Свёрточные сети

Лекция 4

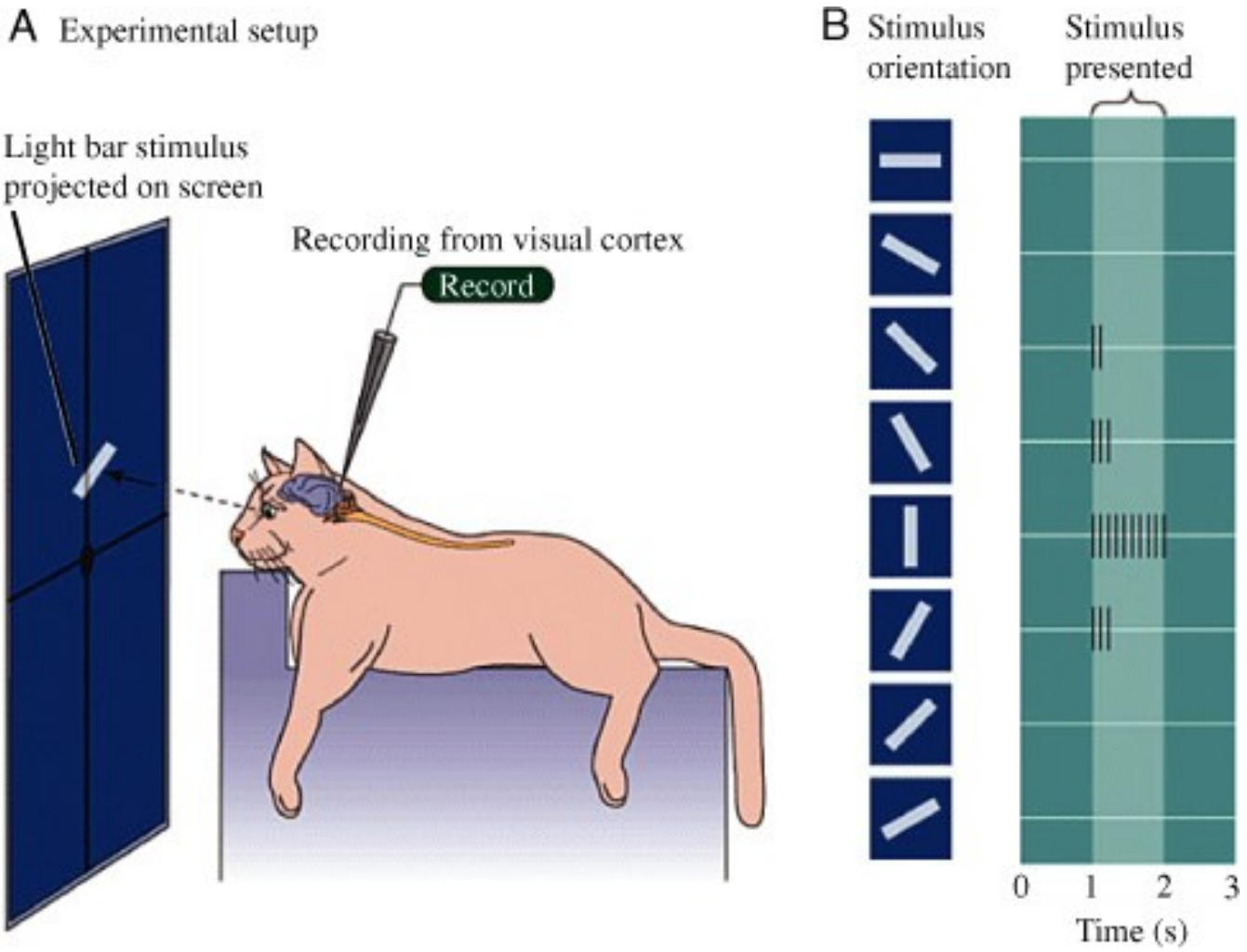
План лекции

- Зрительная кора головного мозга
- Свёртка и субдискретизация
- Реализация в PyTorch
- LeNet-5
- Архитектуры свёрточных сетей

Зрительная кора головного мозга

- Дэвид Хьюбел (David Hubel, 1926-2013) и Торстен Визель (Torsten Wiesel, род. 1924) – Нобелевская премия по физиологии или медицине 1981 года «За открытия, касающиеся принципов переработки информации в нейронных структурах»

Зрительная кора головного мозга

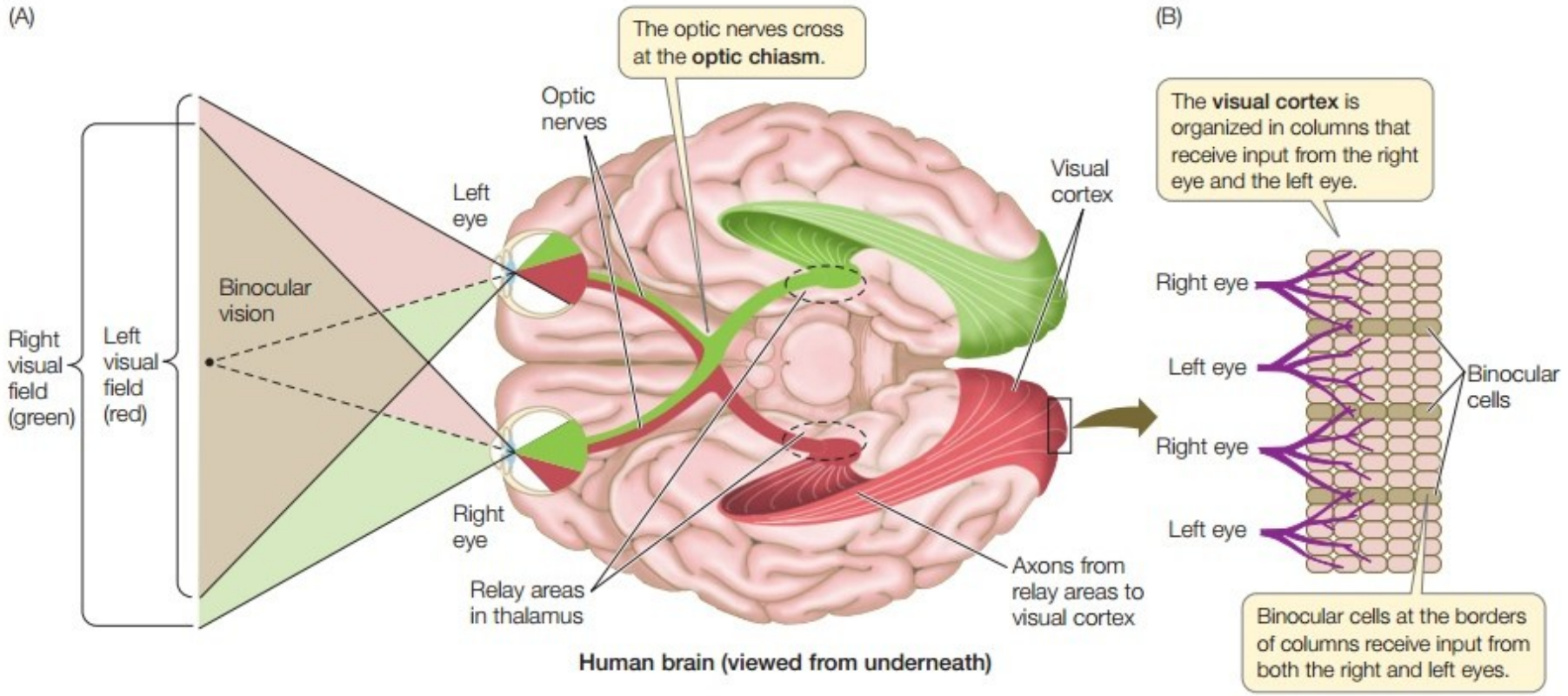


https://www.youtube.com/watch?v=IOHayh06LJ4

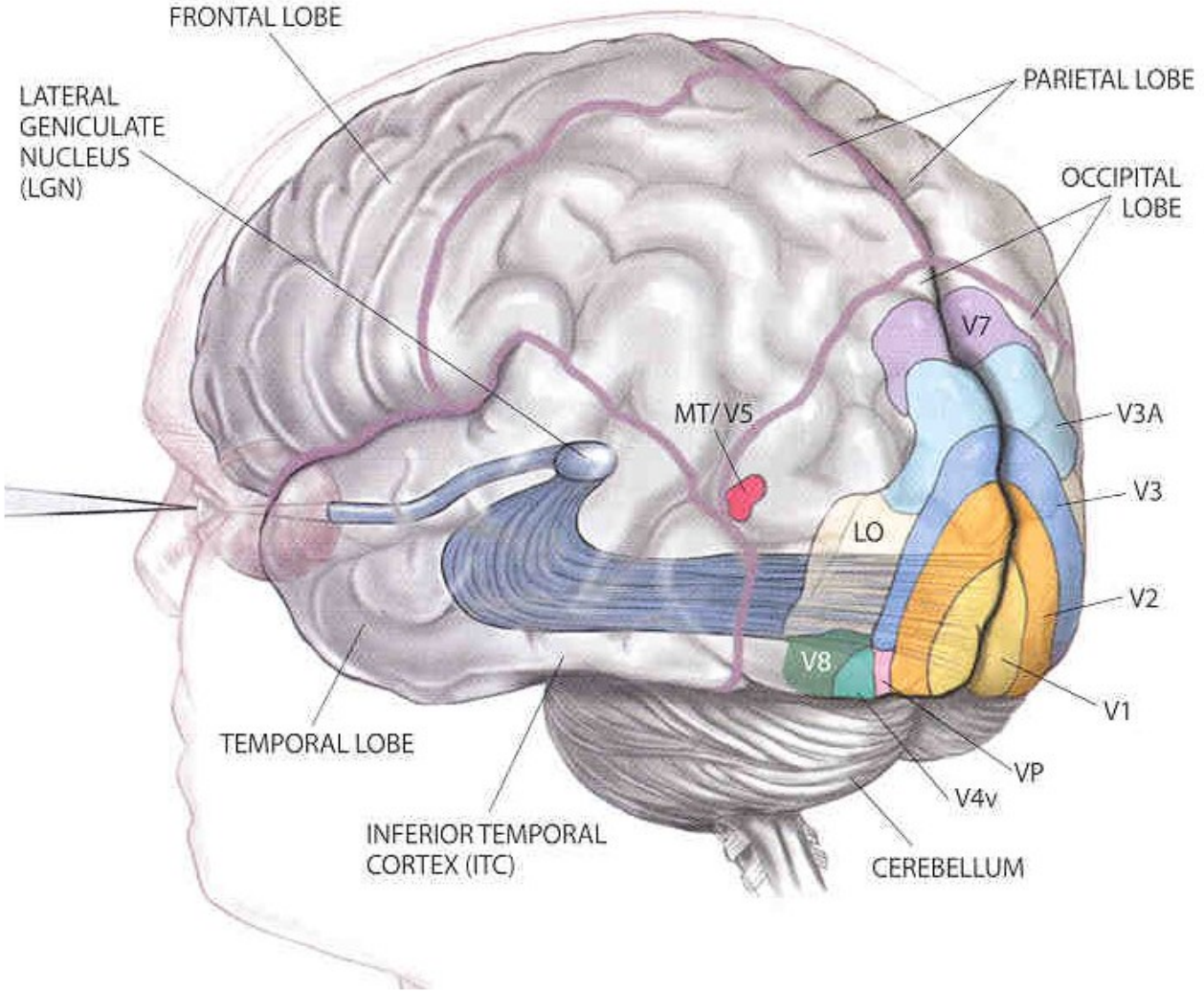
Зрительная кора головного мозга

«Наше первое настоящее открытие случилось совершено неожиданно. На протяжении двух или трёх часов у нас ничего не получалось. Затем постепенно мы начали различать какие-то смутные и непостоянные ответы при стимуляции где-то на границе между центром и периферией сетчатки. Мы как раз вставляли слайд на стекле в виде тёмного пятна в разъём офтальмоскопа, когда внезапно, через аудиомонитор, клетка зарядила как пулемёт. Спустя некоторое время, после небольшой паники, мы выяснили, что же случилось. Конечно, сигнал не имел никакого отношения к тёмному пятну. Во время того, как мы вставляли слайд на стекле, его край отбрасывал на сетчатку слабую, но чёткую тень, в виде прямой тёмной линии на светлом фоне. Это было именно то, чего хотела клетка, и, более того, она хотела, чтобы эта линия имела строго определённую ориентацию. Это было неслыханно. Сейчас даже трудно подумать и представить себе, насколько далеко мы были от какой-либо идеи относительно того, какую роль могут играть клетки коры в обычной жизни животного» (1959 г.)

Зрительная кора головного мозга



Зрительная кора головного мозга



Свёрточные сети (convolutional neural networks) – история

- Neocognitron (Kunihiko Fukushima, 1979)
  - свёрточная сеть, основанная на идеях Хьюбела и Визеля
  - обучение без учителя, распознавание символов
- Time Delay Neural Network (TDNN, Alex Waibel, 1987)
  - свёрточная сеть для распознавания речи
- Yann LeCun et al. (1989)
  - свёрточная сеть LeNet-5 в современном виде для распознавания рукописных цифр
  - алгоритм обратного распространения ошибки

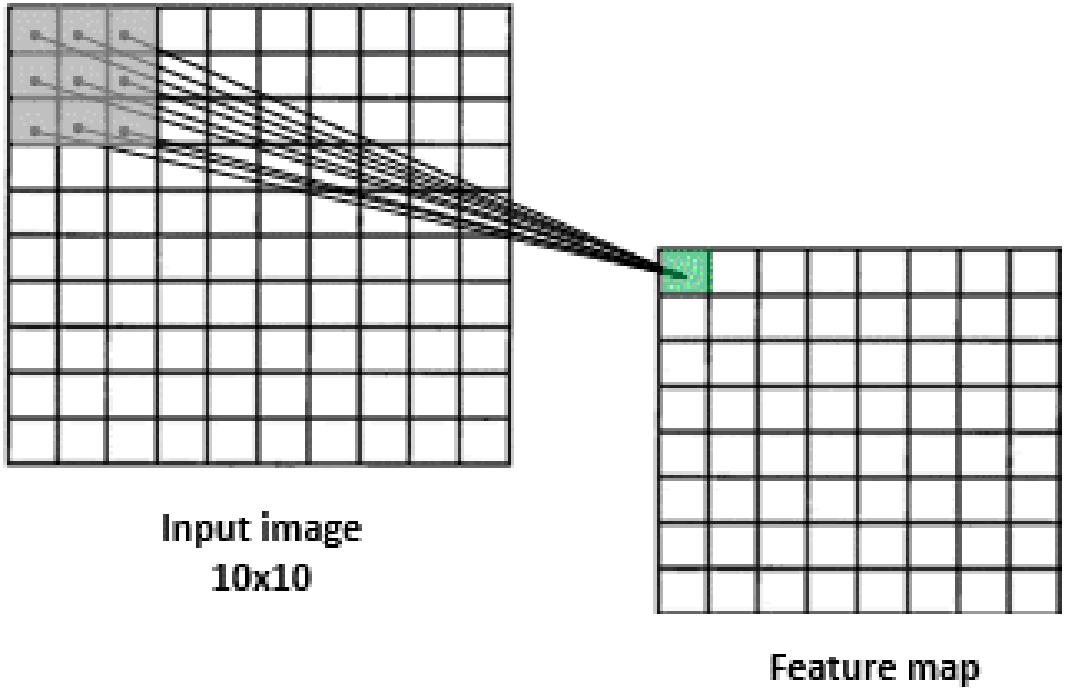
Свёртка

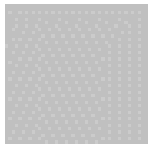
- Многие виды данных имеют внутреннюю структуру:
  - 1-мерную – порядок слов в текстах, порядок звуковых сигналов в речи
  - 2-мерную – взаимное расположение пикселей в изображениях
  - 3-мерную – двумерные изображения с каналами, объемные изображения
- Полносвязные сети не позволяют явно задавать порядок, отличный от одномерного
- В сверточных сетях порядок учитывается при помощи *операции свёртки*

