

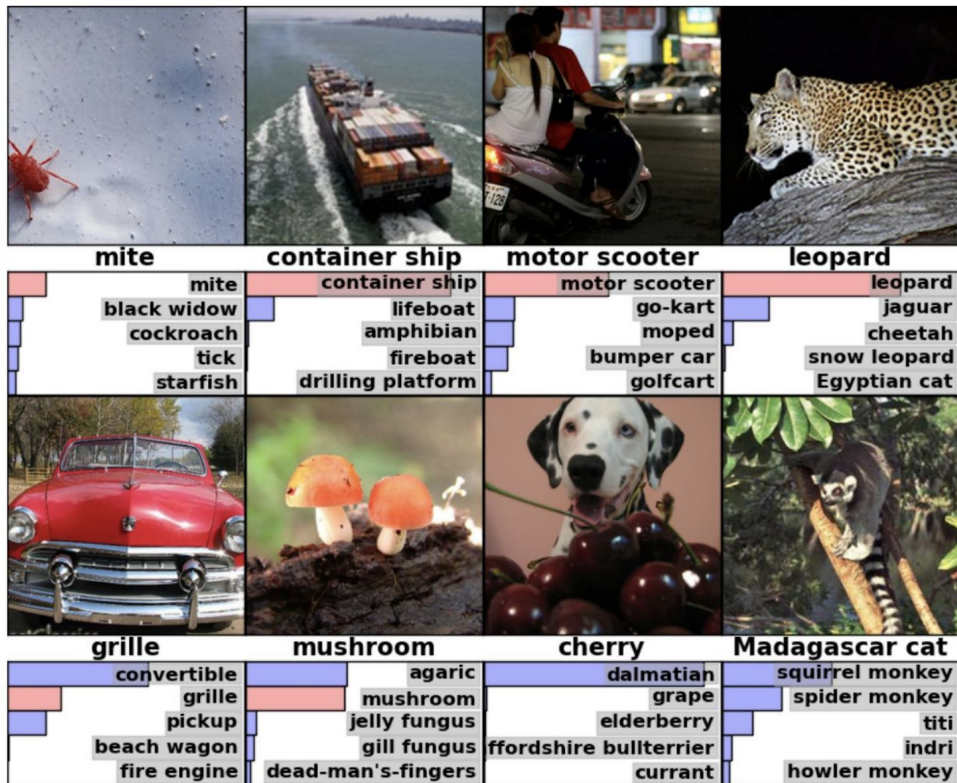
Свёртки. Свёрточные сети в Pytorch. Сегментация.

Практикум на ЭВМ для 317 группы, весна 2021
Кафедра ММП ВМК МГУ

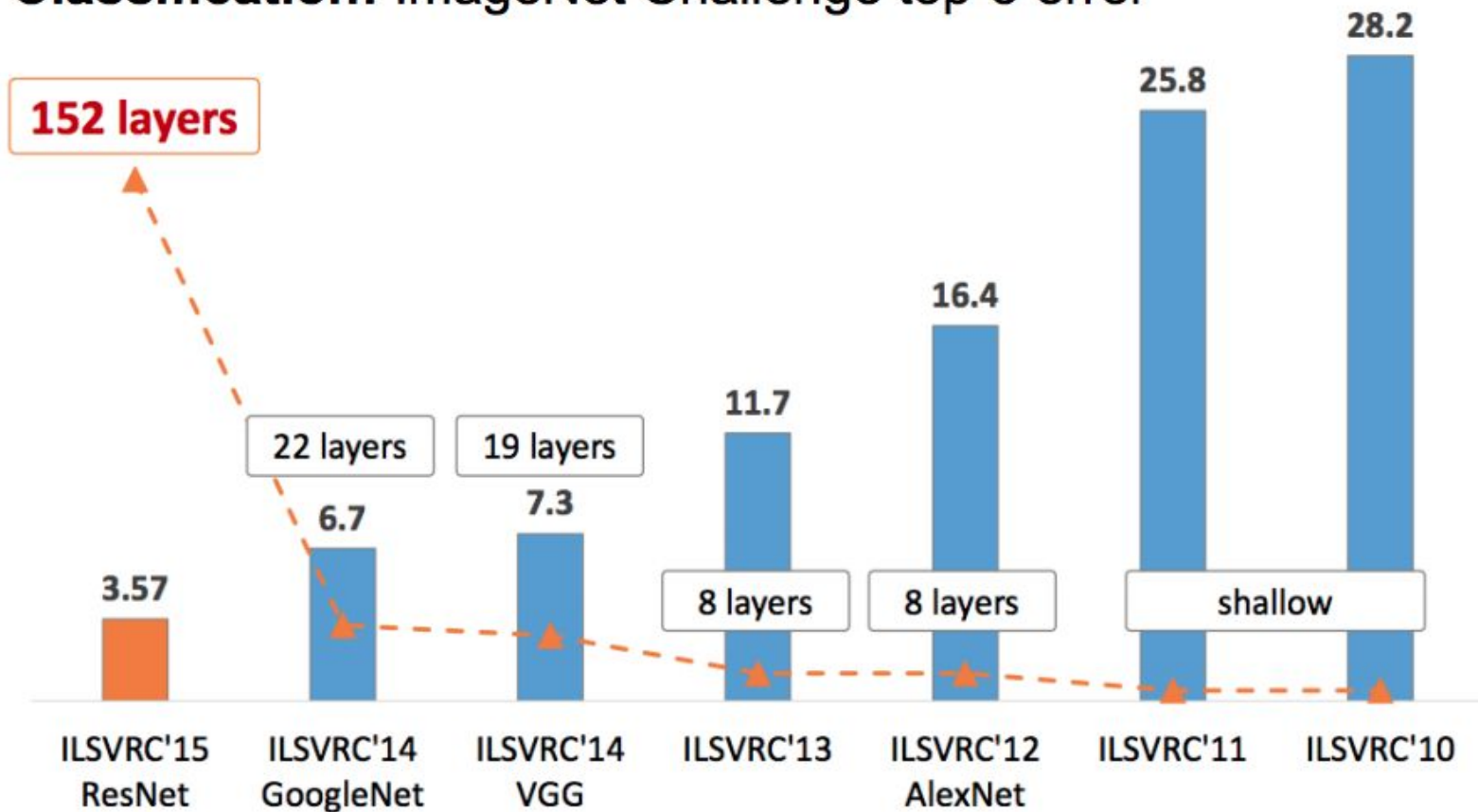
ImageNet Challenge

IMAGENET

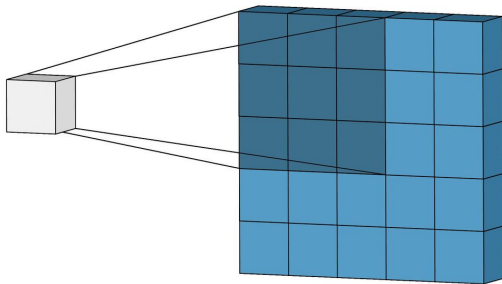
- 1,000 object classes (categories).
- Images:
 - 1.2 M train
 - 100k test.



