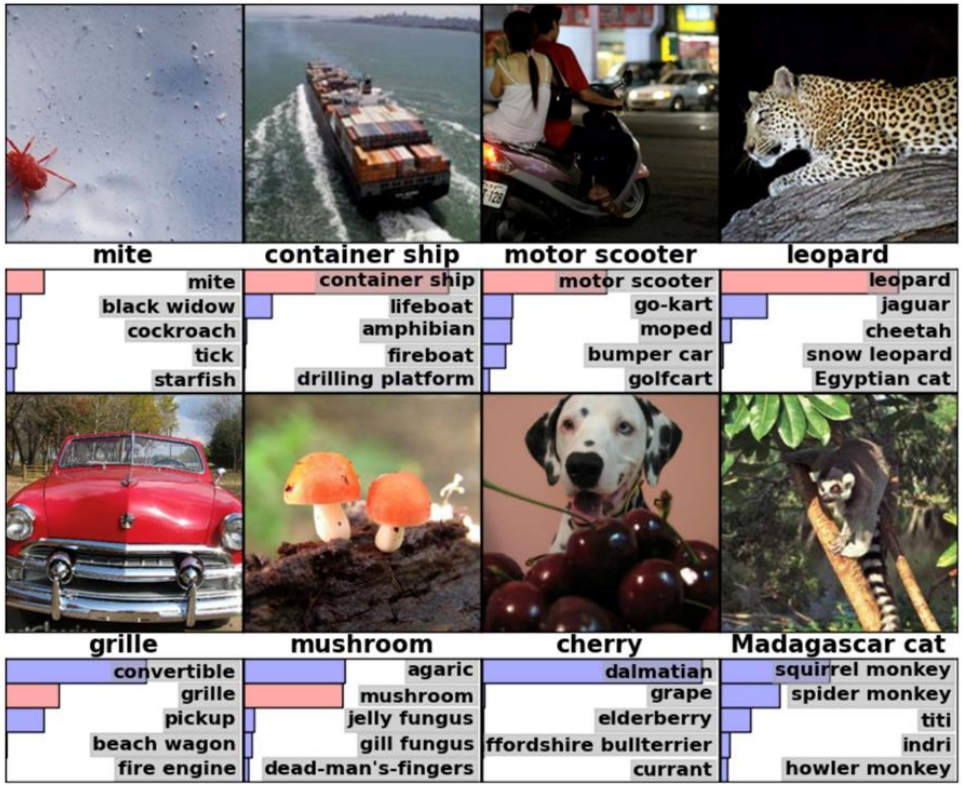
Свёртки. Свёрточные сети в Pytorch. Сегментация.

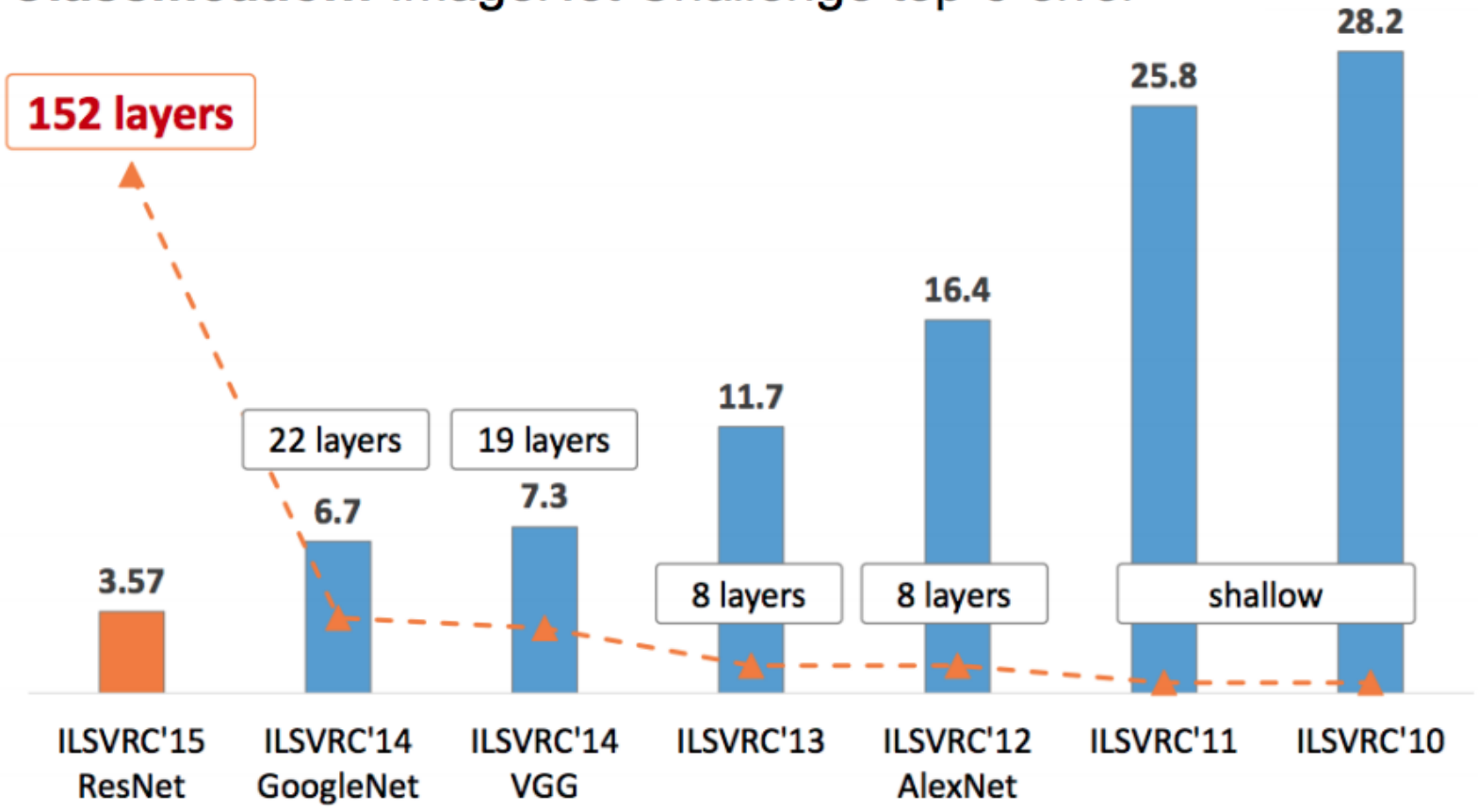
Практикум на ЭВМ для 317 группы, весна 2022
Кафедра ММП ВМК МГУ

ImageNet Challenge

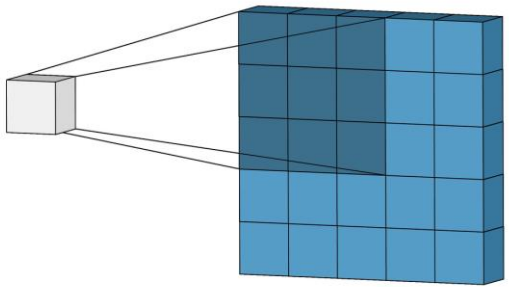


- 1,000 object classes (categories).
- Images:
  - 1.2 M train
  - 100k test.

