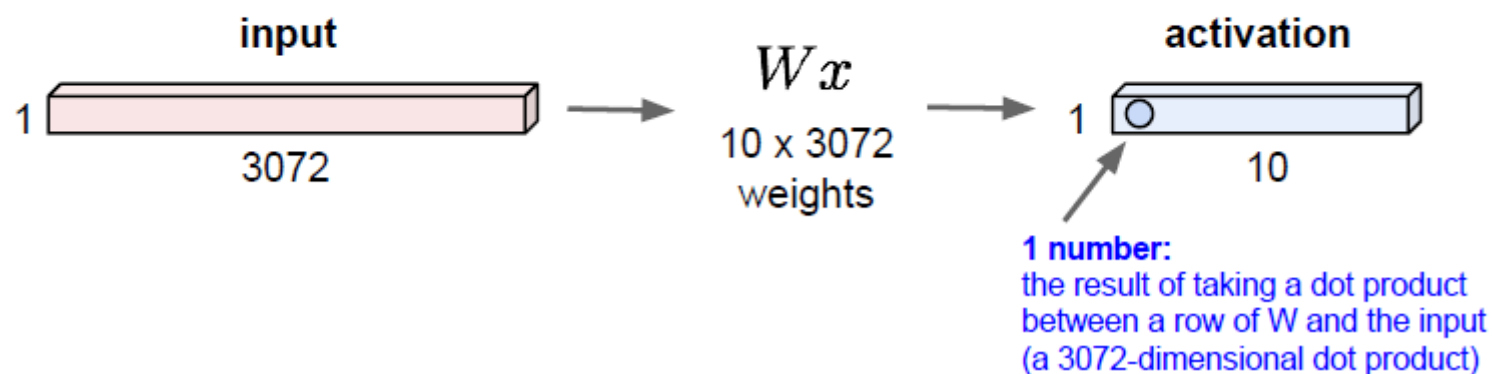


- уменьшение размеров
- на каждой карте действует независимо

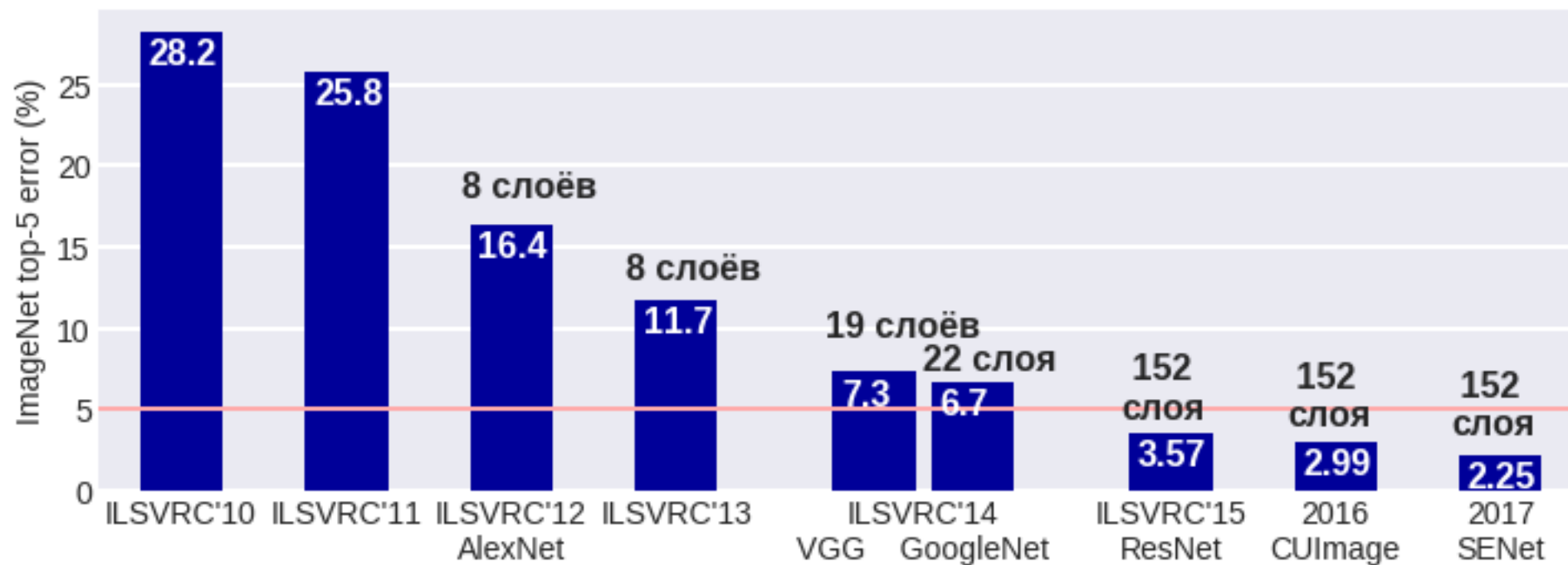
Полносвязный слой

32x32x3 image -> stretch to 3072 x 1

Each neuron
looks at the full
input volume

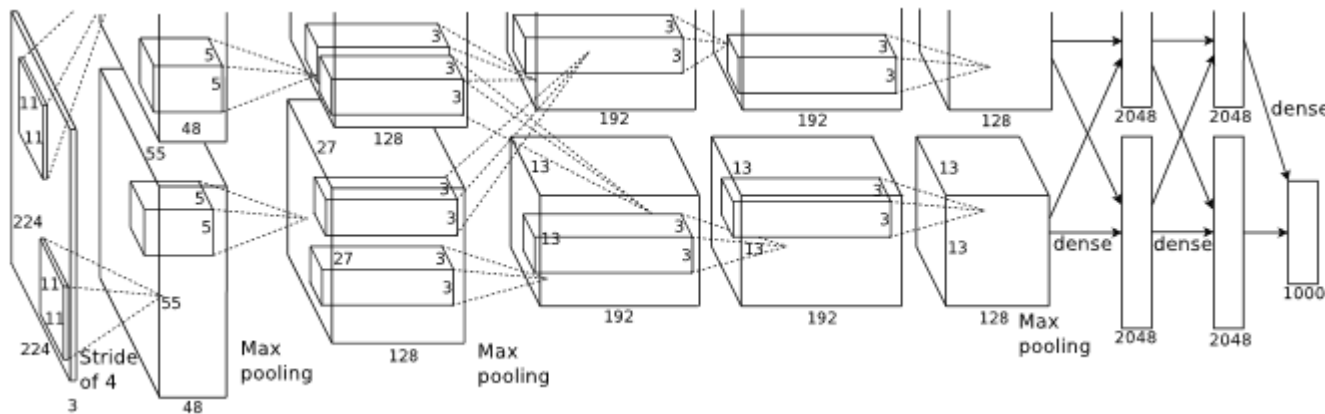


Революция в машинном обучении

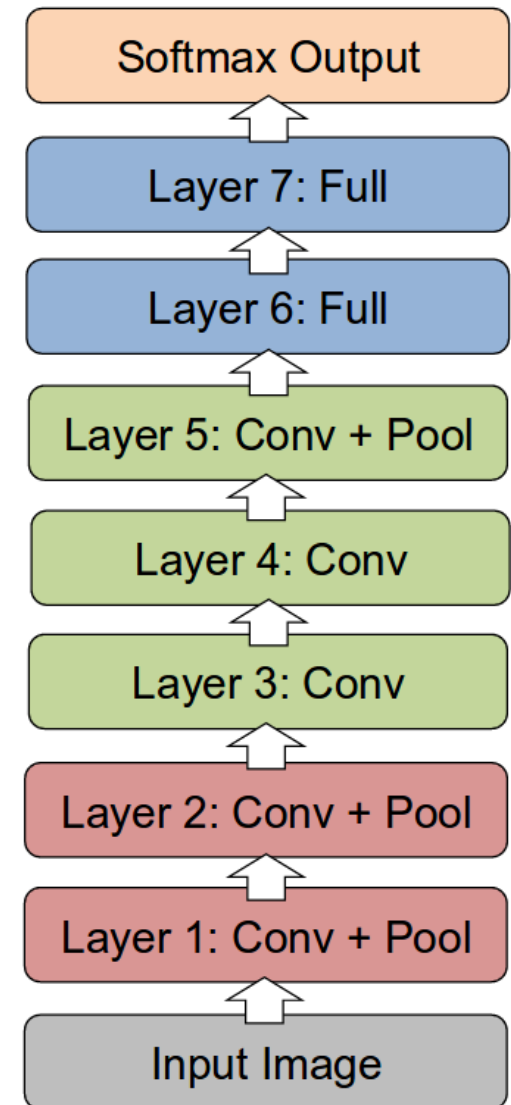


ошибка человека – 5.1

AlexNet (2012)



- ReLu
- Max-pooling
- Полно-связные слои
- Data augmentation
 - Dropout 0.5
- Batch size = 128
- SGD Momentum 0.9
- 60M параметров
- 650K нейронов
- 1 неделя на 2 GPU
- 7 скрытых слоёв



AlexNet (2012)

