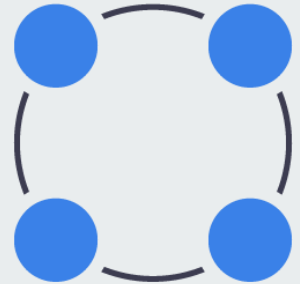




# Машинное обучение

Лекция 9. Сверточные нейронные сети. Начало

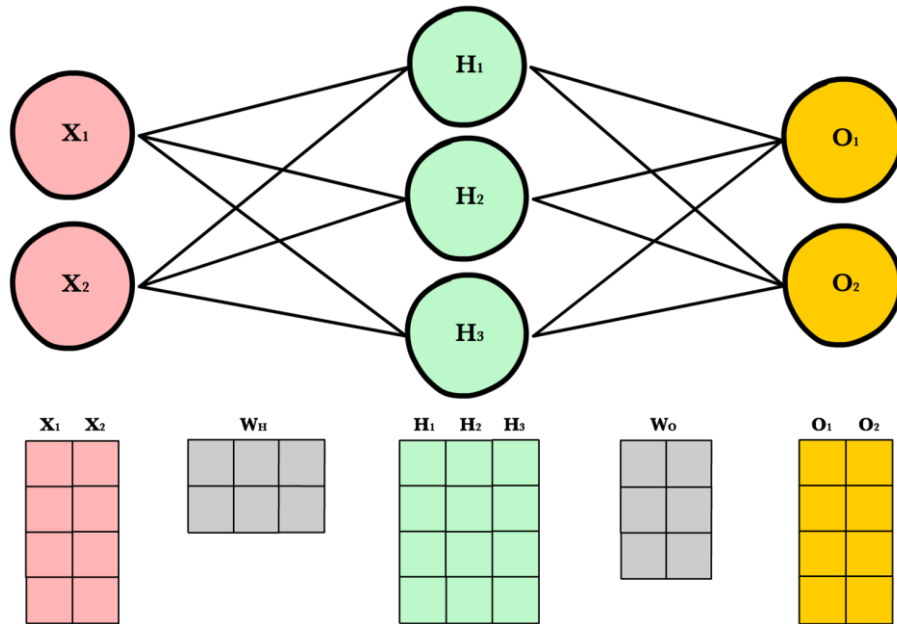
(09.04.2022)





Начало.

# Полносвязные нейронные сети (FC, MLP)





14,197,122 images, 21841 synsets indexed

[Explore](#) [Download](#) [Challenges](#) [Publications](#) [Updates](#) [About](#)

Not logged in. [Login](#) | [Signup](#)

База данных изображений, поделенных на 1000 классов



# ImageNet Timeline



До 2012 года в обработке картинок использовались НЕ нейронные подходы.

Подходы, основанные на нейросетях, существовали, но считались неспособными превзойти классические алгоритмы.

# HOG (Histogram of Oriented Gradients)



121	10	78	96	125
48	152	68	125	111
145	78	85	89	65
154	214	56	200	66
214	87	45	102	45



Градиенты для выделенного пикселя:

по OY:  $68 - 56 = 8$

по OX:  $89 - 78 = 11$

# HOG (Histogram of Oriented Gradients)



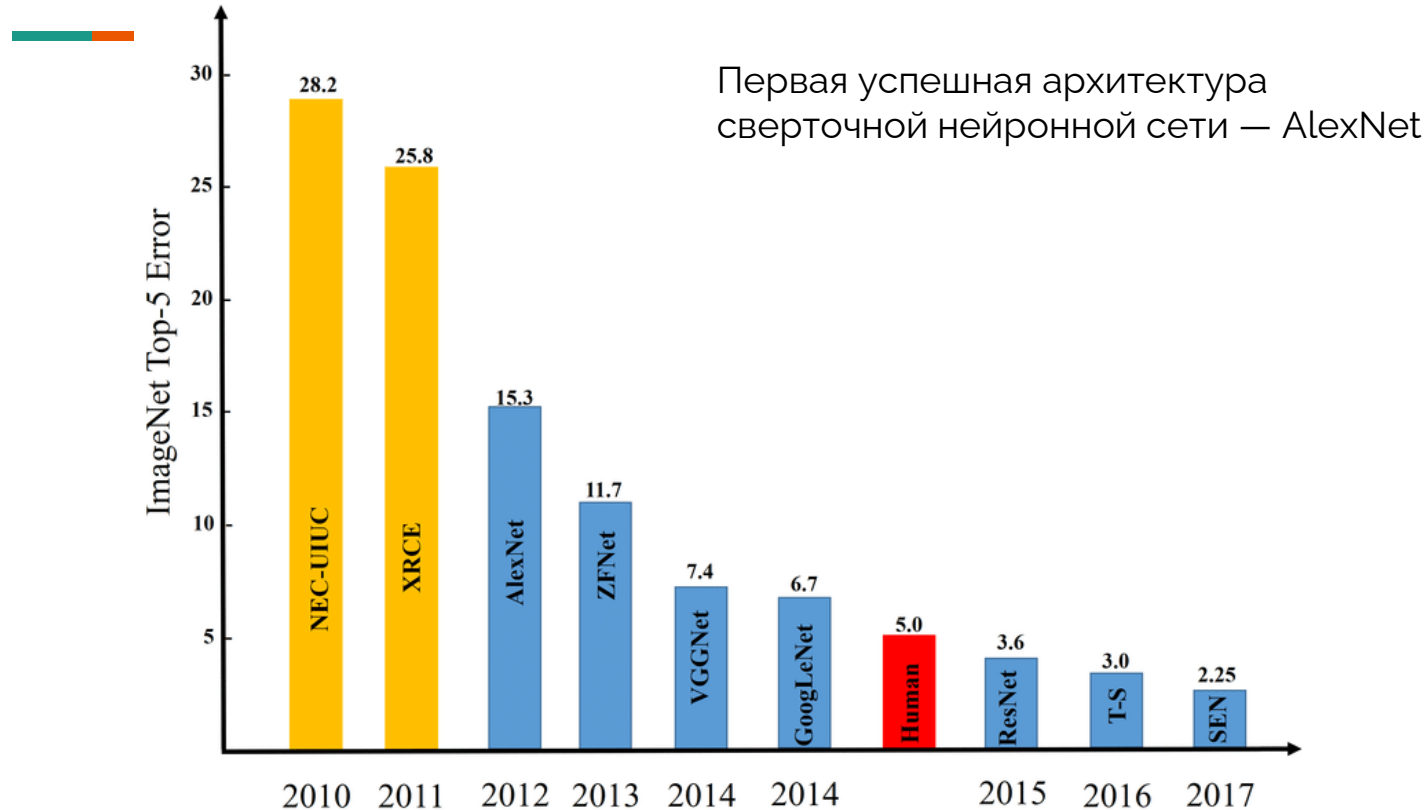
121	10	78	96	125
48	152	68	125	111
145	78	85	89	65
154	214	56	200	66
214	87	45	102	45



Градиенты будут отличаться для пикселей, находящихся на границе изображения и внутри объектов.