Classification: ImageNet Challenge top-5 error



Свертка



CONV2D

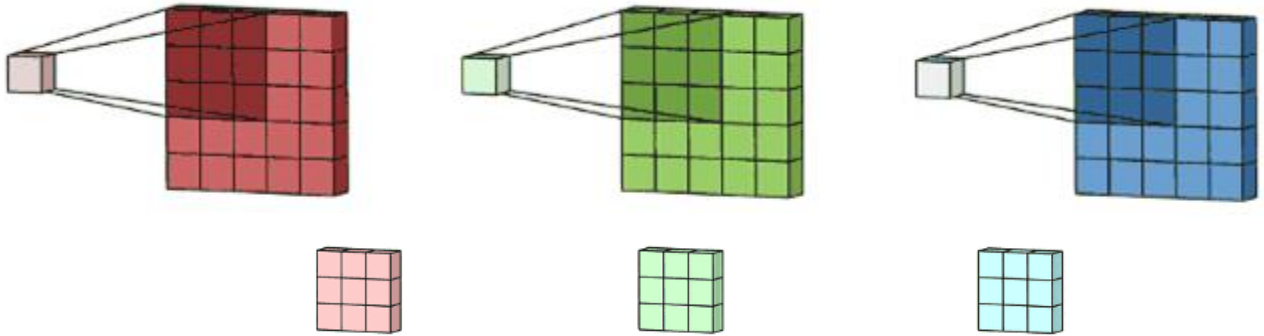
CLASS `torch.nn.Conv2d(`*`in_channels: int, out_channels: int, kernel_size: Union[T, Tuple[T, T]], stride: Union[T, Tuple[T, T]] = 1, padding: Union[T, Tuple[T, T]] = 0, dilation: Union[T, Tuple[T, T]] = 1, groups: int = 1, bias: bool = True, padding_mode: str = 'zeros'`*`)` [SOURCE]

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:

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

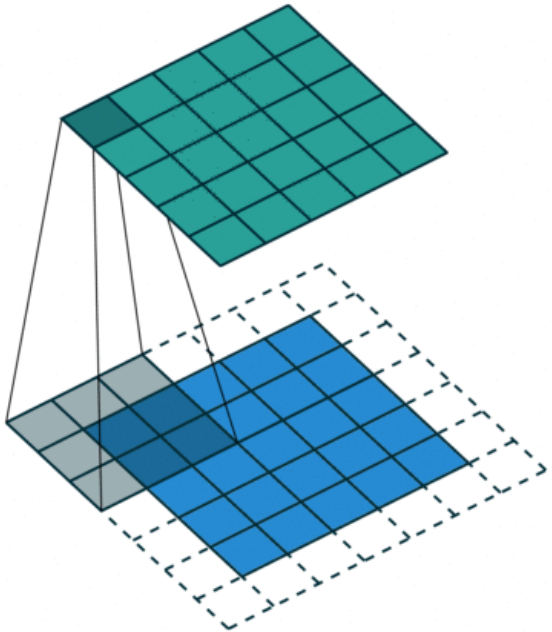
Свертка



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

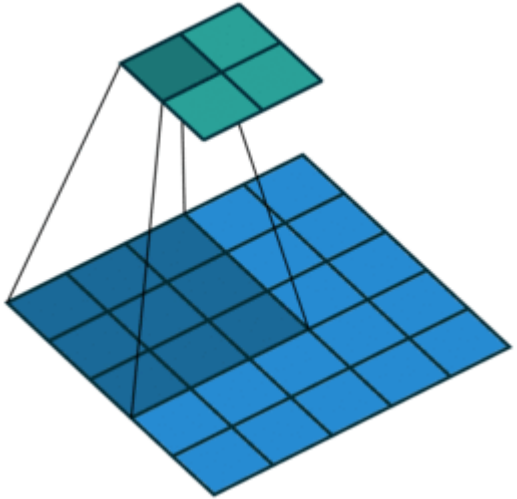
CLASS `torch.nn.Conv2d(in_channels: int, out_channels: int, kernel_size: Union[T, Tuple[T, T]], stride: Union[T, Tuple[T, T]] = 1, padding: Union[T, Tuple[T, T]] = 0, dilation: Union[T, Tuple[T, T]] = 1, groups: int = 1, bias: bool = True, padding_mode: str = 'zeros')` [SOURCE]

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

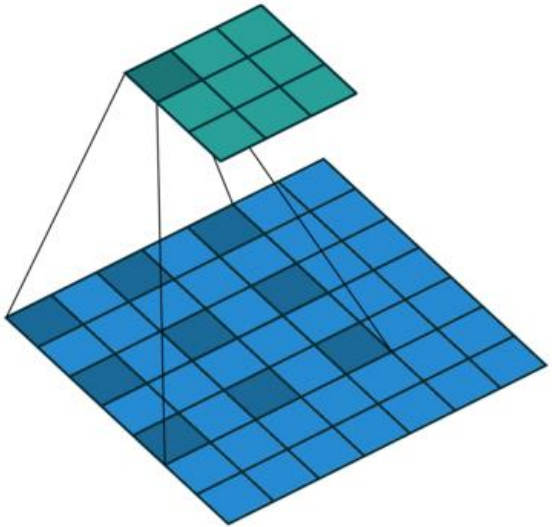
CLASS `torch.nn.Conv2d(in_channels: int, out_channels: int, kernel_size: Union[T, Tuple[T, T]], stride: Union[T, Tuple[T, T]] = 1, padding: Union[T, Tuple[T, T]] = 0, dilation: Union[T, Tuple[T, T]] = 1, groups: int = 1, bias: bool = True, padding_mode: str = 'zeros')` [SOURCE]

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

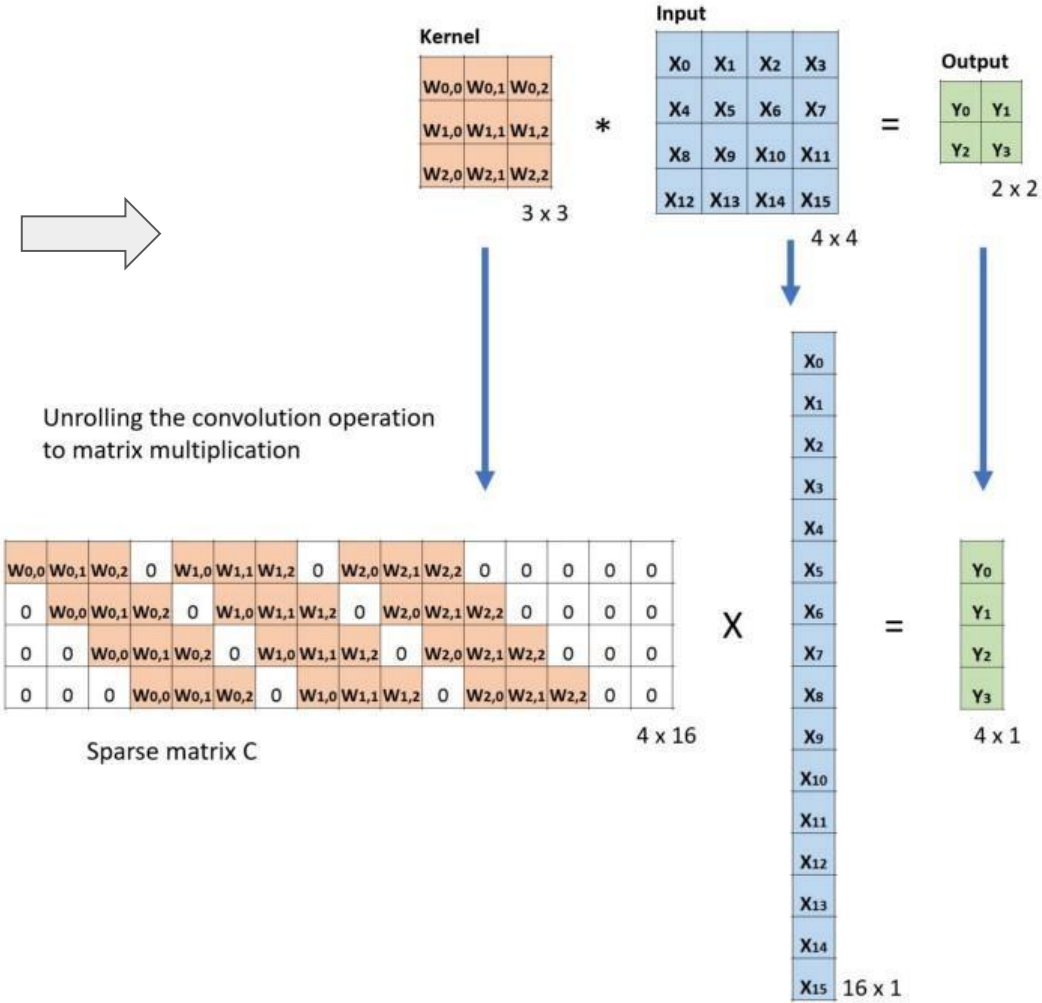
CLASS `torch.nn.Conv2d(in_channels: int, out_channels: int, kernel_size: Union[T, Tuple[T, T]], stride: Union[T, Tuple[T, T]] = 1, padding: Union[T, Tuple[T, T]] = 0, dilation: Union[T, Tuple[T, T]] = 1, groups: int = 1, bias: bool = True, padding_mode: str = 'zeros')` [SOURCE]