свёртки 3×3, 5×5, 11×11

7 CNN ансамбль: 18.2% → 15.4%

Интересно

**Убрать 16М параметров (последний полносвязный слой) –
качество 1.1% ↓**

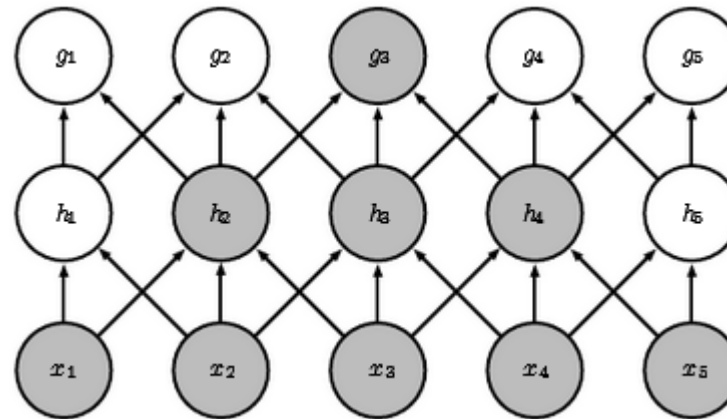
**Убрать 50М параметров (2 последних полносвязных слоя) –
качество 5.7% ↓**

Убрали 1М параметр (3 и 4 слоя) – качество 3.0% ↓

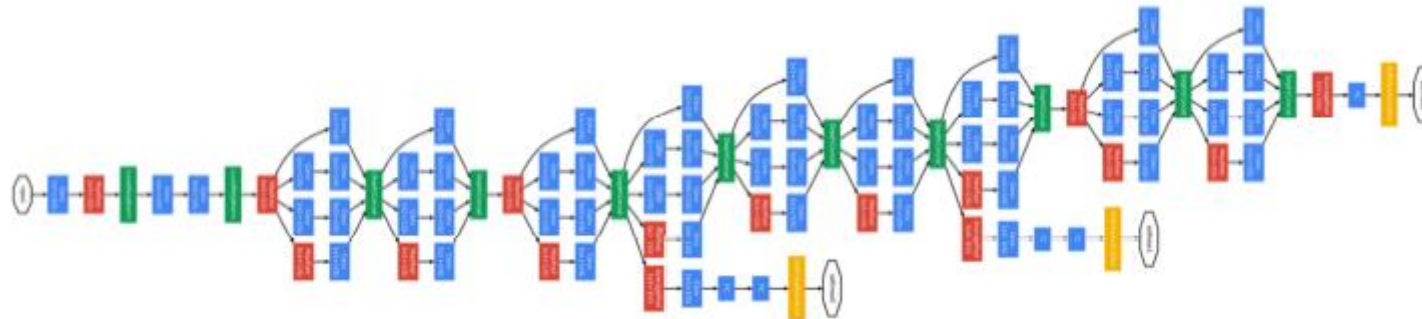
Убрать несколько слоёв (3, 4, 6, 7) – качество 33.5% ↓

-
- The diagram illustrates the VGG16 and VGG19 architectures. VGG16 consists of 16 layers: an input layer, three stages of convolutional layers (3x3 conv) and pooling layers, and a final Softmax layer. VGG19 consists of 19 layers: an input layer, three stages of convolutional layers (3x3 conv) and pooling layers, and a final Softmax layer. The layers are color-coded: red for Softmax, green for FC, blue for Pool, orange for 3x3 conv, and grey for Input.
- | VGG16 | VGG19 |
|---------------|---------------|
| Softmax | Softmax |
| FC 1000 | FC 1000 |
| FC 4096 | FC 4096 |
| FC 4096 | FC 4096 |
| Pool | Pool |
| 3x3 conv, 512 | 3x3 conv, 512 |
| 3x3 conv, 512 | 3x3 conv, 512 |
| 3x3 conv, 512 | 3x3 conv, 512 |
| Pool | Pool |
| 3x3 conv, 512 | 3x3 conv, 512 |
| 3x3 conv, 512 | 3x3 conv, 512 |
| 3x3 conv, 512 | 3x3 conv, 512 |
| Pool | Pool |
| 3x3 conv, 256 | 3x3 conv, 256 |
| 3x3 conv, 256 | 3x3 conv, 256 |
| Pool | Pool |
| 3x3 conv, 128 | 3x3 conv, 128 |
| 3x3 conv, 128 | 3x3 conv, 128 |
| Pool | Pool |
| 3x3 conv, 64 | 3x3 conv, 64 |
| 3x3 conv, 64 | 3x3 conv, 64 |
| Input | Input |
- VGG16
- VGG19

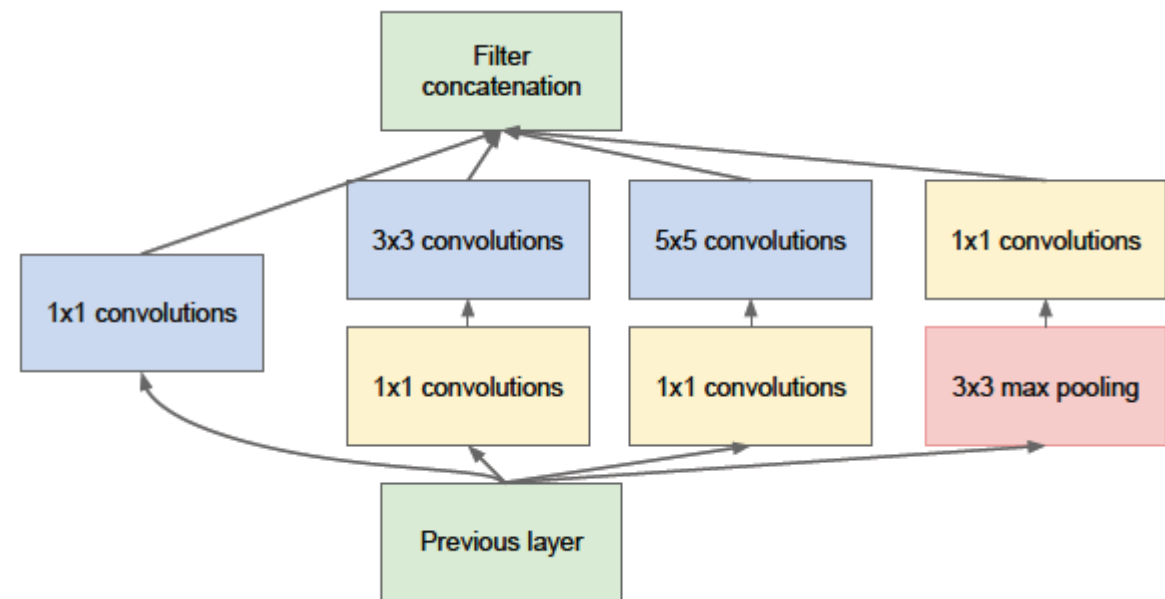
Идея каскада свёрток



GoogLeNet (2014)



- «конструктор» НС
 - 22 слоя – нет полносвязных
- Модуль «Inception»
- 5M параметров (меньше!)
- дополнительные выходы классификации (для протекания градиента)

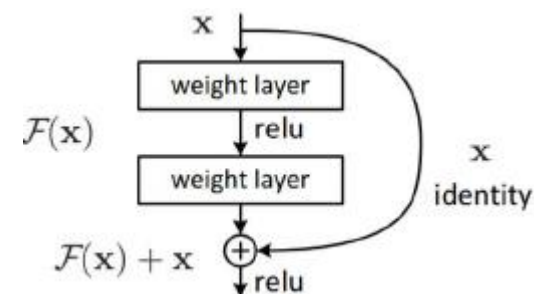


1×1-свёртки существенно уменьшают число параметров!
... а идея была (синие блоки) – разные свёртки + пулинг

