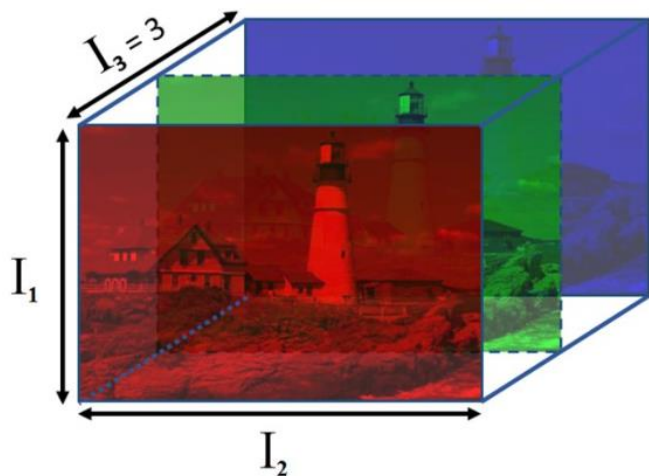


Глубокое обучение для обработки изображений

Лекция 3

Входные данные

Чаще всего изображения представляют собой тензор размерности (H, W, C) . $C = 3$ для RGB, $C = 1$ для grayscale. Для RGB формата значения в каждом канале C_i лежат в диапазоне от 0 до 255, при этом $(0, 0, 0)$ соответствует черному цвету, а $(255, 255, 255)$ – белому.