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

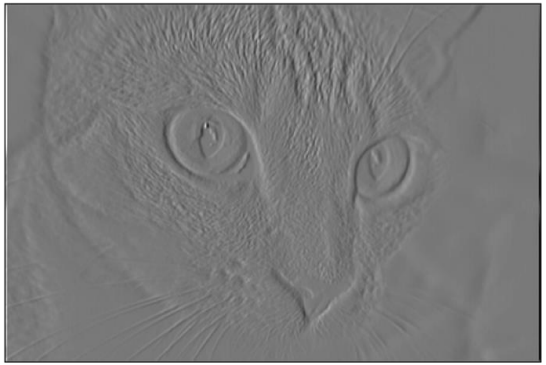


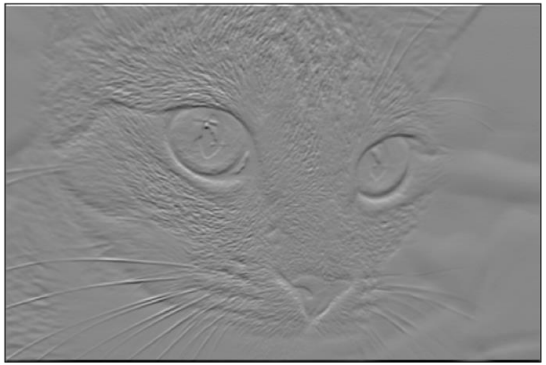
Примеры свертки

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

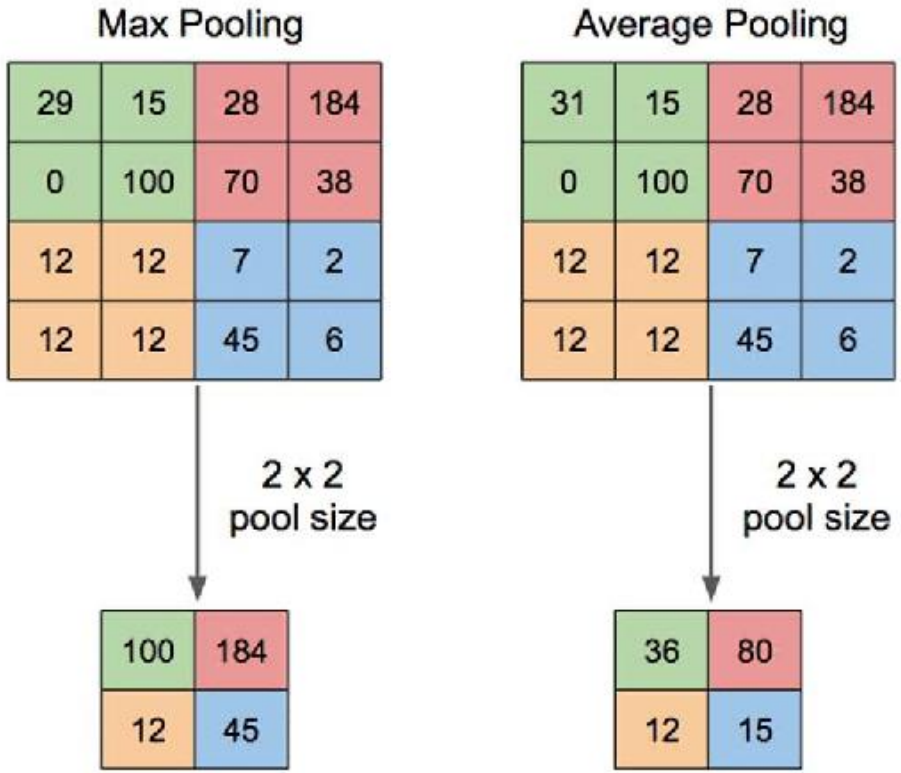




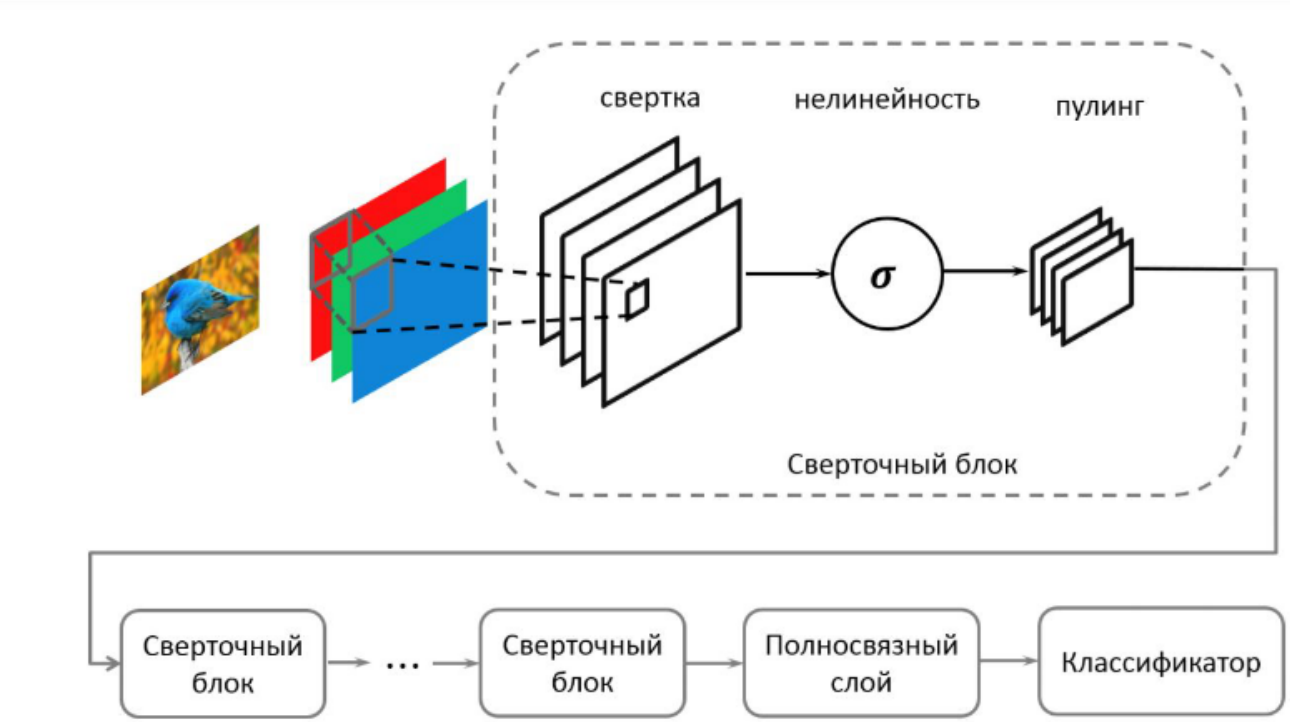
-1	0	1
-2	0	2
-1	0	1

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

Max pooling and average pooling