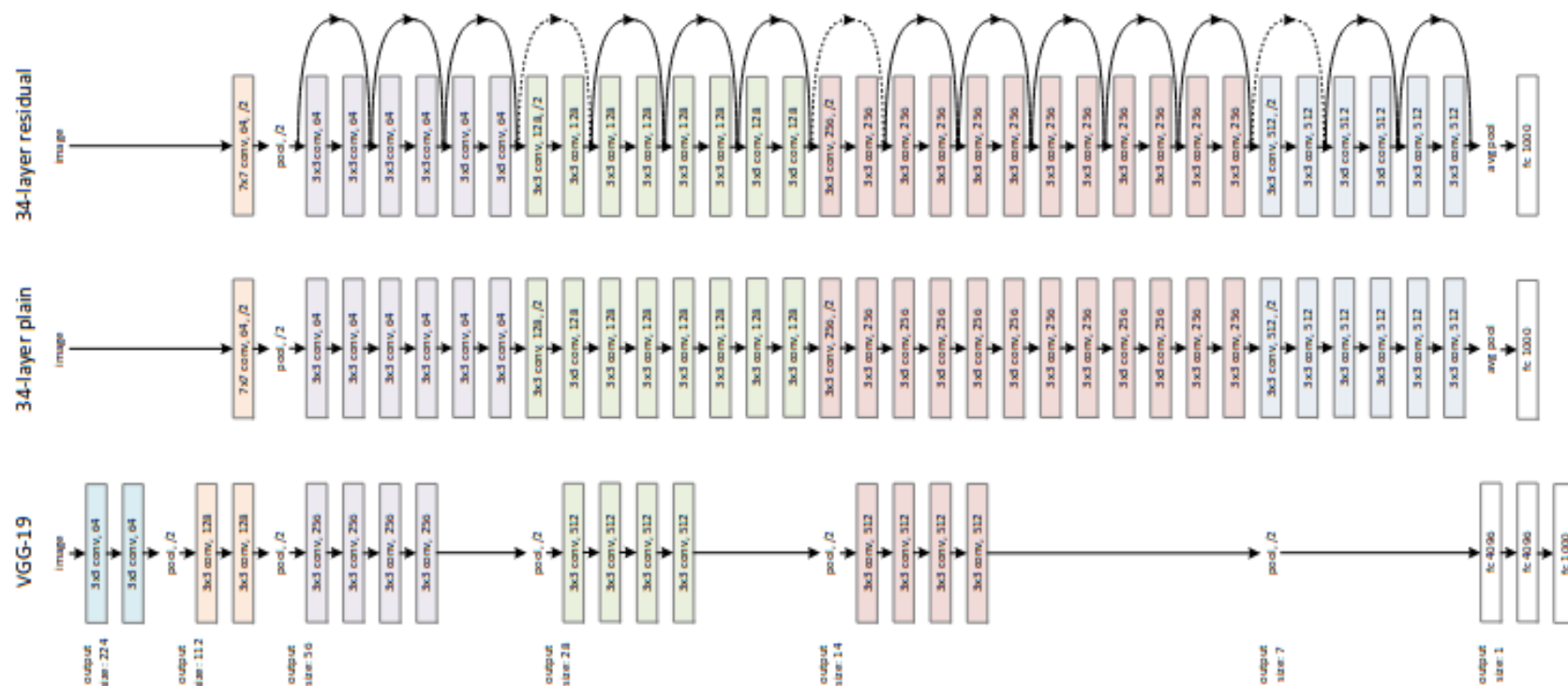
ResNet = Residual Network (2014)

$$y = f(x) + x$$

$$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial y} f'(x) + \frac{\partial L}{\partial y}$$



He, Zhang, Ren, Sun, CVPR 2016

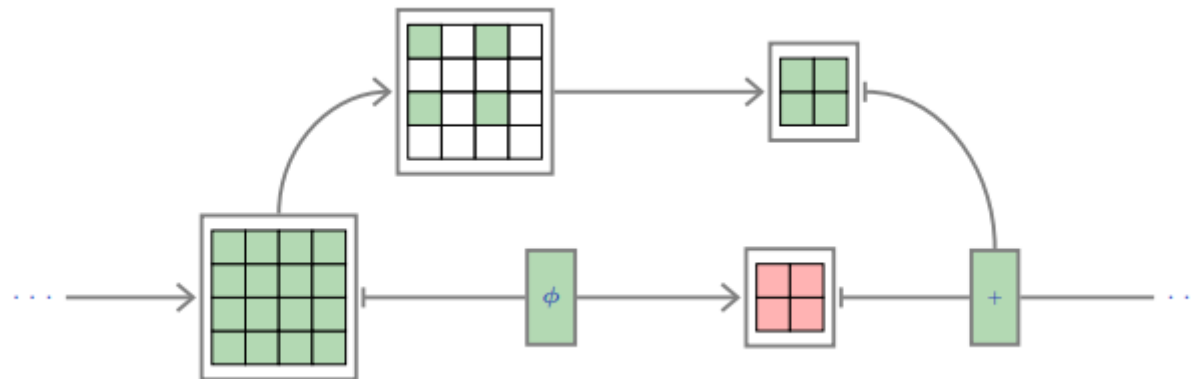


ResNet = Residual Network (2014)

- **152 слоя**
 - **связи проходят через слои**
- **Batch Normalization после каждого CONV-layer**
 - **Умные инициализации весов**
 - **SGD + Momentum (0.9)**
 - **Mini-batch size = 256**
 - **Нет Dropout!**

Просто добавление слоёв не помогает!
Добавлять надо по-умному...

Проблемы с прокидыванием в свёрточных слоях



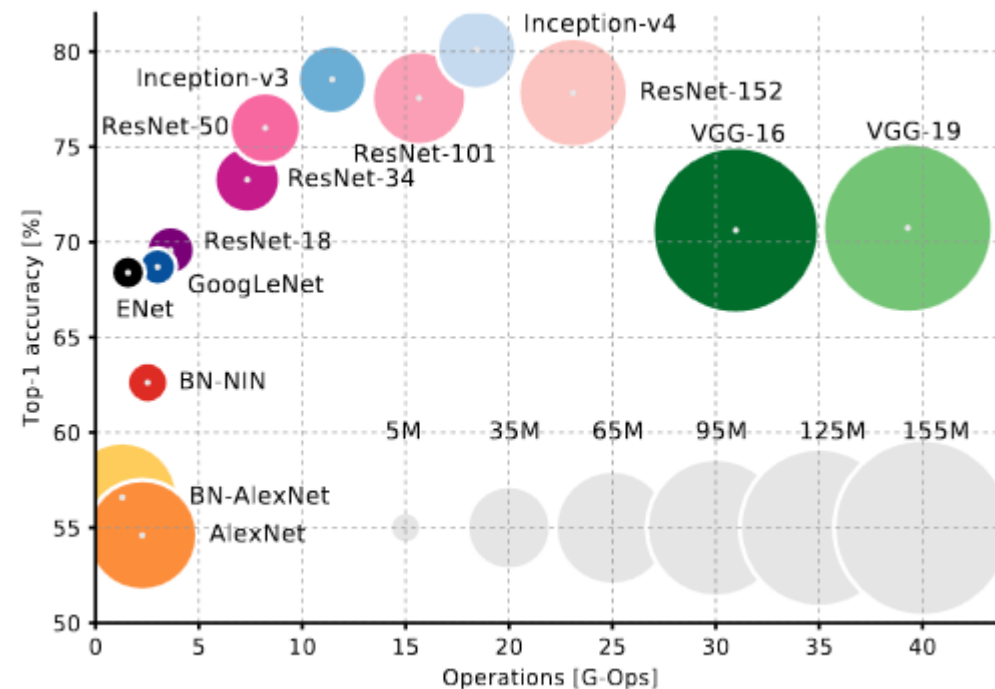
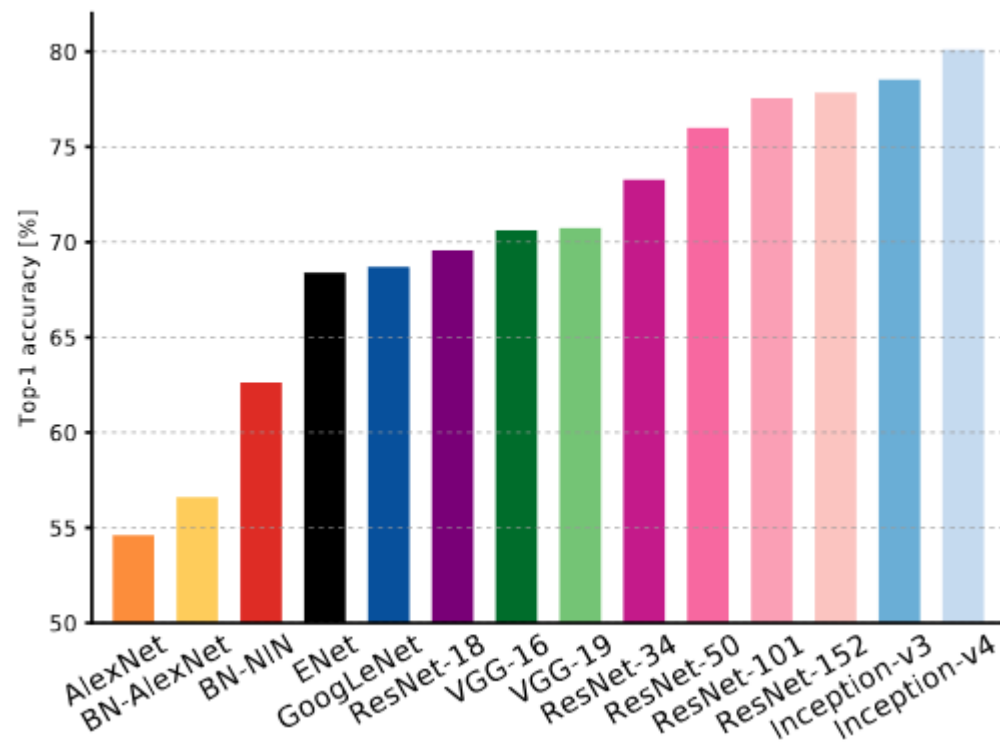
1) размеры уменьшаются, поэтому не совсем прямая связь

2) изменяется число каналов!

добавить нули
есть ещё способ... ;)

<https://fleuret.org/ee559/>

Inception-v4: Resnet + Inception!