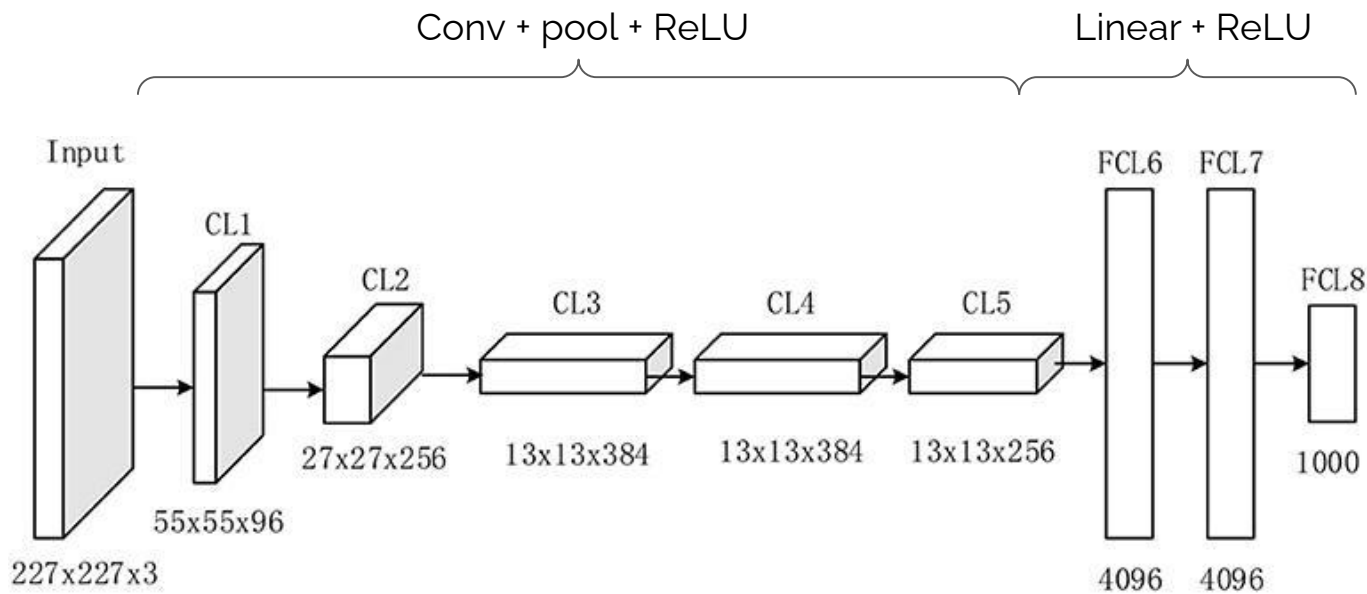
Далее на полученных фичах обучим классификатор изображений (например, логистическая регрессия).

# ImageNet Timeline

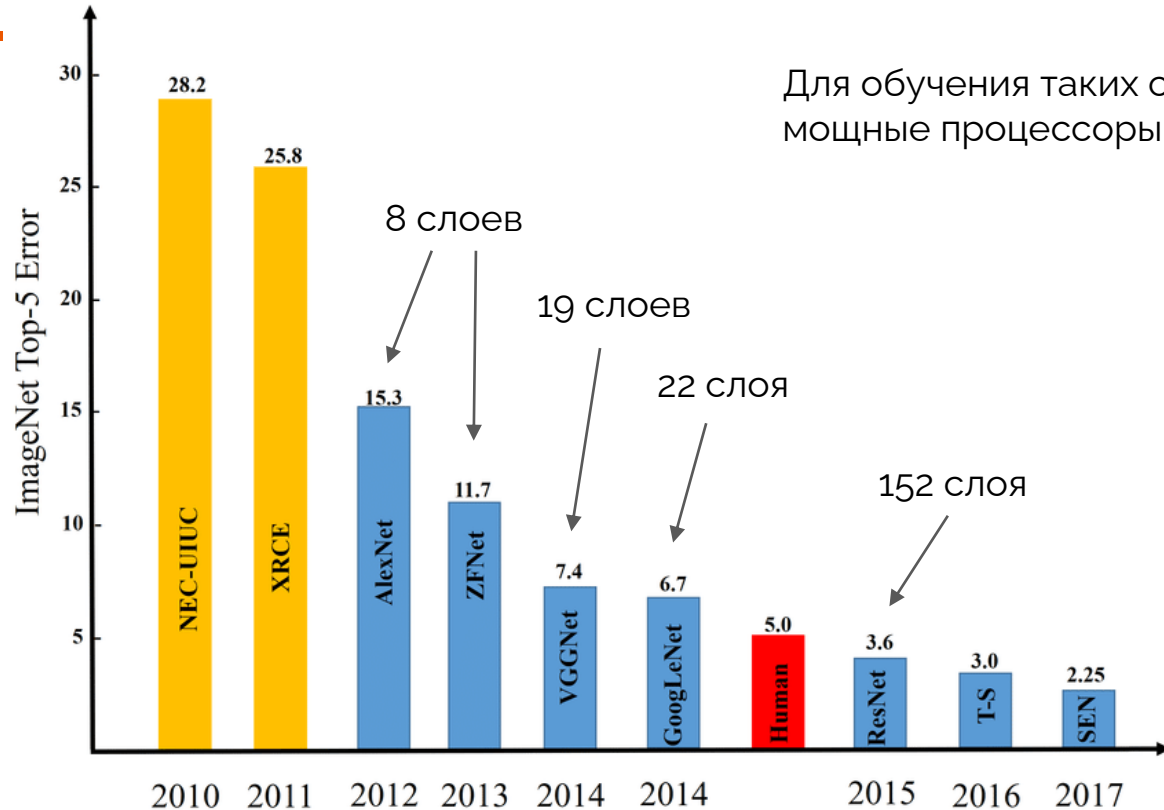




# AlexNet



# ImageNet Timeline

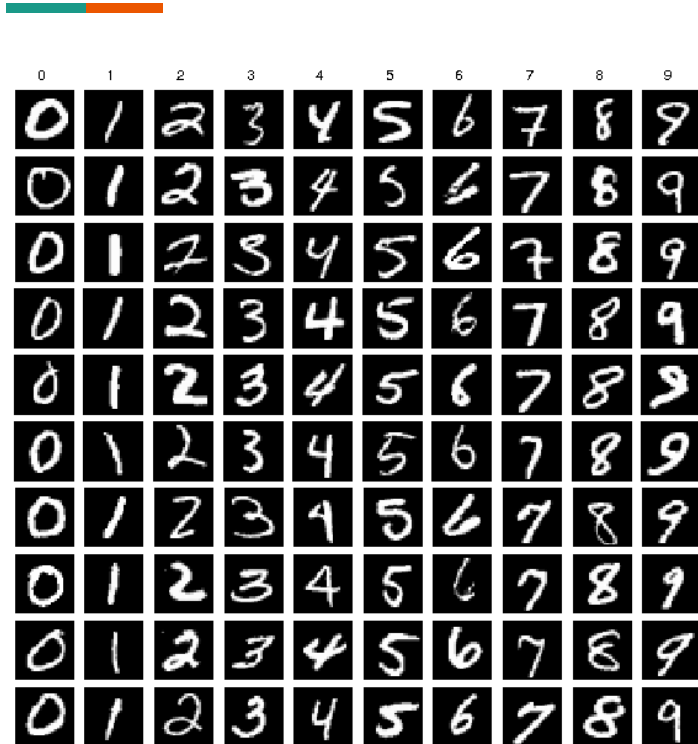


Для обучения таких сетей нужны мощные процессоры (GPU)