Свёртка

- *Свёртка* (convolution) – линейное преобразование входных данных на основе ядра свёртки
- *Ядро свёртки* (kernel, filter, convolution matrix, mask) – небольшая матрица весовых коэффициентов, которая последовательно умножается на фрагменты входных данных, формируя в результате *карту признаков* (feature map)
  - Размеры ядер: 1x1, 3x3, 5x5, 7x7, 11x11, …
  - Говорят, что веса «разделяются» (share) между всеми фрагментами входных данных
- В сети присутствует несколько ядер свёртки; столько же получается карт признаков
- *Локальное рецептивное поле* (Local Receptive Field) – область входных данных, совпадающая по размерам с ядром свёртки

Свёртка

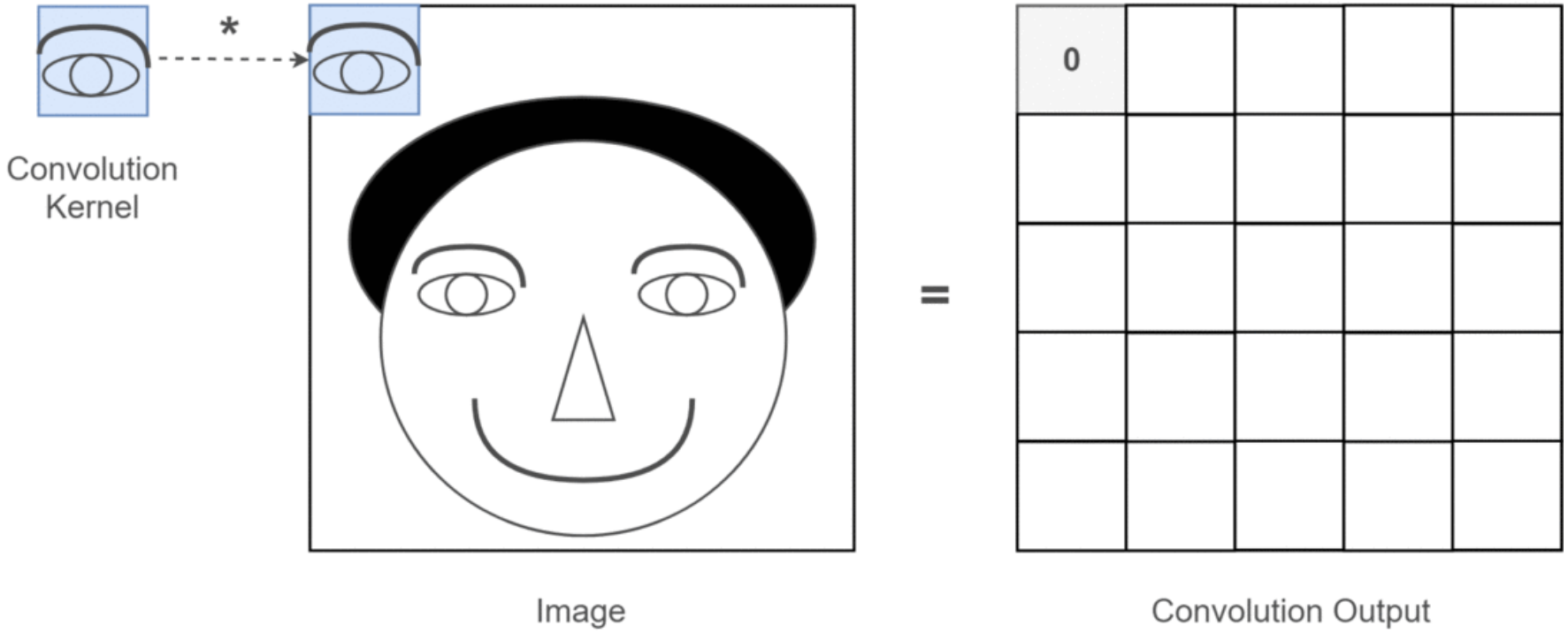




Kernel

Hidden neuron

Свёртка



Свёртка

- Обозначим:

  x^l – входная карта признаков в слое l,

  размер ядра: $2d + 1$,

  матрица весов ядра W,

  размер матрицы W: $(2d + 1) \times (2d + 1)$

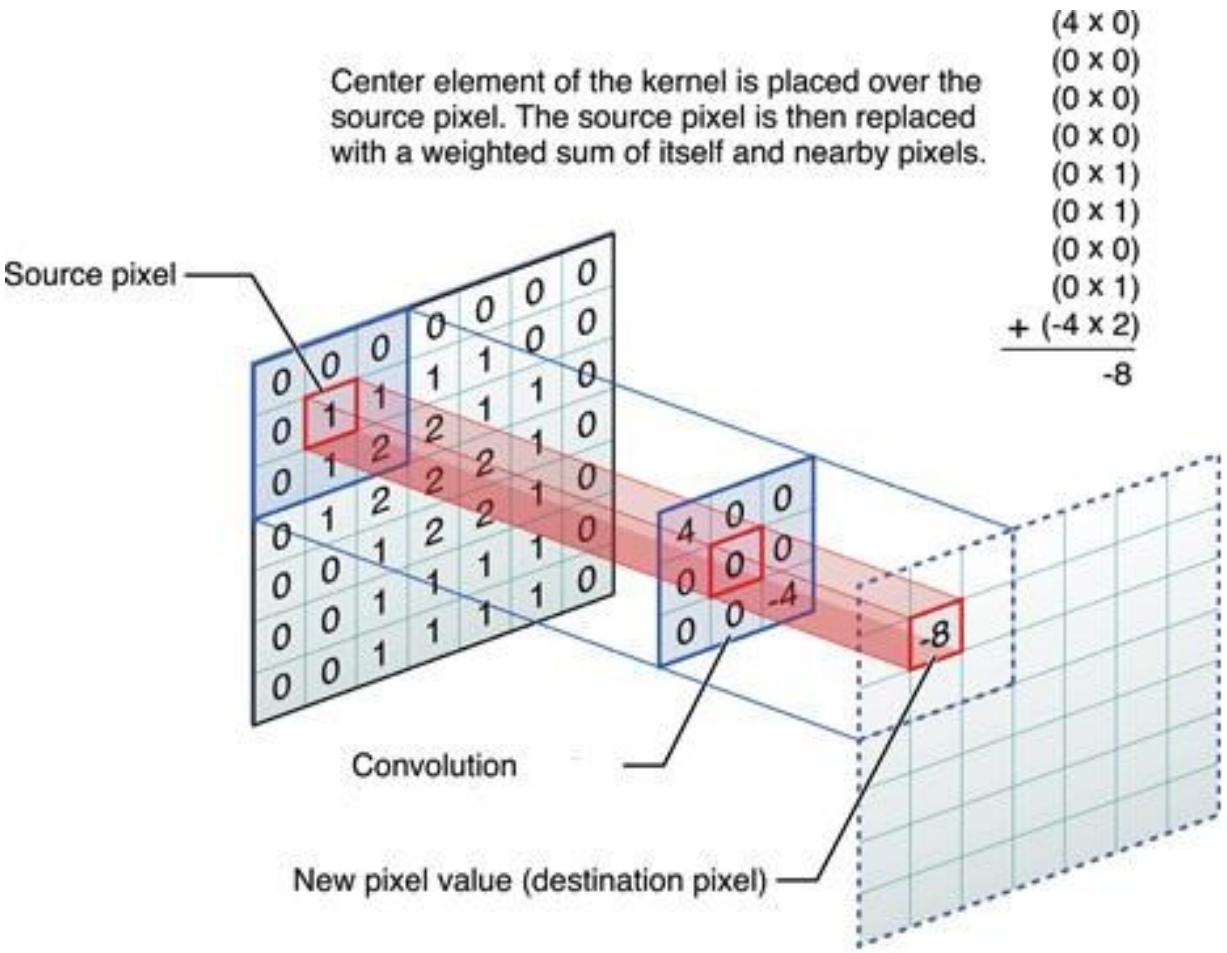
- Тогда результат свёртки в слое l:

$$y_{i,j}^l = \sum_{-d \le a,b \le d} W_{a,b} x_{i+a,j+b}^l$$

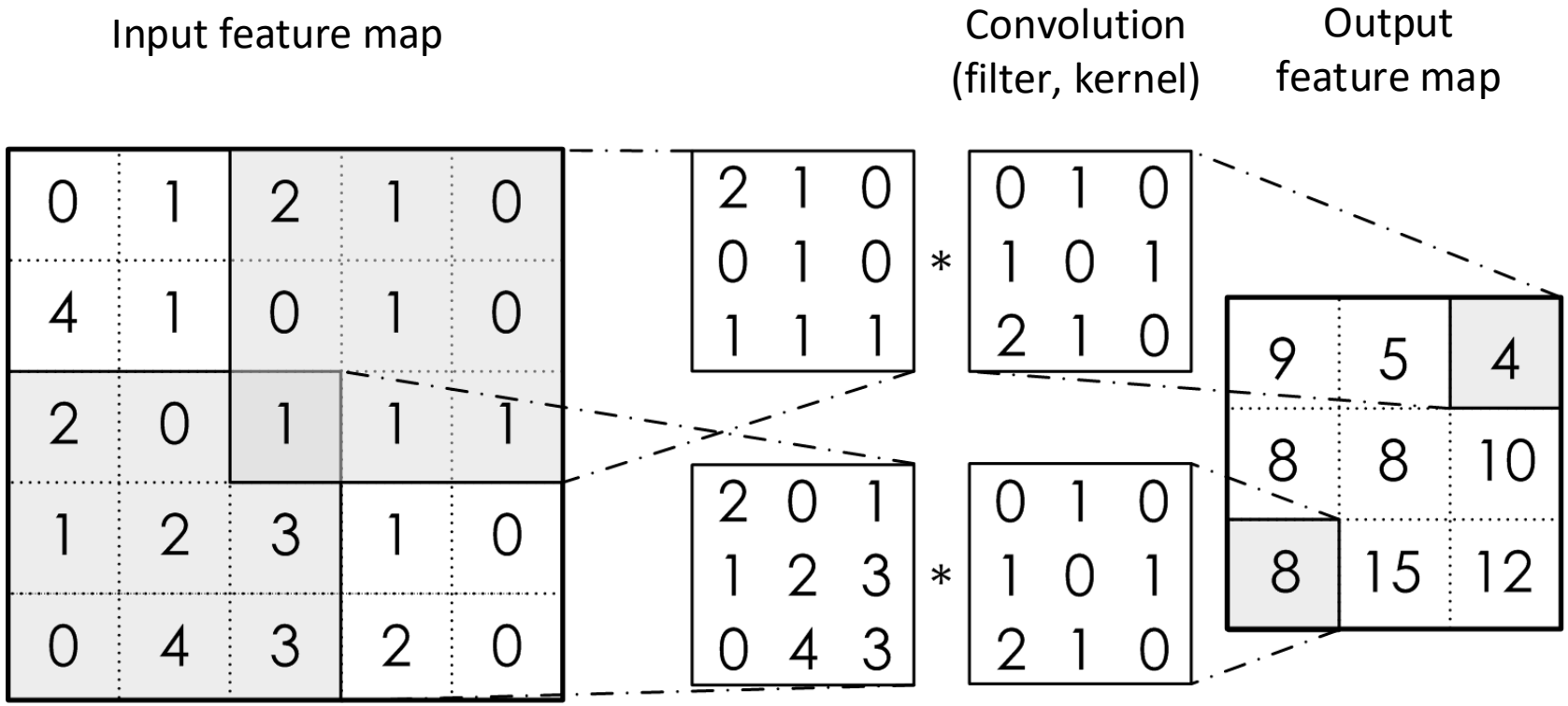
- После этого следует нелинейность, например, ReLU:

$$z_{i,j}^l = h\left(y_{i,j}^l\right)$$

Свёртка


Center element of the kernel is placed over the source pixel. The source pixel is then replaced with a weighted sum of itself and nearby pixels.
(4 x 0)
(0 x 0)
(0 x 0)
(0 x 0)
(0 x 1)
(0 x 1)
(0 x 0)
(0 x 1)
+ (-4 x 2)
-8
Source pixel
Convolution
New pixel value (destination pixel)

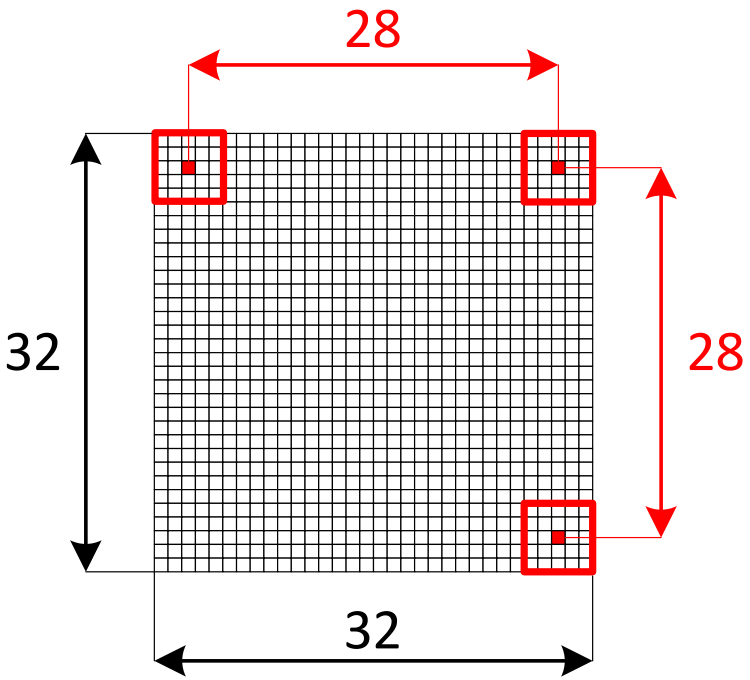
Свёртка


Input feature map
Convolution
(filter, kernel)
Output
feature map
0 1 2 1 0
4 1 0 1 0
2 0 1 1 1
1 2 3 1 0
0 4 3 2 0
2 1 0
0 1 0
1 1 1
*
0 1 0
1 0 1
2 1 0
2 0 1
1 2 3
0 4 3
*
0 1 0
1 0 1
2 1 0
9 5 4
8 8 10
8 15 12

Свёртка – padding

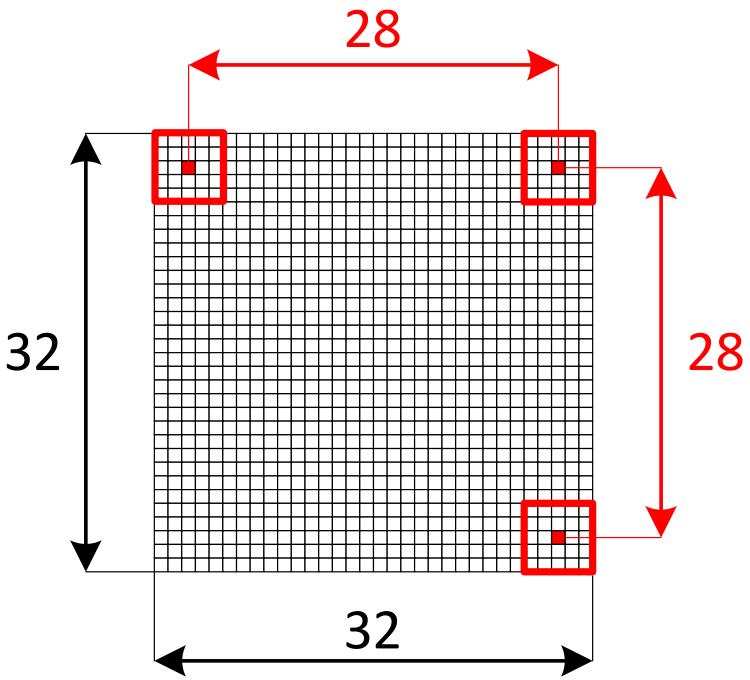
- Заполнение (padding) – определяет, будут ли ядра выходить за границы входных данных
- padding = ‘SAME’ – ядра выходят за границы, размер выходной карты признаков совпадает со входной
- padding = ‘VALID’ – ядра не выходят за границы, размер выходной карты признаков меньше входной