Classification: ImageNet Challenge top-5 error



Свертка



CONV2D

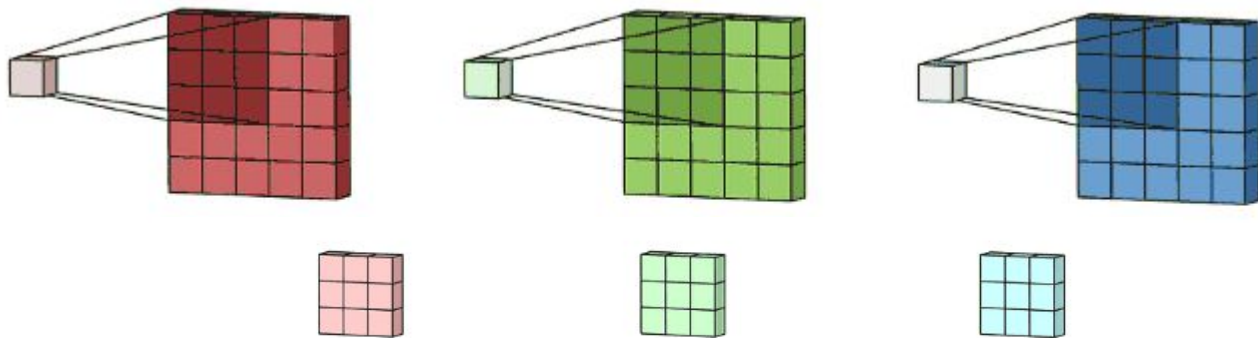
```
CLASS torch.nn.Conv2d(in_channels: int, out_channels: int, kernel_size: Union[T, Tuple[T, T]], stride: Union[T, Tuple[T, T]] = 1, padding: Union[T, Tuple[T, T]] = 0, dilation: Union[T, Tuple[T, T]] = 1, groups: int = 1, bias: bool = True, padding_mode: str = 'zeros') [SOURCE]
```

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_{in}, H, W) and output $(N, C_{\text{out}}, H_{\text{out}}, W_{\text{out}})$ can be precisely described as:

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

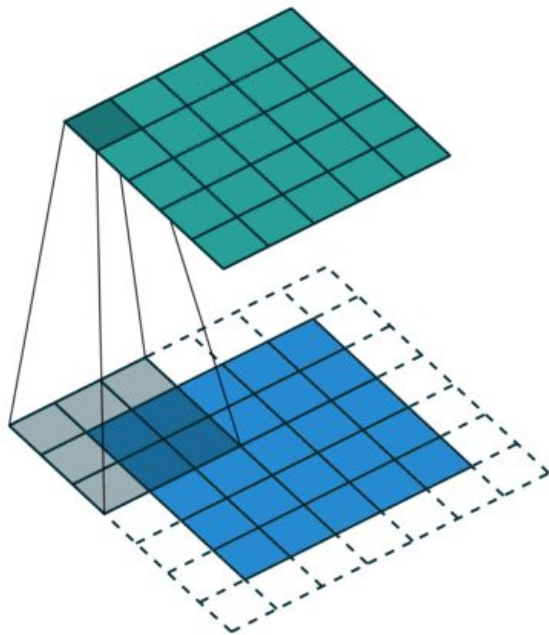
Свертка



Операция свертки
многоканального изображения
K с ядром U:

$$V(x, y, t) = \sum_{i=x-\delta}^{x+\delta} \sum_{j=y-\delta}^{y+\delta} \sum_{s=1}^S K(i - x + \delta, j - y + \delta, s, t) \cdot U(i, j, s)$$

Свертка



CONV2D

```
CLASS torch.nn.Conv2d(in_channels: int, out_channels: int, kernel_size: Union[T,  
    Tuple[T, T]], stride: Union[T, Tuple[T, T]] = 1, padding: Union[T, Tuple[T, T]] =  
    0, dilation: Union[T, Tuple[T, T]] = 1, groups: int = 1, bias: bool = True,  
    padding_mode: str = 'zeros')
```

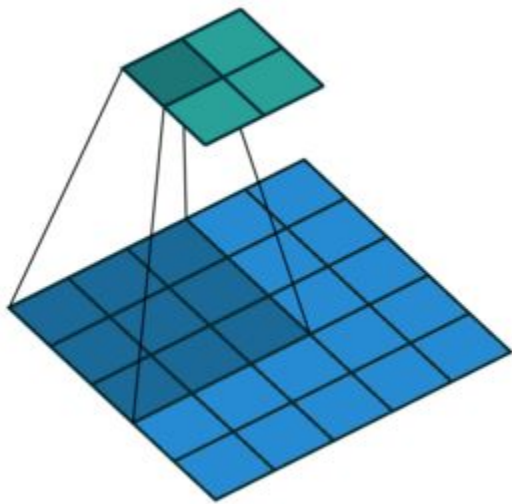
[SOURCE]

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_{in}, H, W) and output $(N, C_{\text{out}}, H_{\text{out}}, W_{\text{out}})$ can be precisely described as:

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

Свертка



CONV2D

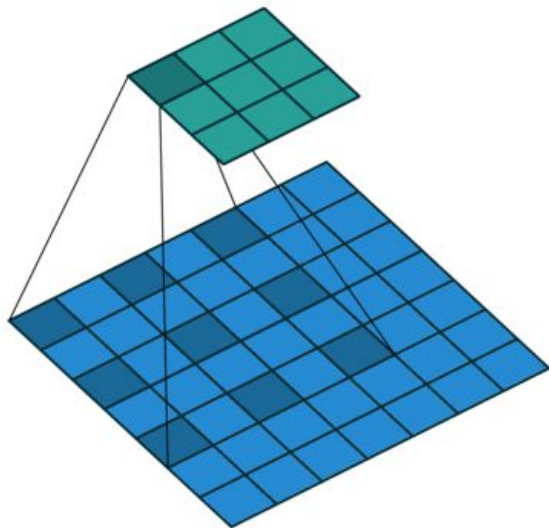
```
CLASS torch.nn.Conv2d(in_channels: int, out_channels: int, kernel_size: Union[T, Tuple[T, T]], stride: Union[T, Tuple[T, T]] = 1, padding: Union[T, Tuple[T, T]] = 0, dilation: Union[T, Tuple[T, T]] = 1, groups: int = 1, bias: bool = True, padding_mode: str = 'zeros') [SOURCE]
```

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_{in}, H, W) and output $(N, C_{\text{out}}, H_{\text{out}}, W_{\text{out}})$ can be precisely described as:

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

Свертка



CONV2D

```
CLASS torch.nn.Conv2d(in_channels: int, out_channels: int, kernel_size: Union[T,  
    Tuple[T, T]], stride: Union[T, Tuple[T, T]] = 1, padding: Union[T, Tuple[T, T]] =  
    0, dilation: Union[T, Tuple[T, T]] = 1, groups: int = 1, bias: bool = True,  
    padding_mode: str = 'zeros')
```

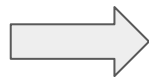
[SOURCE]