---

# Сверточные нейронные сети

# Датасет MNIST



Задача классификации на 10 классов черно-белых изображений размера 28 на 28

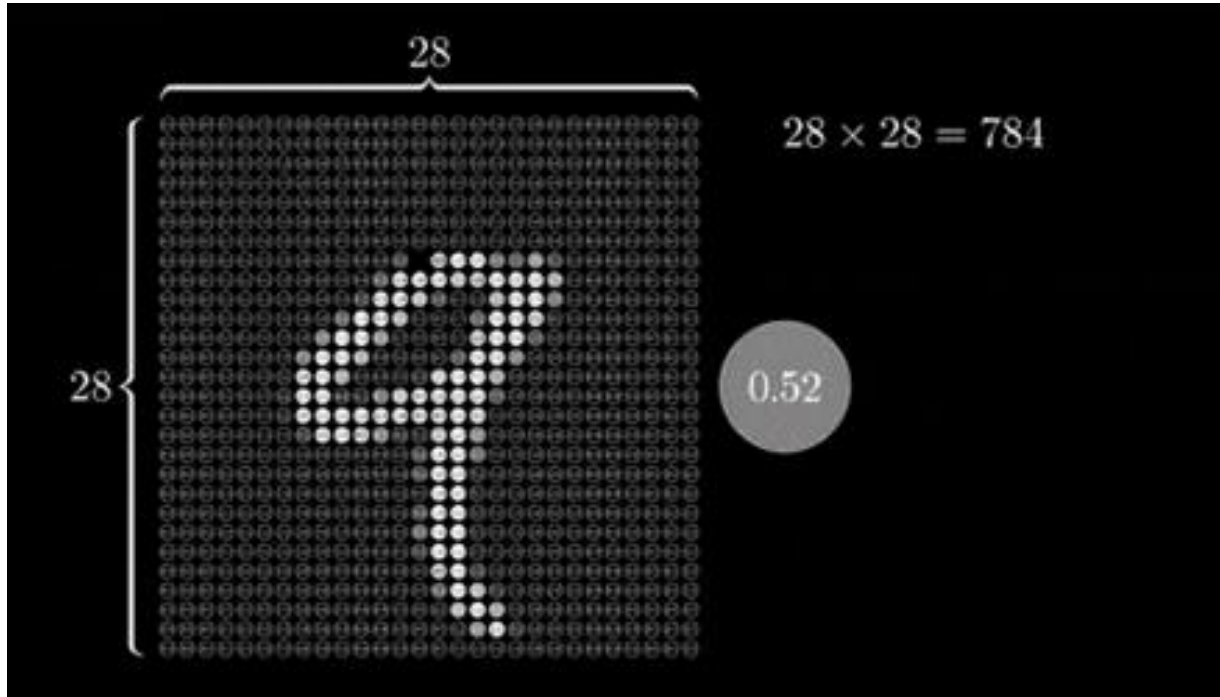
■

[illegible]

28

28

Как решать эту задачу при помощи полносвязных слоев?



# Классификация картинок полносвязной сетью:



Недостатки:

1. Слишком много нейронов в 1 слое сети
2. Ломаются пространственные отношения на картинке, которые могли бы помочь сети в задаче классификации

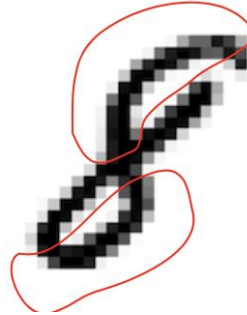
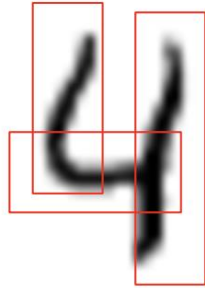
Что отличает четверку от восьмерки?

4

8



Что отличает четверку от восьмерки?



У четверки преимущественно горизонтальные и вертикальные линии, у восьмерки линии плавные

# Фильтры



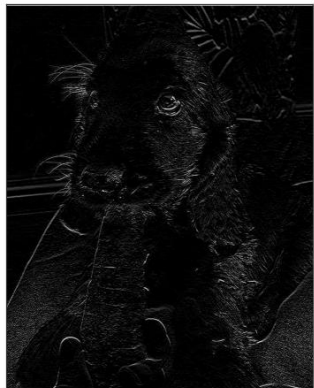
Почти как в инстаграме



0	0	0
0	1	0
0	0	0



# Фильтры



Изображение

28

28

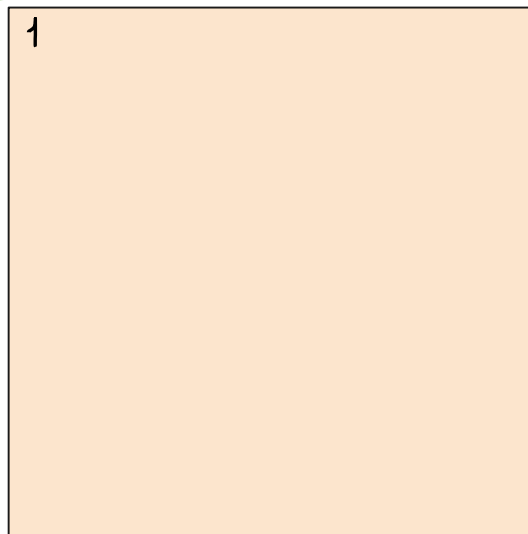
Ядро (фильтр)

1	2	3
-4	7	4
2	-5	1

0 0 0  
0 0 0  
0 0 1

$$0 \cdot 1 + 0 \cdot 2 + \dots + 1 \cdot 1 = 1$$

1	2	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
-4	7	4	0	1	12	0	11	39	137	37	0	152	147	84	0	0	0	0	0
2	-5	1	0	0	0	41	160	250	255	235	162	255	238	206	11	13	0	0	0
0	0	0	16	9	9	150	251	45	21	184	159	154	255	233	40	0	0	0	0
10	0	0	0	0	0	145	146	3	10	0	11	124	253	255	107	0	0	0	0
0	0	3	0	4	15	236	216	0	0	38	109	247	240	169	0	11	0	0	0
1	0	2	0	0	0	253	253	23	62	224	241	255	164	0	5	0	0	0	0
6	0	0	4	0	3	252	250	228	255	255	234	112	28	0	2	17	0	0	0
0	2	1	4	0	21	255	253	251	255	172	31	8	0	1	0	0	0	0	0
0	0	4	0	163	225	251	255	229	120	0	0	0	0	0	11	0	0	0	0
0	0	21	162	255	255	254	255	126	6	0	10	14	6	0	0	9	0	0	0
3	79	242	255	141	66	255	245	189	7	8	0	0	5	0	0	0	0	0	0
26	221	237	98	0	67	251	255	144	0	8	0	0	7	0	0	11	0	0	0
125	255	141	0	87	244	255	208	3	0	0	13	0	1	0	1	0	0	0	0
145	248	228	116	235	255	141	34	0	11	0	1	0	0	0	1	3	0	0	0
85	237	253	246	255	210	21	1	0	1	0	0	6	2	4	0	0	0	0	0
6	23	112	157	114	32	0	0	0	0	2	0	8	0	7	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



0 0 0  
0 0 0  
0 1 0

1 (stride)