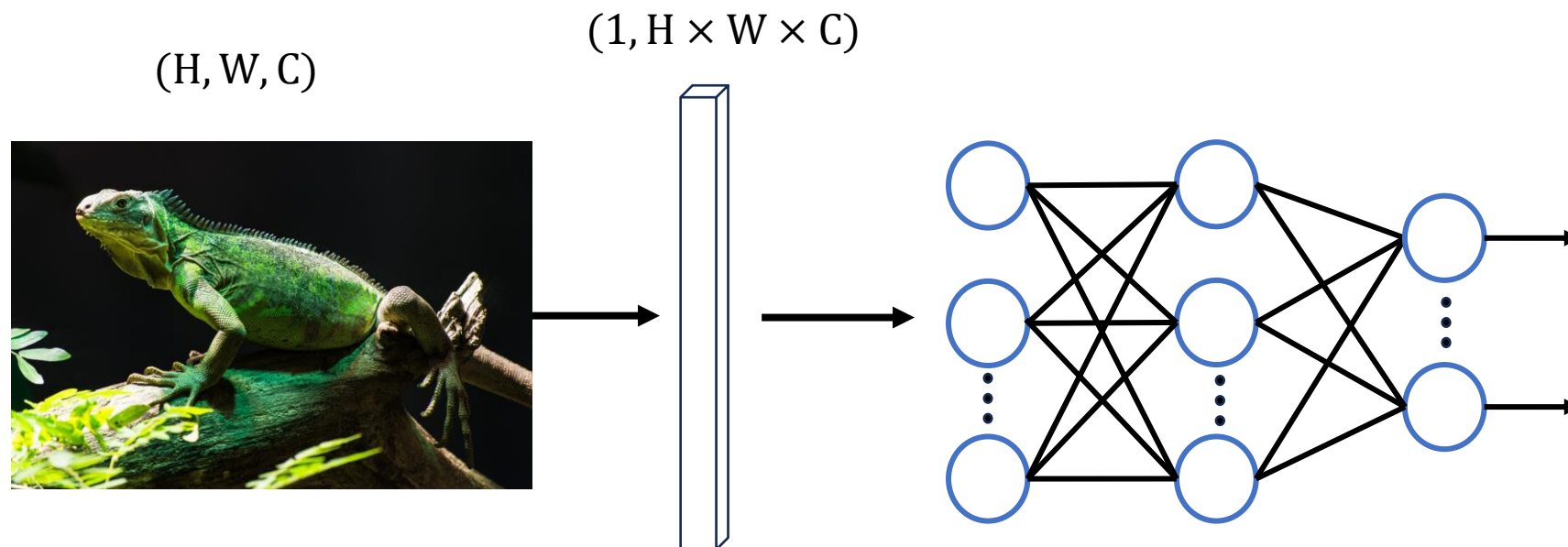


Представление
изображения в виде
тензора 3-го порядка



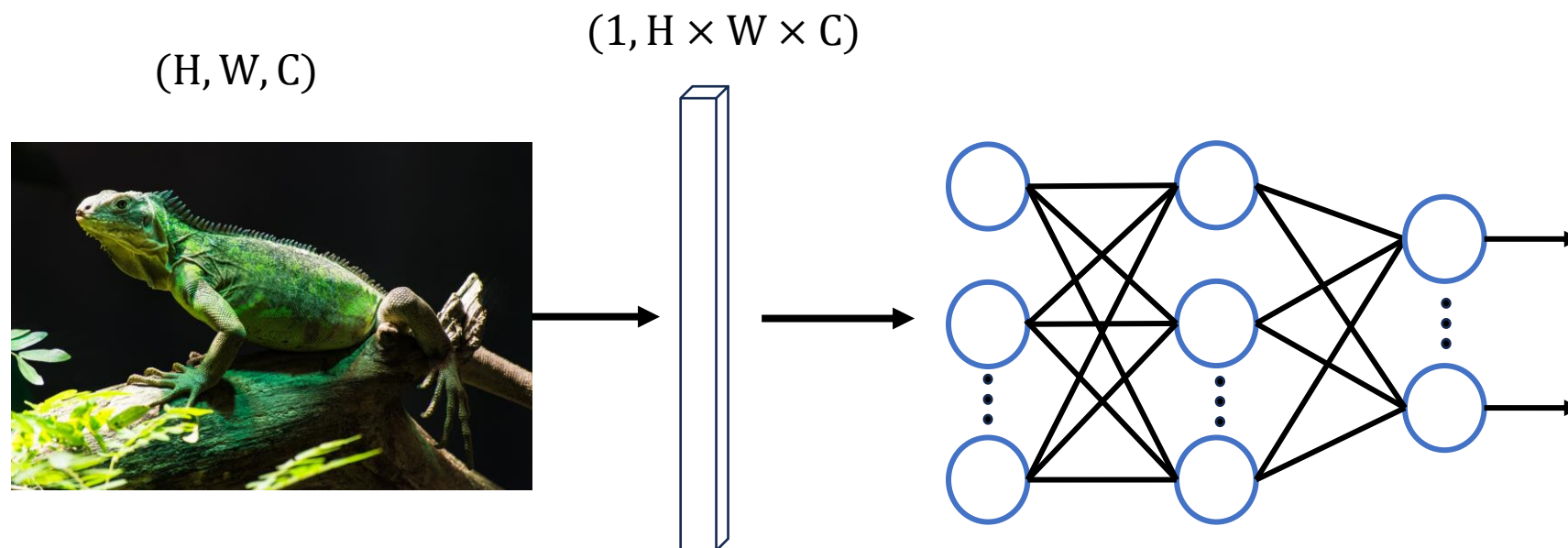
RGB (слева) и grayscale
(справа) изображения

Использование полносвязных сетей



Мы можем, например, преобразовать картинку в вектор и подать на вход нейронной сети

Использование полносвязных сетей

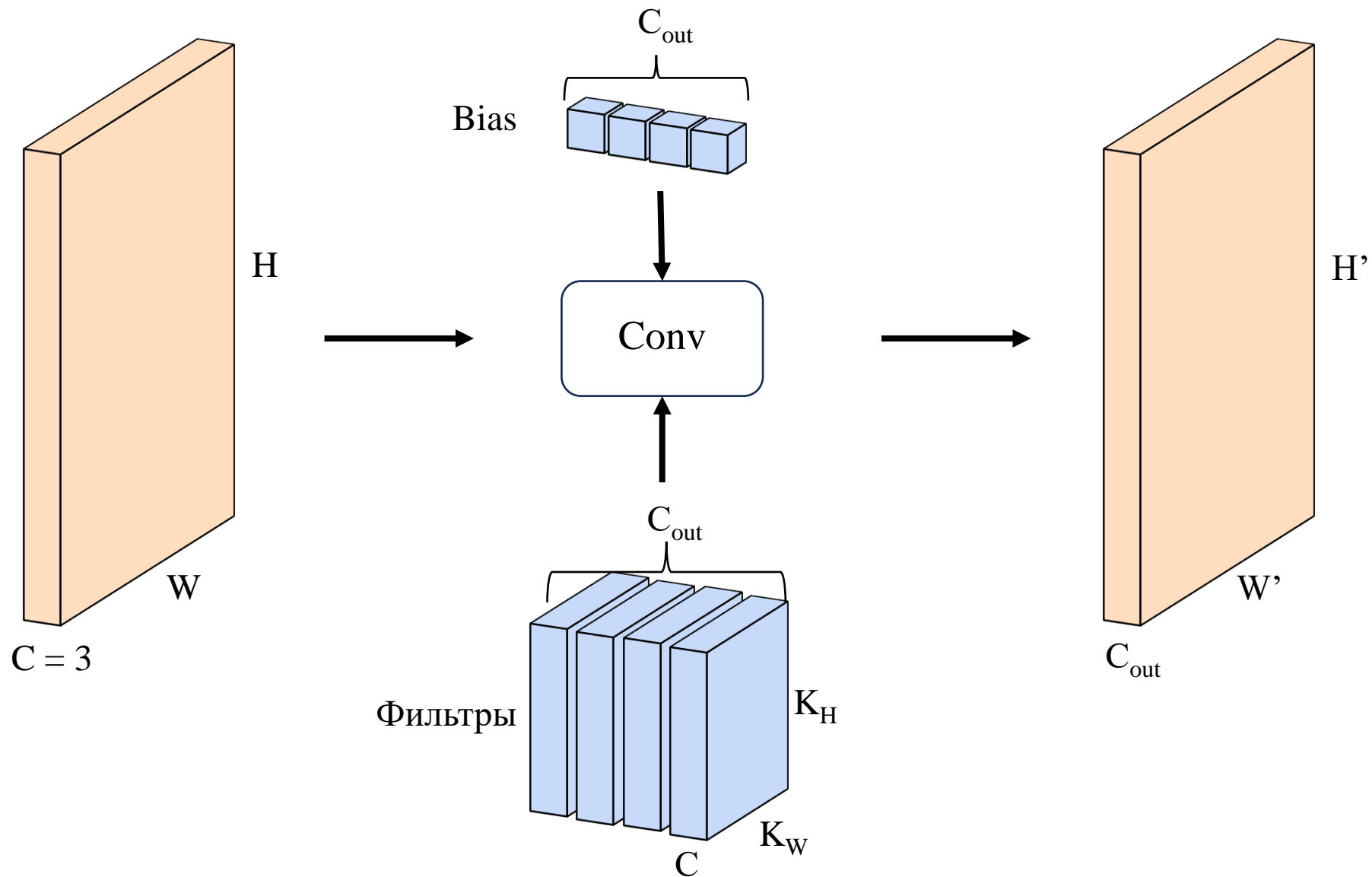


Мы можем, например, преобразовать картинку в вектор и подать на вход нейронной сети