Свёртка – padding

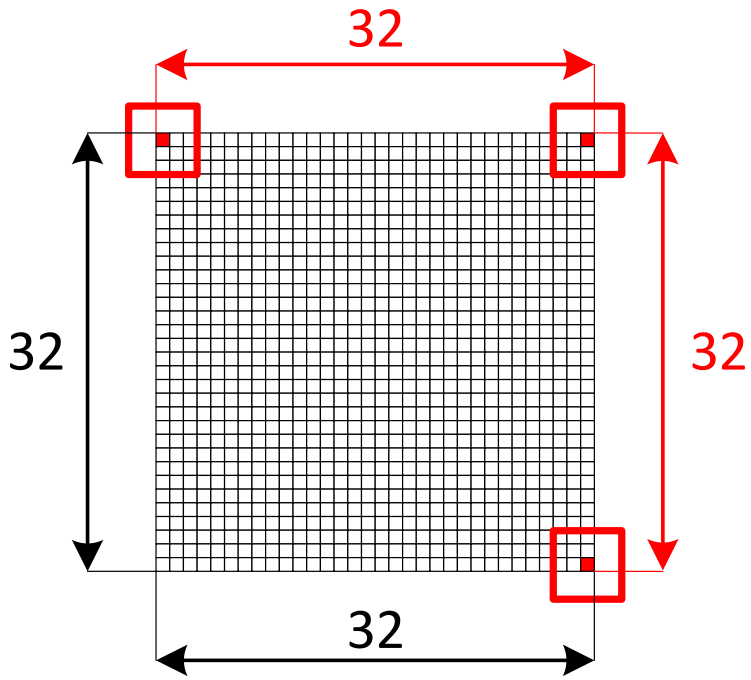


`padding = “valid”`

Свёртка – padding



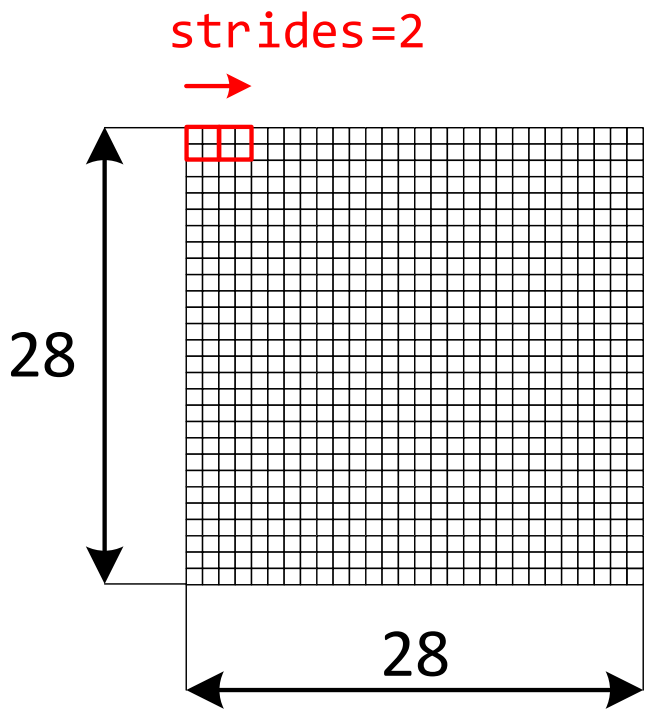
padding = “valid”



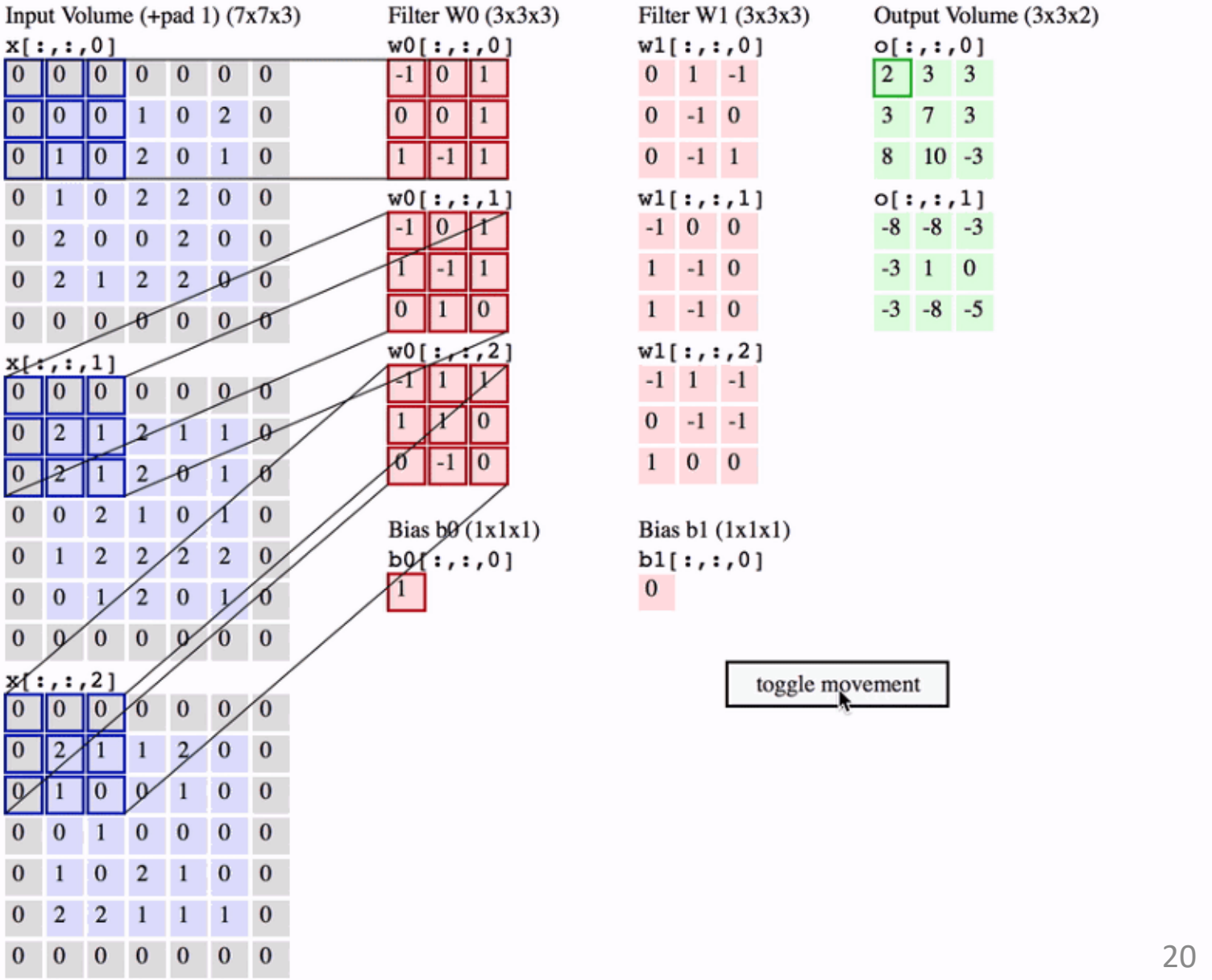
padding = “same”

Свёртка – stride

- Шаг (stride) – определяет расстояние, на которое сдвигаются ядра

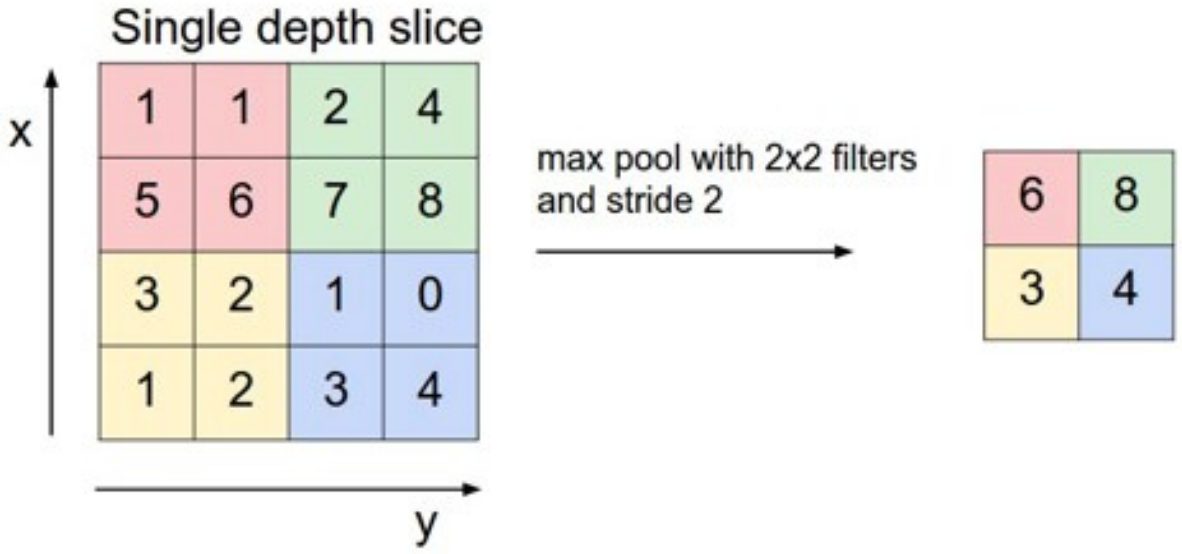


Свёртка – 3 канала

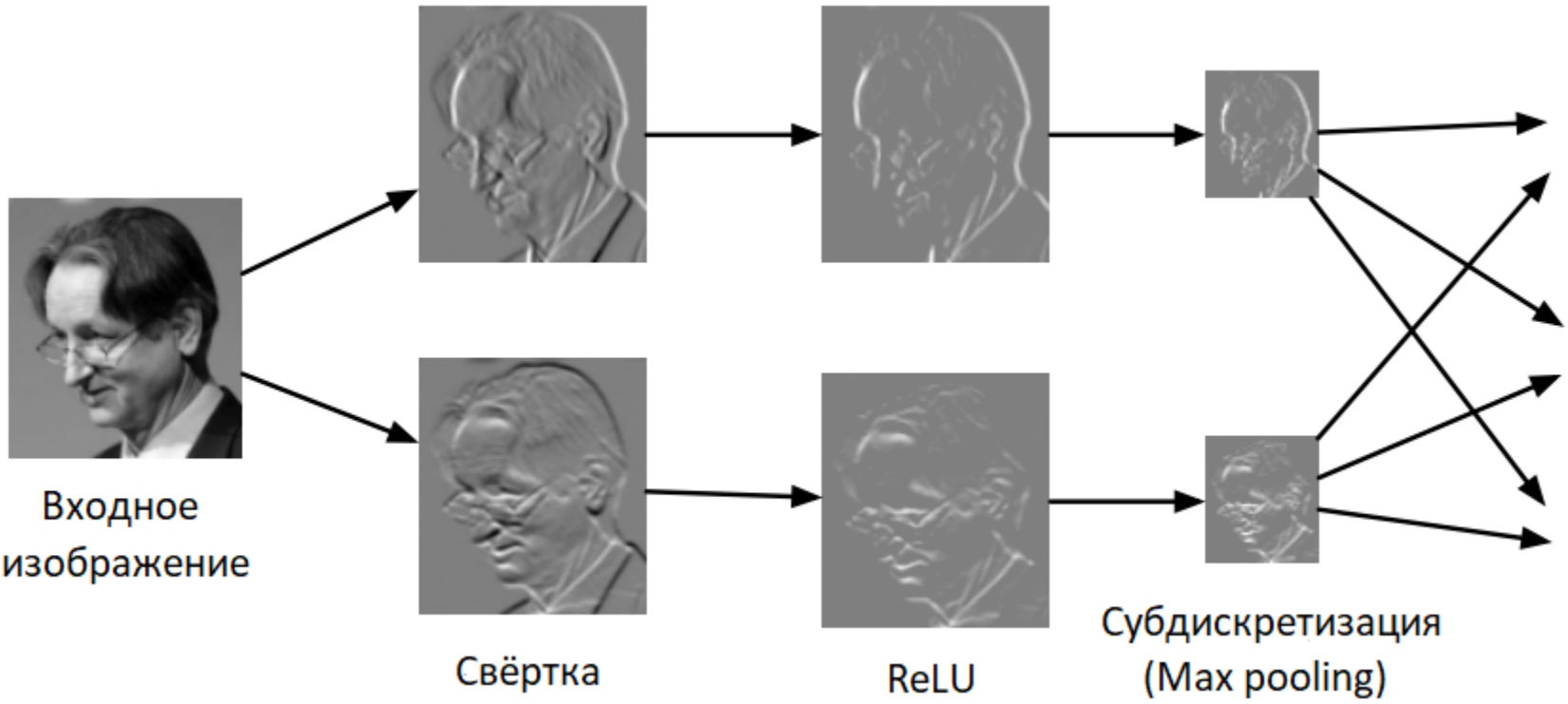


Субдискретизация (pooling)

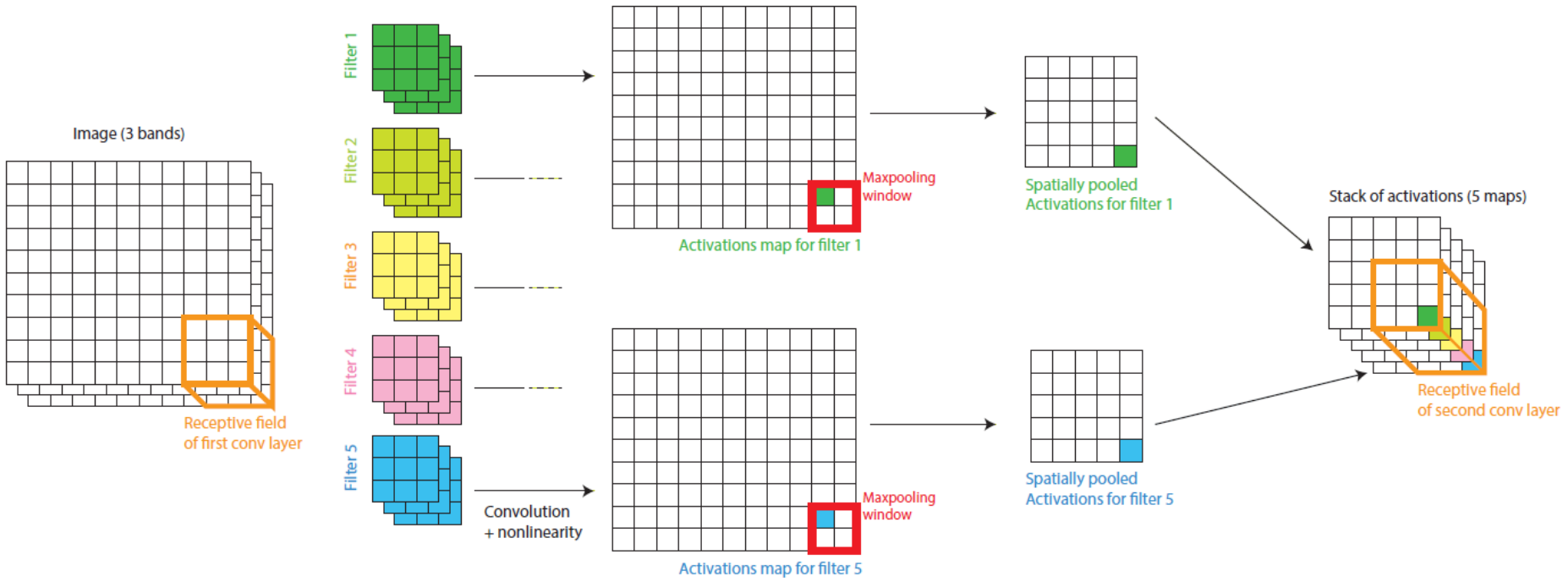
- Субдискретизация (pooling, subsampling) – сокращение размерности карты признаков за счет выделения наиболее важных признаков
- Как правило, используется *Max pooling*, но может быть *Average pooling*