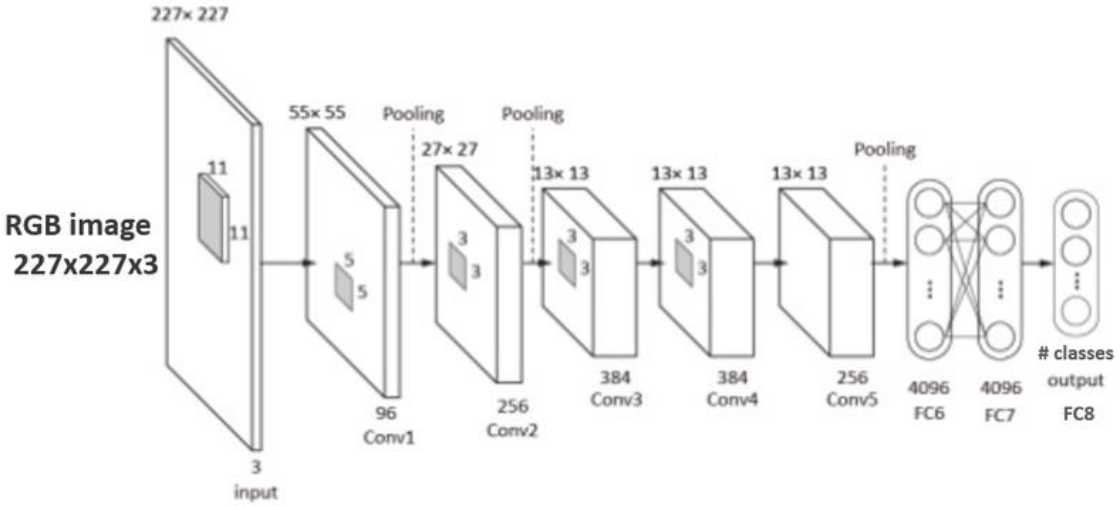
Простая схема сверточной сети



Известные архитектуры сверточных сетей для классификации

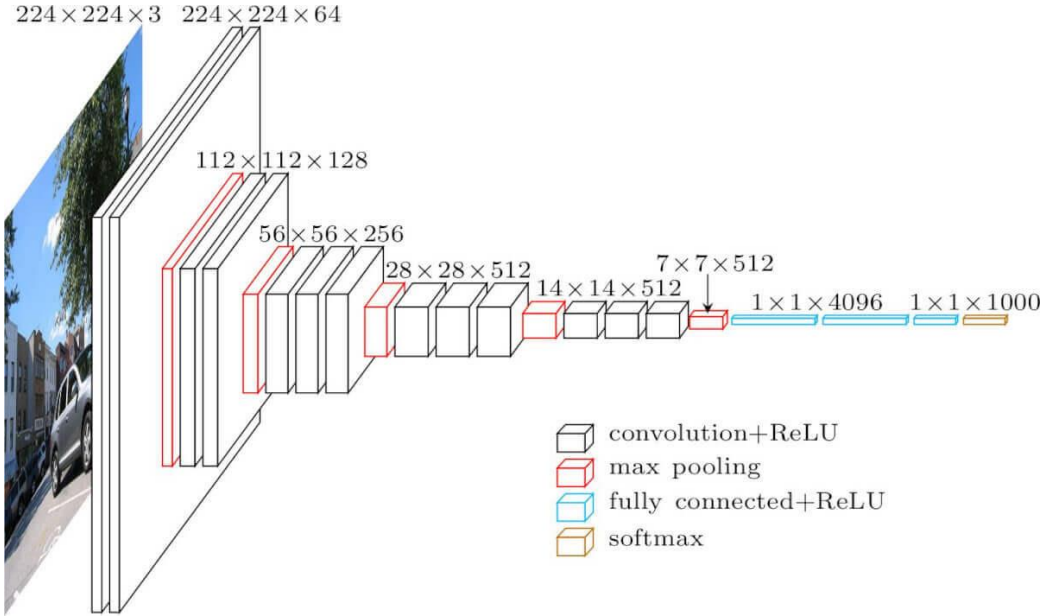
- AlexNet
- VGG
- Inception
- ResNet
- https://pytorch.org/vision/0.8/models.html

AlexNet



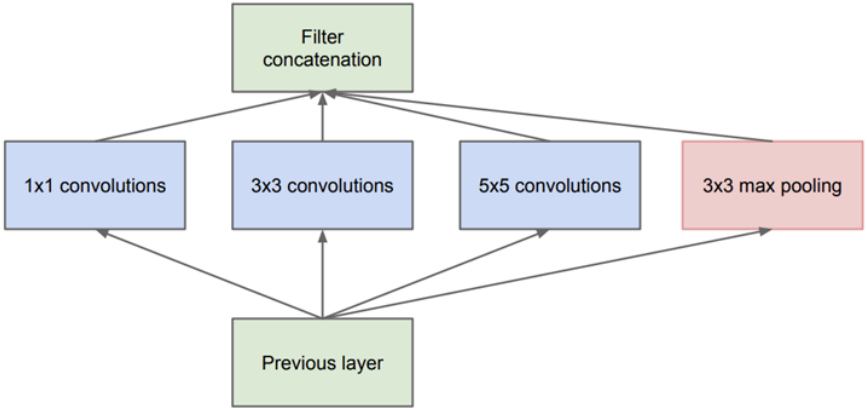
- ReLU
- Dropout

VGG

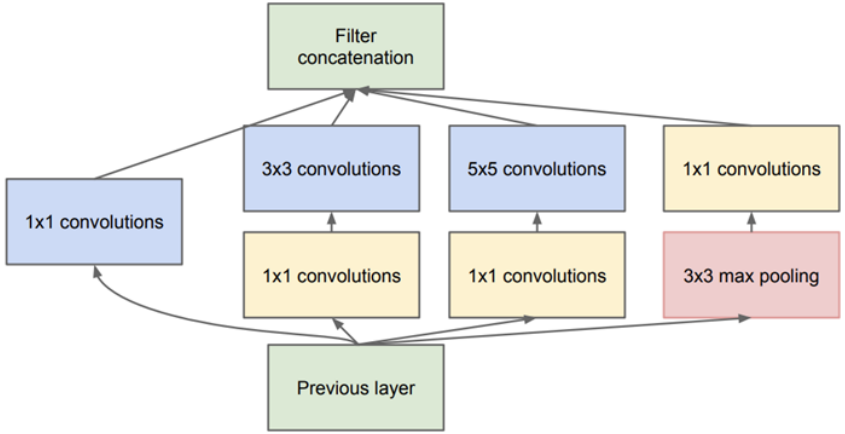


- Design of deeper networks (roughly twice as deep as AlexNet)