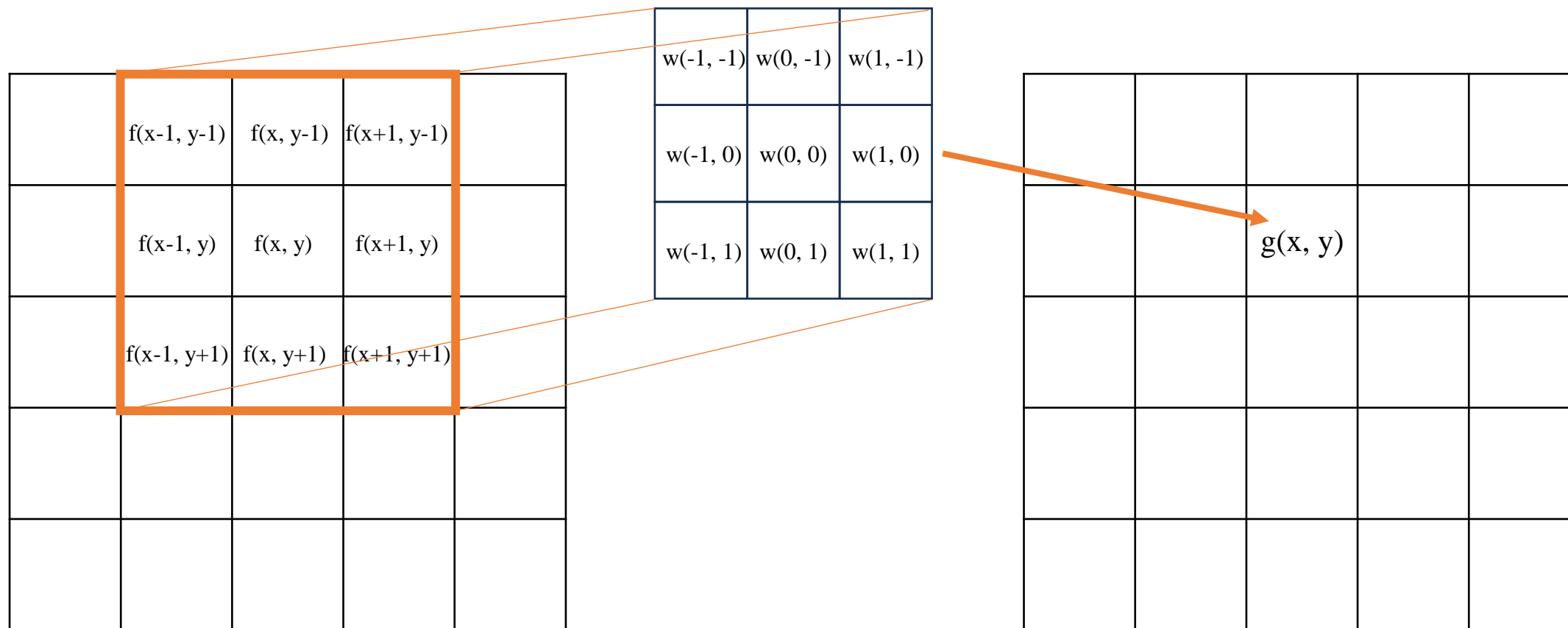
Однако, такой подход обладает следующими недостатками:

- Очень много обучаемых параметров. Размерность данных уже в первом слое будет равна $(H \times W \times C \times X_1)$
- Использование такой архитектуры никак не учитывает структуры входного изображения