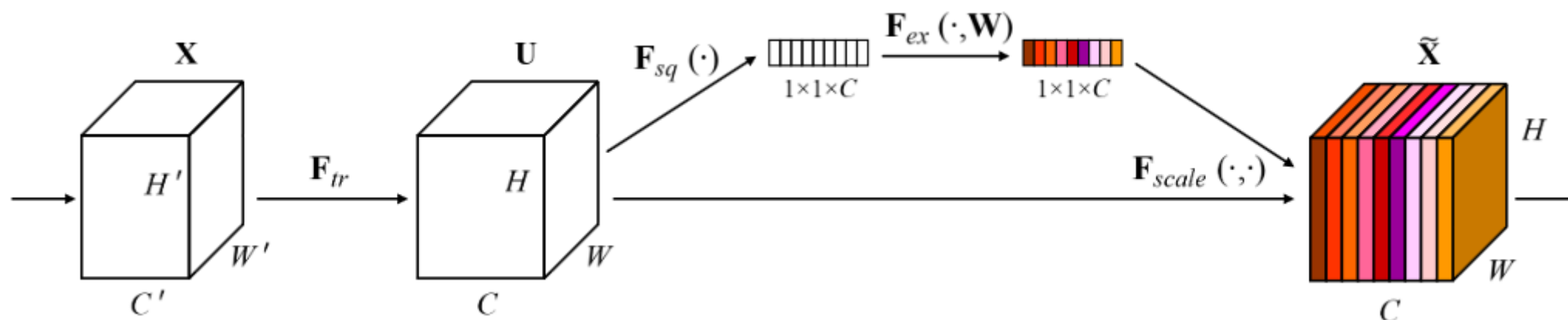


A. Canziani, A. Paszke, E. Culurciello, « An Analysis of Deep Neural Network Models for Practical Applications», 2017 <https://arxiv.org/pdf/1605.07678.pdf>

SENet (Squeeze-and-Excitation Network) 2017

Раньше: трансформация $F_{tr} : X_{H' \times W' \times C'} \rightarrow U_{H \times W \times C}$
(например, свёртка)

Теперь: «Squeeze-and-Excitation» (SE) block $F_{tr} \oplus \dots$



сжатие (squeeze) – агрегация по каналам

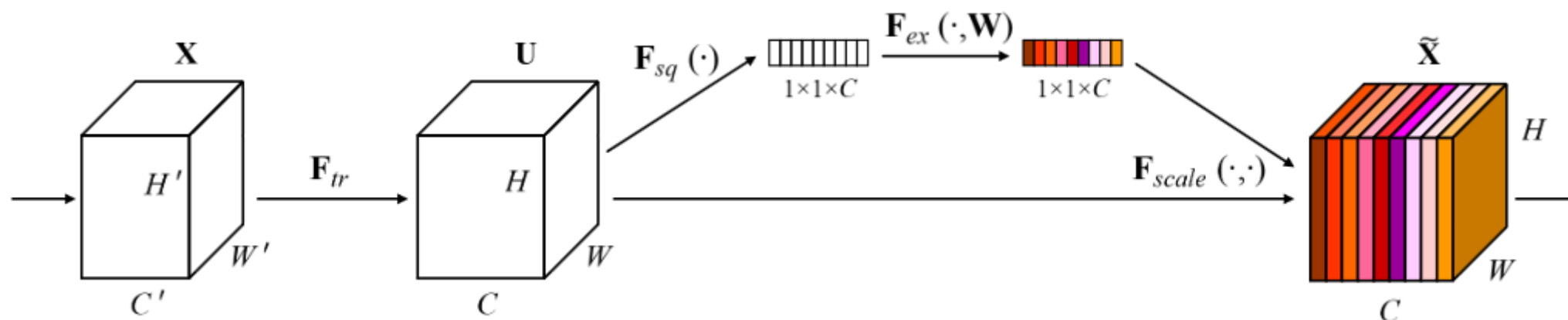
$$F_{sq} : \| u_{h,w,c} \|_{H \times W \times C} \rightarrow \left\| \frac{1}{HW} \sum_{w=1}^W \sum_{h=1}^H u_{h,w,c} \right\|_C$$

Ж. Ну и др. «Squeeze-and-Excitation Networks», 2018 <https://arxiv.org/pdf/1709.01507.pdf>

SENet (Squeeze-and-Excitation Network) 2017

Раньше: трансформация $F_{tr} : X_{H' \times W' \times C'} \rightarrow U_{H \times W \times C}$
(например, свёртка)

Теперь: «Squeeze-and-Excitation» (SE) block $F_{tr} \oplus \dots$



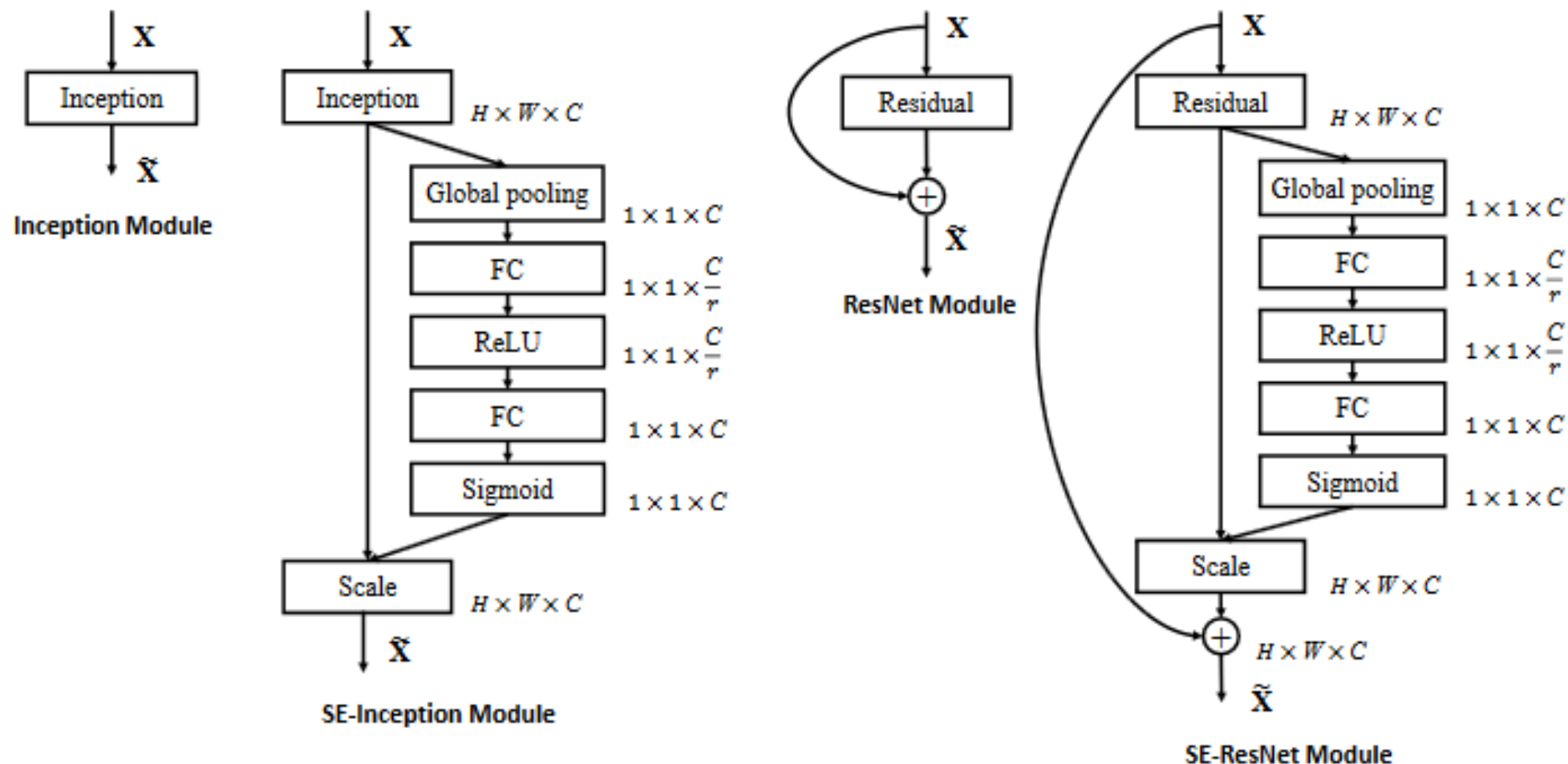
возбуждение (excitation) – адаптивная калибровка

$$F_{ex} = \sigma(W_{C \times k} \text{ReLU}(V_{k \times C} z_C))$$

$$F_{scale} : \|u_{h,w,c}\|_{H \times W \times C} \rightarrow \|u_{h,w,c} F_{ex}(z)_c\|_C$$

SENet (Squeeze-and-Excitation Network) 2017

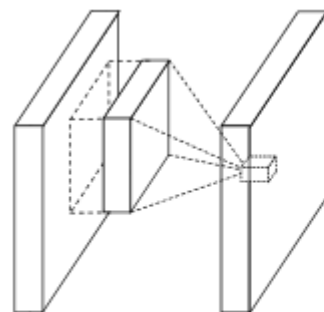
Можно переделать «старые сети»



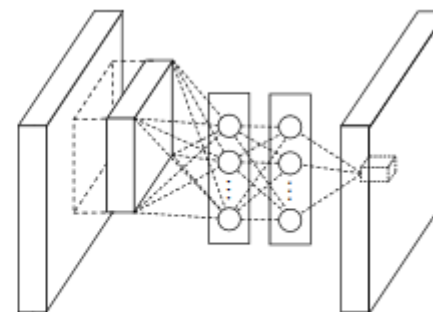
Динамическая перекалибровка признаков позволяет «увеличивать» важные признаки и «уменьшать» неважные

Какие архитектуры ещё надо знать...