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

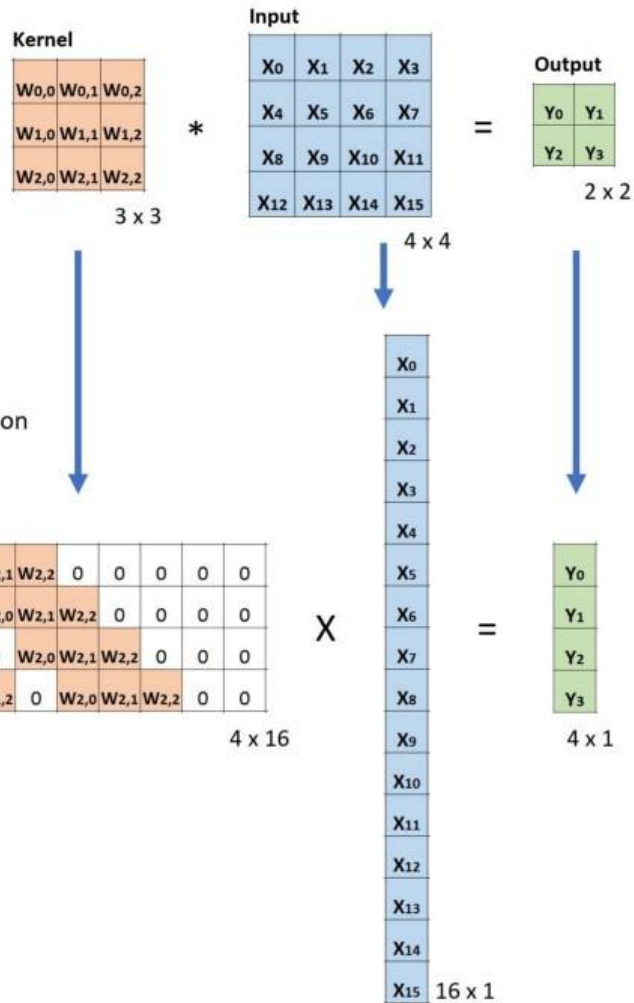
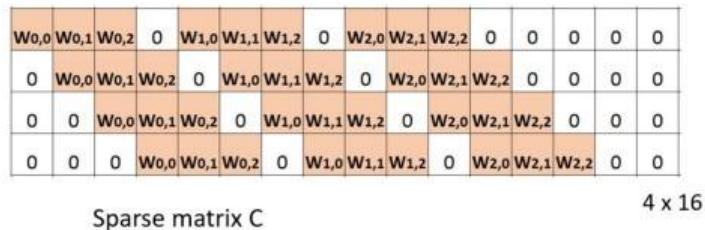
In the simplest case, the output value of the layer with input size (N, C_{in}, H, W) and output $(N, C_{\text{out}}, H_{\text{out}}, W_{\text{out}})$ can be precisely described as:

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



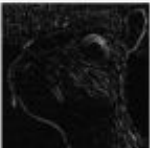

Свертка реализуется как матричное умножение






Unrolling the convolution operation
to matrix multiplication

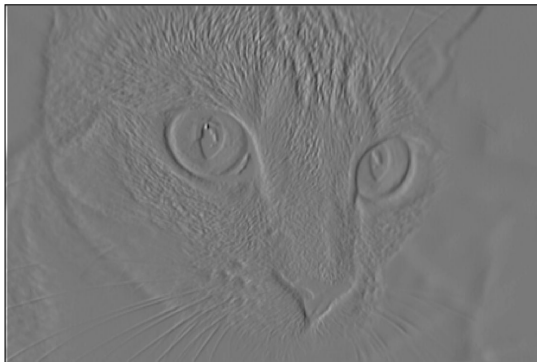


Примеры свертки

Operation	Filter	Convolved Image
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	

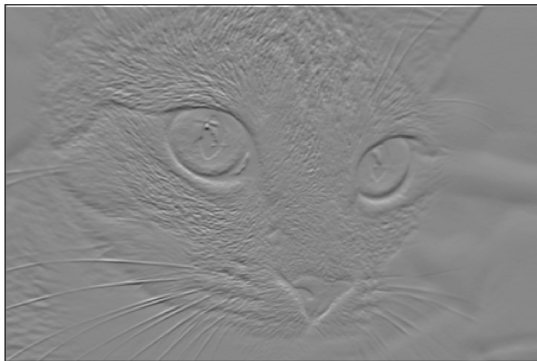
Operation	Filter	Convolved Image
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur (approximation)	$\frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	