Inception



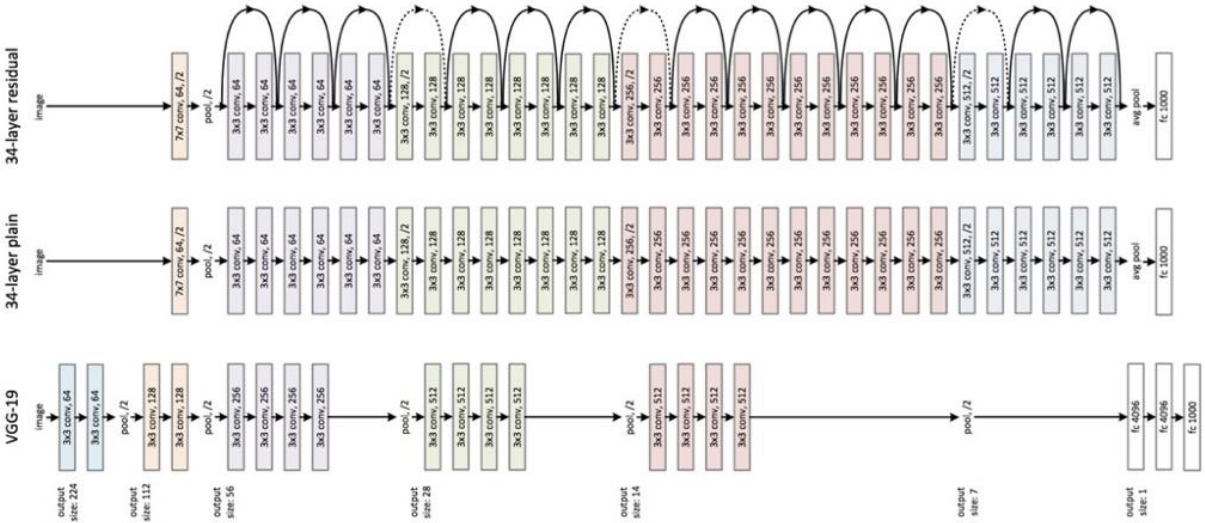
(a) Inception module, naïve version

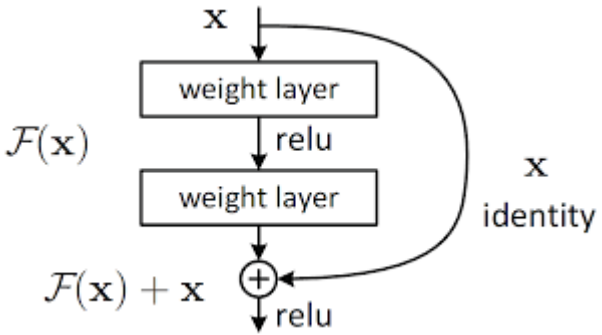


(b) Inception module with dimension reductions

Figure 2: Inception module

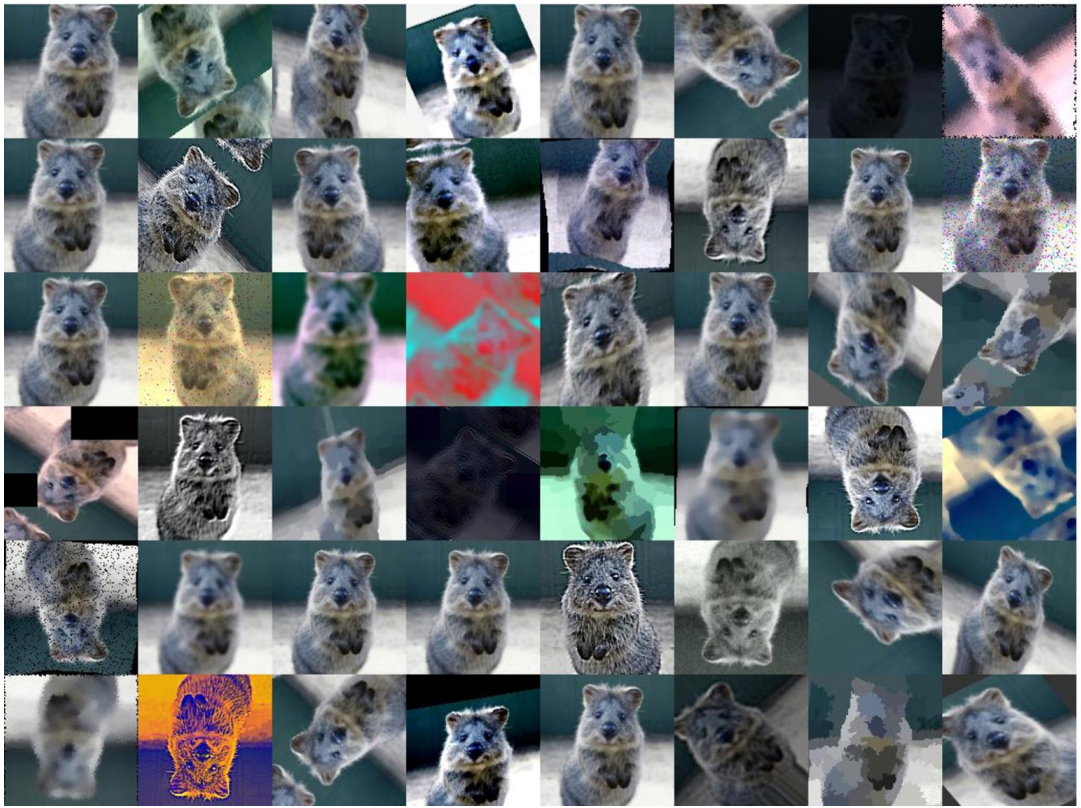
ResNet


34-layer residual
34-layer plain
VGG-19
image
7x7 conv, 64, /2
pool, /2
3x3 conv, 64
3x3 conv, 128, /2
3x3 conv, 256, /2
3x3 conv, 512, /2
avg pool
fc 1000
3x3 conv, 128
3x3 conv, 256
3x3 conv, 512
fc 4096
output size: 224
output size: 112
output size: 56
output size: 28
output size: 14
output size: 7
output size: 1

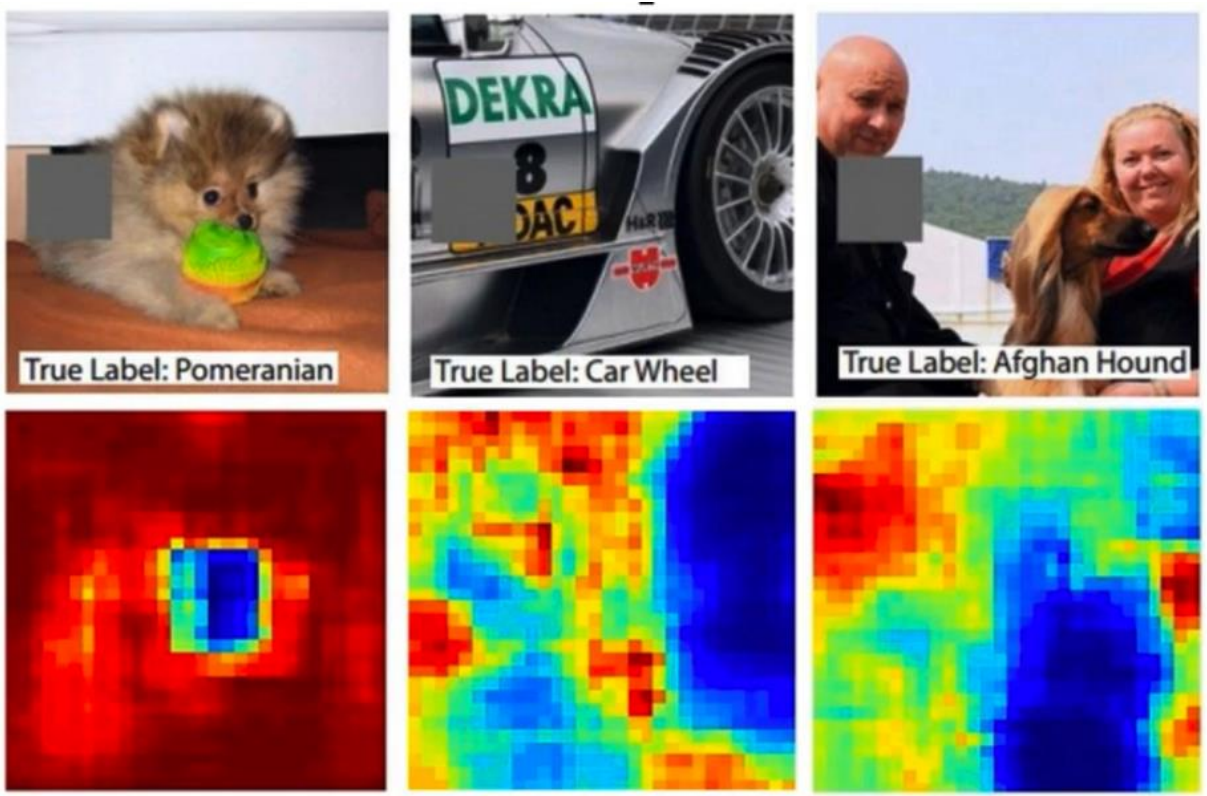

x
weight layer
relu
weight layer
$\mathcal{F}(\mathbf{x})$
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu
x
identity