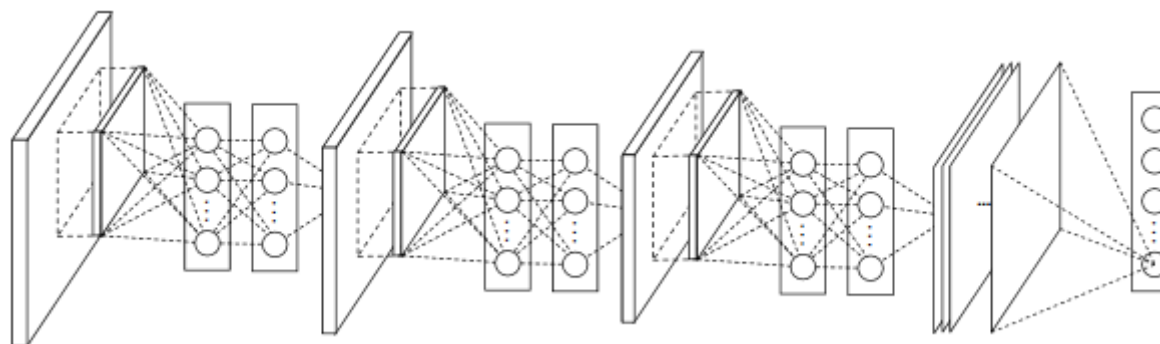
Network in Network (NiN)



(a) Linear convolution layer



(b) Mlpconv layer



полносвязность ~ свёртки 1×1 внутри свёртки
глобальный пулинг

Min Lin « Network in Network (NiN) » 2014, <https://arxiv.org/pdf/1312.4400.pdf>

Deep Networks with Stochastic Depth

- **Во время обучения: случайно удаляем подмножество слоёв**
(используем менее глубокую сеть во время обучения)
- **«Прокидывание» тождественной функции**

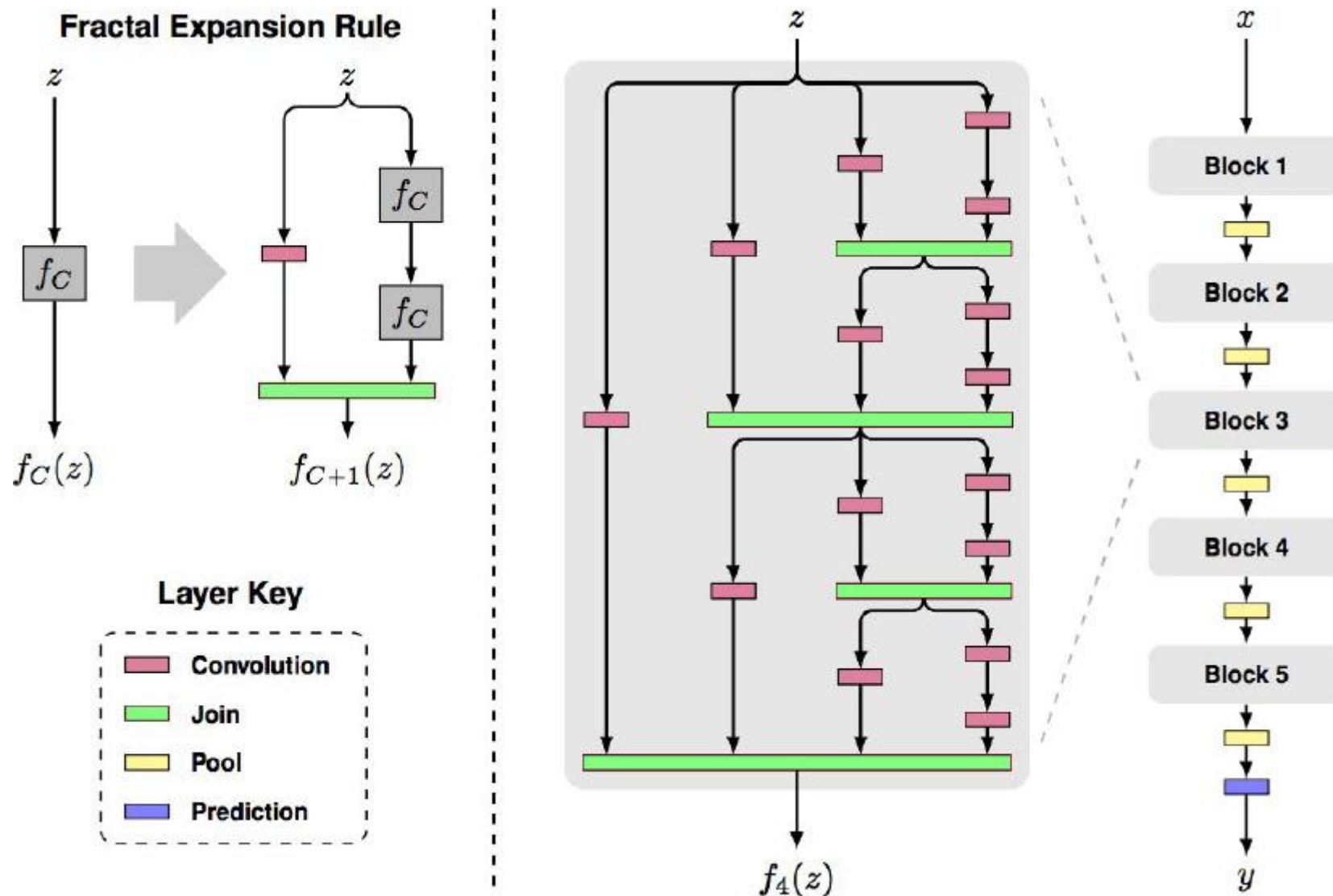
[Gao Huang, 2016 <https://arxiv.org/abs/1603.09382>]

FractalNet: Ultra-Deep Neural Networks without Residuals

Фрактальная архитектура с короткими и длинными связями
Обучение со случайным выбрасыванием связей

[Gustav Larsson 2017 <https://arxiv.org/abs/1605.07648>]

Фрактальные сети



Densely Connected Convolutional Networks (DenseNets)

Блоки в которых слой соединён с каждым последующим

[Gao Huang 2016 <https://arxiv.org/abs/1608.06993>]

Обычная сеть: $z_i = H_i(z_{i-1})$

где z_i выход i -го слоя.

ResNet: $z_i = H_i(z_{i-1}) + z_{i-1}$

DensNet: $z_i = H_i([z_{i-1}, z_{i-2}, \dots, z_0])$

H = BN + ReLU + convolution + dropout

число признаков линейно вырастает...

ResNeXt

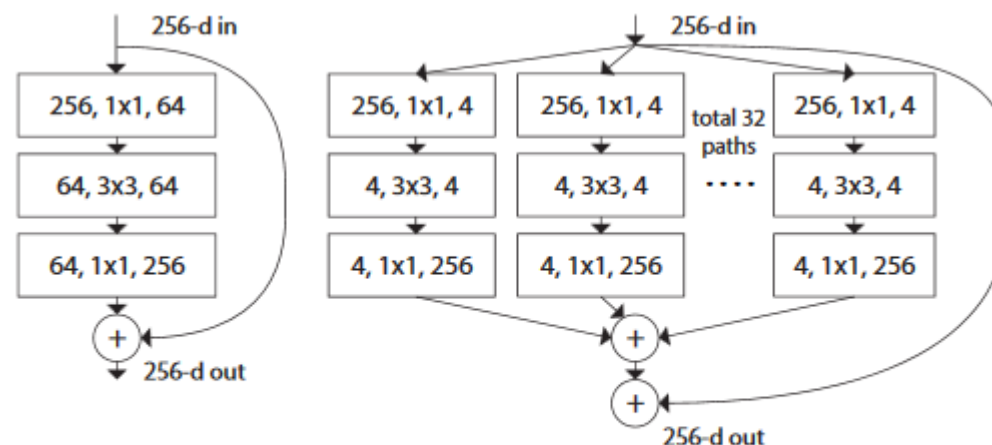


Figure 1. Left: A block of ResNet [14]. Right: A block of ResNeXt with cardinality = 32, with roughly the same complexity. A layer is shown as (# in channels, filter size, # out channels).