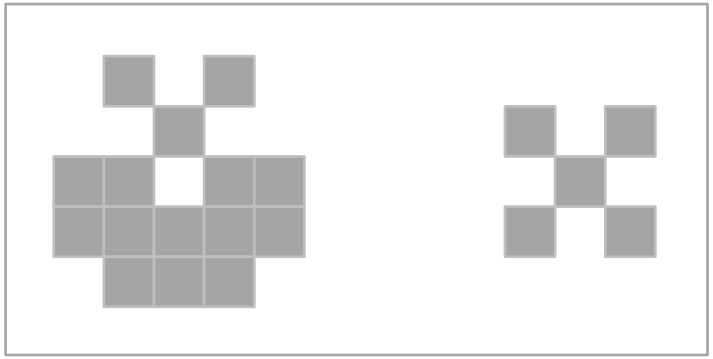
Convolution + ReLU + Max pooling



Convolution + ReLU + Max pooling



Пример



Исходное изображение

0	1	0	1	0
0	0	1	0	0
1	1	0	1	1
1	1	1	1	1
0	1	1	1	0

Сверточные фильтры первого слоя

1	0	0
0	1	0
0	0	1

1	1	1
1	1	1
1	1	1

0	0	1
0	1	0
1	0	0

Карты признаков

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

1	2	3	2	1
3	4	5	4	3
4	6	6	6	4
5	7	8	7	5
3	5	6	5	3

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

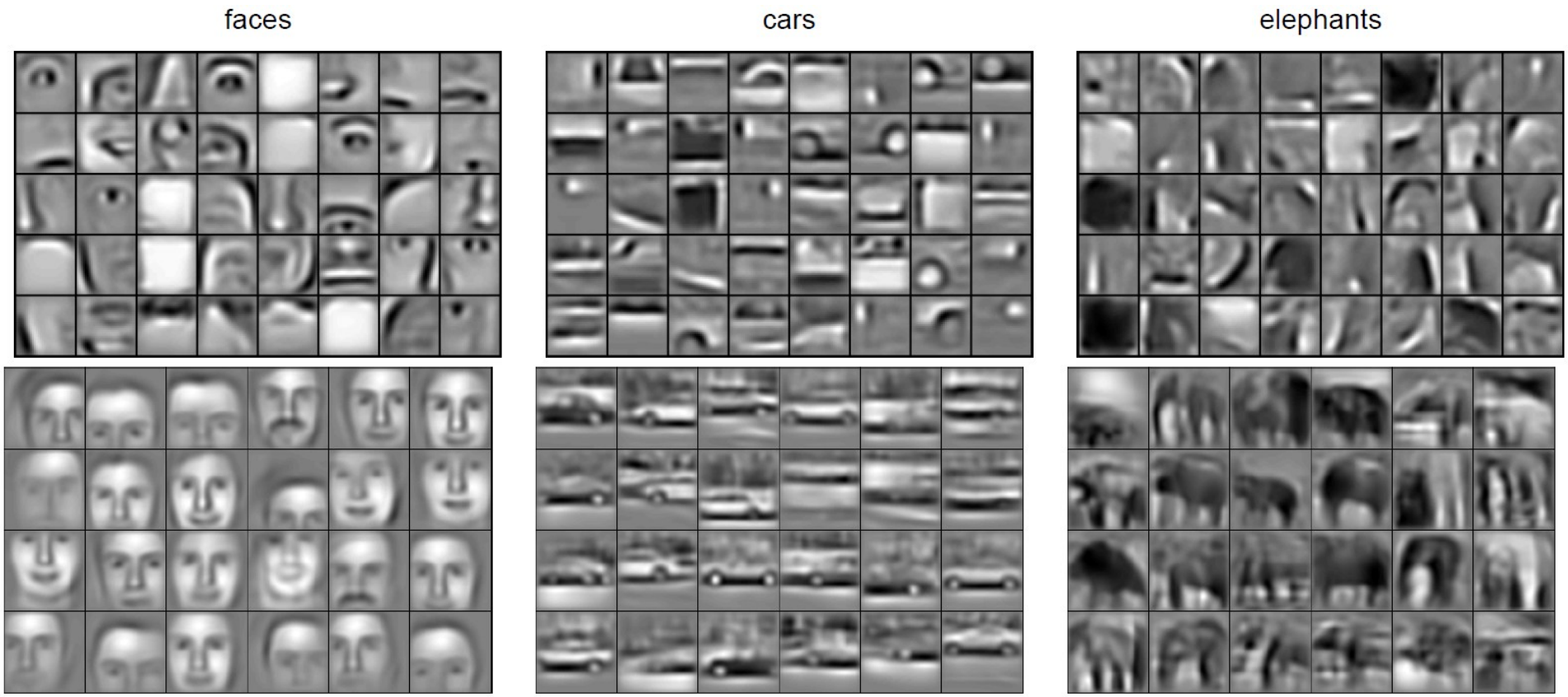
Сверточный фильтр второго слоя по каналам

1
-1
1

Результат

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

Пример



Lee H., Grosse R., Ranganath R., Ng A. Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations // ICML-2009

Демо

- CNN Explainer:
  - https://poloclub.github.io/cnn-explainer/
- A guide to convolution arithmetic for deep learning
  - https://github.com/vdumoulin/conv_arithmetic/

Особенности свёрточных сетей

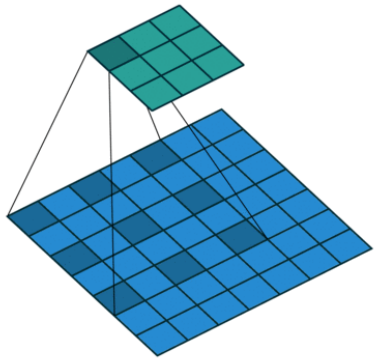
- Свёртка сохраняет структуру входа
- Свёртка выделяет локальные признаки
- Объем данных на входе фактически увеличивается (одно изображение превращается во множество фрагментов)
- Свёртка обладает свойством разреженности – значение нейрона следующего слоя зависит от небольшого количества нейронов текущего слоя
- Свёртка многократно переиспользует одни и те же веса:
  - придает устойчивость к переносам
  - сокращает количество весов (форма регуляризации)
- Взаимосвязь между удаленными фрагментами может быть учтена в глубоких свёрточных сетях

Реализация в PyTorch

```
torch.nn.Conv2d(in_channels, out_channels, kernel_size, stride=1,
padding=0, dilation=1, groups=1, bias=True, padding_mode='zeros')
```

- https://pytorch.org/docs/stable/generated/torch.nn.Conv2d.html

- `in_channels` – количество входных каналов
- `out_channels` – количество ядер (фильтров, выходных каналов)
- `kernel_size` – размер ядра
- `stride` – шаг сдвига ядер
- `padding` – количество точек заполнения со всех сторон входного изображения
- `dilation` (растяжение) – расстояние между точками ядра
- `kernel_size`, `stride`, `padding`, `dilation` – могут быть `int` или `tuple`: `(int, int)`



Реализация в PyTorch

```
torch.nn.Conv2d(in_channels, out_channels, kernel_size, stride=1,
padding=0, dilation=1, groups=1, bias=True, padding_mode='zeros')
```

- Размерность входа: (N, C_{in}, H, W)
- Размерность выхода: $(N, C_{out}, H_{out}, W_{out})$
- Выход слоя:

$$out\left(N_i, C_{out_j}\right) = bias\left(C_{out_j}\right) + \sum_{k=0}^{C_{in}-1} weight\left(C_{out_j}, k\right) * input(N_i, k)\,,$$

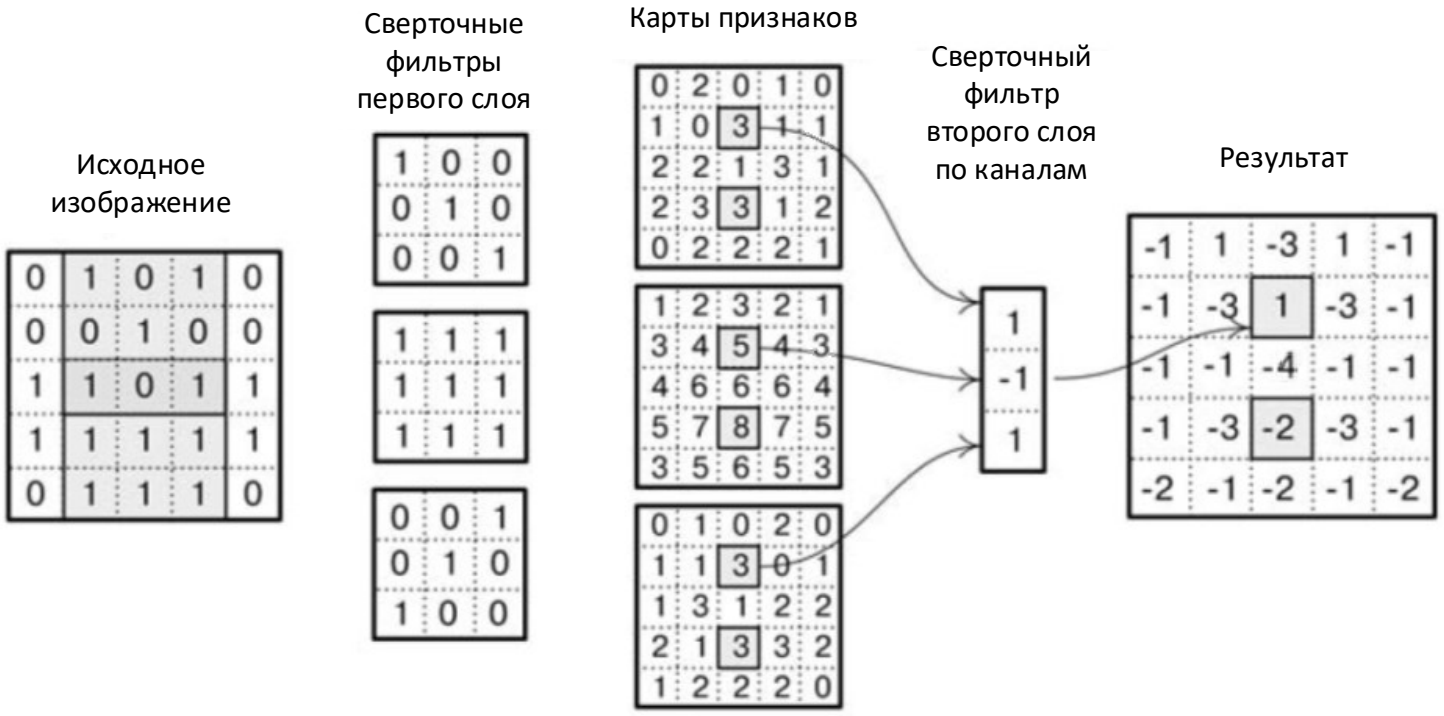
где N – размер батча, C – количество каналов, H – высота изображения, W – ширина изображения

- Веса:
  - `Conv2d.weight`: размерность $(C_{out}, C_{in}, kernel\_size[0], kernel\_size[1])$
  - `Conv2d.bias`: размерность C_{out}

Реализация в PyTorch

```
torch.nn.Conv2d(in_channels, out_channels, kernel_size, stride=1,
padding=0, dilation=1, groups=1, bias=True, padding_mode='zeros’)
```

- См. пример “convolution.ipynb”



Реализация в PyTorch

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

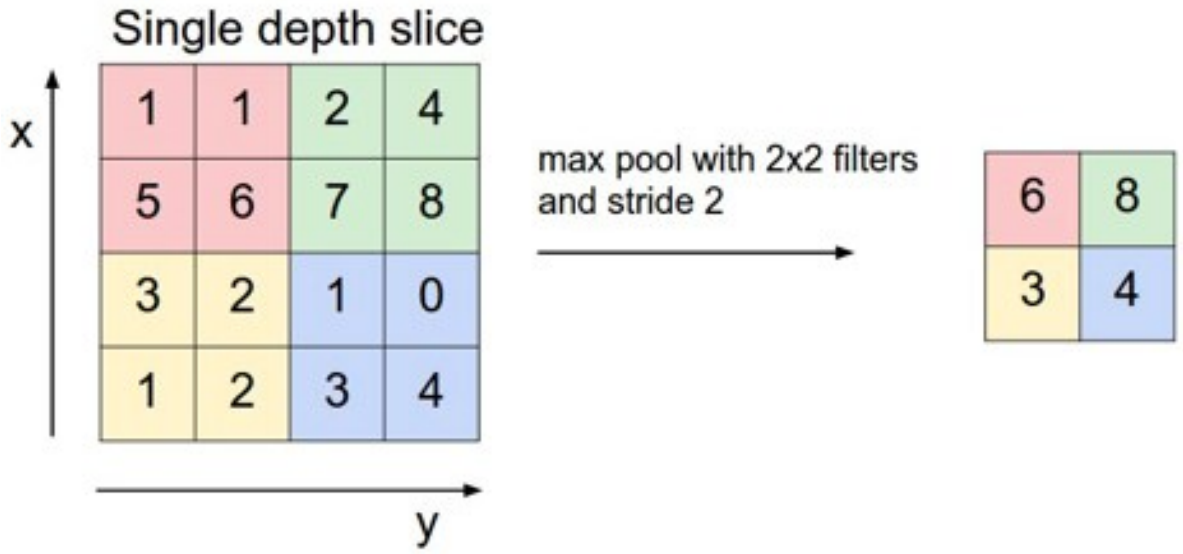
- https://pytorch.org/docs/stable/generated/torch.nn.MaxPool2d.html

- `kernel_size` – размер ядра
- `stride` – шаг сдвига ядер: `default value = kernel_size`
- `padding` – количество точек заполнения со всех сторон входного изображения
- `dilation` (растяжение) – расстояние между точками ядра
- `kernel_size`, `stride`, `padding`, `dilation` – могут быть `int` или `tuple: (int, int)`
- `return_indices` – возвращать ли индексы максимальных элементов
- `ceil_mode` – использовать округление `ceil` вместо `floor` для вычисления размерности выхода

Реализация в PyTorch

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

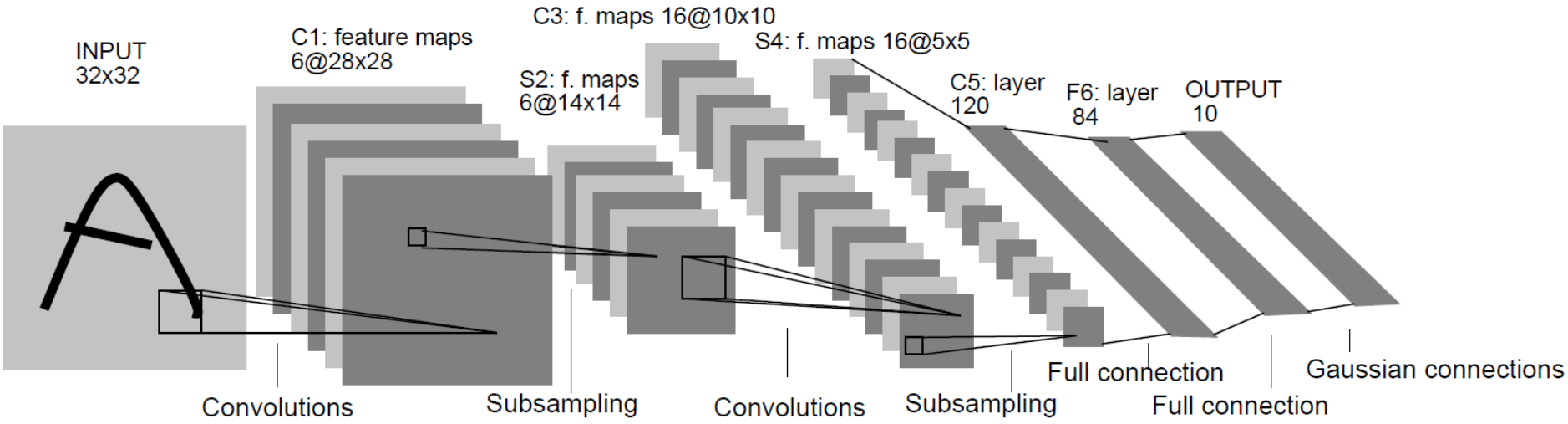
- См. пример “max_pooling.ipynb”