Примеры свертки

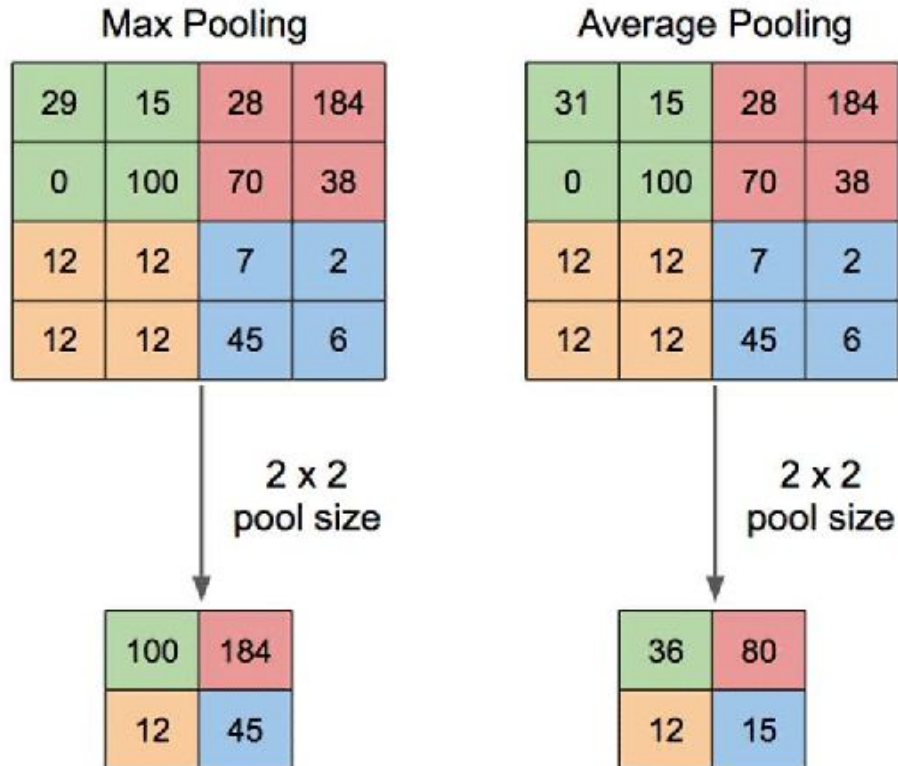


-1	0	1
-2	0	2
-1	0	1

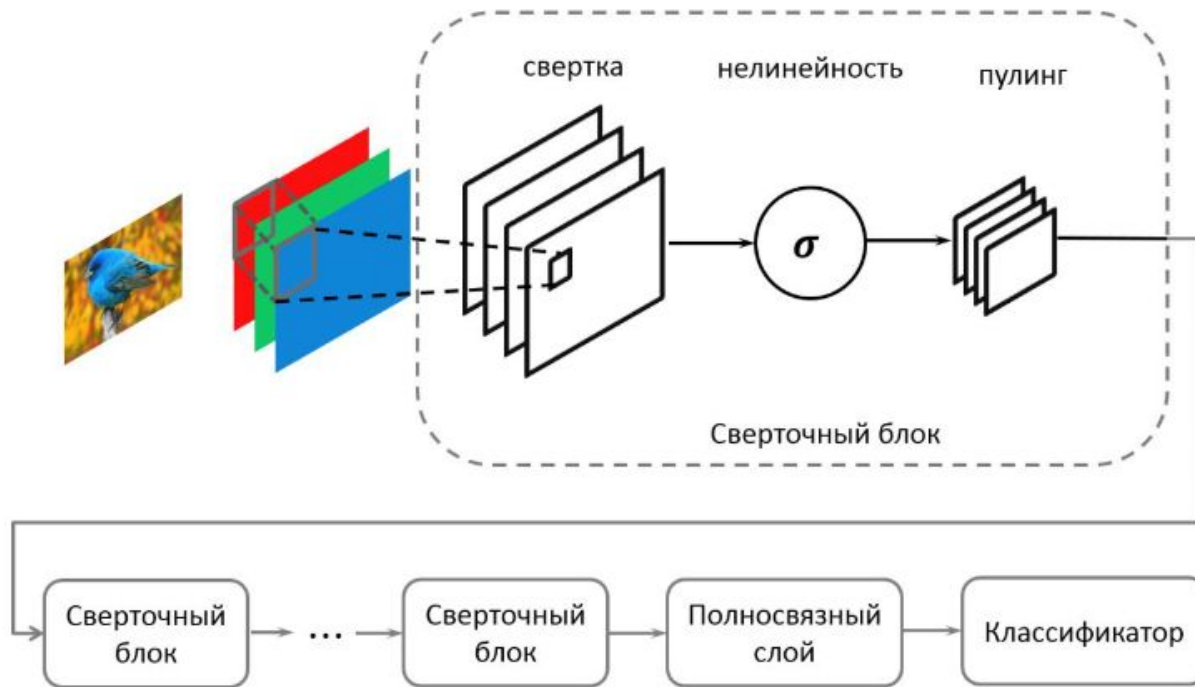
1	2	1
0	0	0
-1	-2	-1



Max pooling and average pooling



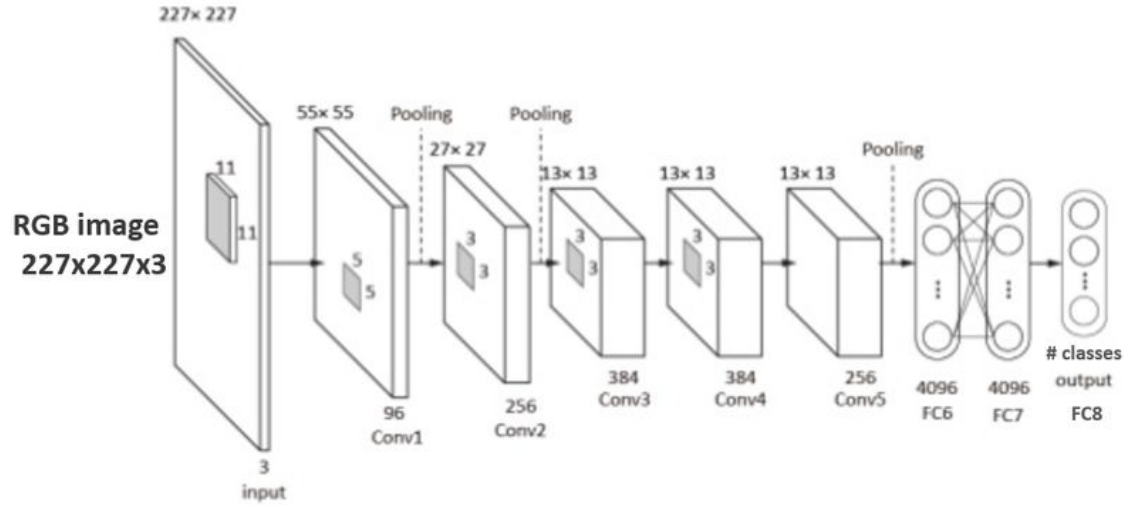
Простая схема сверточной сети



Известные архитектуры сверточных сетей для классификации

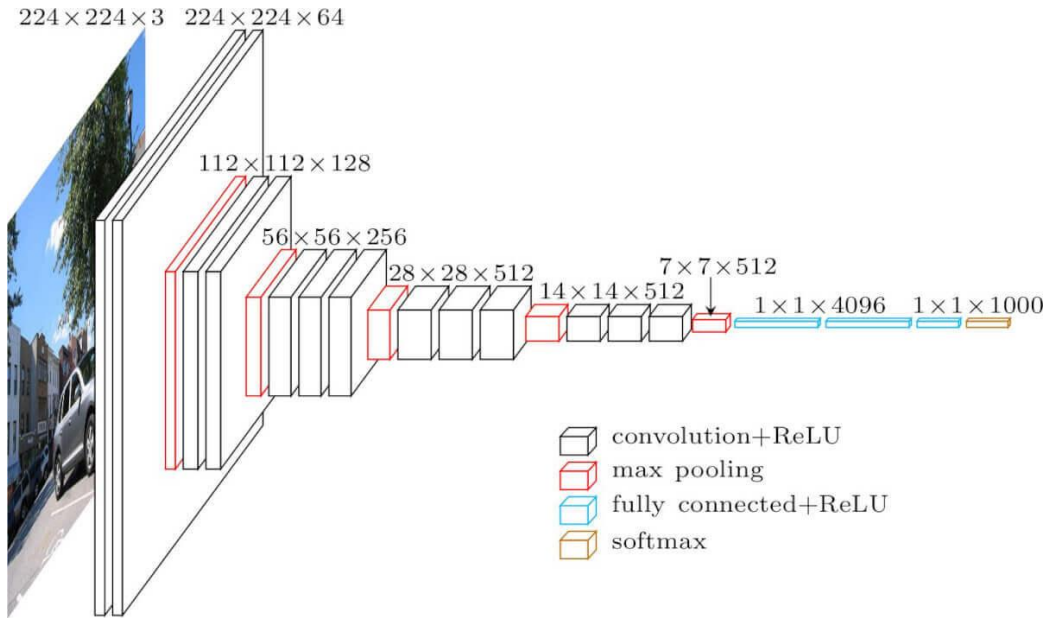
- AlexNet
- VGG
- Inception
- ResNet
- <https://pytorch.org/vision/0.8/models.html>