0	1	2	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	-4	7	4	1	12	0	11	39	137	37	0	152	147	84	0	0	0	0	0
0	2	-5	1	0	0	41	160	250	255	235	162	255	238	206	11	13	0	0	0
0	0	0	16	9	9	150	251	45	21	184	159	154	255	233	40	0	0	0	0
10	0	0	0	0	0	145	146	3	10	0	11	124	253	255	107	0	0	0	0
0	0	3	0	4	15	236	216	0	0	38	109	247	240	169	0	11	0	0	0
1	0	2	0	0	0	253	253	23	62	224	241	255	164	0	5	0	0	0	0
6	0	0	4	0	3	252	250	228	255	255	234	112	28	0	2	17	0	0	0
0	2	1	4	0	21	255	253	251	255	172	31	8	0	1	0	0	0	0	0
0	0	4	0	163	225	251	255	229	120	0	0	0	0	0	11	0	0	0	0
0	0	21	162	255	255	254	255	126	6	0	10	14	6	0	0	9	0	0	0
3	79	242	255	141	66	255	245	189	7	8	0	0	5	0	0	0	0	0	0
26	221	237	98	0	67	251	255	144	0	8	0	0	7	0	0	11	0	0	0
125	255	141	0	87	244	255	208	3	0	0	13	0	1	0	1	0	0	0	0
145	248	228	116	235	255	141	34	0	11	0	1	0	0	0	1	3	0	0	0
85	237	253	246	255	210	21	1	0	1	0	0	6	2	4	0	0	0	0	0
6	23	112	157	114	32	0	0	0	0	2	0	8	0	7	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1 -5

```

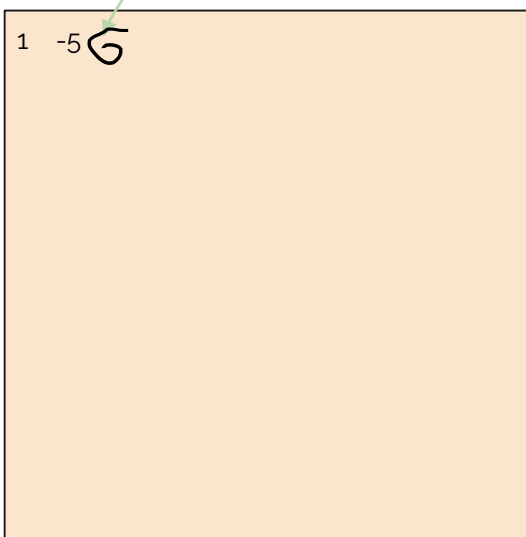
0 0 0
0 0 1
1 0 0

```


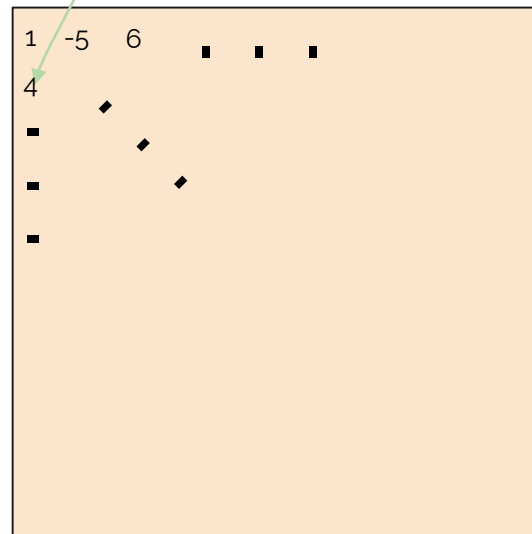
1 (stride)



0	0	1	2	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	-4	7	4	12	0	11	39	137	37	0	152	147	84	0	0	0	0	0
0	0	2	-5	1	0	41	160	250	255	235	162	255	238	206	11	13	0	0	0
0	0	0	16	9	9	150	251	45	21	184	159	154	255	233	40	0	0	0	0
10	0	0	0	0	0	145	146	3	10	0	11	124	253	255	107	0	0	0	0
0	0	3	0	4	15	236	216	0	0	38	109	247	240	169	0	11	0	0	0
1	0	2	0	0	0	253	253	23	62	224	241	255	164	0	5	0	0	0	0
6	0	0	4	0	3	252	250	228	255	255	234	112	28	0	2	17	0	0	0
0	2	1	4	0	21	255	253	251	255	172	31	8	0	1	0	0	0	0	0
0	0	4	0	163	225	251	255	229	120	0	0	0	0	0	11	0	0	0	0
0	0	21	162	255	255	254	255	126	6	0	10	14	6	0	0	9	0	0	0
3	79	242	255	141	66	255	245	189	7	8	0	0	5	0	0	0	0	0	0
26	221	237	98	0	67	251	255	144	0	8	0	0	7	0	0	11	0	0	0
125	255	141	0	87	244	255	208	3	0	0	13	0	1	0	1	0	0	0	0
145	248	228	116	235	255	141	34	0	11	0	1	0	0	0	1	3	0	0	0
85	237	253	246	255	210	21	1	0	1	0	0	6	2	4	0	0	0	0	0
6	23	112	157	114	32	0	0	0	0	2	0	8	0	7	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

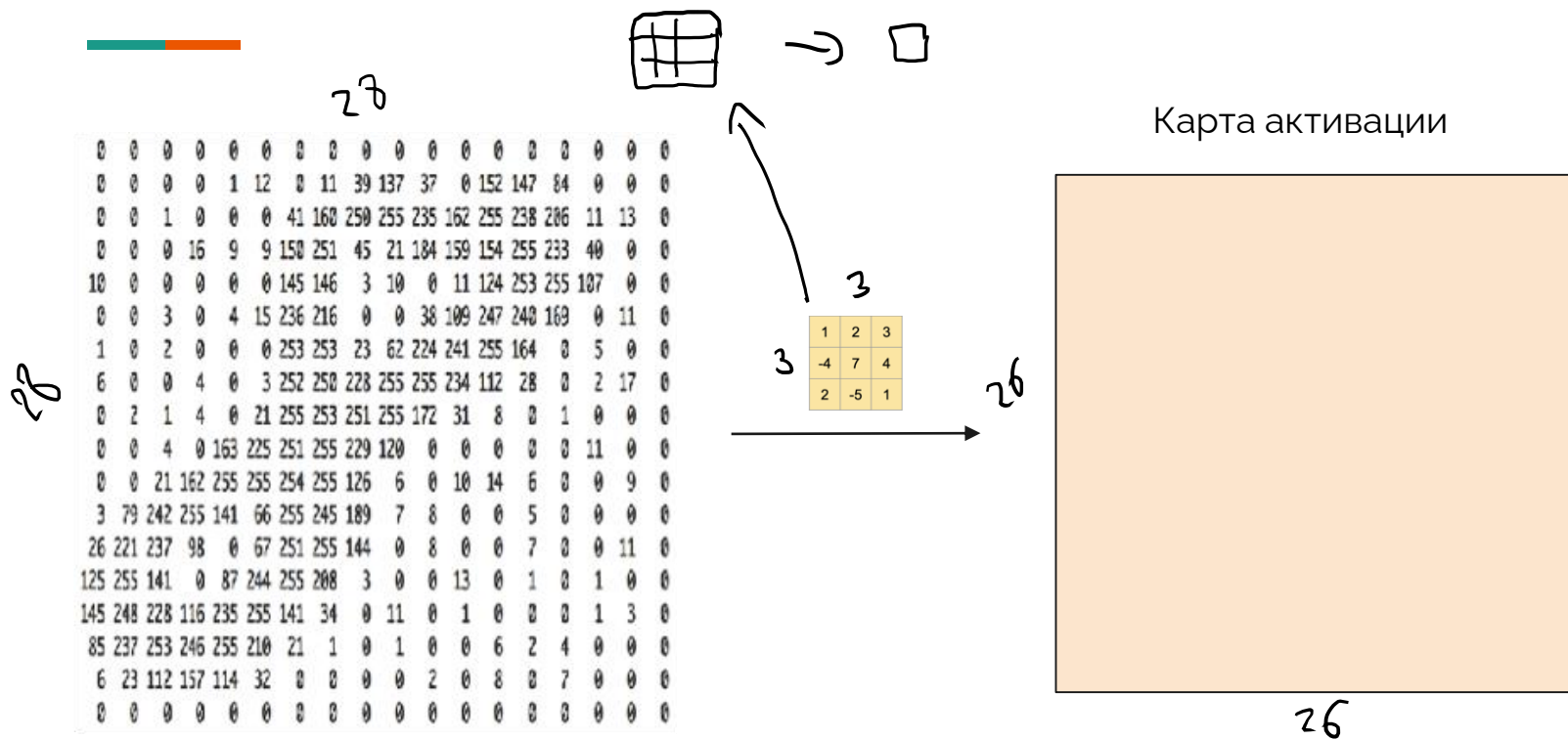




[illegible]