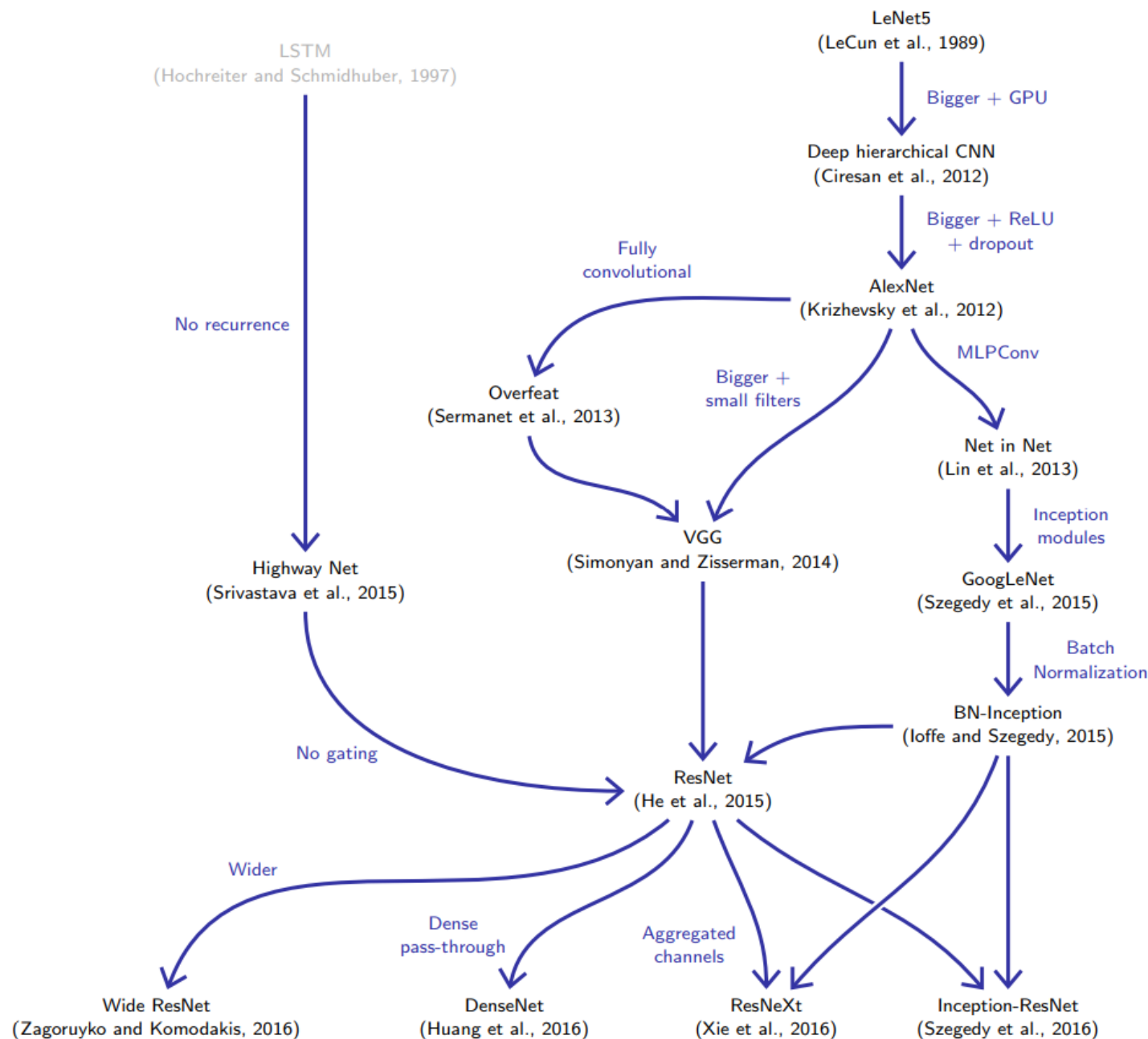
**такое же число параметров как в ResNet,
но разнести их по 32 разным путям
тут блок – conv + ИТ + ReLU
используется bottleneck!!!**

Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, Kaiming He
Aggregated Residual Transformations for Deep Neural Networks //
<https://arxiv.org/abs/1611.05431>

SqueezeNet – Сжатие AlexNet

сеть	подход	размер	отношение	Top1-точность	Top5-точность
AlexNet		240MB	1x	57.2%	80.3%
AlexNet	SVD	48MB	2x	56.0%	79.4%
AlexNet	Deep Compression	7MB	35x	57.2%	80.3%
SqueezeNet		5MB	50x	57.5%	80.3%
SqueezeNet	Deep Compression	0.5MB	510x	57.5%	80.3%

Forrest N. Iandola « SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size»2017 <https://arxiv.org/abs/1602.07360>



<https://fleuret.org/ee559/ee559-slides-7-2-image-classification.pdf>

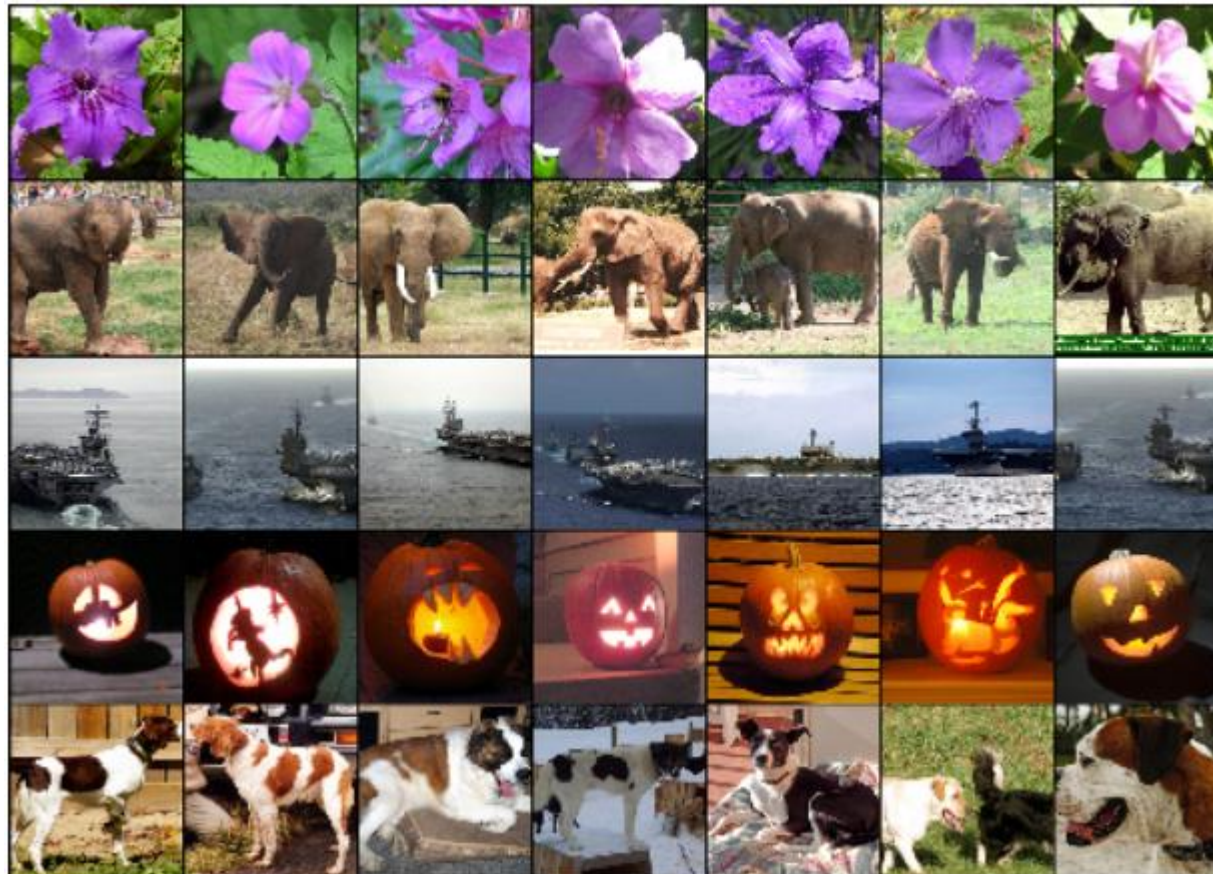
За чем можно наблюдать в NN

- **параметры: фильтры как картинки**
- **внутренние активации – как картинки**
- **распределения активаций (на отдельных объектах)**
- **производные по входу**
- **входы, максимизирующие какой-то ответ**
- **«adversarial samples»**

Визуализация – как понять, чему сеть учится...

Последний полносвязный слой

Можно смотреть соседей в этом признаковом пространстве



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

Визуализация – как понять, чему сеть учится...

Последний полносвязный слой

Применить уменьшение размерности... t-SNE в R^2



<https://cs.stanford.edu/people/karpathy/cnnembed/>

Визуализация – как понять, чему сеть учится...