AlexNet



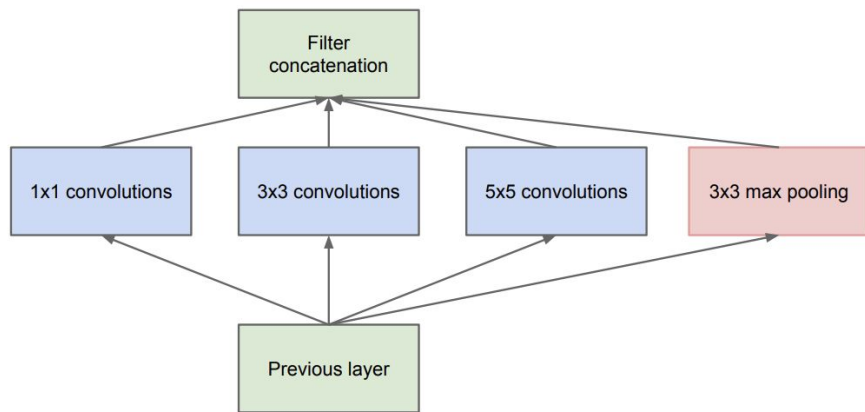
- ReLU
- Dropout

VGG

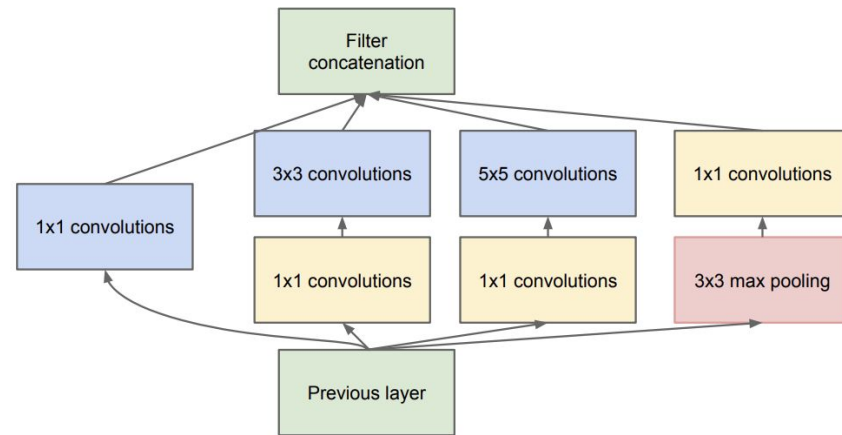


- Design of deeper networks (roughly twice as deep as AlexNet)

Inception



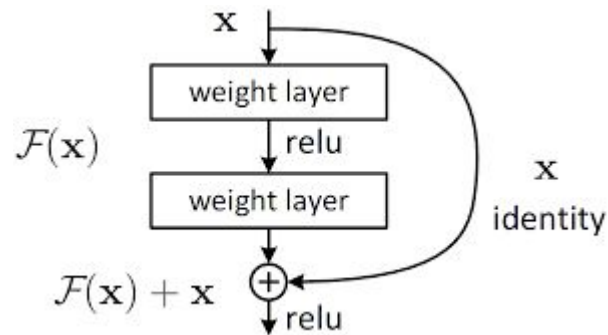
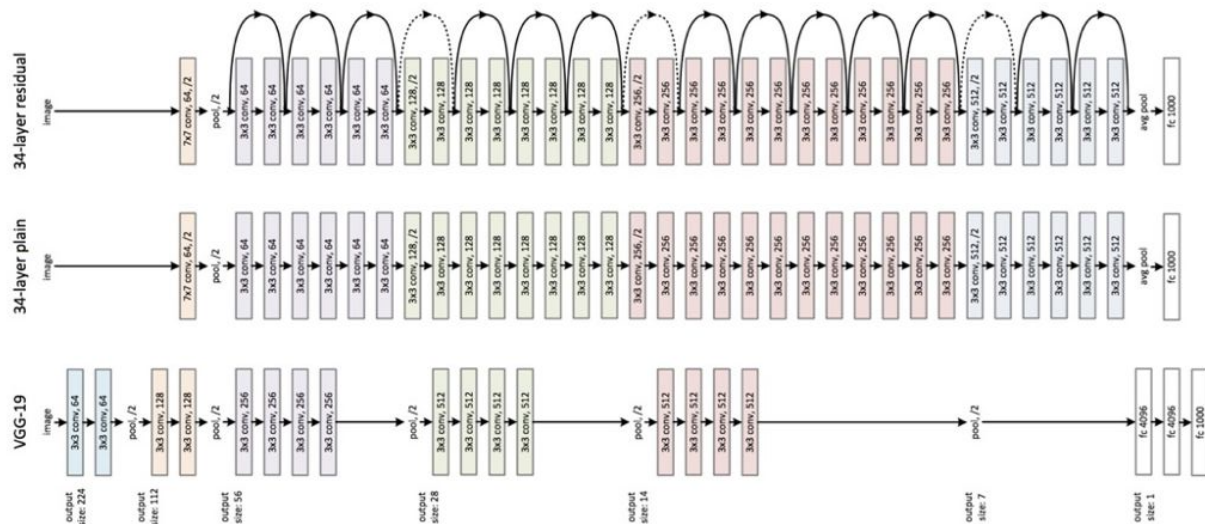
(a) Inception module, naïve version