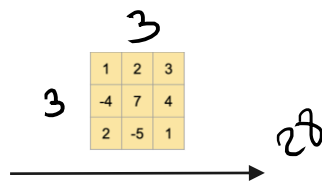
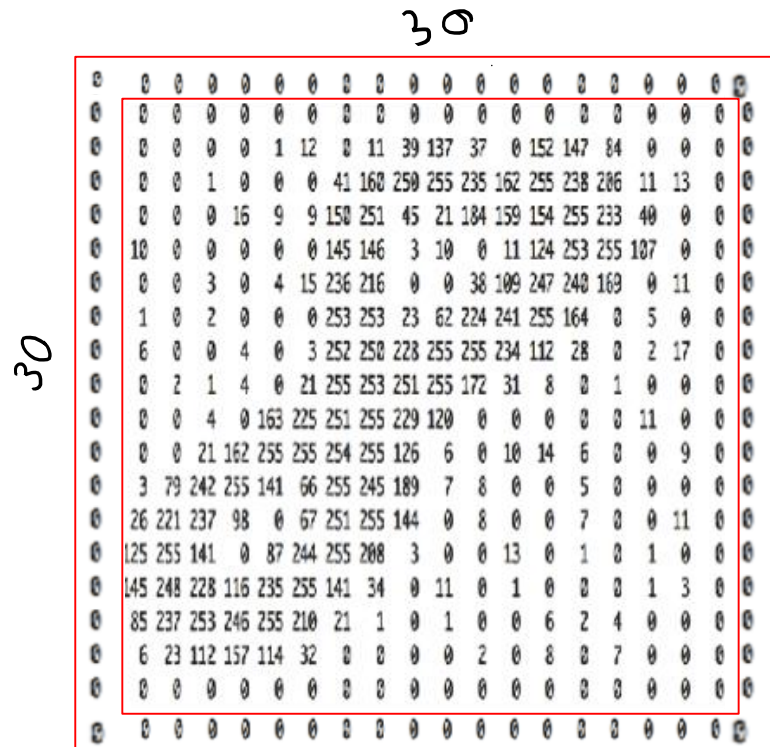


Какой будет размер у изображения после фильтра?

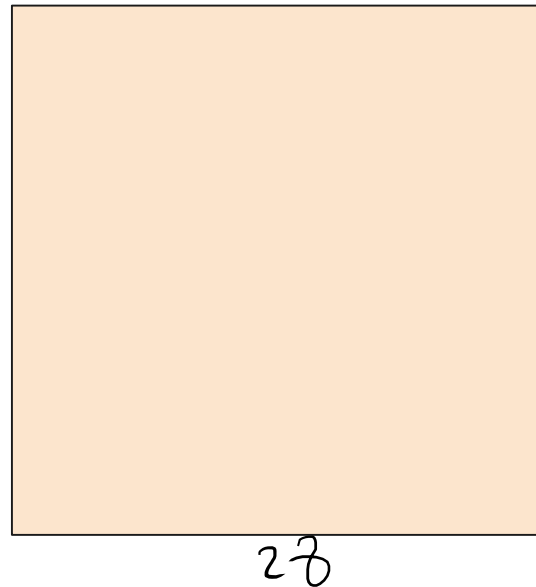


Padding = 1

Используется для манипуляции размерами карт активаций



Карта активации

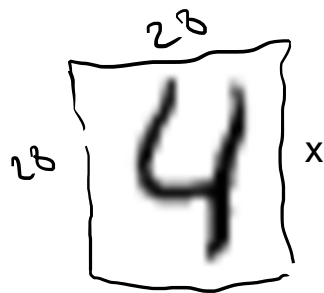


1 <sub>x1</sub>	1 <sub>x0</sub>	1 <sub>x1</sub>	0	0
0 <sub>x0</sub>	1 <sub>x1</sub>	1 <sub>x0</sub>	1	0
0 <sub>x1</sub>	0 <sub>x0</sub>	1 <sub>x1</sub>	1	1
0	0	1	1	0
0	1	1	0	0

Изображение

4		

Feature map



Фильтр, который реагирует  
на вертикальные линии

-1	2	-1
-1	2	-1
-1	2	-1



Активация сильнее, на карте  
активации большие числа

34	55	64	73	23
13	15	23	-86	-96
12	-3	0.4	71	19
11	14	17	-35	19

-1	-1	2
-1	2	-1
2	-1	-1

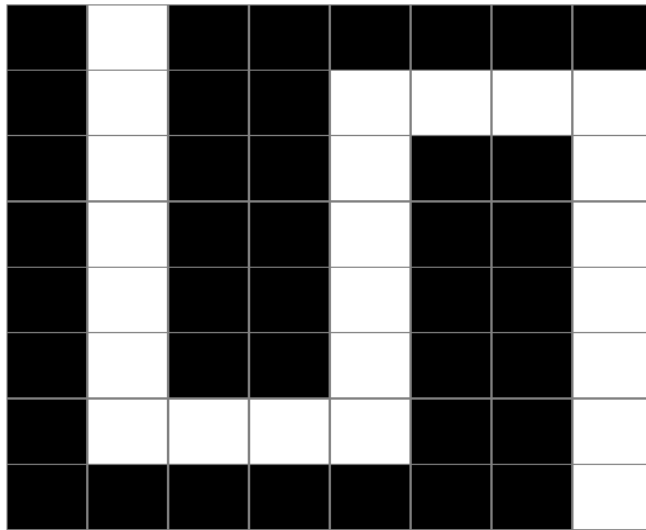


Фильтр, который реагирует  
на изогнутые линии

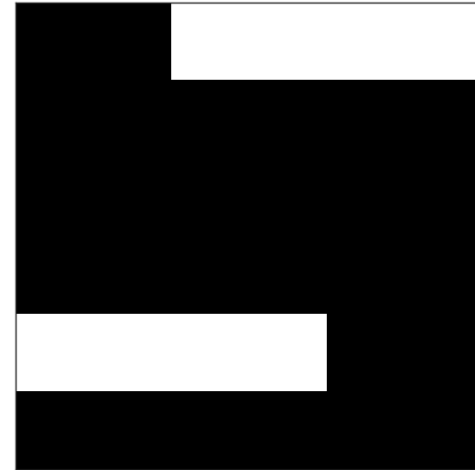
0.01	-0.2	1.8	2	-1.5
3	-0.5	6	-7	0.4
4	5	-0.8	-5	-1
1.2	0.5	3	-3	0.4

Активация слабее, на карте  
активации маленькие числа

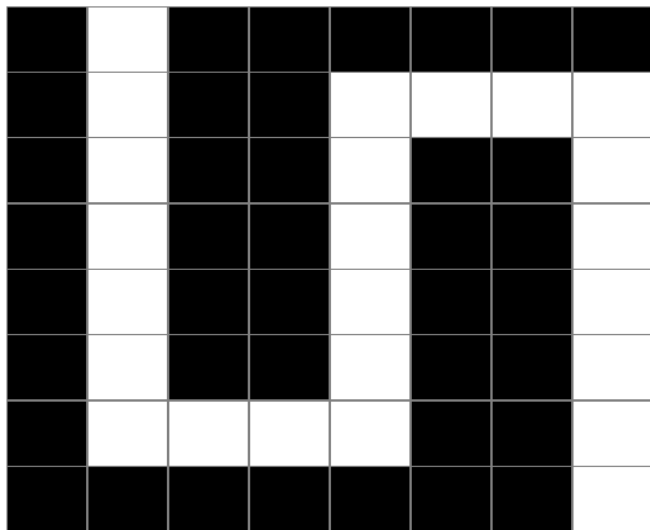
# Фильтр, реагирующий на горизонтальные линии