Аугментации

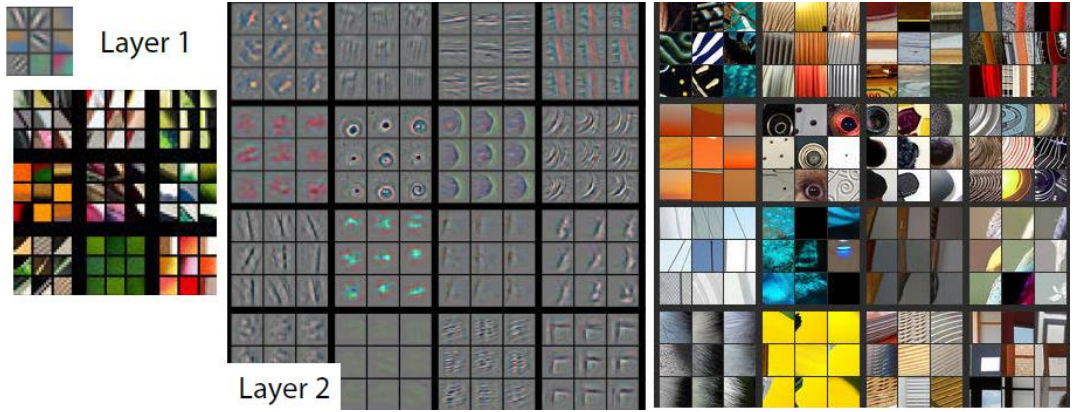
- Поворот
- Сдвиг
- Отражение
- Удаление части
- Обрезка
- Размытие
- Добавление шума
- Изменение яркости
- Изменение контраста
- Изменение порядка каналов
- etc

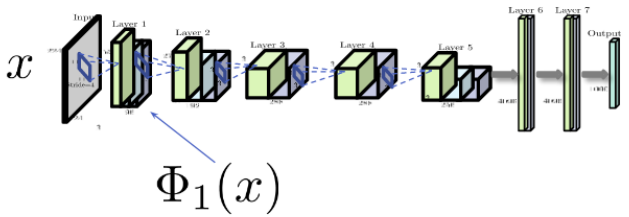


Окклюзия изображений



Чувствительность фильтров

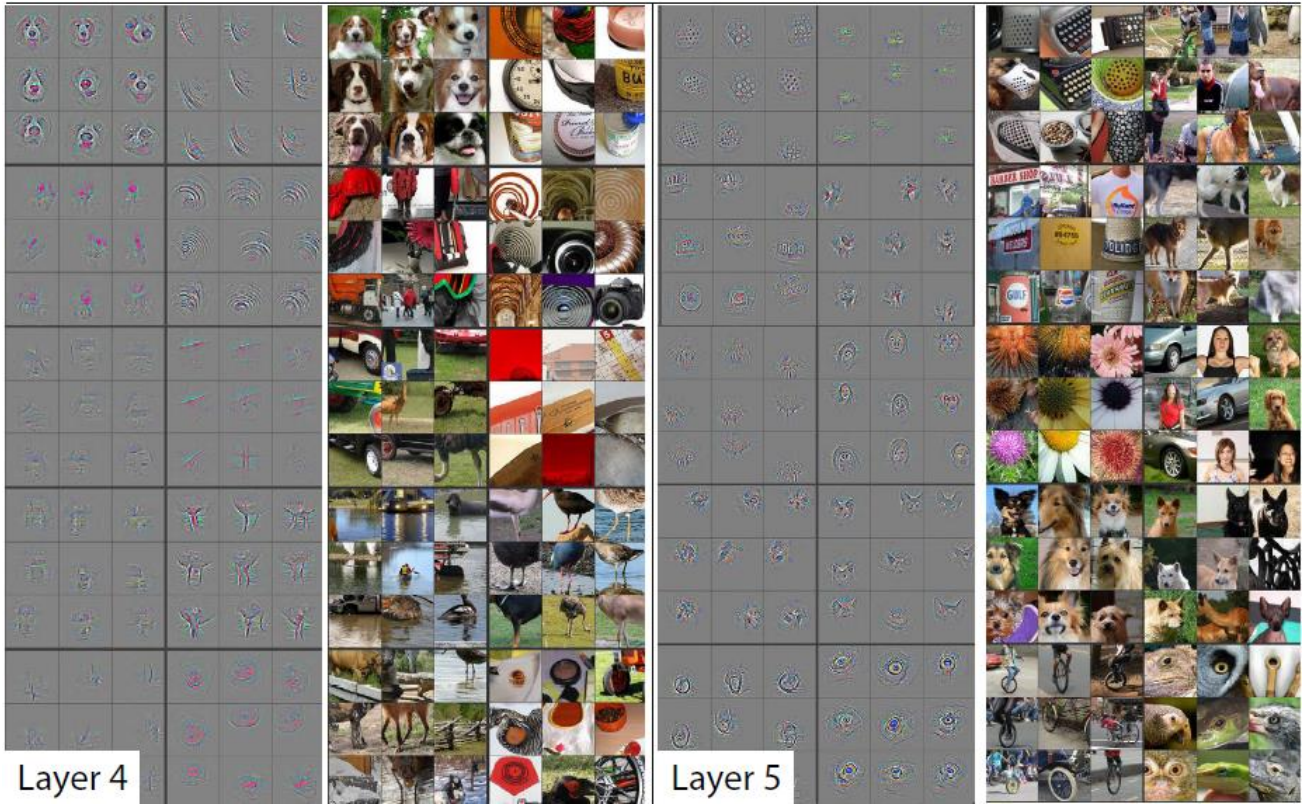




Для свёртки t выделяем картинки, на которые она реагирует сильнее всего:

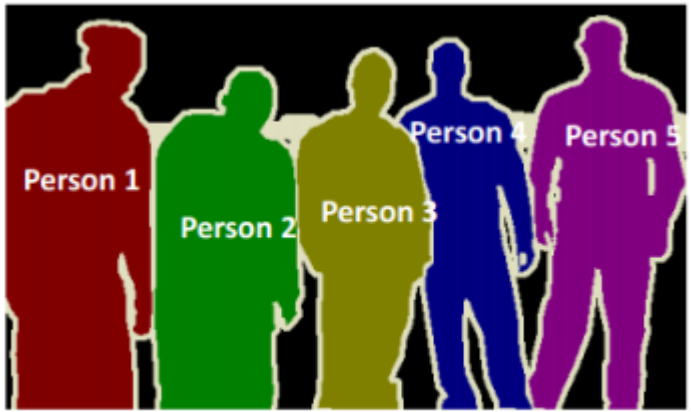
$$\arg\max_{x \in S} \max_{p,q} \Phi_1(x)^t[p,q]$$