Свёрточный слой

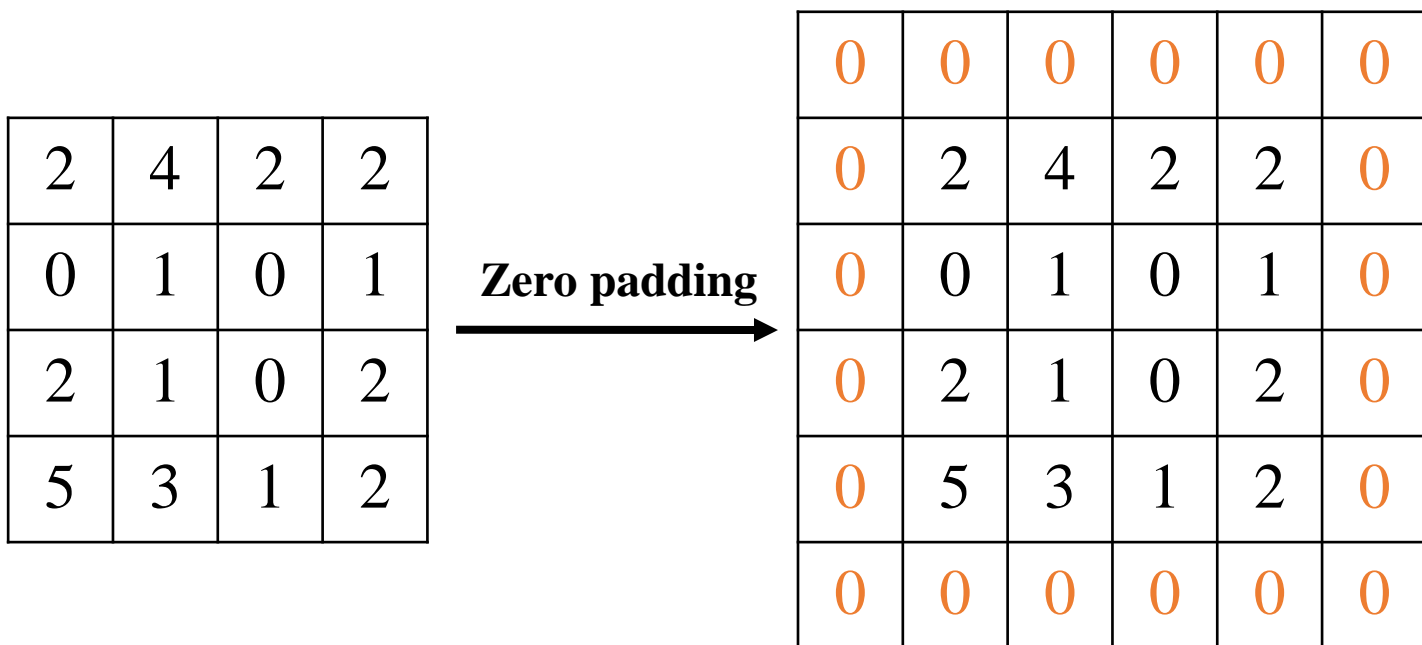


Свёрточный слой



$$g(x, y, c_{out}) = \sum_{c_{in}} \sum_{m, n} f(x + m, y + n, c_{in}) w(m, n, c_{in}, c_{out}) + b(c_{out})$$

Padding



Если найти описанным образом отфильтрованное изображение, то оно будет меньшего размера, нежели исходное. Чтобы сохранить размер изображения, его дополняют пикселями по краям, например, нулями или иными значениями

Пример картинки, дополненной нулями на краях

Pooling

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

Max pool, 2x2 filter



2	2
5	4

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

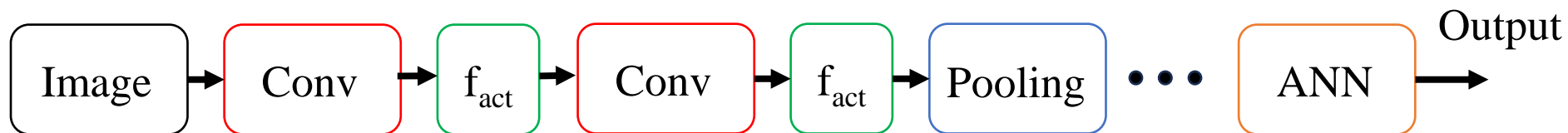
Average pool, 2x2 filter



0.25	0.75
1.25	1.75

Слой **pooling** используется для уменьшения размеров промежуточных представлений в свёрточной нейронной сети

Пример архитектуры CNN

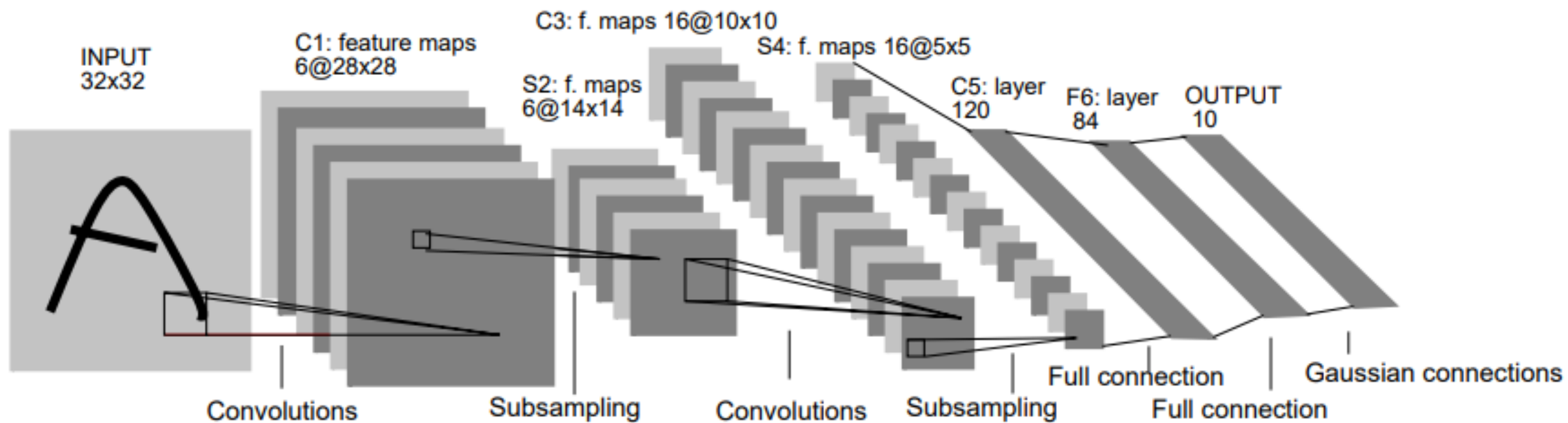


Архитектуры CNN бывают разные, но некоторые общие закономерности используются почти везде. Так, за **свёрточным** слоем идет **функция активации**; также периодически используется **pooling** слой.