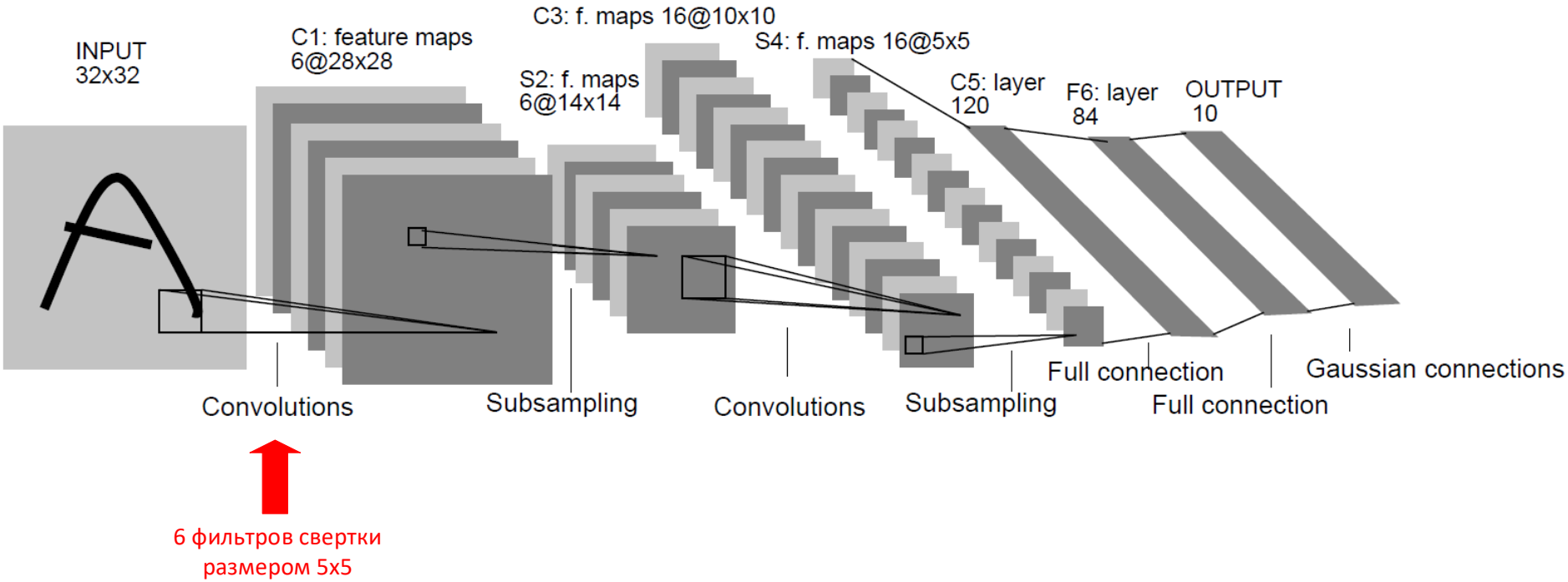
LeNet-5

- LeNet-5 – одна из первых свёрточных нейронных сетей, разработанная Яном Лекуном (Yann LeCun) с коллегами в 1989 году и показавшая лучшие результаты на MNIST (99.2%)
  - LeCun Y., Boser B., Denker J.S., Henderson D., Howard R.E., Hubbard W., Jackel L.D. Backpropagation applied to handwritten zip code recognition. Neural Computation, 1989. Vol. 1(4). P. 541–551
  - LeCun Y., Bottou L., Bengio Y., Haffner P. Gradient-based learning applied to document recognition. Proceedings of the IEEE. 1998. Vol. 86(11). P. 2278–2324
  - 60К параметров

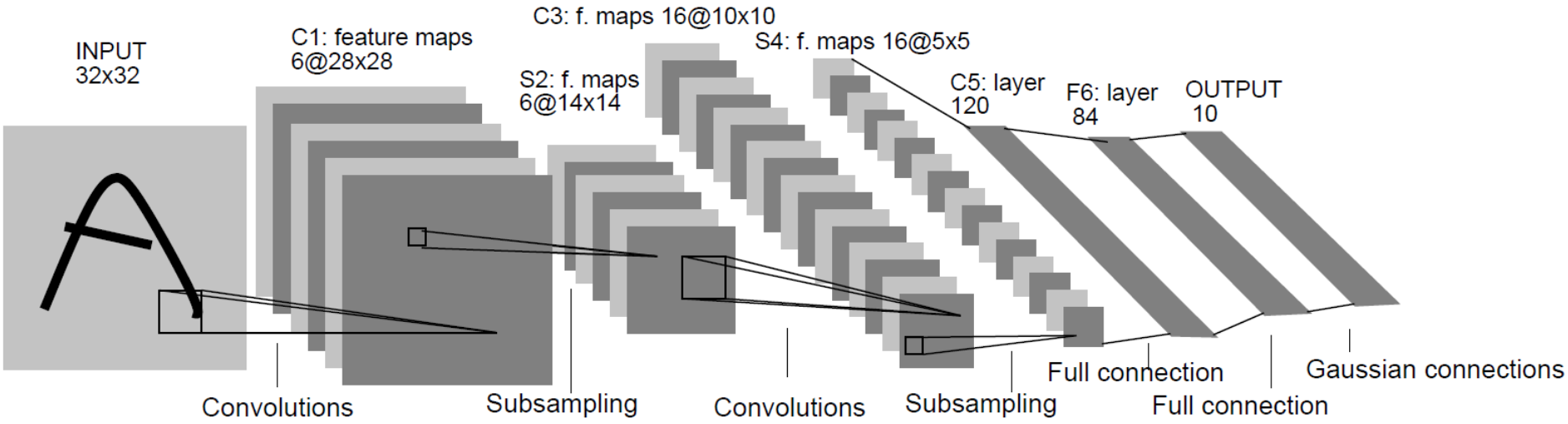
LeNet-5


INPUT
32x32
C1: feature maps
6@28x28
S2: f. maps
6@14x14
C3: f. maps 16@10x10
S4: f. maps 16@5x5
C5: layer
120
F6: layer
84
OUTPUT
10
Convolutions
Subsampling
Convolutions
Subsampling
Full connection
Full connection
Gaussian connections

LeNet-5


INPUT
32x32
C1: feature maps
6@28x28
S2: f. maps
6@14x14
C3: f. maps 16@10x10
S4: f. maps 16@5x5
C5: layer
120
F6: layer
84
OUTPUT
10
Convolutions
Subsampling
Convolutions
Subsampling
Full connection
Full connection
Gaussian connections
6 фильтров свертки
размером 5x5

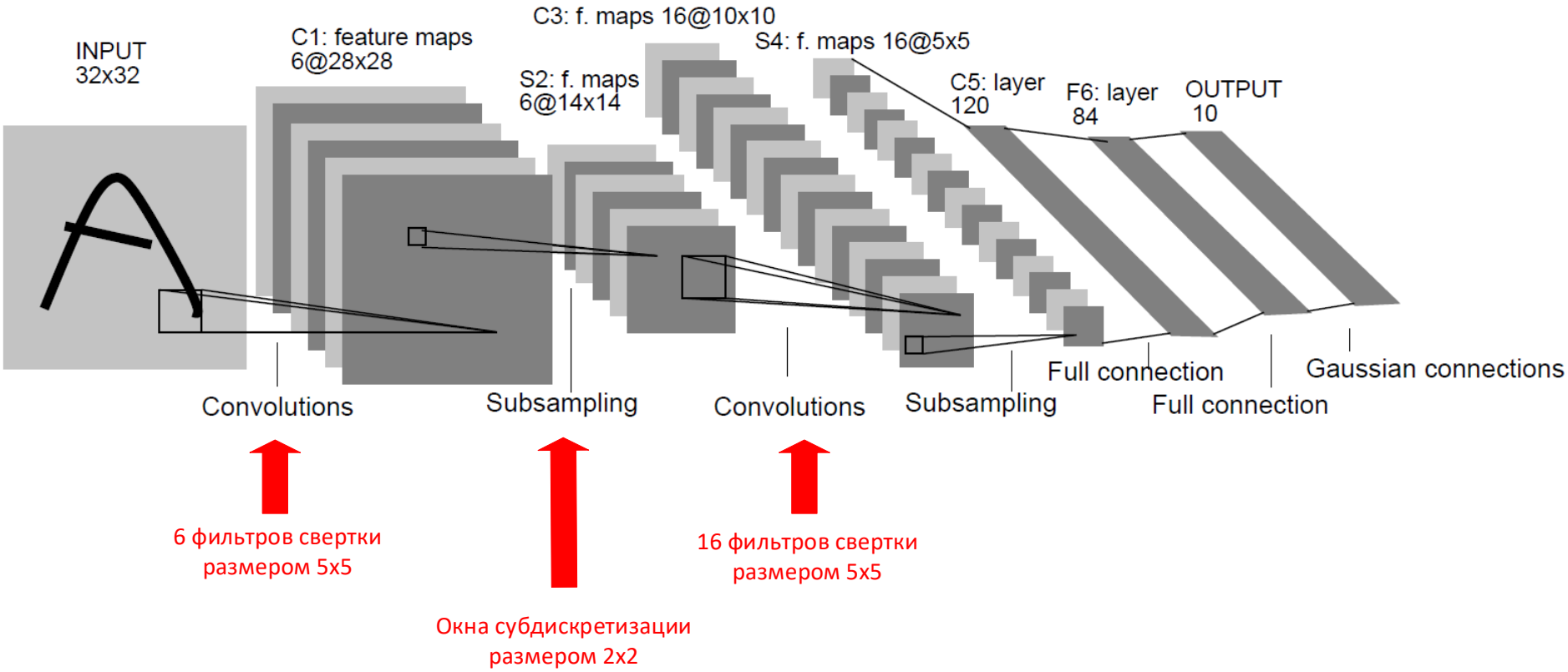
LeNet-5



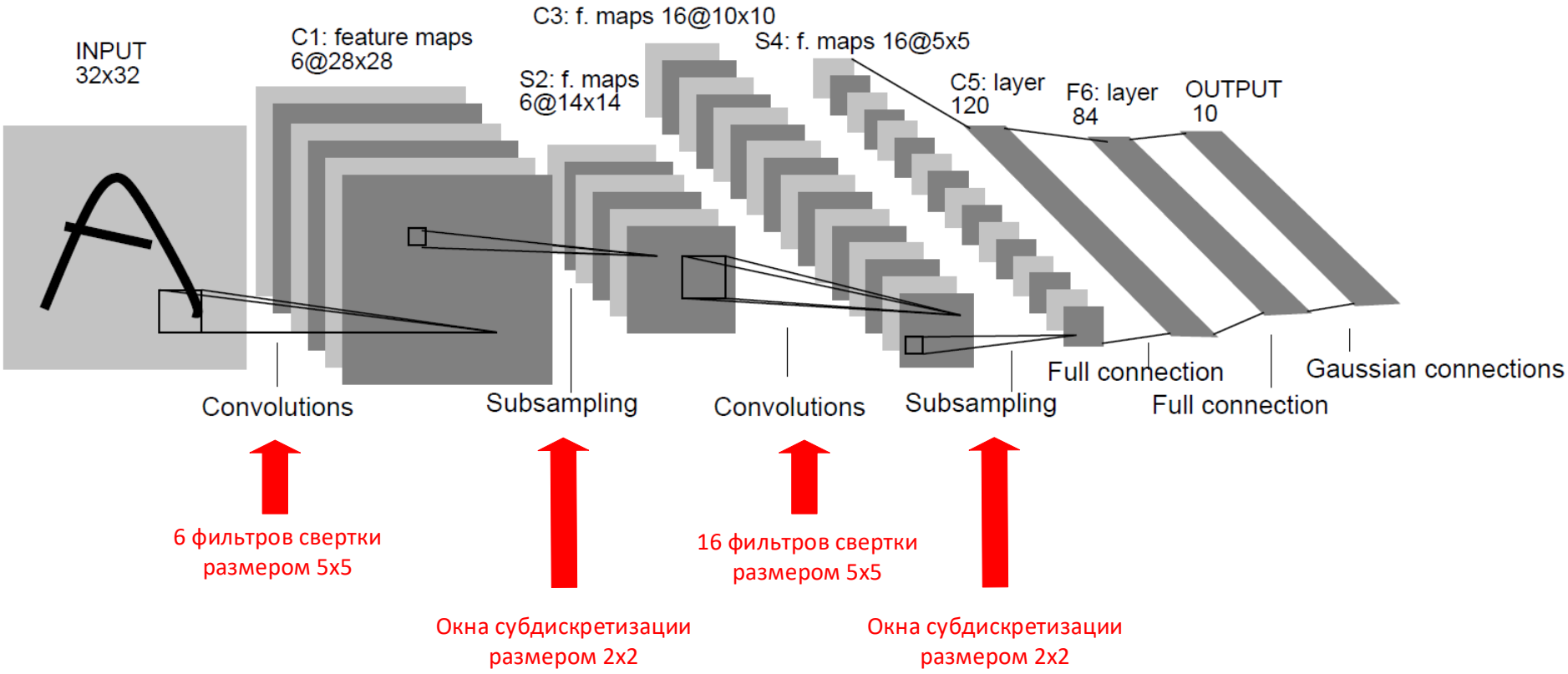
6 фильтров свертки
размером 5x5

Окна субдискретизации
размером 2x2

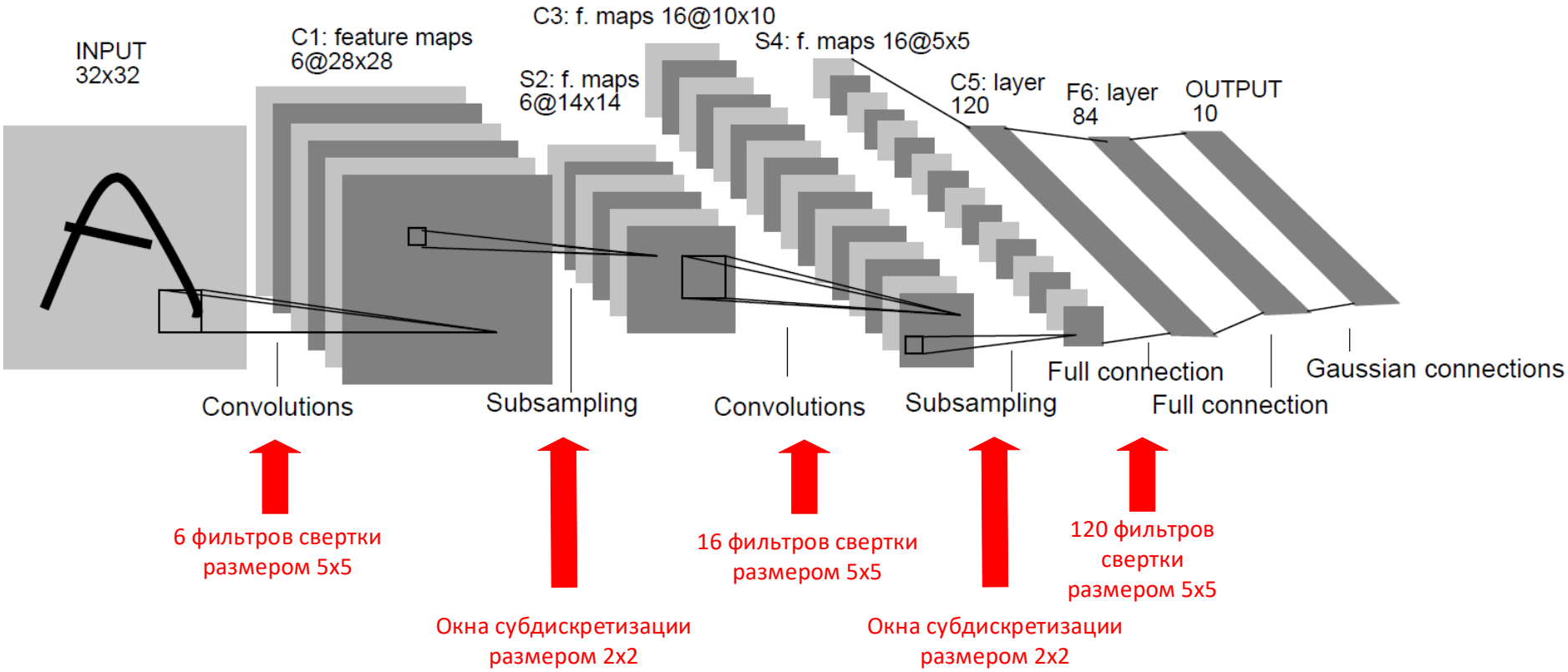
LeNet-5


INPUT
32x32
C1: feature maps
6@28x28
S2: f. maps
6@14x14
C3: f. maps 16@10x10
S4: f. maps 16@5x5
C5: layer
120
F6: layer
84
OUTPUT
10
Convolutions
Subsampling
Convolutions
Subsampling
Full connection
Full connection
Gaussian connections
6 фильтров свертки
размером 5x5
Окна субдискретизации
размером 2x2
16 фильтров свертки
размером 5x5