Чувствительность фильтров

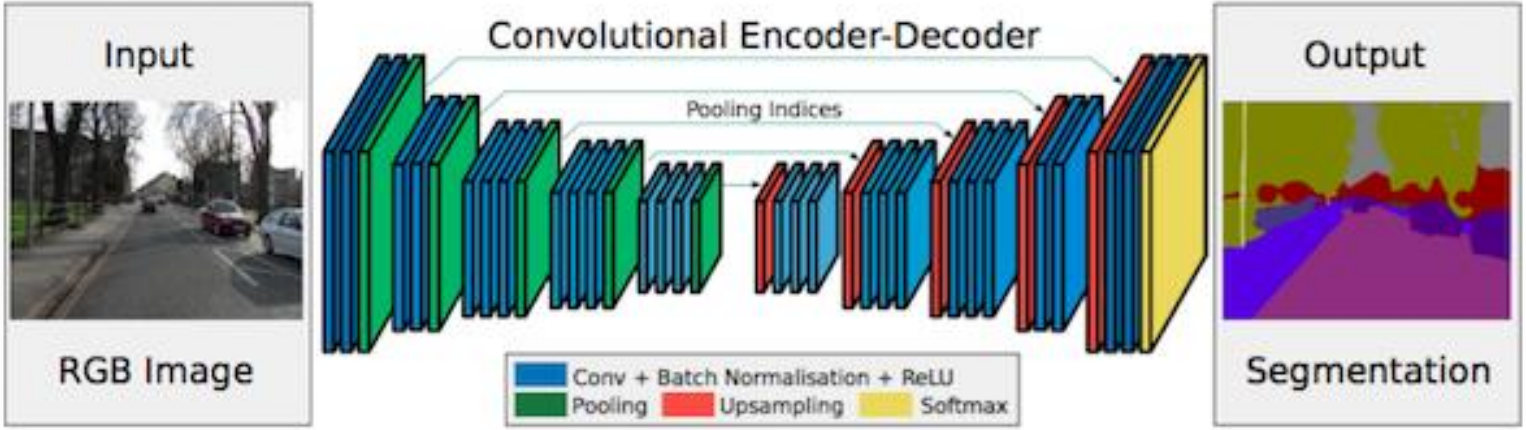


Задача сегментации изображений

Semantic Segmentation



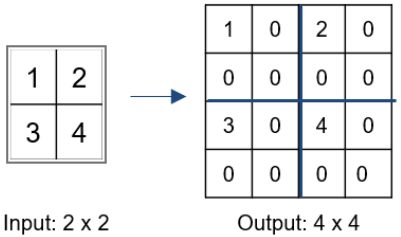
Instance Segmentation



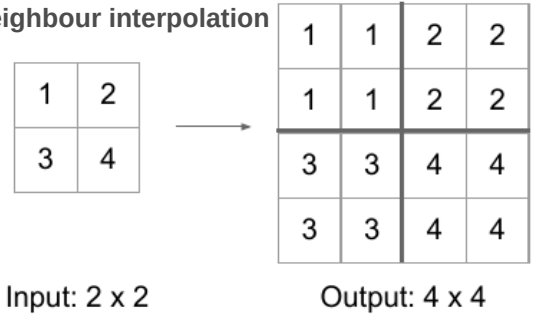
- Проблема: ответ должен быть близок по размеру ко входу
- Проблема: чем глубже тем лучше понимаем что изображено, но забываем контекст
- Receptive field на последних слоях должен быть достаточно большим

