

Компьютерное зрение

Компьютерное зрение

Компьютерное зрение (Computer Vision, CV) - область искусственного интеллекта, задача которой получить полезную информацию из изображений

Компьютерное зрение вокруг нас:

- Face ID
- Поиск по картинке
- Распознавание текста с фото
- Автопилот

Ключевые задачи СВ

Классификация



Локализация



Детектирование



Сегментация



Кошка

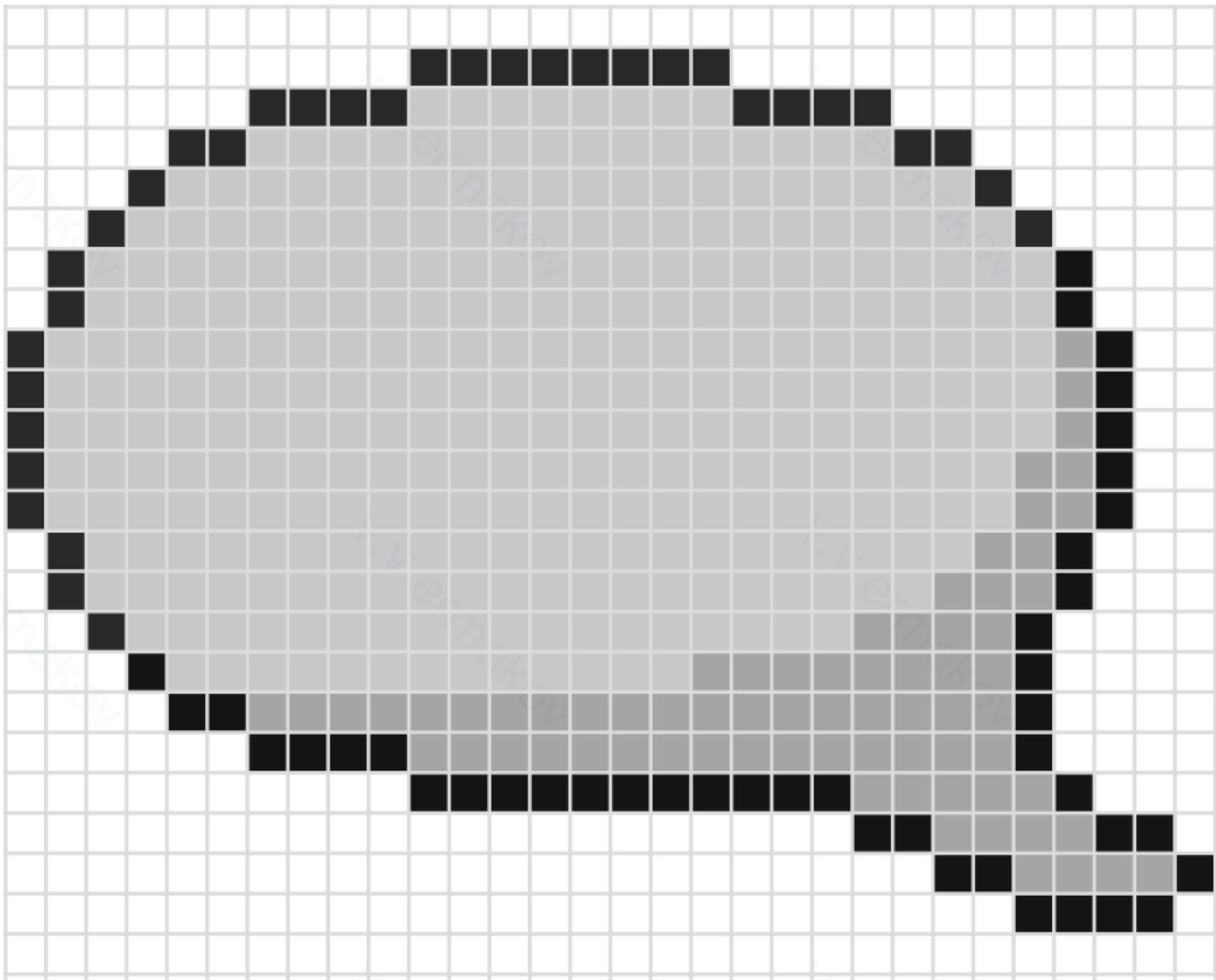
Кошка

Кошка, собака

Кошка, собака

Представление изображения

- Изображение представляется в виде **матрицы ($H \times W$)**
- Каждый элемент матрицы содержит **кодировку цвета** (как правило, от 0 до 255)
- На мониторе это число декодируется в **цвет** и называется **пикселием**