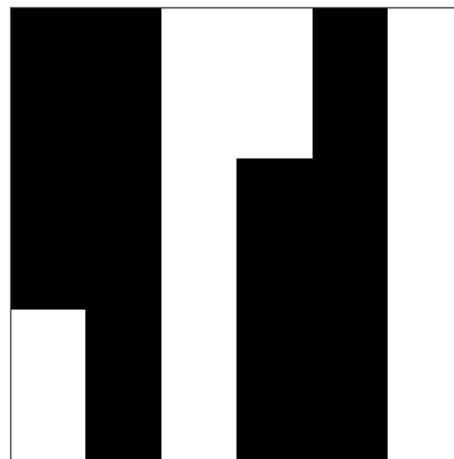
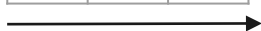
-1	-1	-1
0	0	0
1	1	1



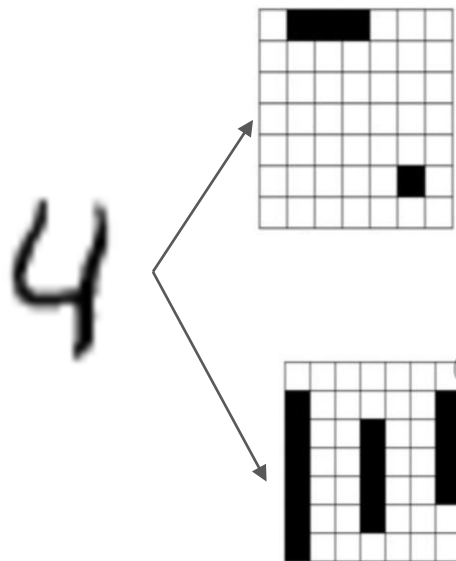
## Фильтр, реагирующий на вертикальные линии



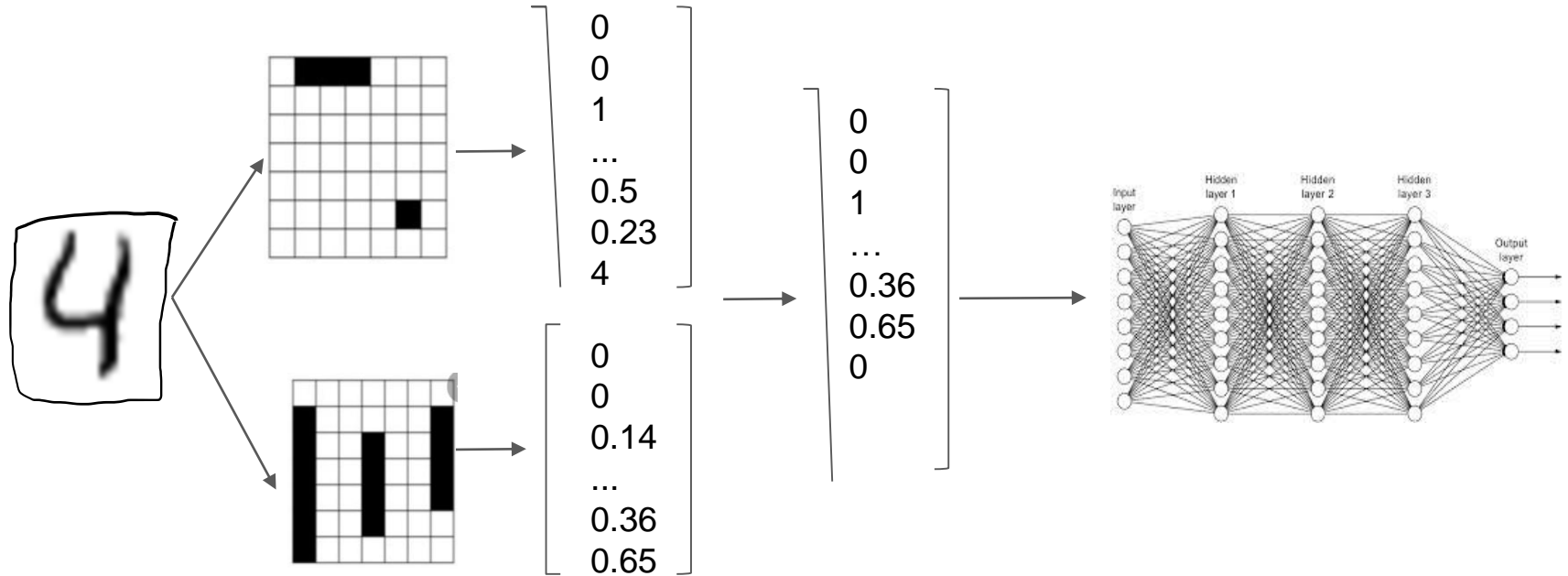
1	0	-1
1	0	-1
1	0	-1



# Можно применять сразу несколько фильтров



После получения карт активаций, мы **развернем все карты в векторы**, **сконкатенируем** и подадим на вход полносвязной сети



свертки

flatten

конкатенация

подача в полносвязную сеть

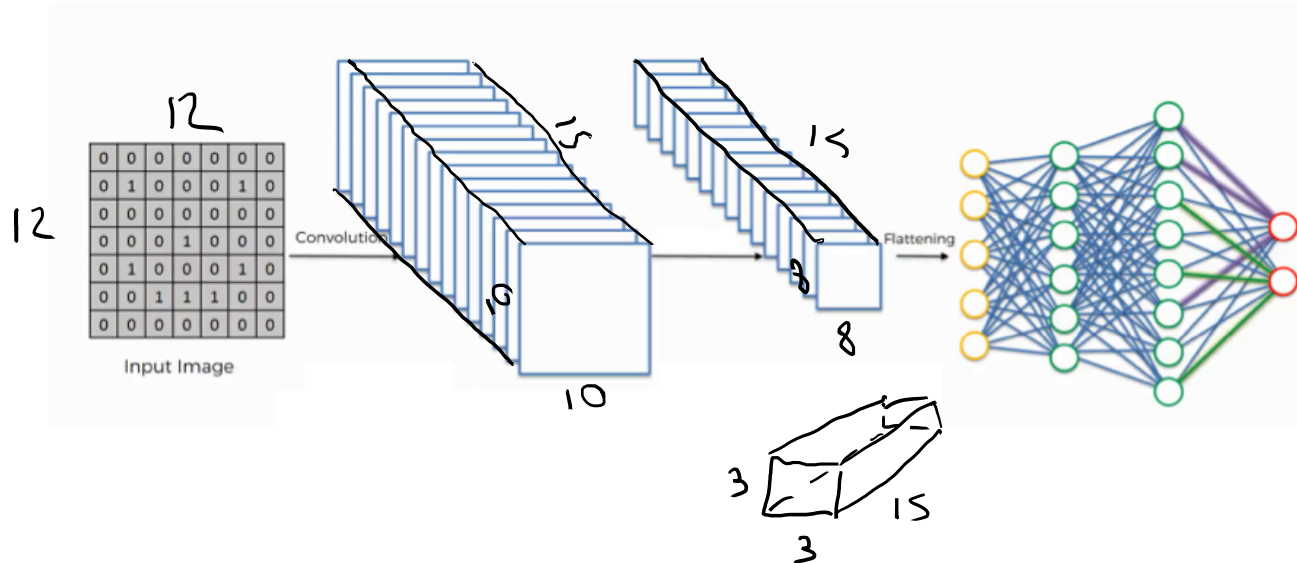


# Сверточная нейросеть



Сверточная сеть  
(извлекает признаки)