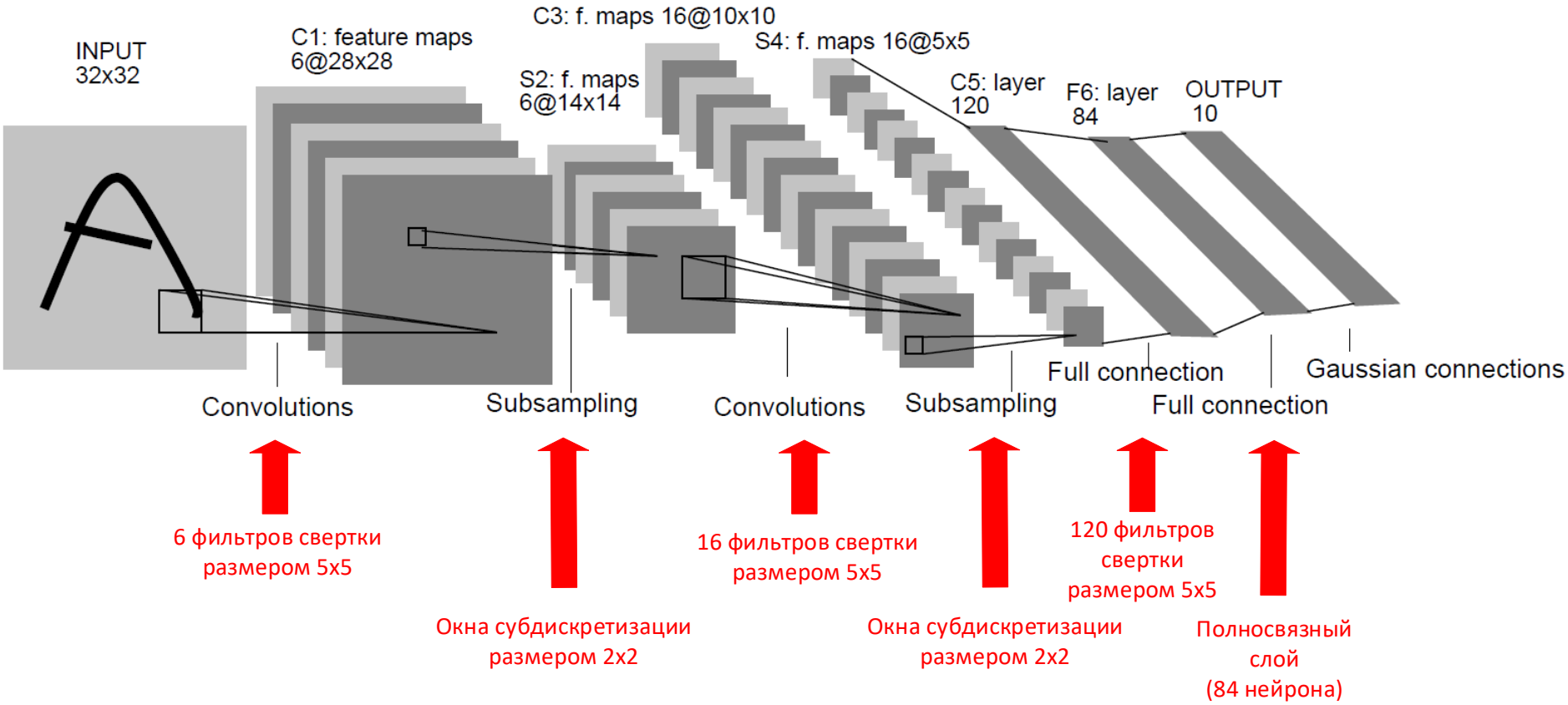
LeNet-5


INPUT
32x32
C1: feature maps
6@28x28
S2: f. maps
6@14x14
C3: f. maps 16@10x10
S4: f. maps 16@5x5
C5: layer
120
F6: layer
84
OUTPUT
10
Convolutions
Subsampling
Convolutions
Subsampling
Full connection
Full connection
Gaussian connections
6 фильтров свертки
размером 5x5
Окна субдискретизации
размером 2x2
16 фильтров свертки
размером 5x5
Окна субдискретизации
размером 2x2

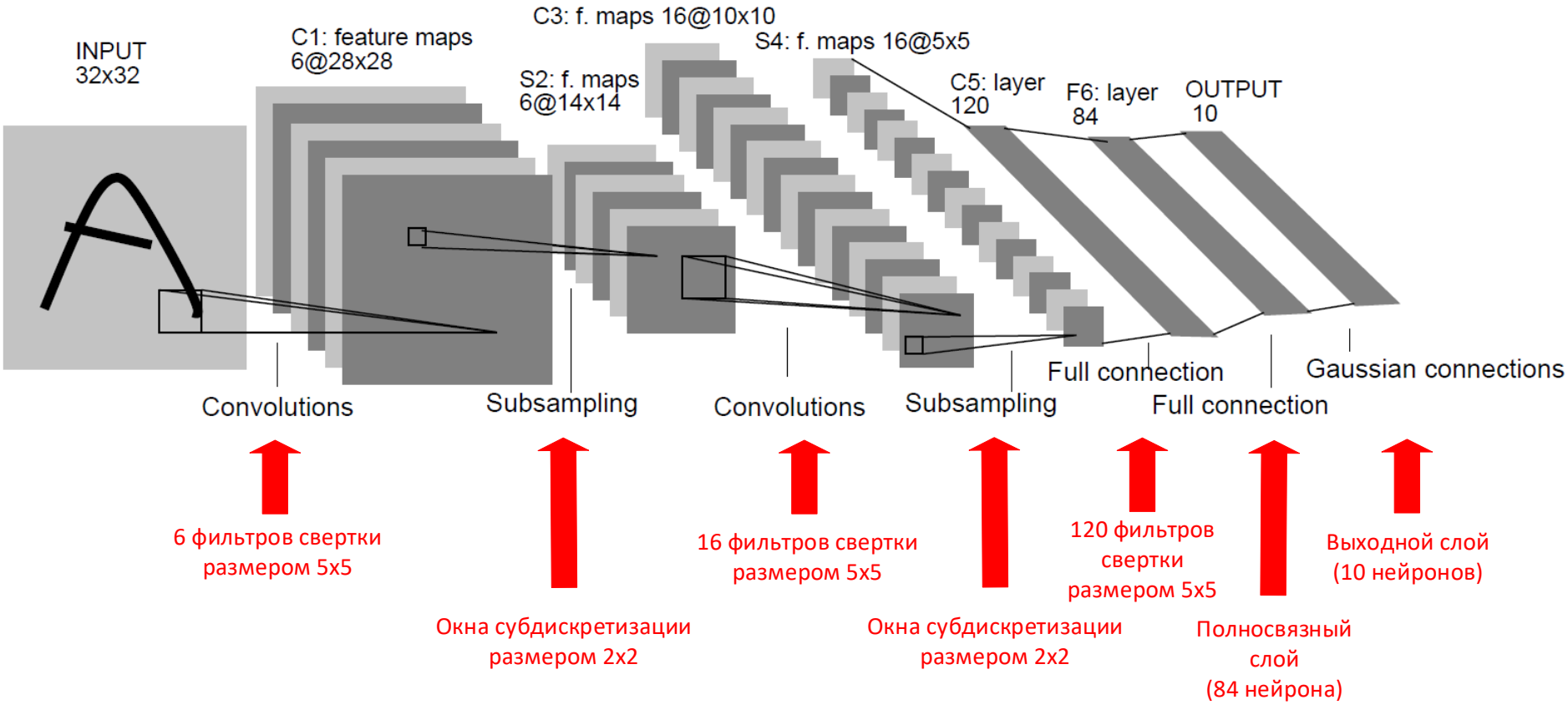
LeNet-5


INPUT
32x32
C1: feature maps
6@28x28
S2: f. maps
6@14x14
C3: f. maps 16@10x10
S4: f. maps 16@5x5
C5: layer
120
F6: layer
84
OUTPUT
10
Convolutions
Subsampling
Convolutions
Subsampling
Full connection
Full connection
Gaussian connections
6 фильтров свертки
размером 5x5
Окна субдискретизации
размером 2x2
16 фильтров свертки
размером 5x5
Окна субдискретизации
размером 2x2
120 фильтров
свертки
размером 5x5

LeNet-5


INPUT
32x32
C1: feature maps
6@28x28
S2: f. maps
6@14x14
C3: f. maps 16@10x10
S4: f. maps 16@5x5
C5: layer
120
F6: layer
84
OUTPUT
10
Convolutions
Subsampling
Convolutions
Subsampling
Full connection
Full connection
Gaussian connections
6 фильтров свертки
размером 5x5
Окна субдискретизации
размером 2x2
16 фильтров свертки
размером 5x5
Окна субдискретизации
размером 2x2
120 фильтров
свертки
размером 5x5
Полносвязный
слой
(84 нейрона)

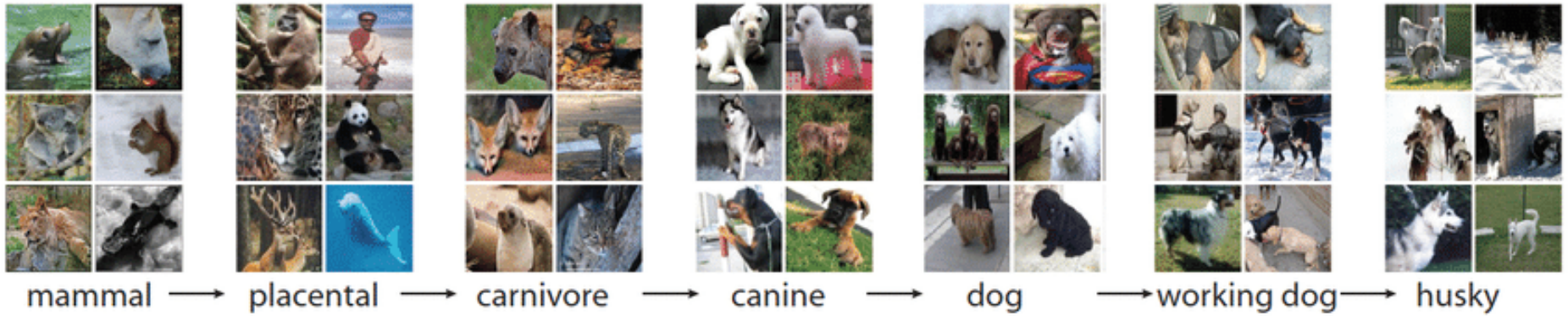
LeNet-5



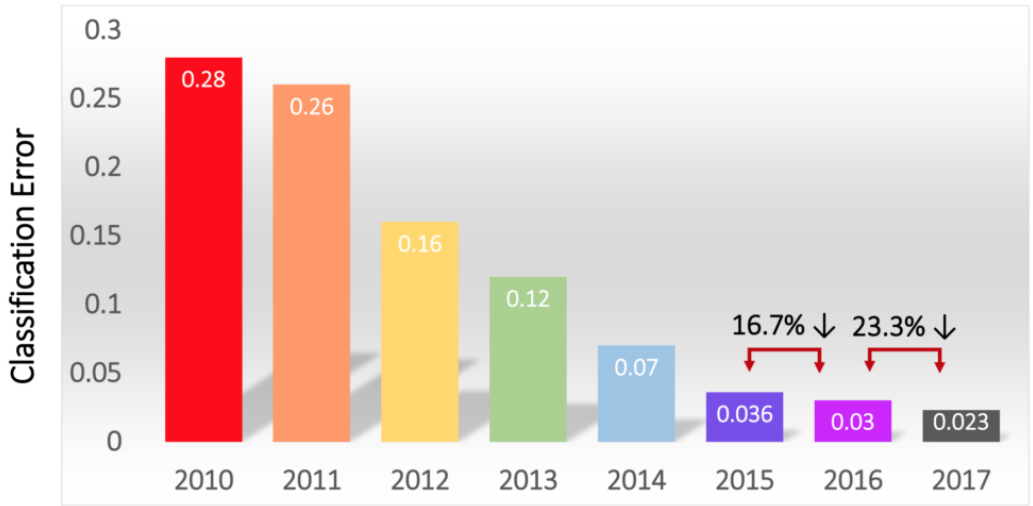
См. “LeNet5_pytorch.ipynb”

ImageNet

- ImageNet – база изображений, организованных в соответствии с иерархией понятий WordNet
  - https://www.image-net.org
- Более 14 млн изображений, более 21 тыс. иерархических категорий (синсетов), в среднем 650 изображений на категорию
- Более миллиона изображений размечено с ограничивающими объект прямоугольниками
- Соревнование: ImageNet Large Scale Visual Recognition Challenge (ILSVRC) (2010–2017)
  - 1,2 млн обучающих изображений, 1000 категорий
  - 50 000 валидационных изображений, 150 000 тестовых изображений
  - https://www.kaggle.com/c/imagenet-object-localization-challenge