(b) Inception module with dimension reductions

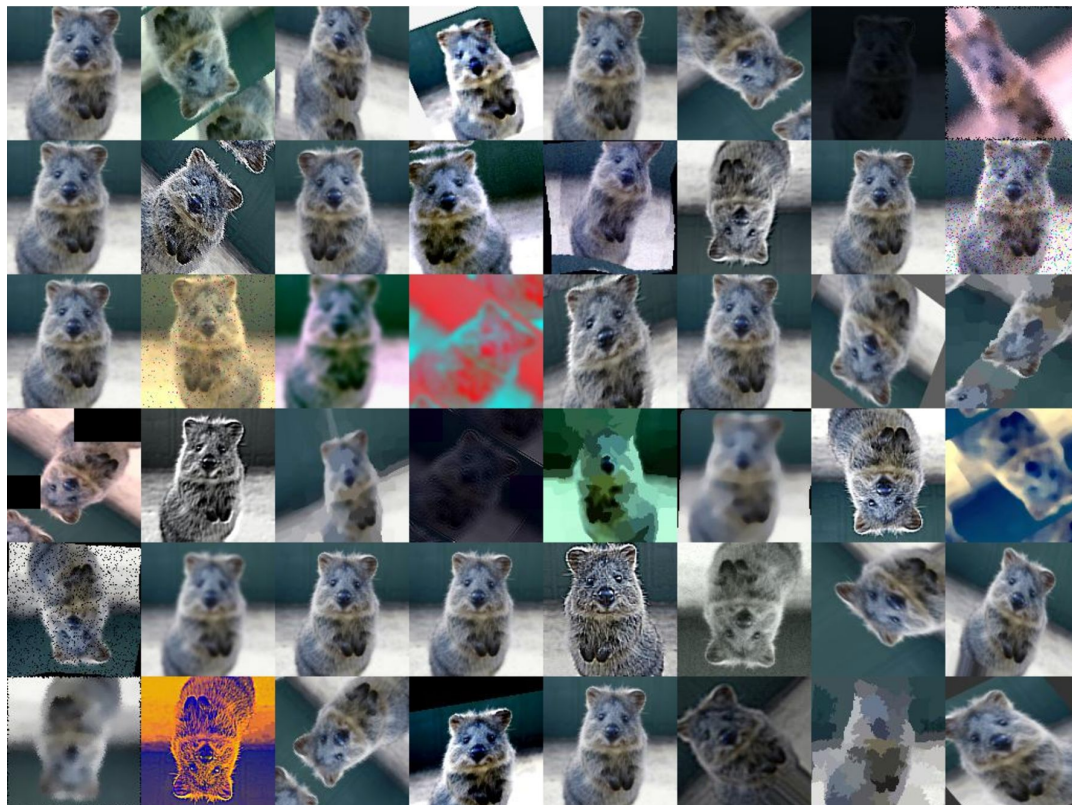
Figure 2: Inception module

ResNet

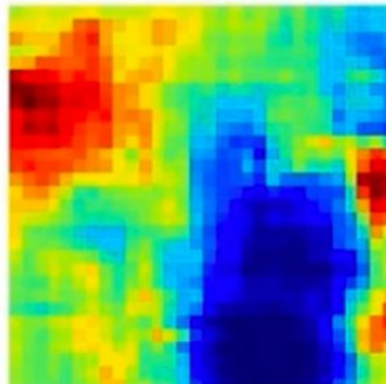
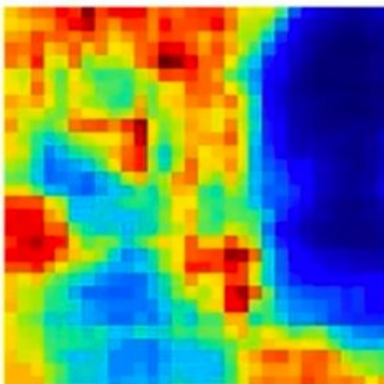
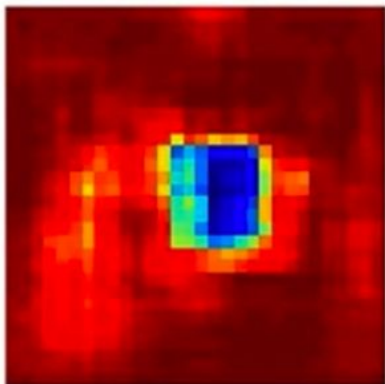


Аугментации

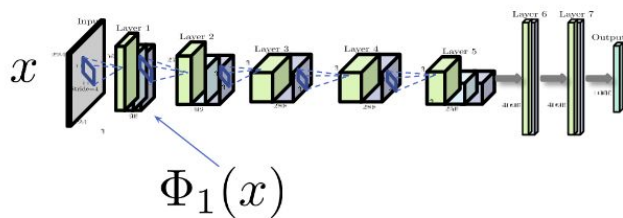
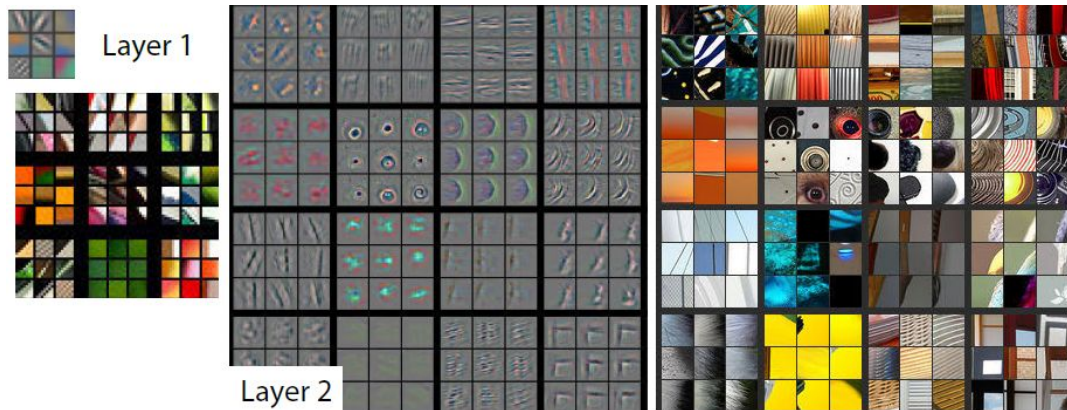
- Поворот
- Сдвиг
- Отражение
- Удаление части
- Обрезка
- Размытие
- Добавление шума
- Изменение яркости
- Изменение контраста
- Изменение порядка каналов
- etc



Окклюзия изображений



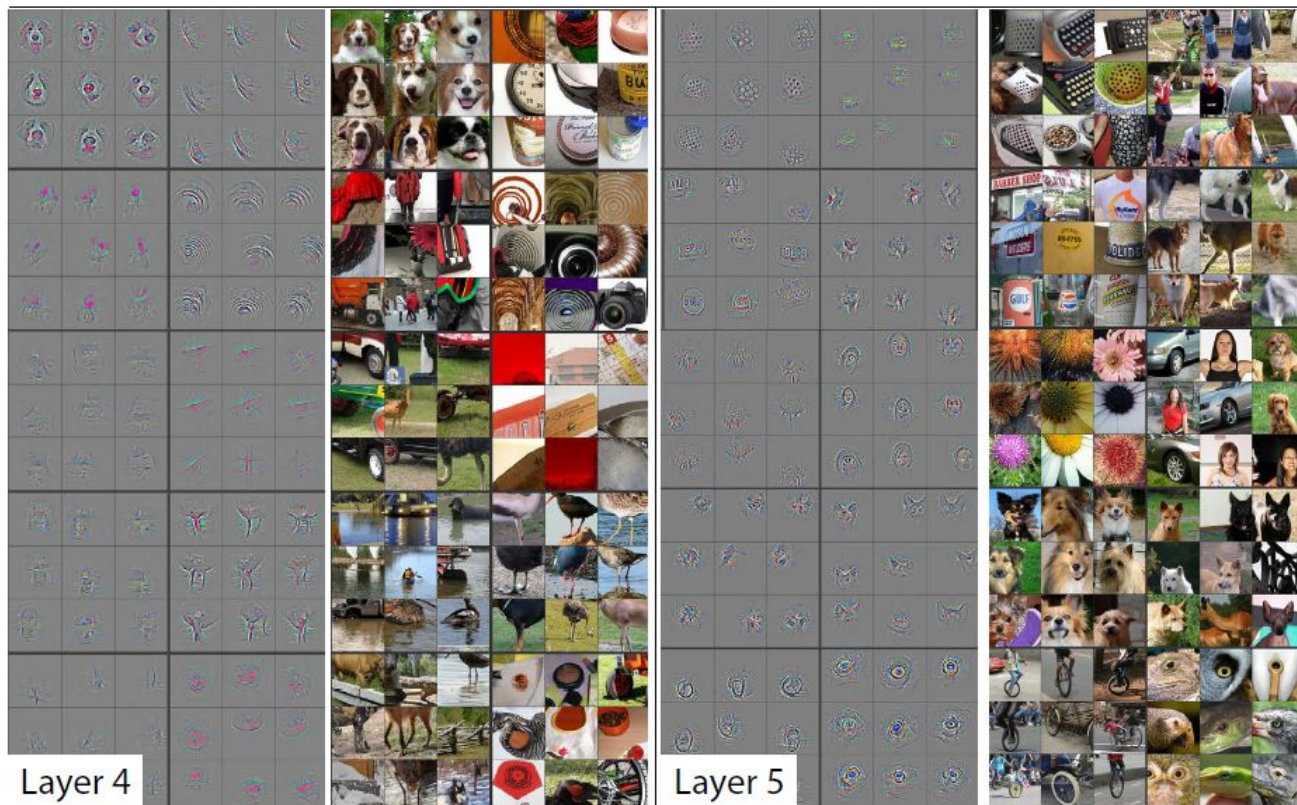
Чувствительность фильтров



Для свёртки t выделяем картинки, на которые она реагирует сильнее всего:

$$\arg \max_{x \in S} \max_{p, q} \Phi_1(x)^t[p, q]$$

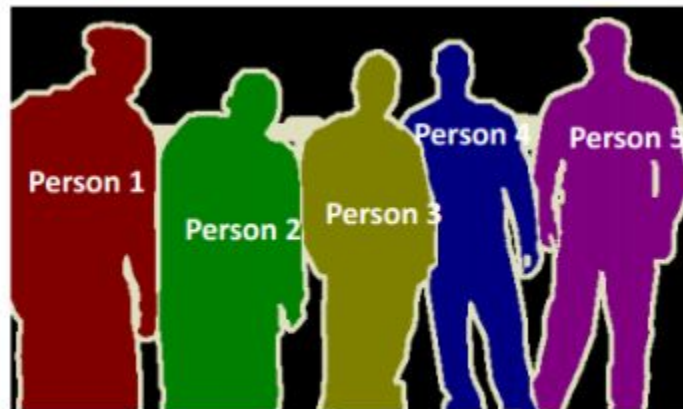
Чувствительность фильтров



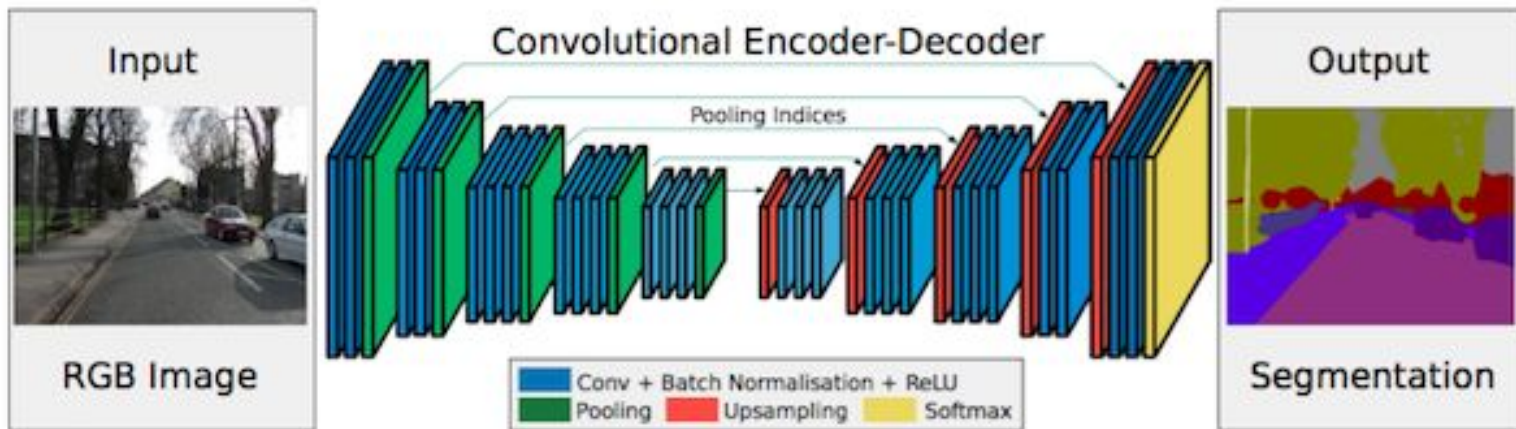
Задача сегментации изображений



Semantic Segmentation



Instance Segmentation



- Проблема: ответ должен быть близок по размеру ко входу
- Проблема: чем глубже тем лучше понимаем что изображено, но забываем контекст
- Resceptive field на последних слоях должен быть достаточно большим

Upsampling

Bed of nails

1	2
3	4

Input: 2 x 2



1	0	2	0
0	0	0	0
3	0	4	0
0	0	0	0

Output: 4 x 4

Nearest neighbour interpolation

1	2
3	4

Input: 2 x 2



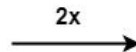
1	1	2	2
1	1	2	2
3	3	4	4
3	3	4	4

Output: 4 x 4

Bilinear Interpolation

10	20
30	40

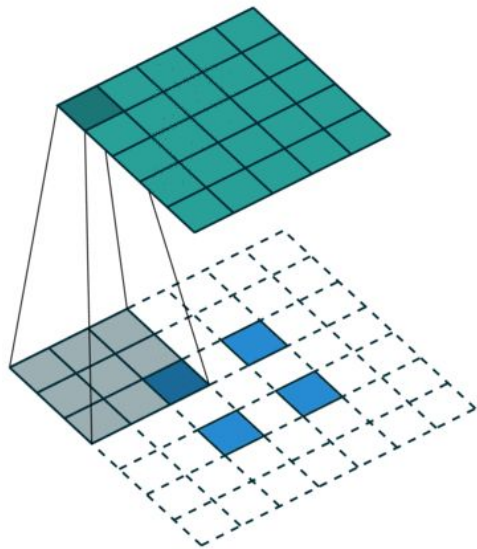
2x2



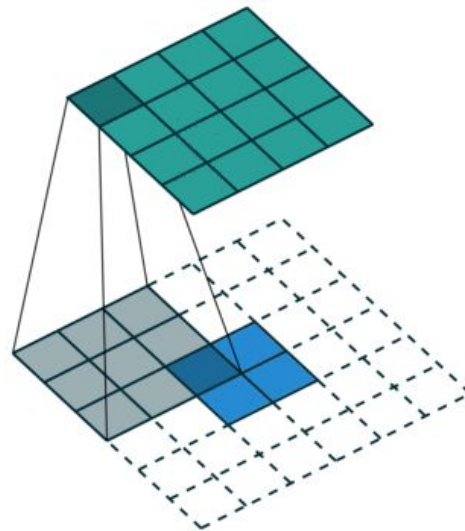
10	12	17	20
15	17	22	25
25	27	32	35
30	32	37	40