Представление изображения

- Изображение представляется в виде **матрицы** ($H \times W$)
 - Каждый элемент матрицы содержит **кодировку цвета** (как правило, от 0 до 255)
 - На мониторе это число декодируется в **цвет** и называется **пикселием**

Пример:

- 0 - черный
 - 180 - серый
 - 255 - белый

Представление изображения

- Изображение представляется в виде матрицы ($H \times W$)
 - Каждый элемент матрицы содержит кодировку цвета (как правило, от 0 до 255)
 - На мониторе это число декодируется в цвет и называется пикселием

Пример:

- 0 - черный
 - 180 - серый
 - 255 - белый

Цветные изображения

- Изображение представляется в виде **тензора ($3 \times H \times W$)**
- Каждый элемент матрицы содержит **кодировку цвета** (как правило, от 0 до 255)
- На мониторе это число декодируется в **цвет** и называется **пикселием**

Пример:

- [0, 0, 0] - черный
- [252, 15, 192] - розовый
- [255, 0, 0] - красный

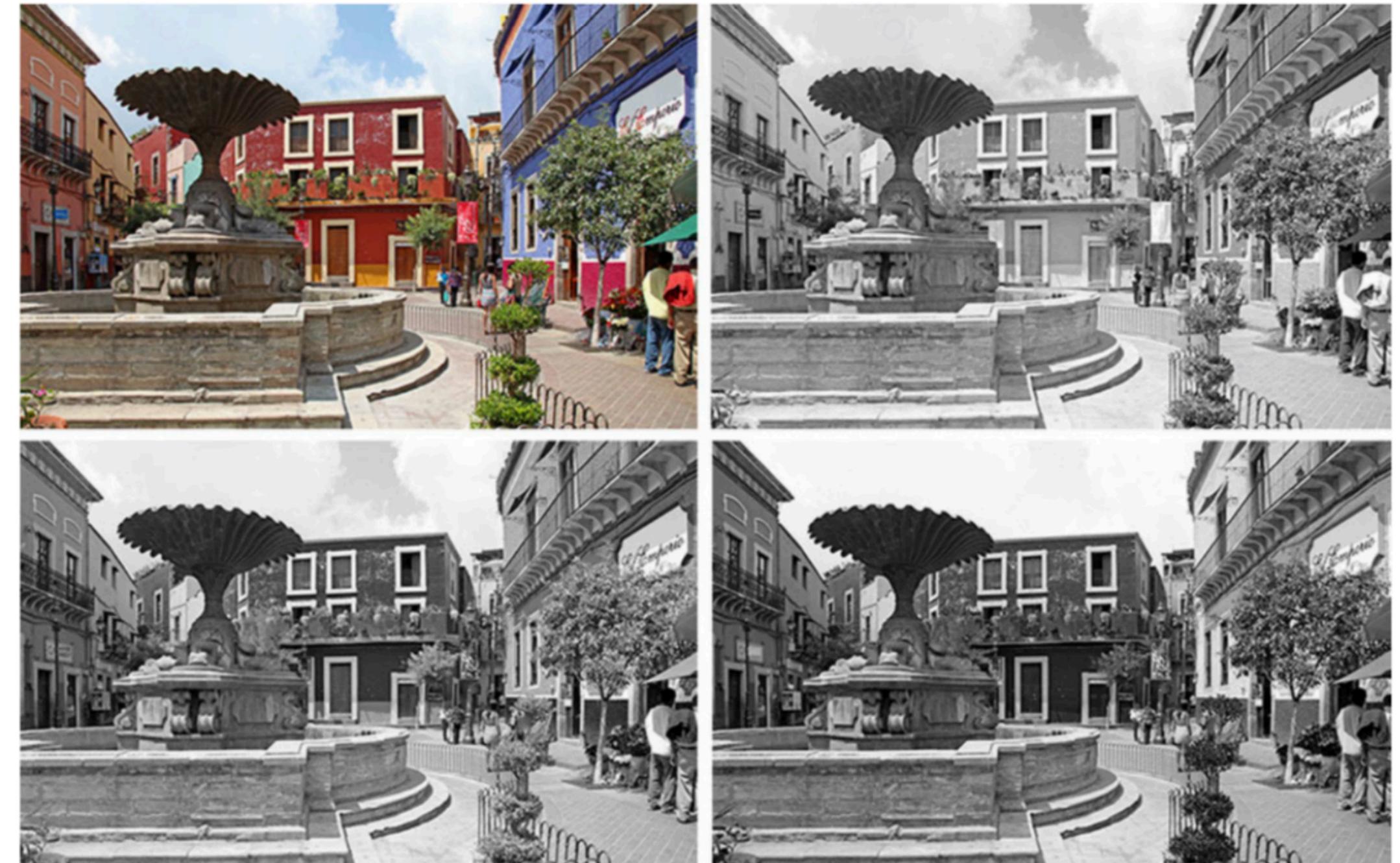


Цветные изображения

- Изображение представляется в виде **тензора ($3 \times H \times W$)**
- Каждый элемент матрицы содержит **кодировку цвета** (как правило, от 0 до 255)
- На мониторе это число декодируется в **цвет** и называется **пикселием**