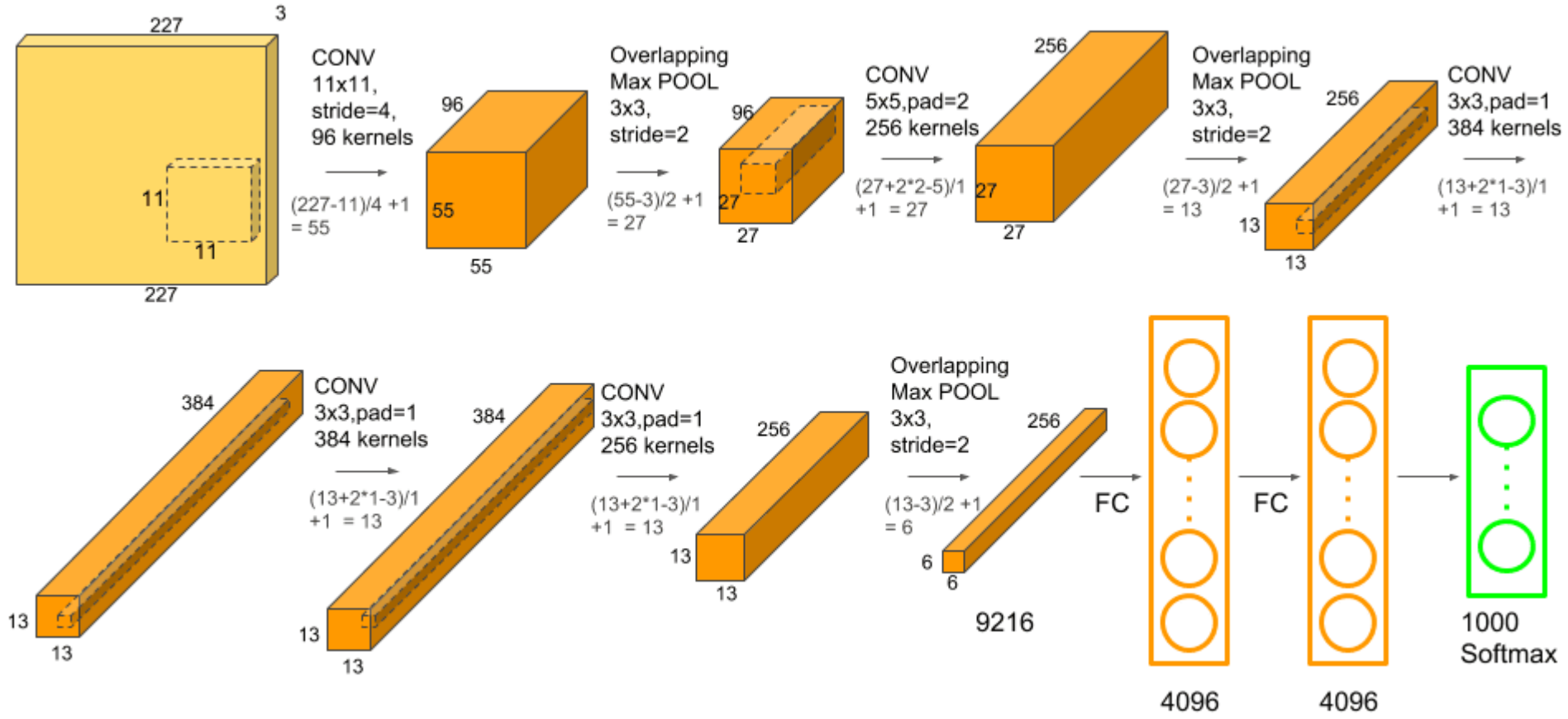

mammal
placental
carnivore
canine
dog
working dog
husky

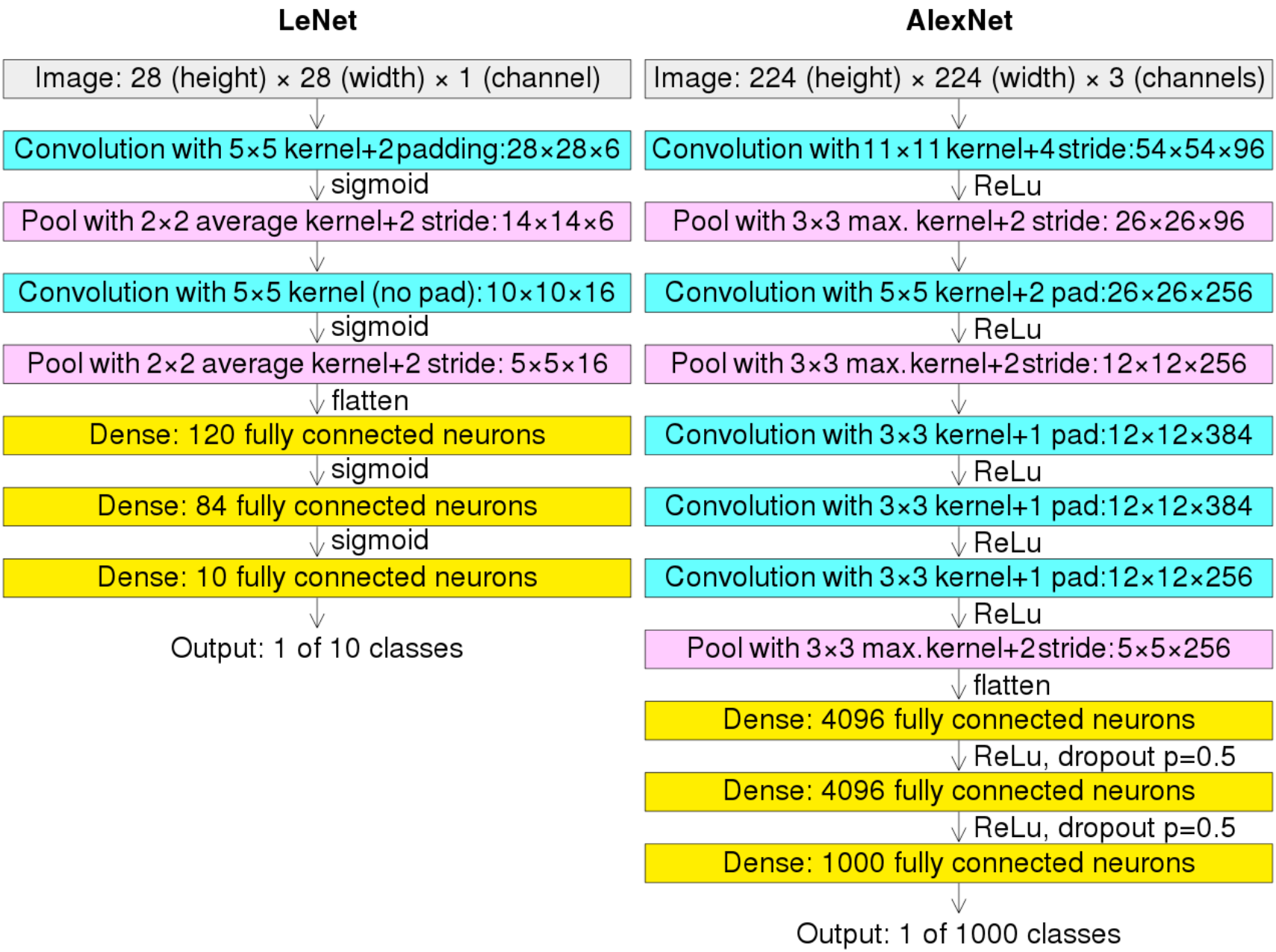

vehicle
craft
watercraft
sailing vessel
sailboat
trimaran


Classification Error
0.3
0.25
0.2
0.15
0.1
0.05
0
0.28
0.26
0.16
0.12
0.07
0.036
0.03
0.023
16.7% ↓
23.3% ↓
2010
2011
2012
2013
2014
2015
2016
2017

AlexNet

- Свёрточная нейронная сеть – победитель ILSVRC-2012, разработанная Алексом Крижевским (Alex Krizhevsky) и др.
  - Krizhevsky Alex, Sutskever Ilya, Hinton Geoffrey. ImageNet Classification with Deep Convolutional Neural Networks // NIPS 2012
  - 60M параметров

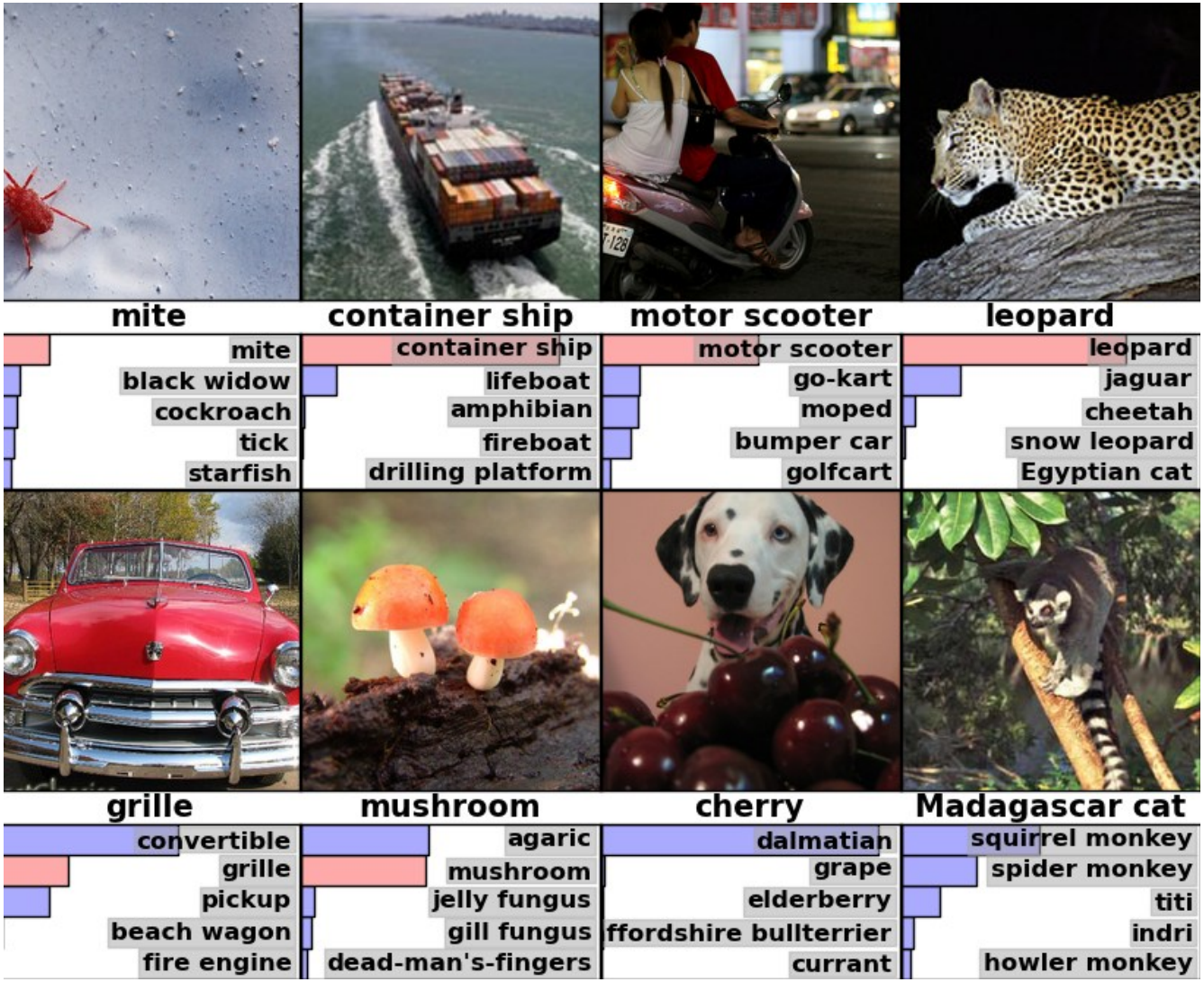



LeNet
Image: 28 (height) × 28 (width) × 1 (channel)
Convolution with 5×5 kernel+2 padding:28×28×6
sigmoid
Pool with 2×2 average kernel+2 stride:14×14×6
Convolution with 5×5 kernel (no pad):10×10×16
sigmoid
Pool with 2×2 average kernel+2 stride: 5×5×16
flatten
Dense: 120 fully connected neurons
sigmoid
Dense: 84 fully connected neurons
sigmoid
Dense: 10 fully connected neurons
Output: 1 of 10 classes
AlexNet
Image: 224 (height) × 224 (width) × 3 (channels)
Convolution with 11×11 kernel+4 stride:54×54×96
ReLu
Pool with 3×3 max. kernel+2 stride: 26×26×96
Convolution with 5×5 kernel+2 pad:26×26×256
ReLu
Pool with 3×3 max. kernel+2 stride:12×12×256
Convolution with 3×3 kernel+1 pad:12×12×384
ReLu
Convolution with 3×3 kernel+1 pad:12×12×384
ReLu
Convolution with 3×3 kernel+1 pad:12×12×256
ReLu
Pool with 3×3 max. kernel+2 stride:5×5×256
flatten
Dense: 4096 fully connected neurons
ReLu, dropout p=0.5
Dense: 4096 fully connected neurons
ReLu, dropout p=0.5
Dense: 1000 fully connected neurons
Output: 1 of 1000 classes

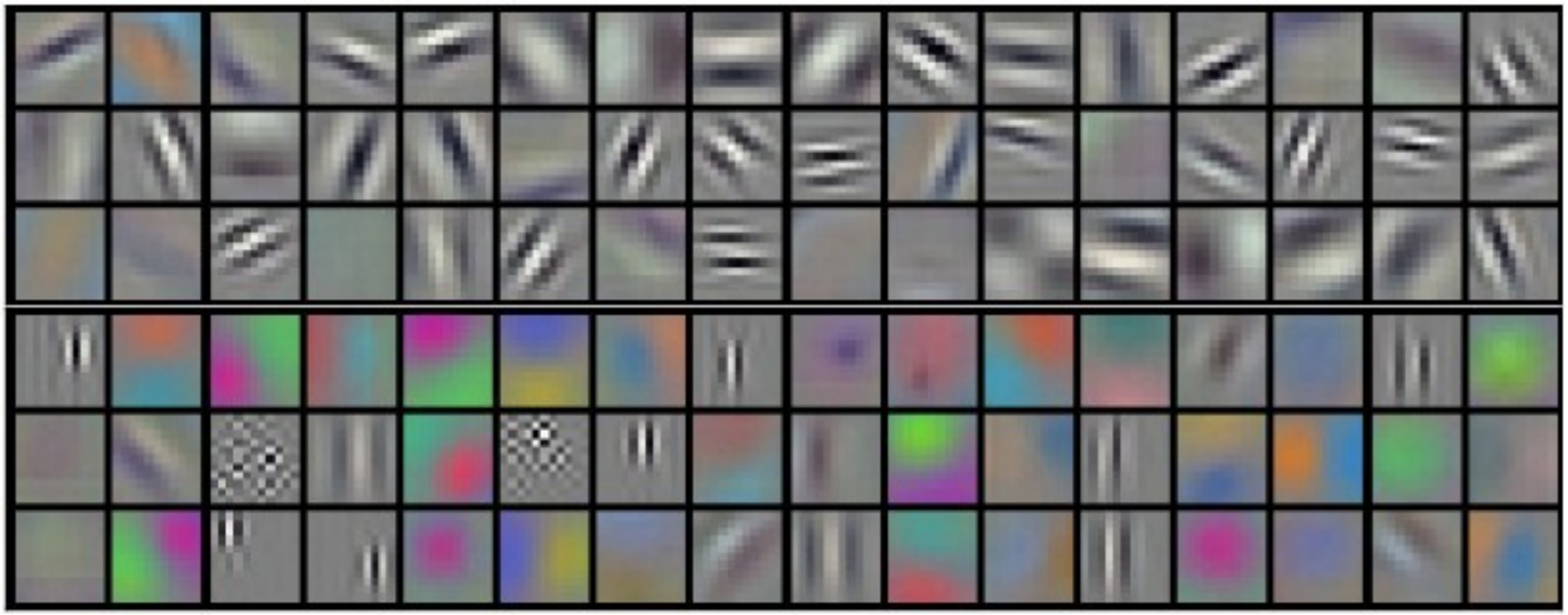
AlexNet

- **Аугментация** (data augmentation) – расширение набора данных за счет изменения существующих данных
  - борьба с переобучением
  - сокращение дисбаланса по классам
- В AlexNet:
  - вырезание фрагментов из исходных изображений
  - отражение фрагментов по горизонтали
  - изменение интенсивности RGB-каналов
  - $\rightarrow$ увеличение набора данных в 2048 раз
  - $\rightarrow$ предотвращение переобучения

AlexNet

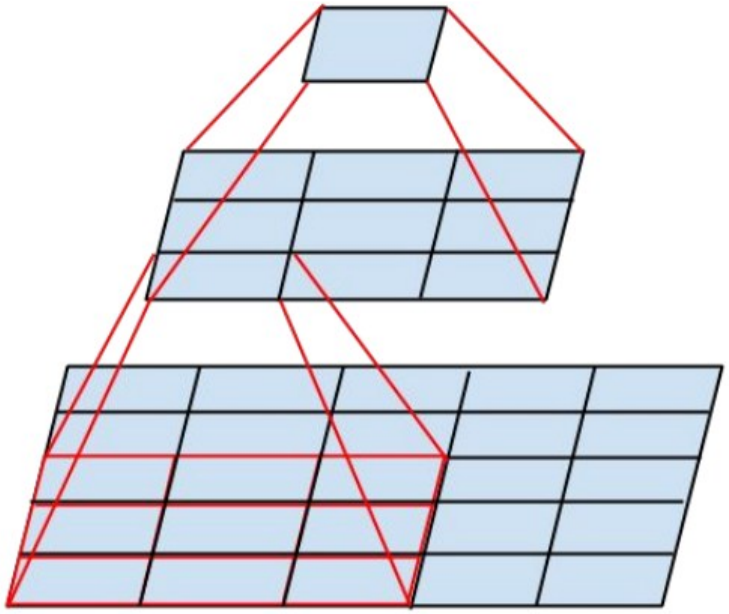

mite
mite
black widow
cockroach
tick
starfish
container ship
container ship
lifeboat
amphibian
fireboat
drilling platform
motor scooter
motor scooter
go-kart
moped
bumper car
golfcart
leopard
leopard
jaguar
cheetah
snow leopard
Egyptian cat
grille
convertible
grille
pickup
beach wagon
fire engine
mushroom
agaric
mushroom
jelly fungus
gill fungus
dead-man's-fingers
cherry
dalmatian
grape
elderberry
ffordshire bullterrier
currant
Madagascar cat
squirrel monkey
spider monkey
titi
indri
howler monkey