4x4

Транспонированная свертка

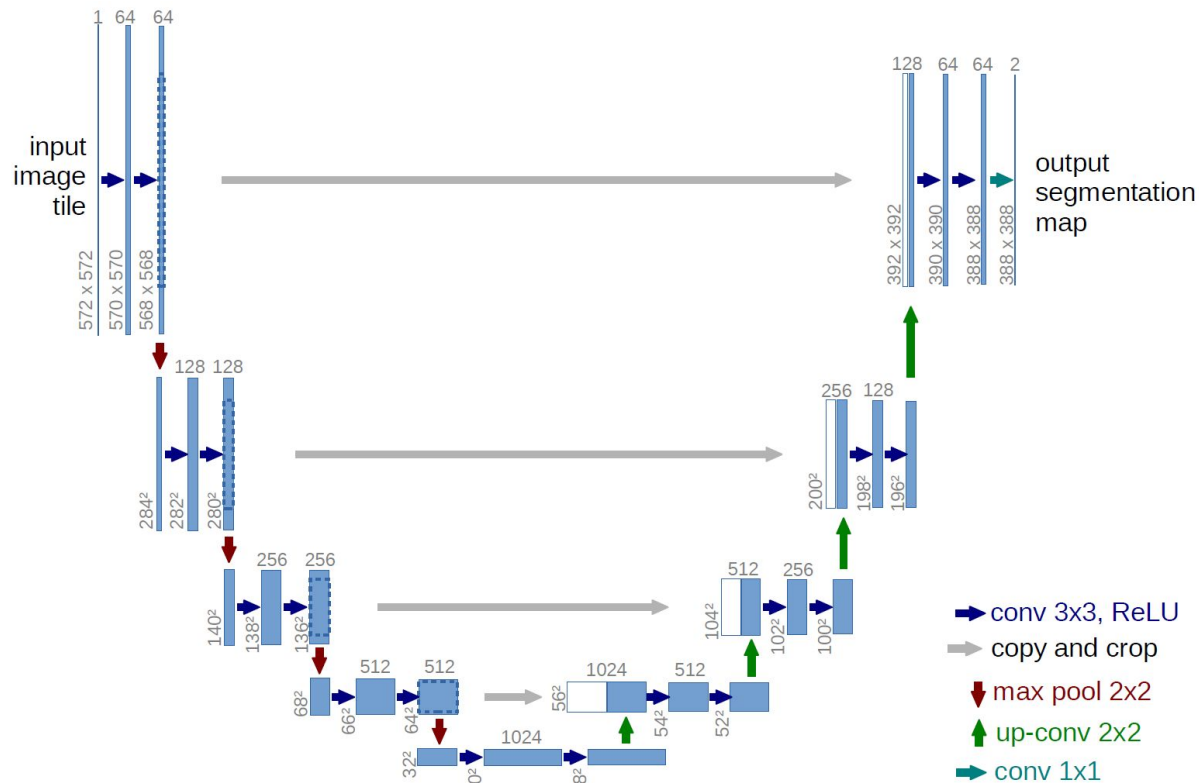


No padding
Stride=2

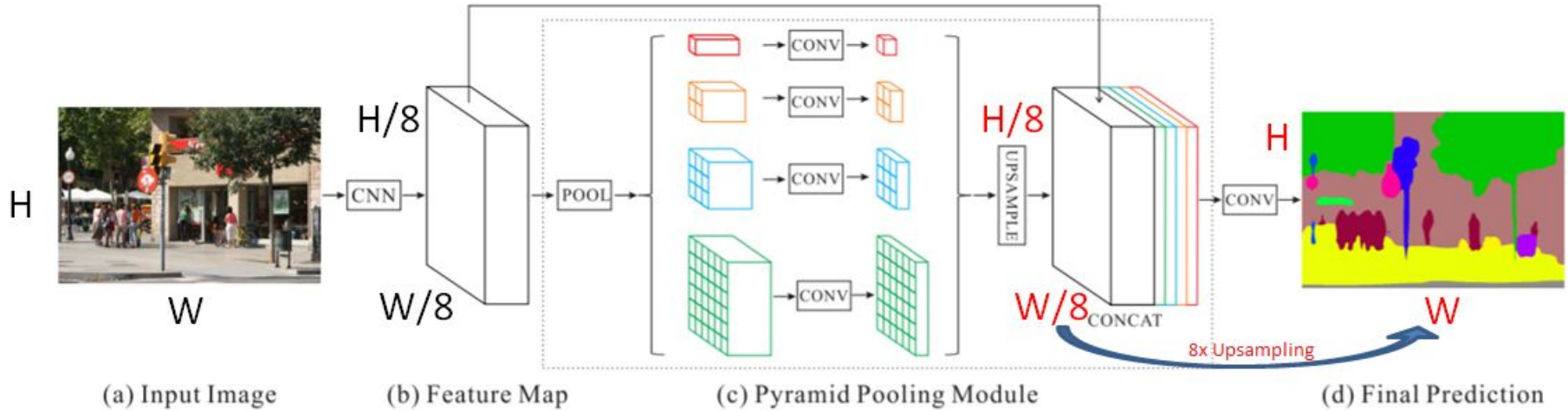


No padding
No strides

U-net

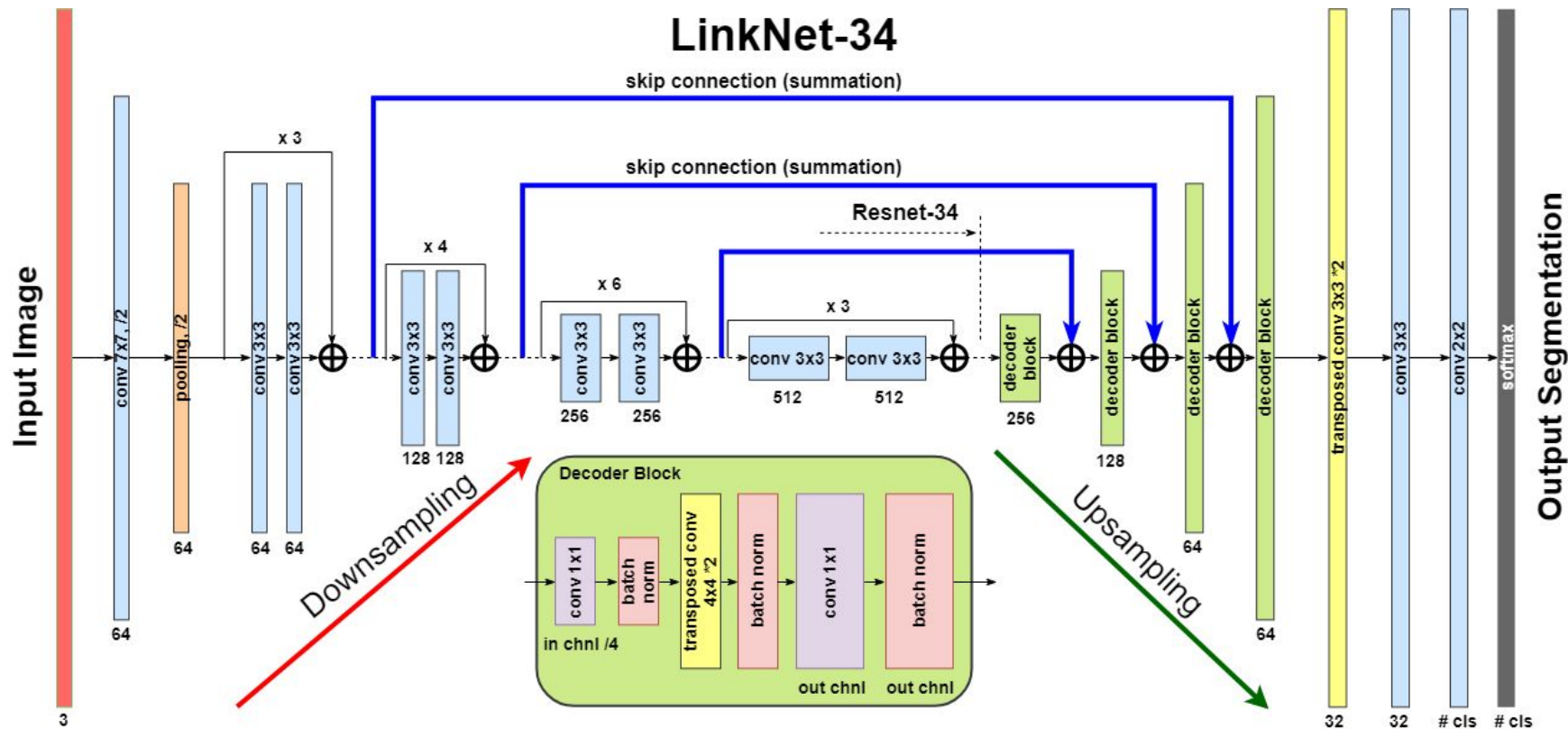


PSP-net



- Dilated convolutions
- Pyramid pooling module

LinkNet



Transfer learning

Transfer Learning