Upsampling

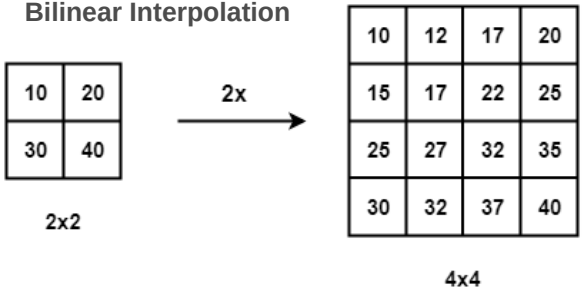
Bed of nails



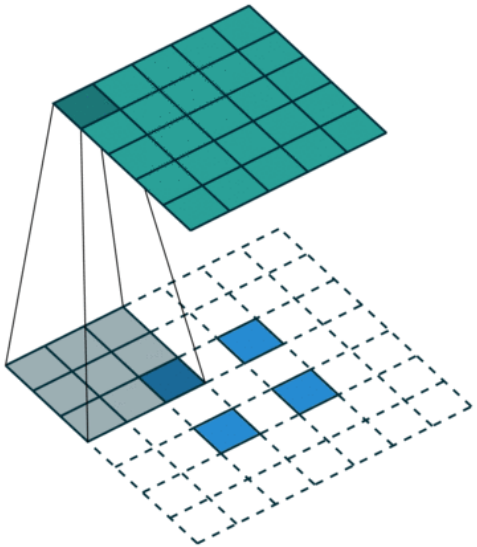
Nearest neighbour interpolation



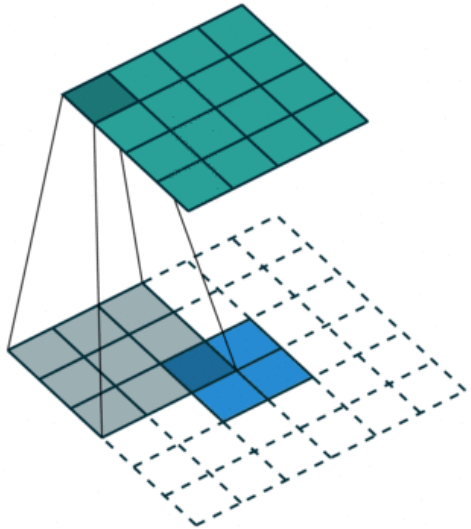
Bilinear Interpolation



Транспонированная свертка

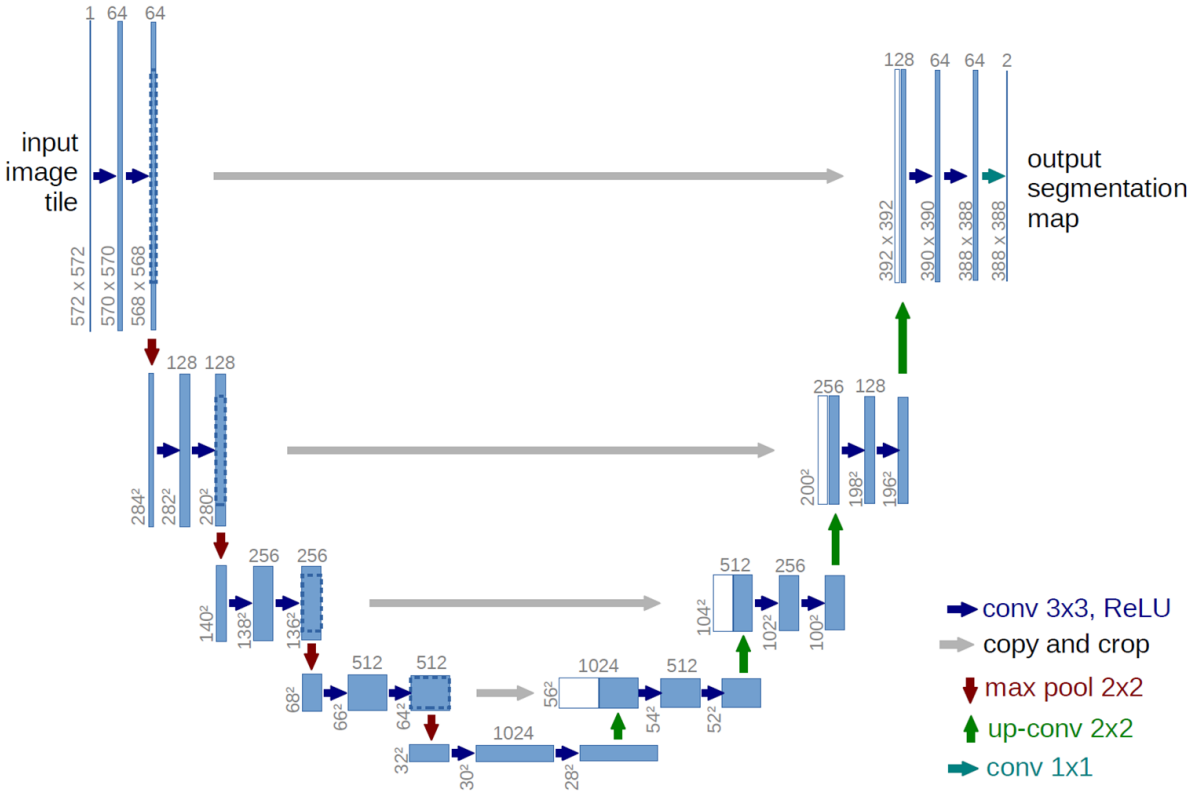


No padding
Stride=2

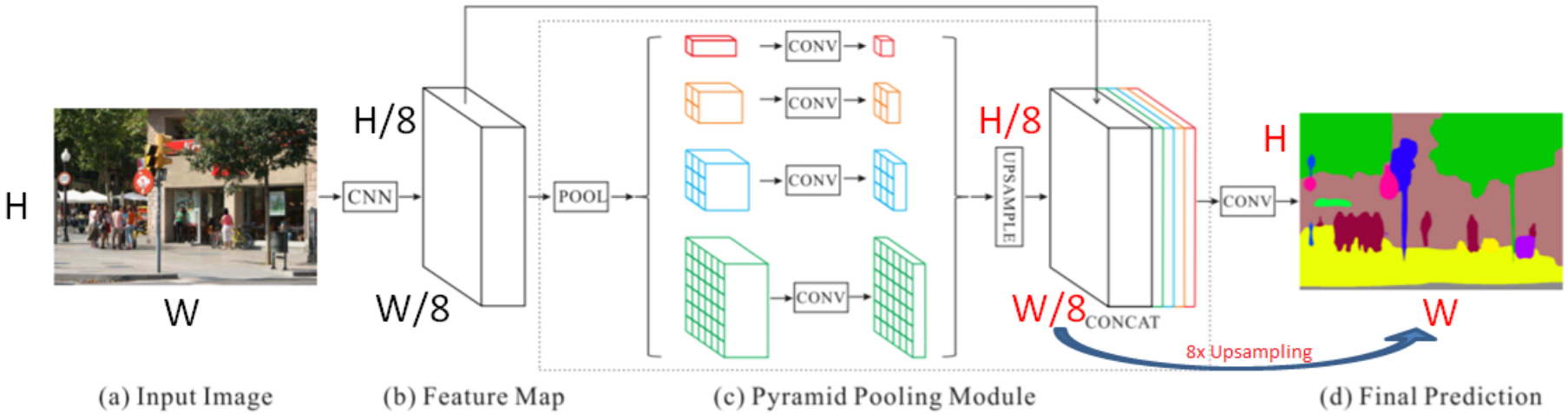


No padding
No strides

U-net

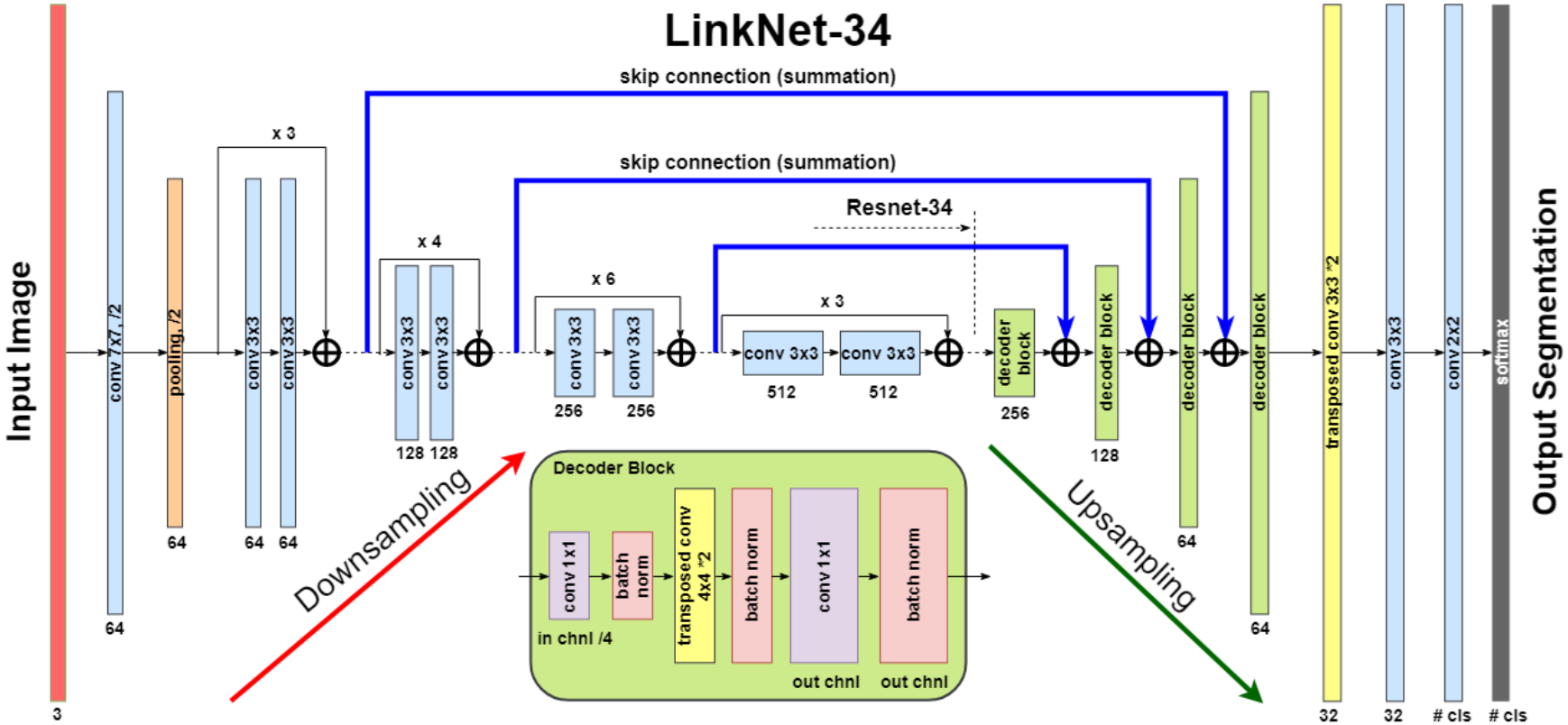

input image tile
output segmentation map
conv 3x3, ReLU
copy and crop
max pool 2x2
up-conv 2x2
conv 1x1

PSP-net



- Dilated convolutions
- Pyramid pooling module

LinkNet


LinkNet-34
skip connection (summation)
skip connection (summation)
Resnet-34
Input Image
Output Segmentation
conv 7x7, /2
pooling, /2
x 3
conv 3x3
conv 3x3
x 4
conv 3x3
conv 3x3
x 6
conv 3x3
conv 3x3
x 3
conv 3x3
conv 3x3
decoder block
decoder block
decoder block
decoder block
transposed conv 3x3 *2
conv 3x3
conv 2x2
softmax
64
64
64
64
128
128
256
256
512
512
256
128
64
64
32
32
cls
cls
3
Downsampling
Upsampling
Decoder Block
conv 1x1
batch norm
transposed conv 4x4 *2
batch norm
conv 1x1
batch norm
in chnl /4
out chnl
out chnl

Transfer learning