В конце почти всегда присутствует **полносвязная сеть (fully connected layer)**, с помощью которой мы и получаем предсказания (в некоторых задачах, например, в сегментации, FC слой отсутствует)

LeNet (1998)

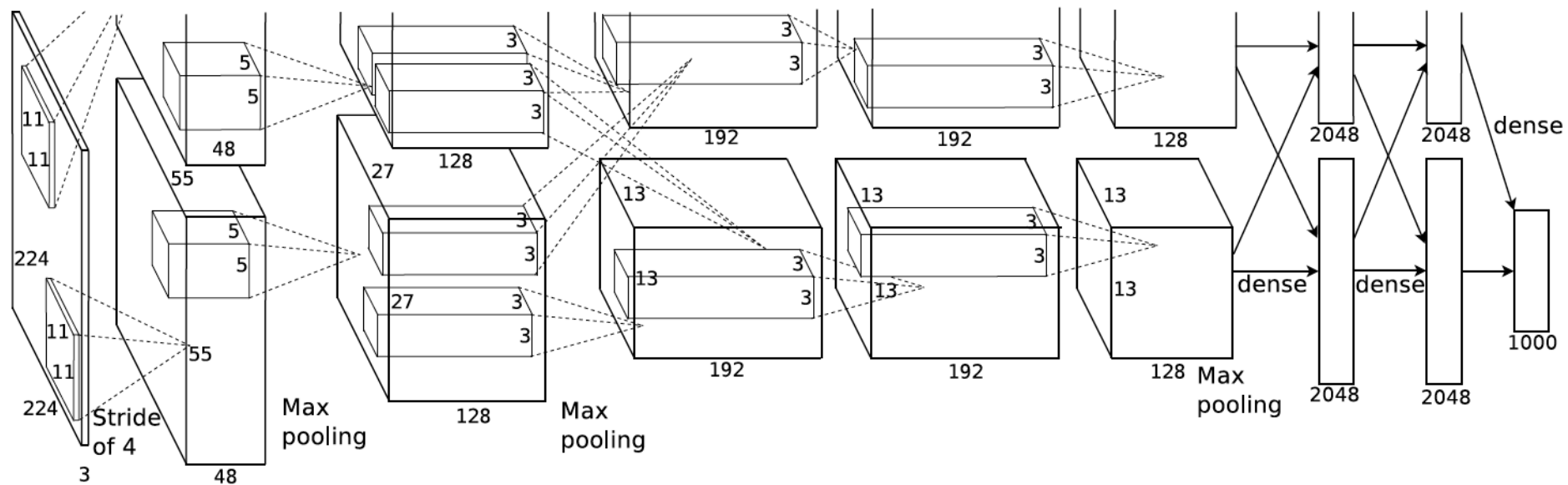
[Статья](#)



AlexNet (2012)

Статья

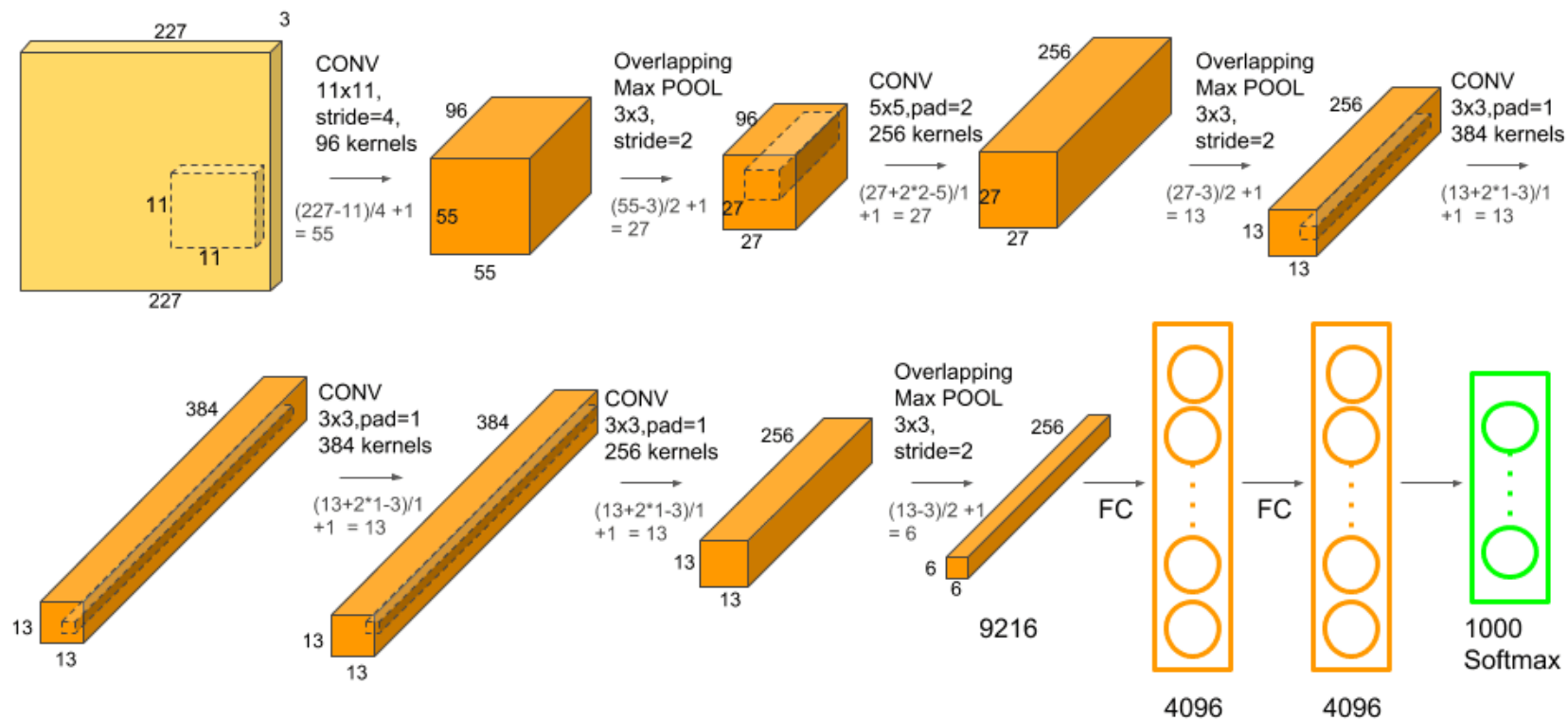
Количество параметров: ~60 миллионов



AlexNet (2012)