Средние слои

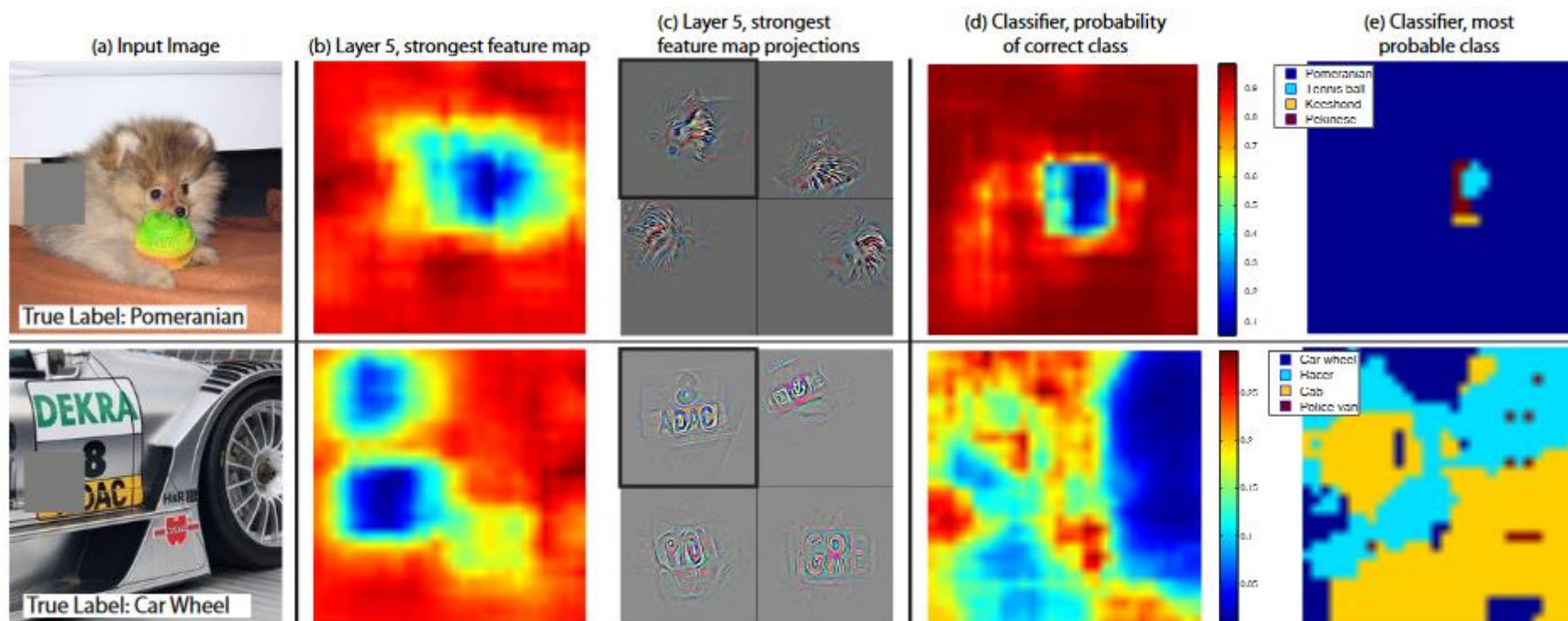
На каких изображениях максимальные значения активаций



<https://arxiv.org/pdf/1412.6806.pdf>

Визуализация – как понять, чему сеть учится...

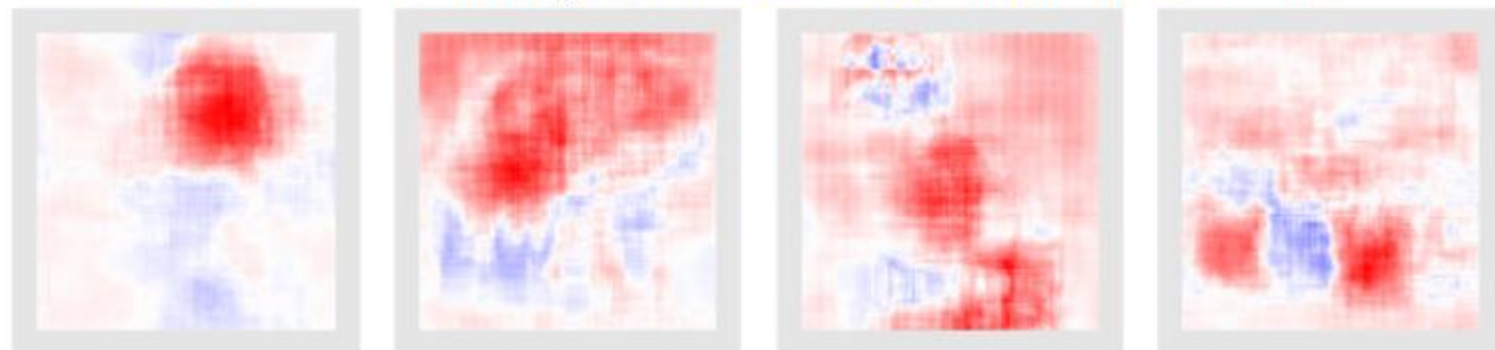
Какие пиксели отвечают за класс – «Occlusion sensitivity»



Закрывать последовательно часть изображения – 2D-гистограмма вероятности принадлежности к заданному классу при закрытии с центром в ij -м пикселе Matthew D Zeiler, Rob Fergus «Visualizing and Understanding Convolutional Networks» <https://arxiv.org/pdf/1311.2901.pdf>

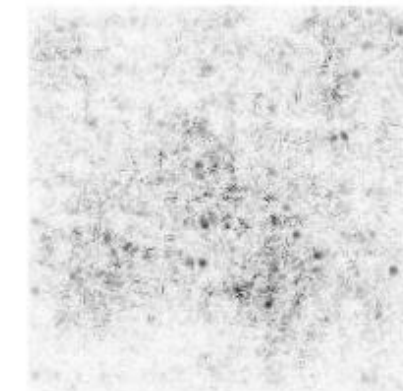
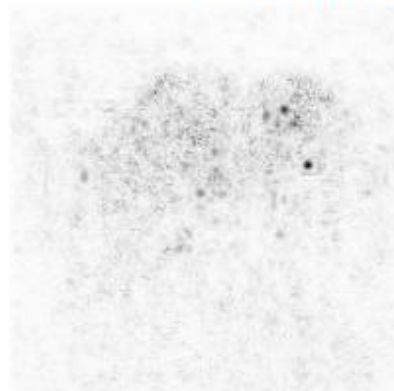
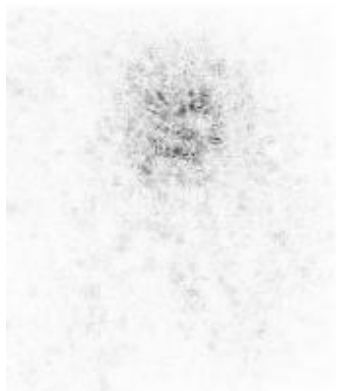
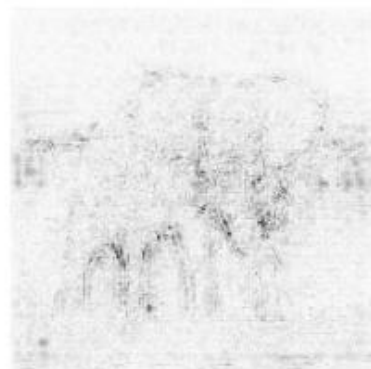
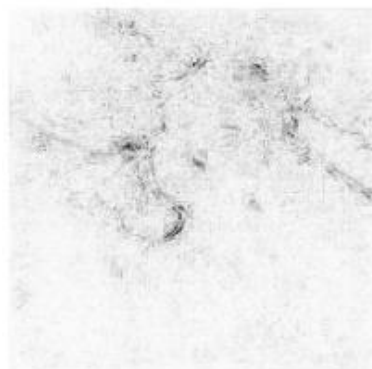
Визуализация – как понять, чему сеть учится...