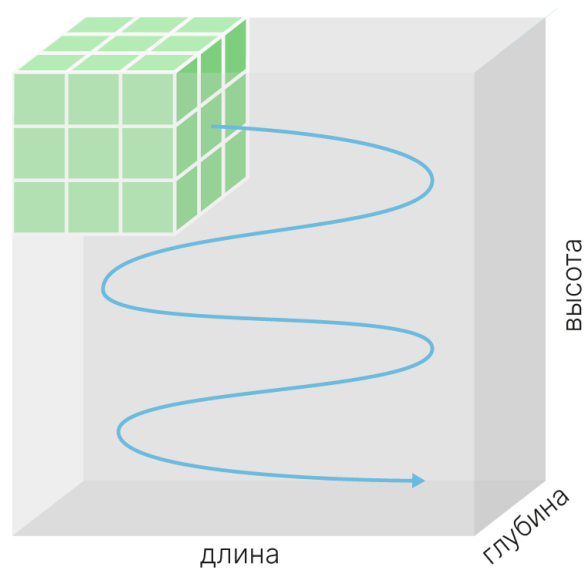
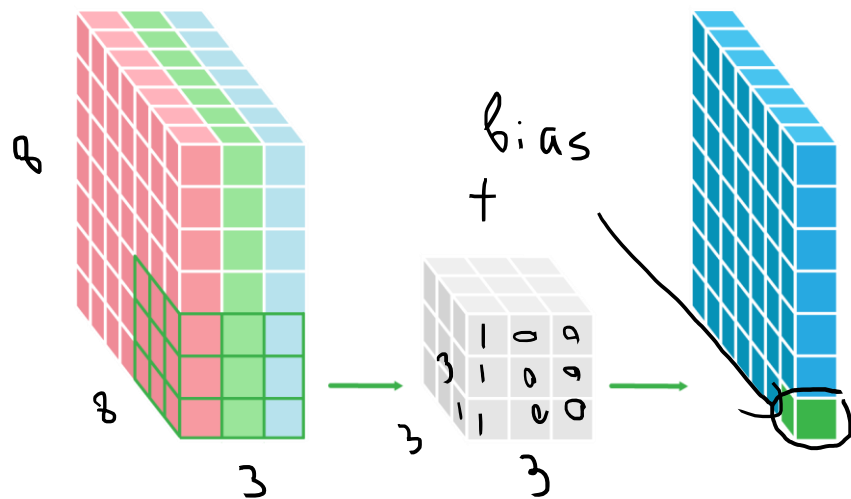
Полносвязная сеть  
(классификатор)



А что делать, если изображение цветное?



# А что делать, если изображение цветное?



---

## Пулинг (pooling)

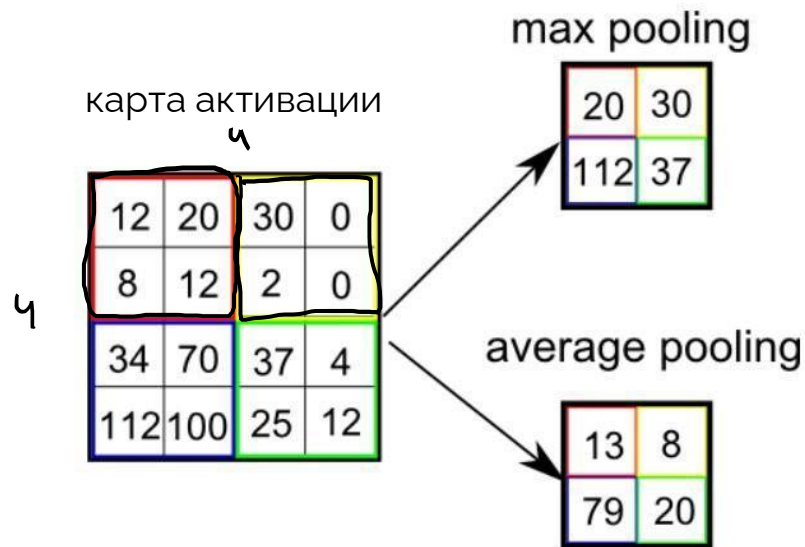
# Pooling

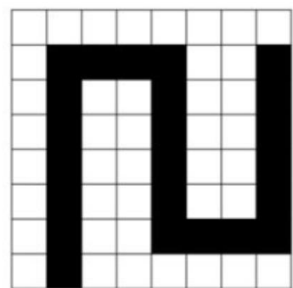


Техника уменьшения размерности (downsampling'a) карт активаций

Используется для:

- уменьшения размерности очень больших изображений
- уменьшения чувствительности свертков к положению объектов на картинке

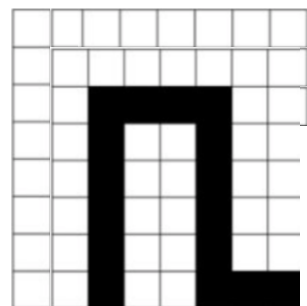
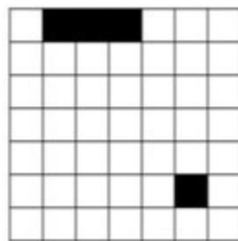