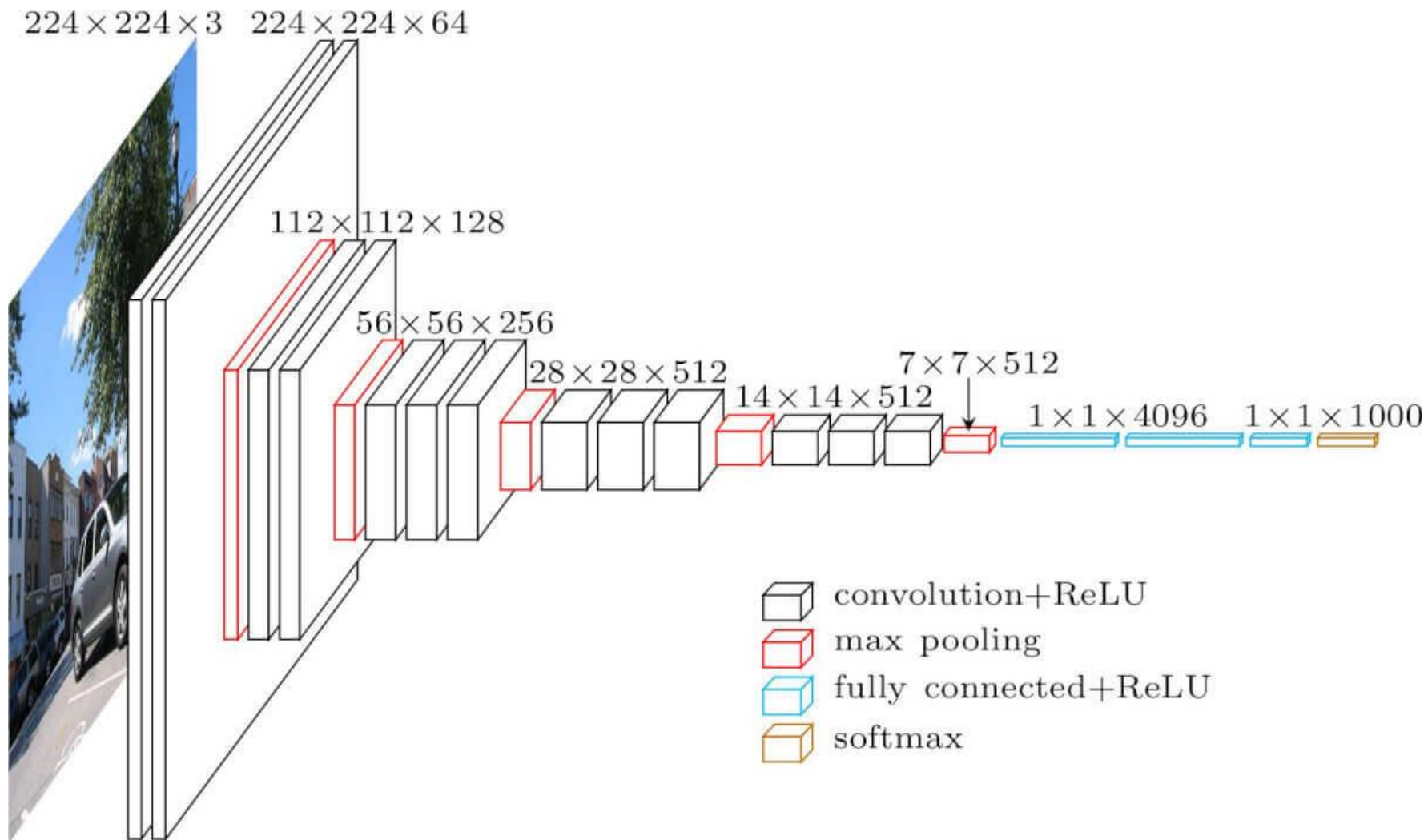
[Статья](#)

Количество параметров: ~60 миллионов



VGG16 (2014)

[Статья](#)