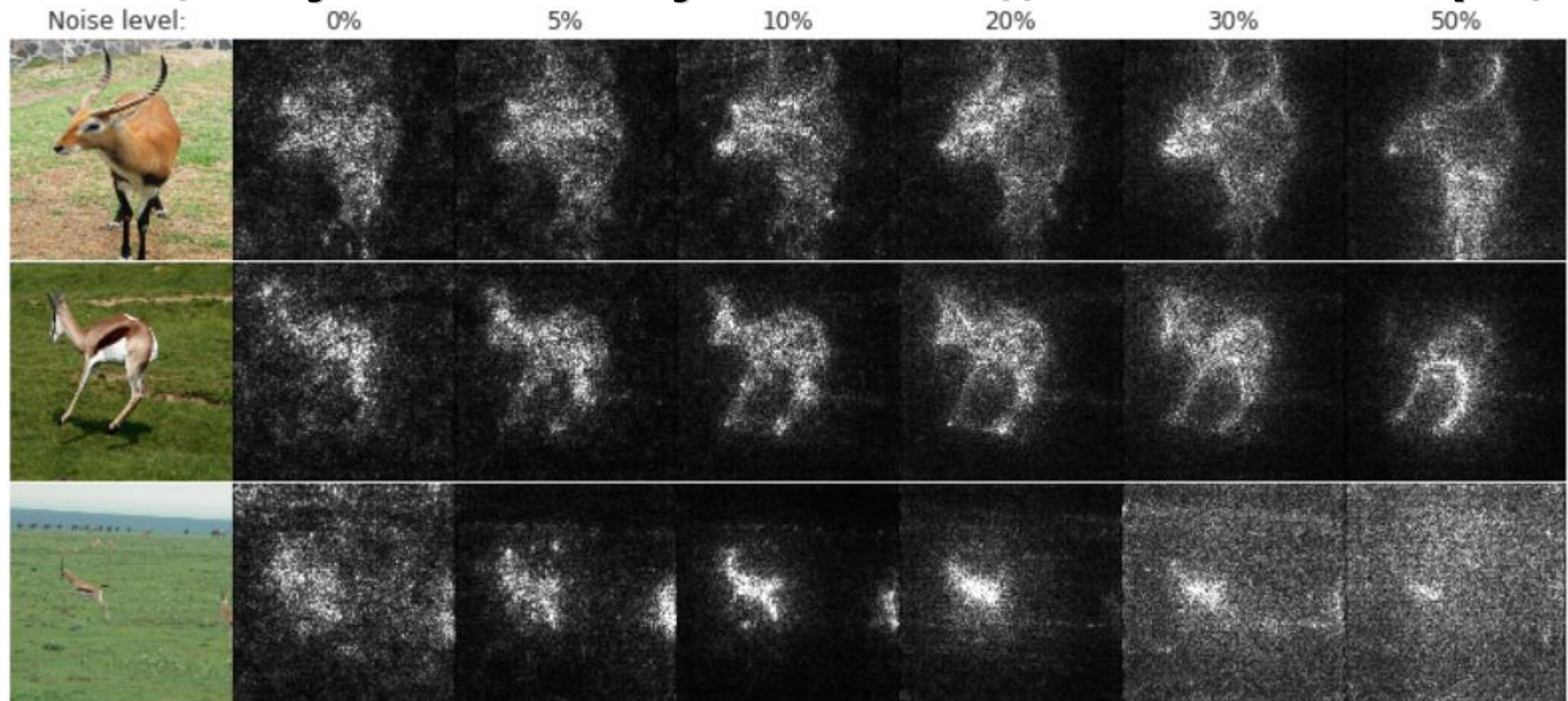
Original images

Occlusion mask 32×32 Occlusion sensitivity, mask 32×32 , stride of 2, AlexNet

«Saliency maps» – градиенты по входу

Gradient, AlexNet

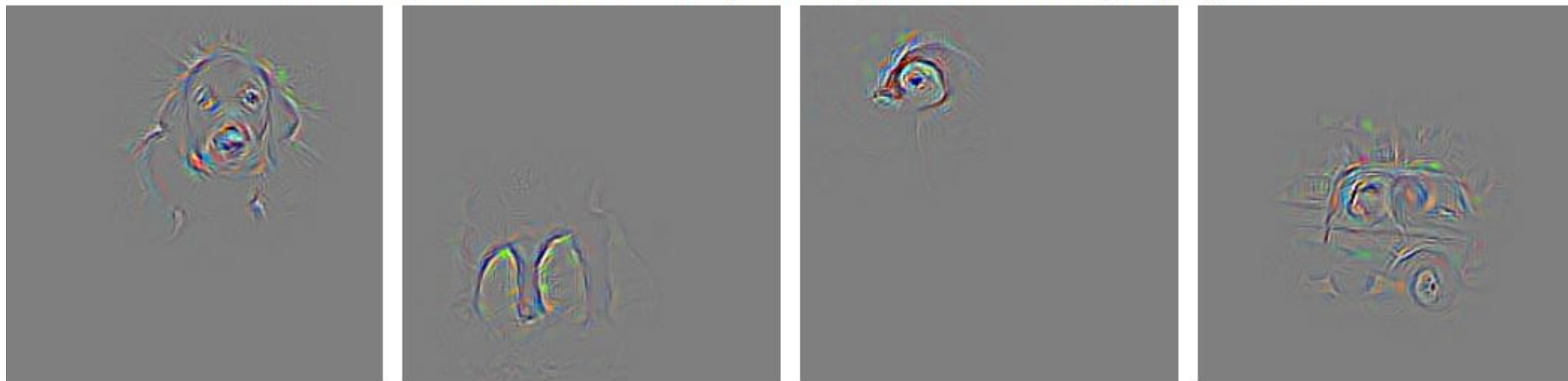
SmoothGrad, AlexNet, $\sigma = \frac{\Delta}{4}$ 

«Saliency maps» – градиенты по входу**с помощью шума можно получать более адекватные иллюстрации**

Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda Viégas, Martin Wattenberg «SmoothGrad: removing noise by adding noise» // <https://arxiv.org/abs/1706.03825>

«guided back-propagation» ~ градиенты по входу

AlexNet, max feature response, guided back-propagation

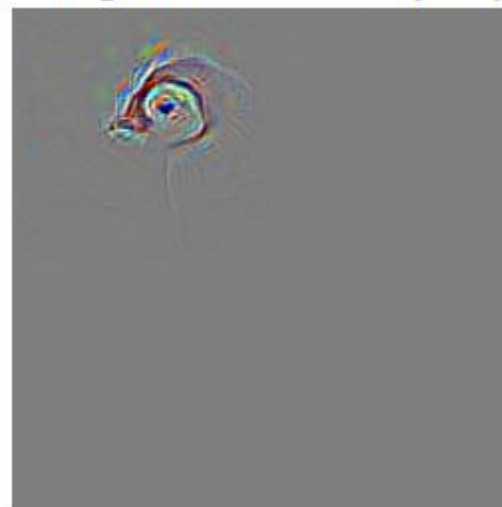


Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, Martin Riedmiller «Striving for Simplicity: The All Convolutional Net» // <https://arxiv.org/abs/1412.6806>

Original images



AlexNet, max feature response, guided back-propagation



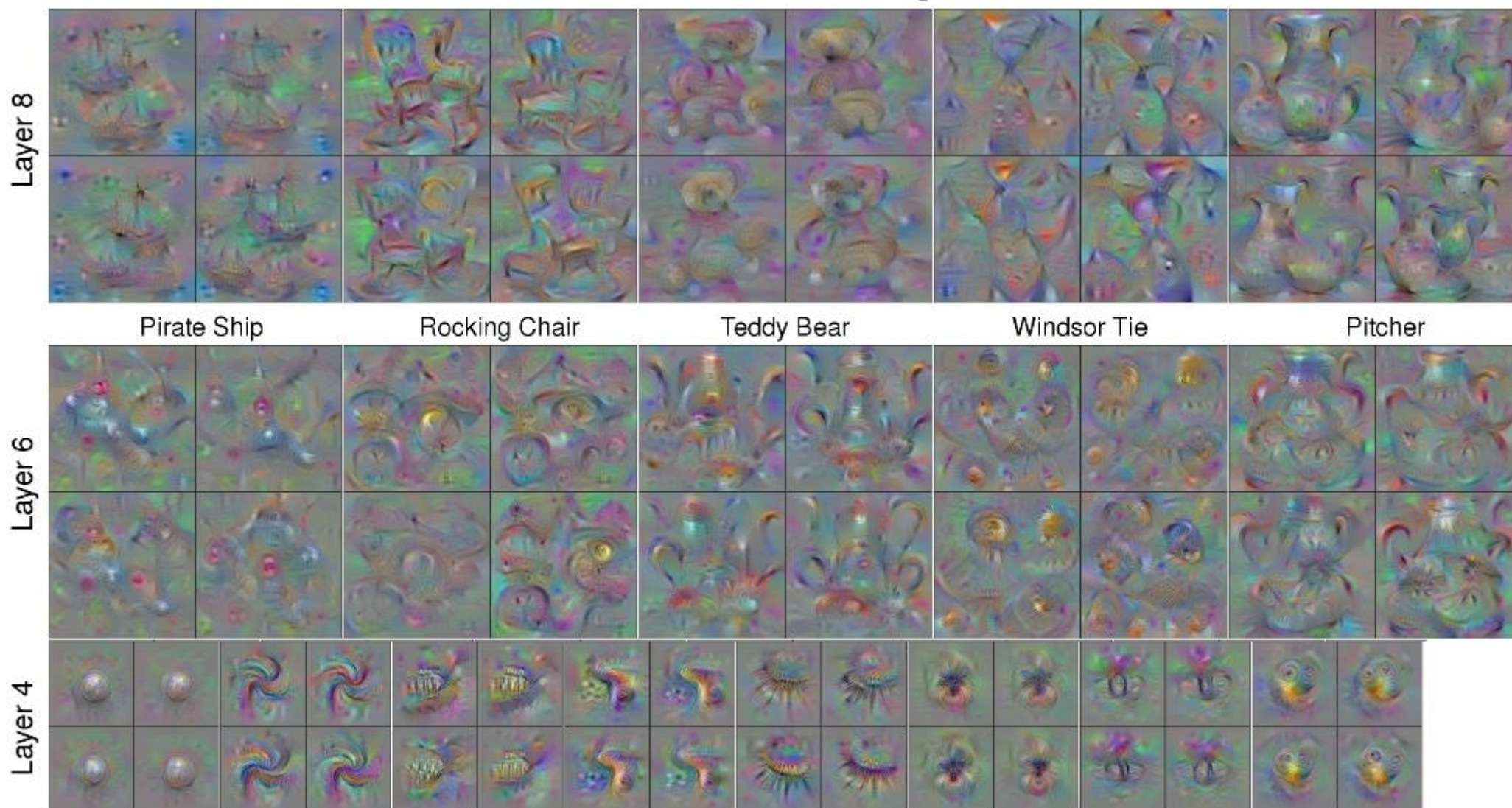
Визуализация – как понять, чему сеть учится...

Отдельные нейроны

**Сгенерировать изображения,
которые максимизируют активацию выделенного нейрона
(методом обратного распространения ошибки)**

Визуализация – как понять, чему сеть учится...

Отдельные нейроны



Jason Yosinski, Jeff Clune, Anh Nguyen, Thomas Fuchs, Hod Lipson «Understanding Neural Networks Through Deep Visualization» // <https://arxiv.org/pdf/1506.06579.pdf>

Что потом обсудим

- **Решение задач компьютерного зрения с помощью DL**
 - **Обманные изображения (Fooling Images)**