AlexNet – первый слой

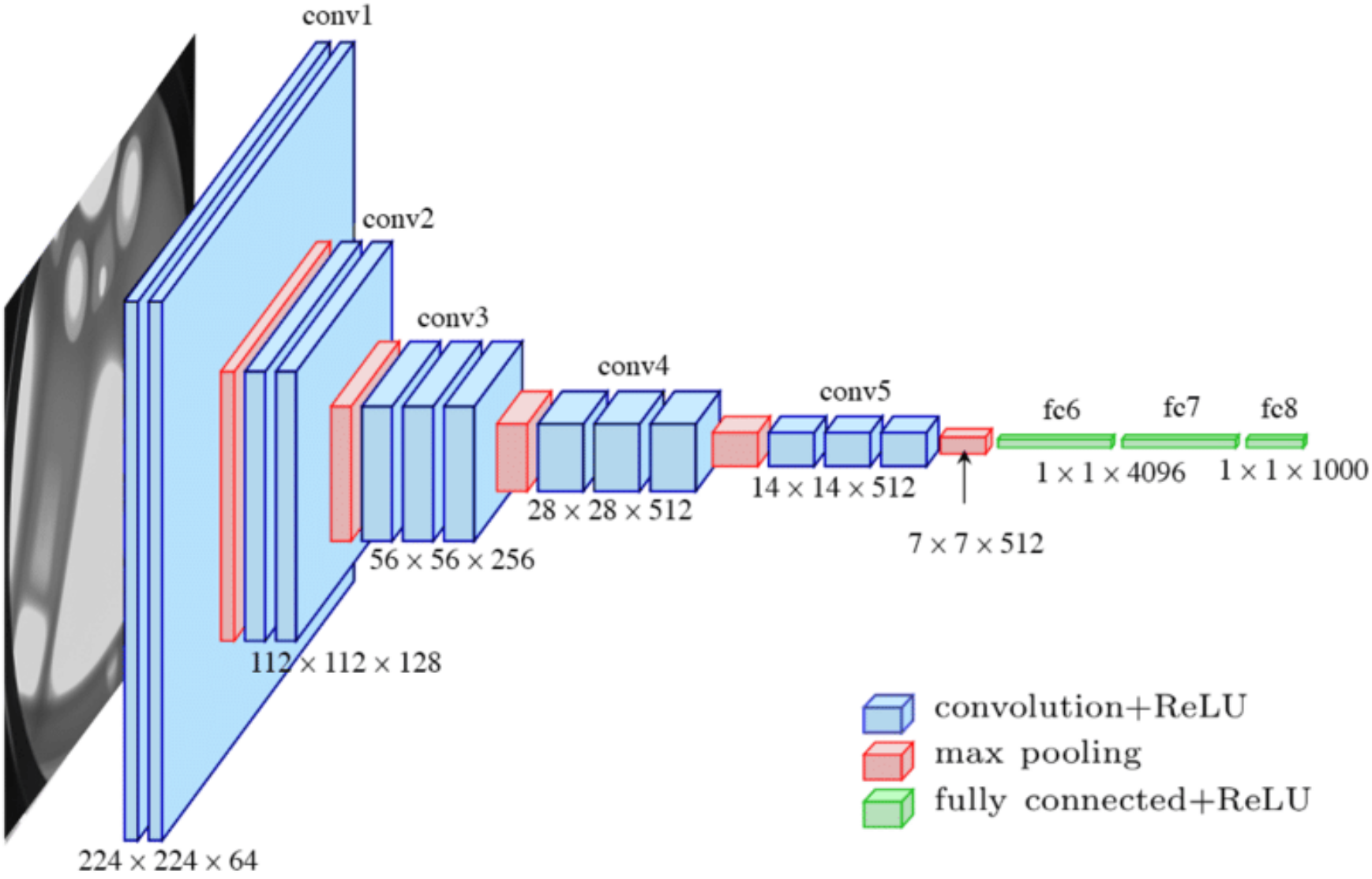


VGG

- VGG – Visual Geometry Group (University of Oxford)
  - Simonyan Karen, Zisserman Andrew. Very Deep Convolutional Networks for Large-Scale Image Recognition // ICLR 2015
  - 138M параметров (VGG-16)
- Большие свёртки (11x11 и 7x7) заменяются на последовательность свёрток 3x3 (меньше весов):



VGG-16

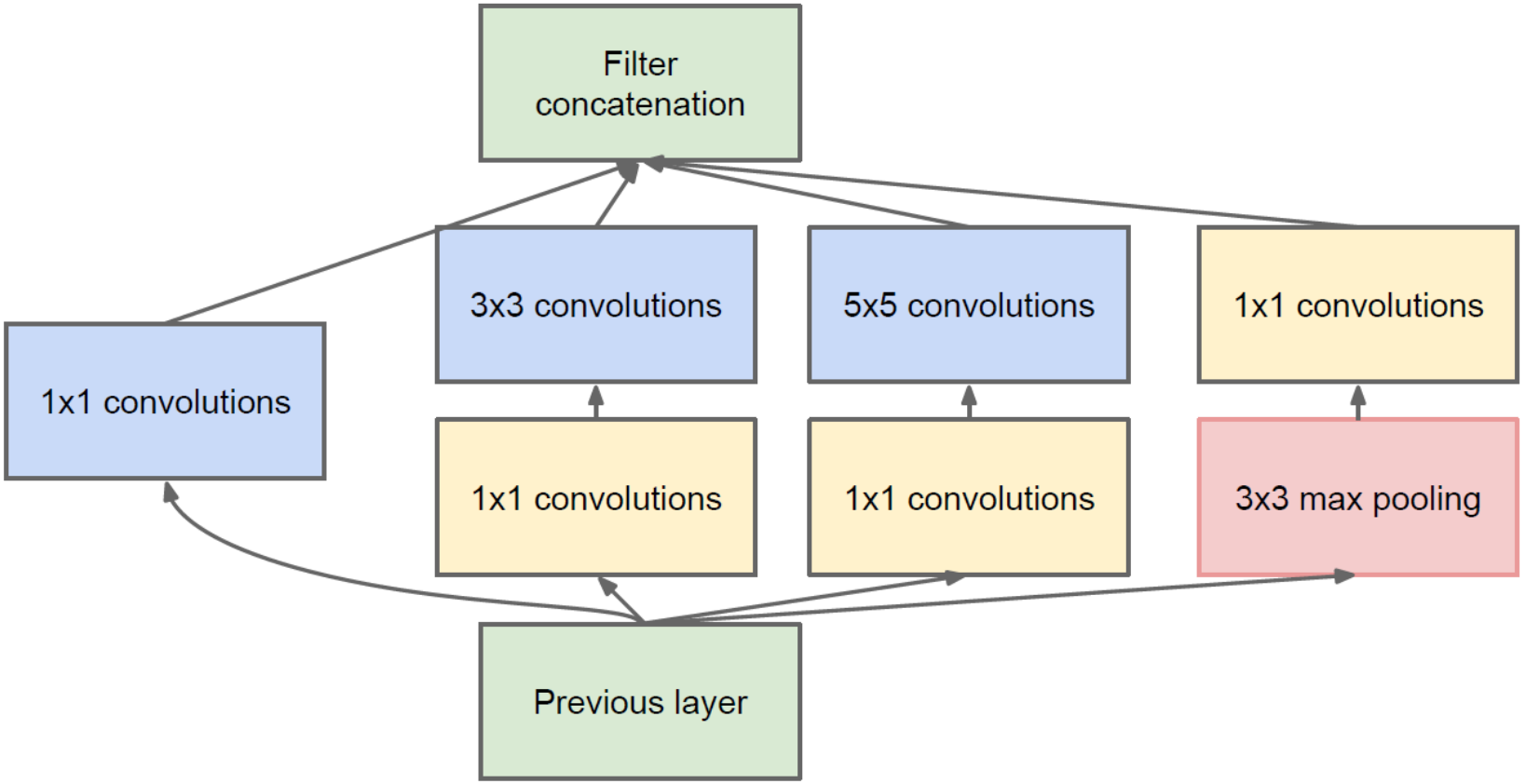

conv1
conv2
conv3
conv4
conv5
fc6
fc7
fc8
224 × 224 × 64
112 × 112 × 128
56 × 56 × 256
28 × 28 × 512
14 × 14 × 512
7 × 7 × 512
1 × 1 × 4096
1 × 1 × 1000
convolution+ReLU
max pooling
fully connected+ReLU

Inception (GoogLeNet)

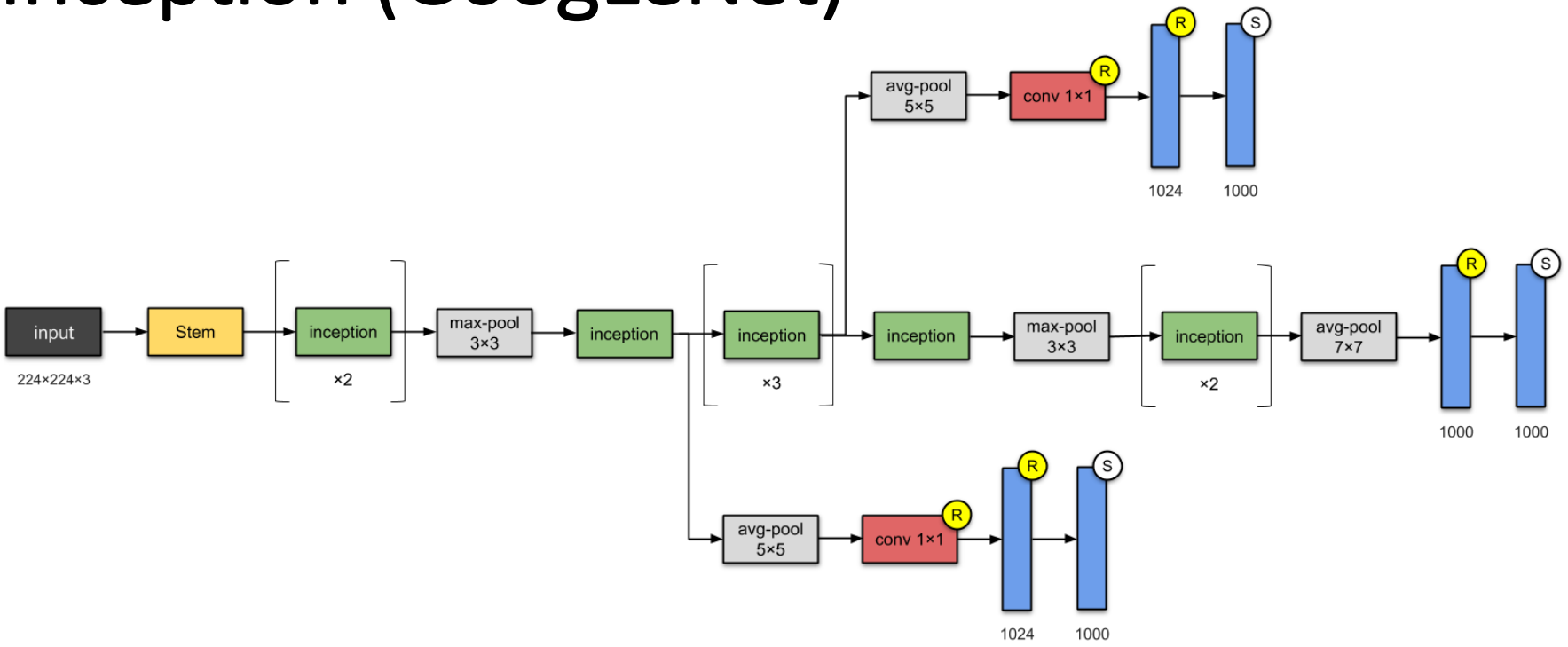
- Inception – нейронная сеть от Google
  - Christian Szegedy et al. Going Deeper with Convolutions // CVPR-2015
  - Christian Szegedy et al. Rethinking the Inception Architecture for Computer Vision // CVPR-2016
- Включает несколько модулей *Inception* (“Network in network”)
- 22 слоя, 6.8M параметров (Inception v1), 24M (Inception v3)
- Разные размеры свёрток на одном уровне (1x1, 3x3, 5x5)
- Свёртки 1x1:
  - нелинейность (ReLU)
  - снижение размерности (сокращение глубины)
- Дополнительные классификаторы для вычисления функции потерь (только в процессе обучения)

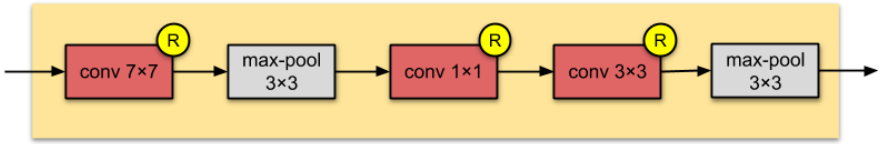
Inception (GoogLeNet)

- Модуль Inception

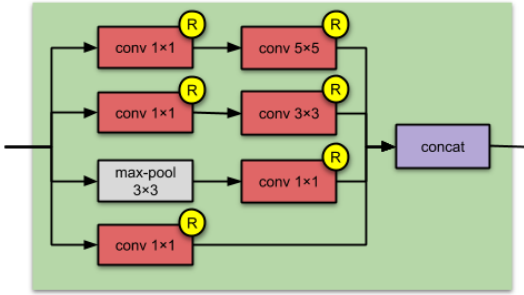


Inception (GoogLeNet)





Stem

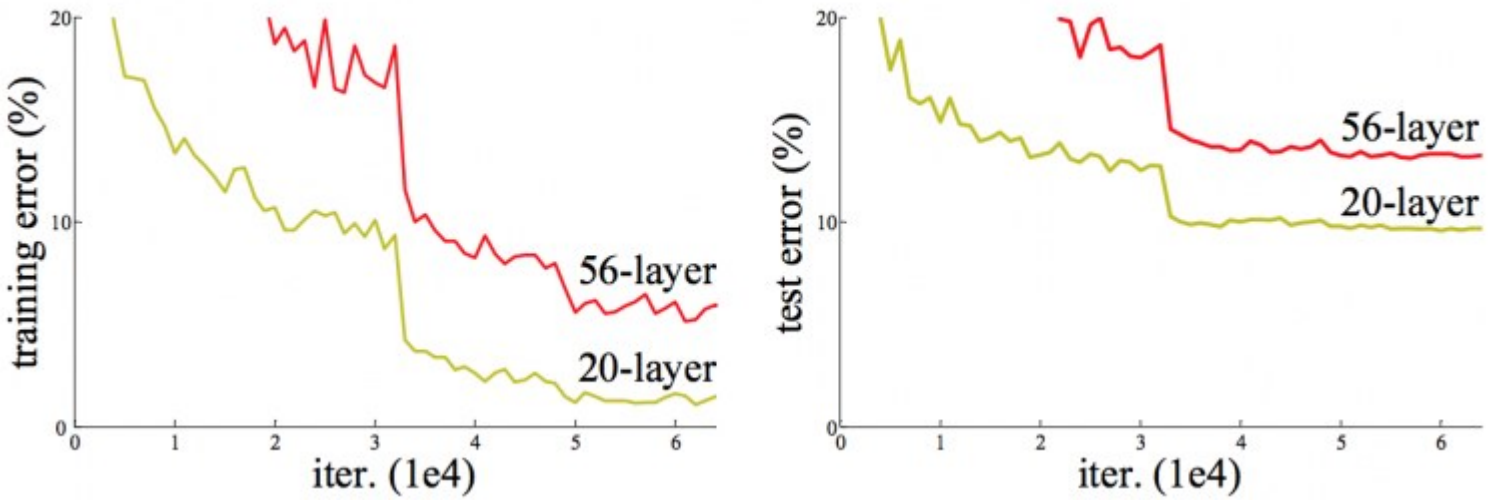


Inception module

https://towardsdatascience.com/illustrated-10-cnn-architectures-95d78ace614

Проблема затухающих градиентов (vanishing gradient problem)

- Решения:
  - функция активации ReLU
  - нормализация по мини-батчам (Иоффе и Сегеди)
  - инициализация весов (Ксавье Глоро, Каймин Хе)
- Но: с увеличением количества слоев ошибка увеличивается не только на тестовом множестве, но и на обучающем (не переобучение)



ResNet

- ResNet (Residual Network) – нейронная сеть от Microsoft Research
  - Kaiming He et al. Deep Residual Learning for Image Recognition // CVPR-2016
- Идея остаточного обучения (residual connections, skip connections):

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

где x – вход слоя, $f(x)$ – функция, которую вычисляет слой

- Если необходимо аппроксимировать функцию $g(x)$, то сеть учится аппроксимировать *остаток*:

$$f(x) = g(x) - x$$

- Градиент:

$$\frac{dy}{dx} = 1 + \frac{df(x)}{dx}$$

ResNet

- ResNet – 34, 50, 101, 152 слоя (152 – 60M параметров)