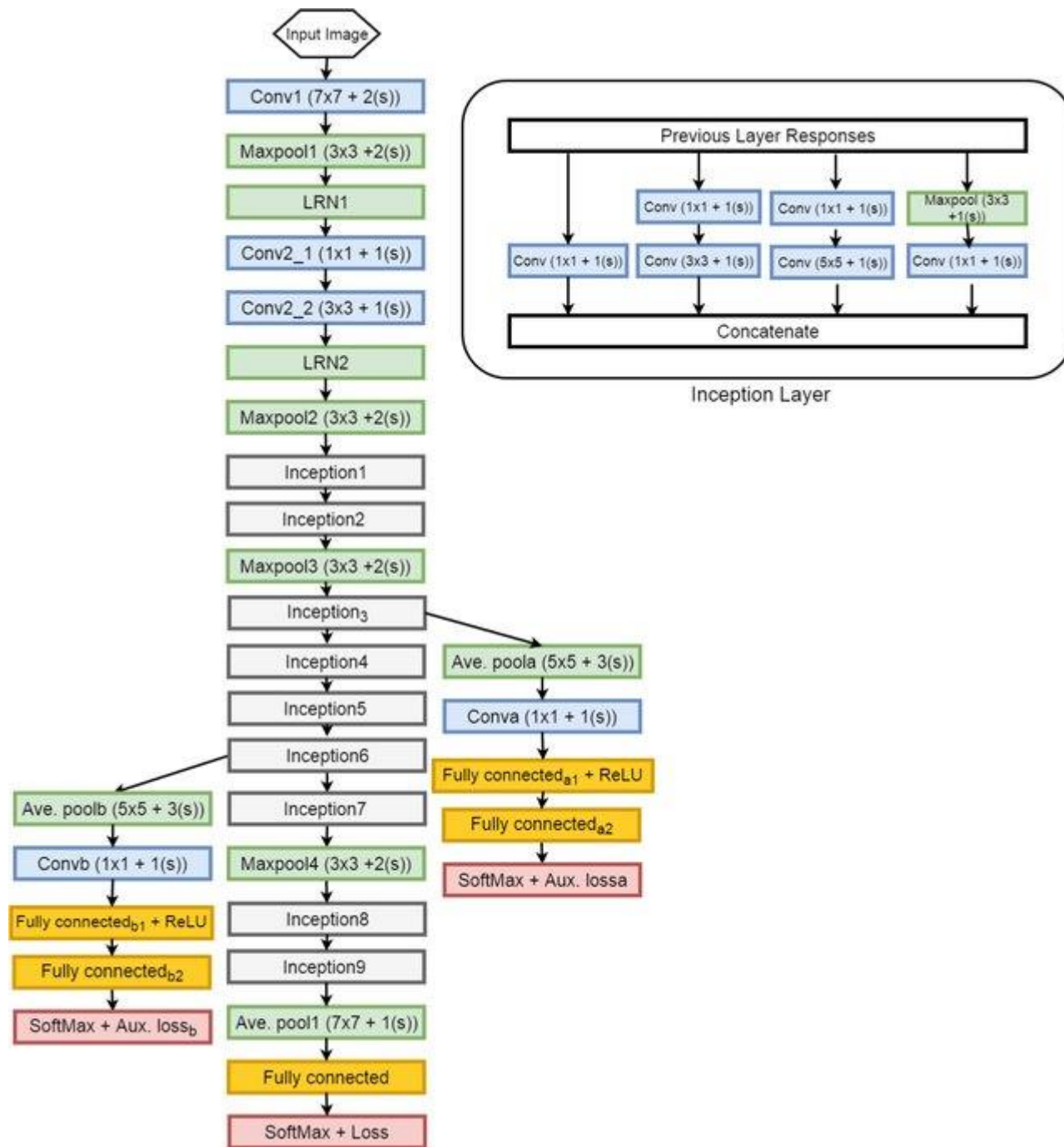


Количество параметров: ~140 миллионов

GoogleNet (Inception) (2014)

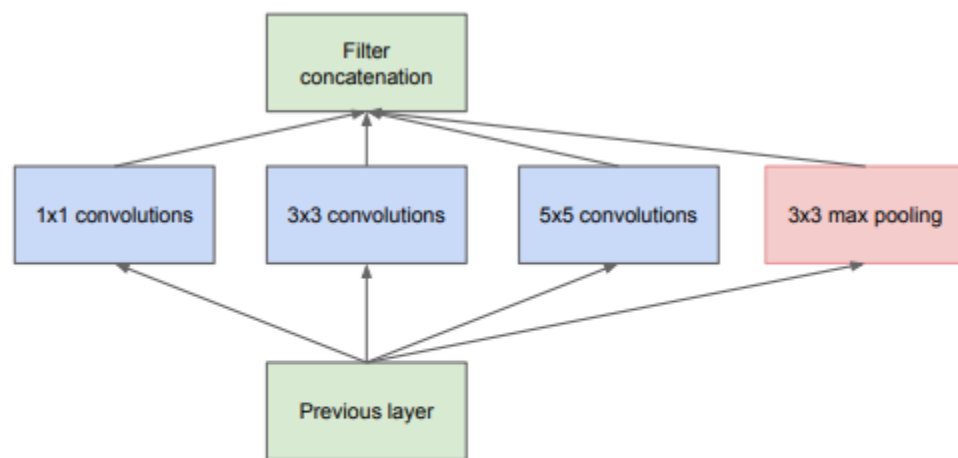
[Статья](#)

Количество параметров: ~5 миллионов

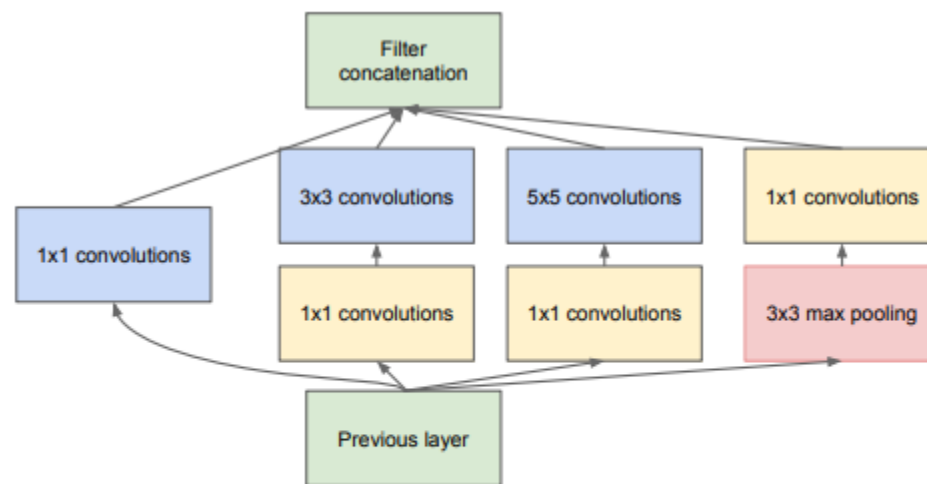


GoogleNet (Inception) (2014)

[Статья](#)



(a) Inception module, naïve version



(b) Inception module with dimension reductions

Figure 2: Inception module

ResNet (2015)