+

-1	-1	-1
0	0	0
1	1	1

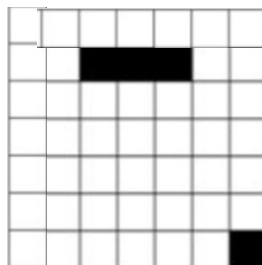
=



+

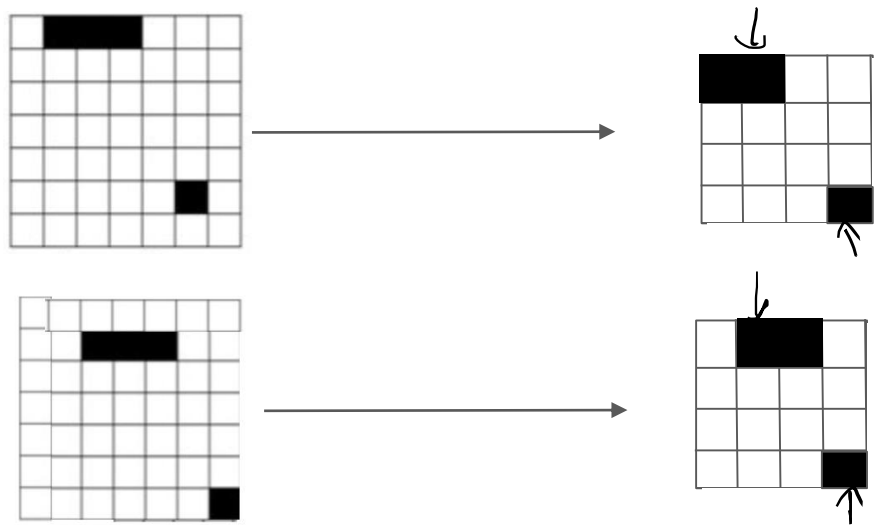
-1	-1	-1
0	0	0
1	1	1

=



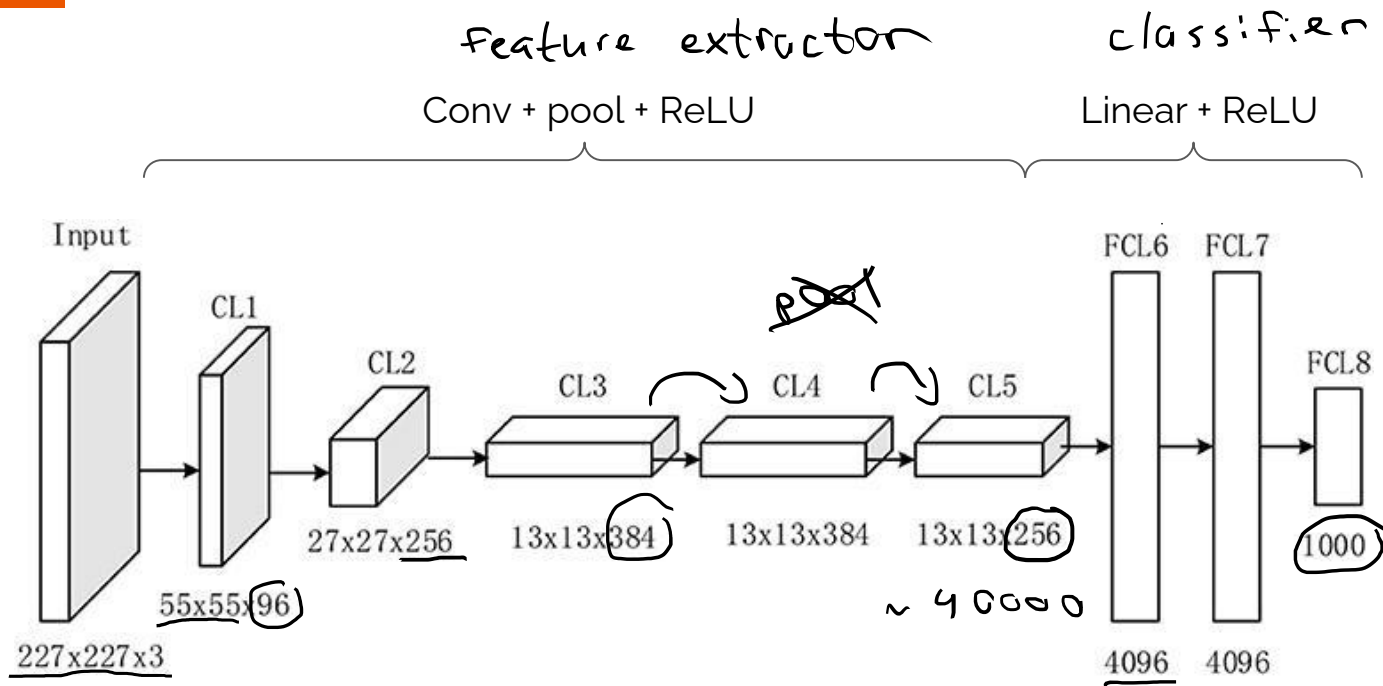
Разница в  
расположении

Результат применения 2x2 MaxPooling'a к картам активаций:



# AlexNet

Convolution == German





---

# Основные параметры слоев

# CONV2D

num C to average. argument  
↓  
macro ops

```
CLASS torch.nn.Conv2d(in_channels, out_channels, kernel_size, stride=1, padding=0,  
dilation=1, groups=1, bias=True, padding_mode='zeros', device=None, dtype=None) [SOURCE]
```

Applies a 2D convolution over an input signal composed of several input planes.

In the simplest case, the output value of the layer with input size  $(N, C_{\text{in}}, H, W)$  and output  $(N, C_{\text{out}}, H_{\text{out}}, W_{\text{out}})$  can be precisely described as:

$$\text{out}(N_i, C_{\text{out}_j}) = \text{bias}(C_{\text{out}_j}) + \sum_{k=0}^{C_{\text{in}}-1} \text{weight}(C_{\text{out}_j}, k) \star \text{input}(N_i, k)$$

where  $\star$  is the valid 2D **cross-correlation** operator,  $N$  is a batch size,  $C$  denotes a number of channels,  $H$  is a height of input planes in pixels, and  $W$  is width in pixels.

This module supports **TensorFloat32**.

- `stride` controls the stride for the cross-correlation, a single number or a tuple.
- `padding` controls the amount of padding applied to the input. It can be either a string {'valid', 'same'} or a tuple of ints giving the amount of implicit padding applied on both sides.

# MAXPOOL2D

```
CLASS torch.nn.MaxPool2d(kernel_size, stride=None, padding=0, dilation=1,
return_indices=False, ceil_mode=False) [SOURCE]
```

Applies a 2D max pooling over an input signal composed of several input planes.

In the simplest case, the output value of the layer with input size  $(N, C, H, W)$ , output  $(N, C, H_{out}, W_{out})$  and `kernel_size`  $(kH, kW)$  can be precisely described as:

$$out(N_i, C_j, h, w) = \max_{m=0, \dots, kH-1} \max_{n=0, \dots, kW-1} input(N_i, C_j, stride[0] \times h + m, stride[1] \times w + n)$$

If `padding` is non-zero, then the input is implicitly padded with negative infinity on both sides for `padding` number of points. `dilation` controls the spacing between the kernel points. It is harder to describe, but this [link](#) has a nice visualization of what `dilation` does.

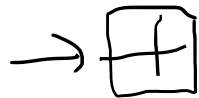
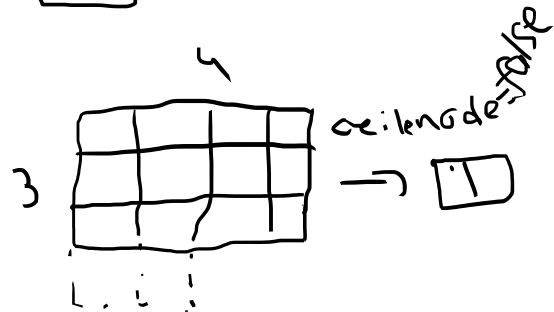
## • NOTE

When `ceil_mode=True`, sliding windows are allowed to go off-bounds if they start within the left padding or the input. Sliding windows that would start in the right padded region are ignored.

The parameters `kernel_size`, `stride`, `padding`, `dilation` can either be:

- a single `int` – in which case the same value is used for the height and width dimension
- a `tuple` of two ints – in which case, the first `int` is used for the height dimension, and the second `int` for the width dimension

pooling



ceil mode = True

# AVGPOOL2D

```
CLASS torch.nn.AvgPool2d(kernel_size, stride=None, padding=0, ceil_mode=False,  
count_include_pad=True, divisor_override=None) [SOURCE]
```

Applies a 2D average pooling over an input signal composed of several input planes.

In the simplest case, the output value of the layer with input size  $(N, C, H, W)$ , output  $(N, C, H_{out}, W_{out})$  and `kernel_size`  $(kH, kW)$  can be precisely described as:

$$out(N_i, C_j, h, w) = \frac{1}{kH * kW} \sum_{m=0}^{kH-1} \sum_{n=0}^{kW-1} input(N_i, C_j, stride[0] \times h + m, stride[1] \times w + n)$$

If `padding` is non-zero, then the input is implicitly zero-padded on both sides for `padding` number of points.

## • NOTE

When `ceil_mode=True`, sliding windows are allowed to go off-bounds if they start within the left padding or the input. Sliding windows that would start in the right padded region are ignored.

The parameters `kernel_size`, `stride`, `padding` can either be:

- a single `int` – in which case the same value is used for the height and width dimension
- a `tuple` of two ints – in which case, the first `int` is used for the height dimension, and the second `int` for the width dimension

---

**Очень важная информация!**

## Как посчитать размер карты активации после свертки?

Довольно важно понимать, какой будет размер карты активации (высота / ширина) после применения свертки. Для этого есть простая формула:

$$H_{out} = \frac{(H_{in} - K + 2P)}{S} + 1 = 26$$

$$W_{out} = \frac{(W_{in} - K + 2P)}{S} + 1 = 26$$



mmm tasty

mmm tasty

mmm tasty

mmm tasty

sadness