[Статья](#)

Количество параметров: ~25 миллионов
для ResNet-50

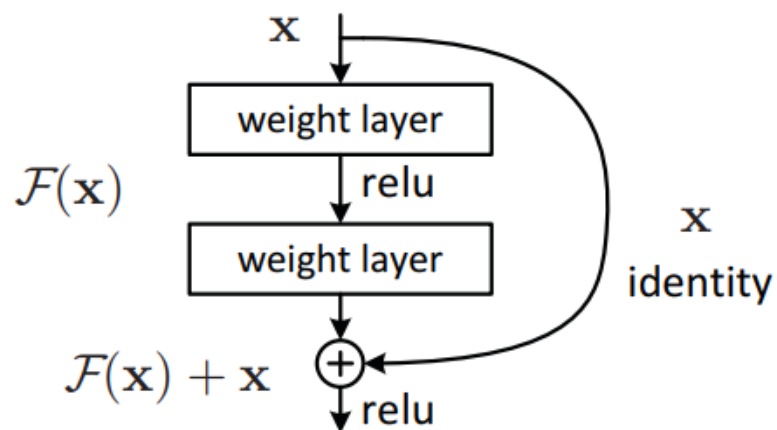
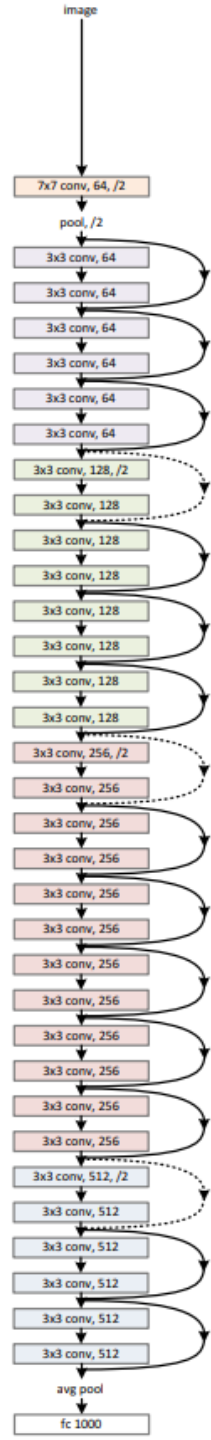


Figure 2. Residual learning: a building block.



EfficientNet (2019)

[Статья](#)

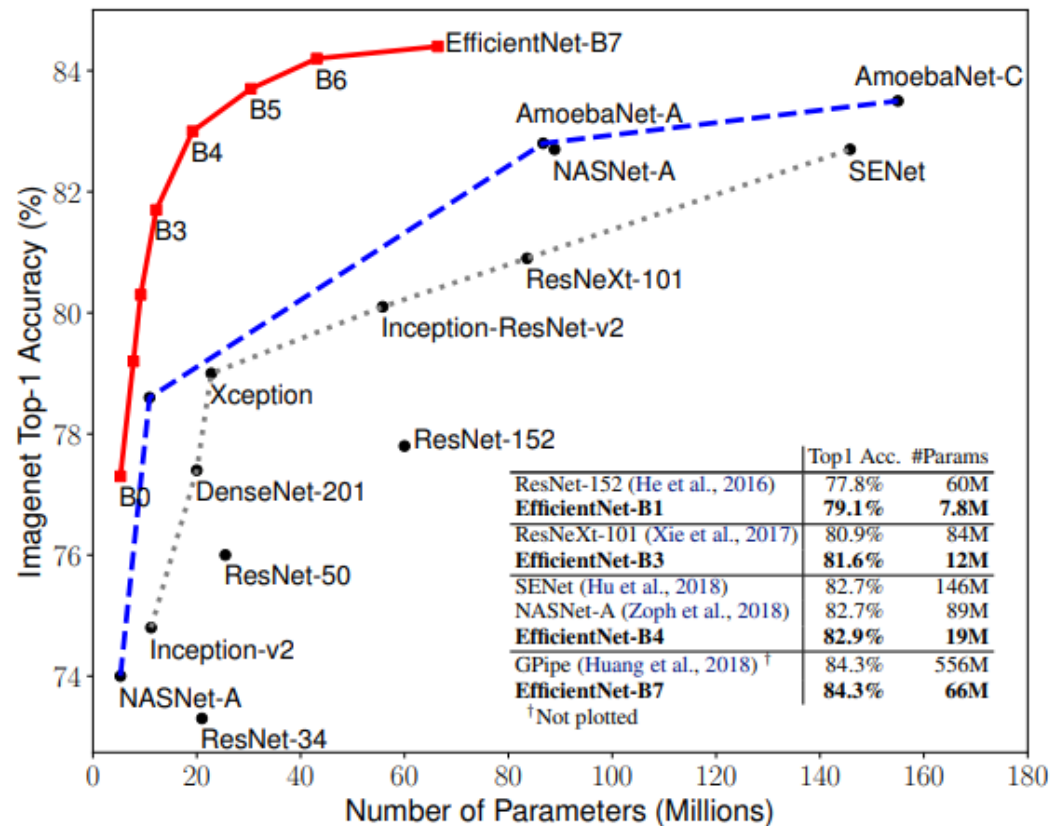


Figure 1. Model Size vs. ImageNet Accuracy. All numbers are for single-crop, single-model. Our EfficientNets significantly outperform other ConvNets. In particular, EfficientNet-B7 achieves new state-of-the-art 84.3% top-1 accuracy but being 8.4x smaller and 6.1x faster than GPipe. EfficientNet-B1 is 7.6x smaller and 5.7x faster than ResNet-152. Details are in Table 2 and 4.