Пример:

- [0, 0, 0] - черный
- [252, 15, 192] - розовый
- [255, 0, 0] - красный



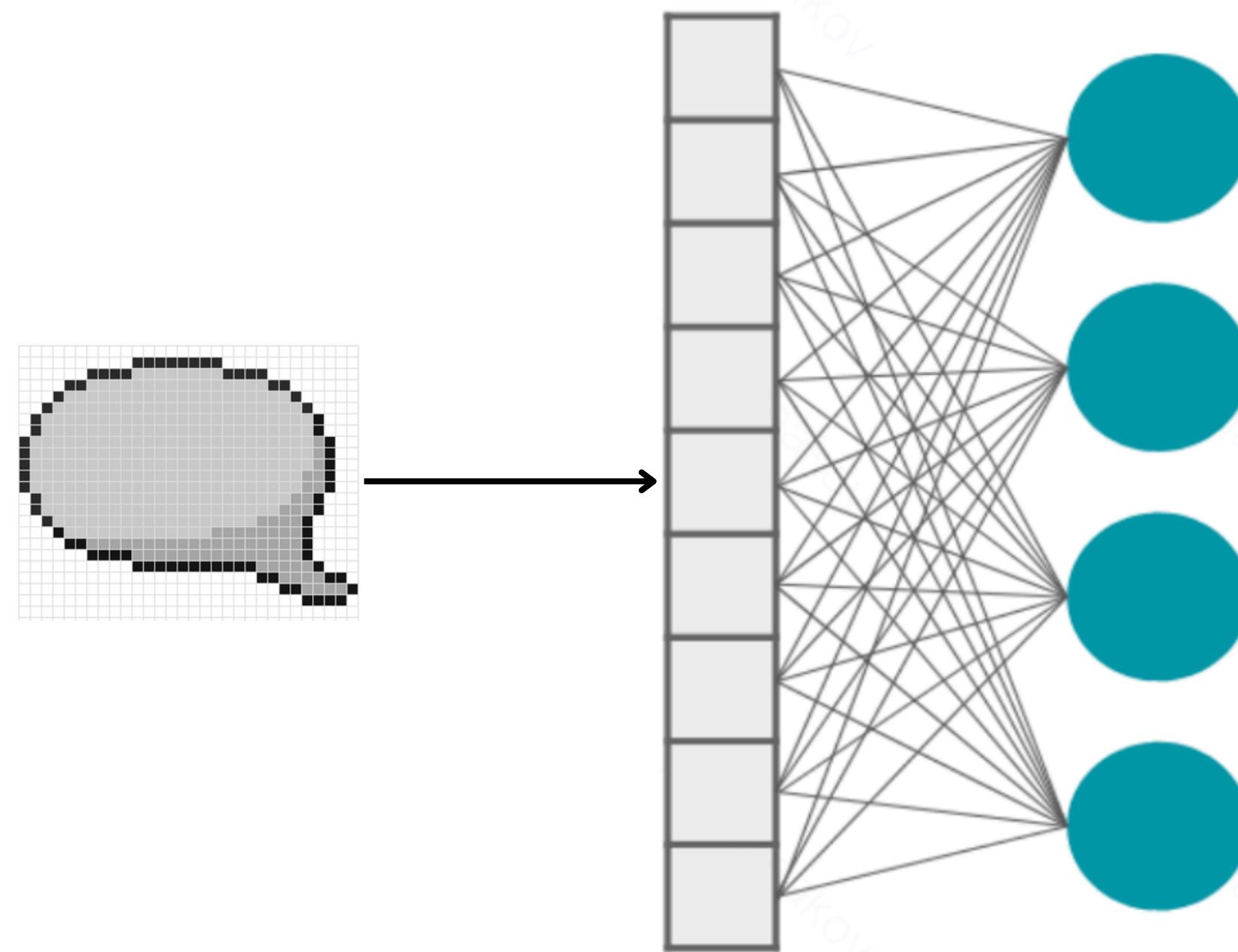
Green

Blue

Как обработать изображение?

Первая идея - использовать
полносвязную нейронную
сеть:

- развернем изображение в
один вектор
- подадим на вход сети



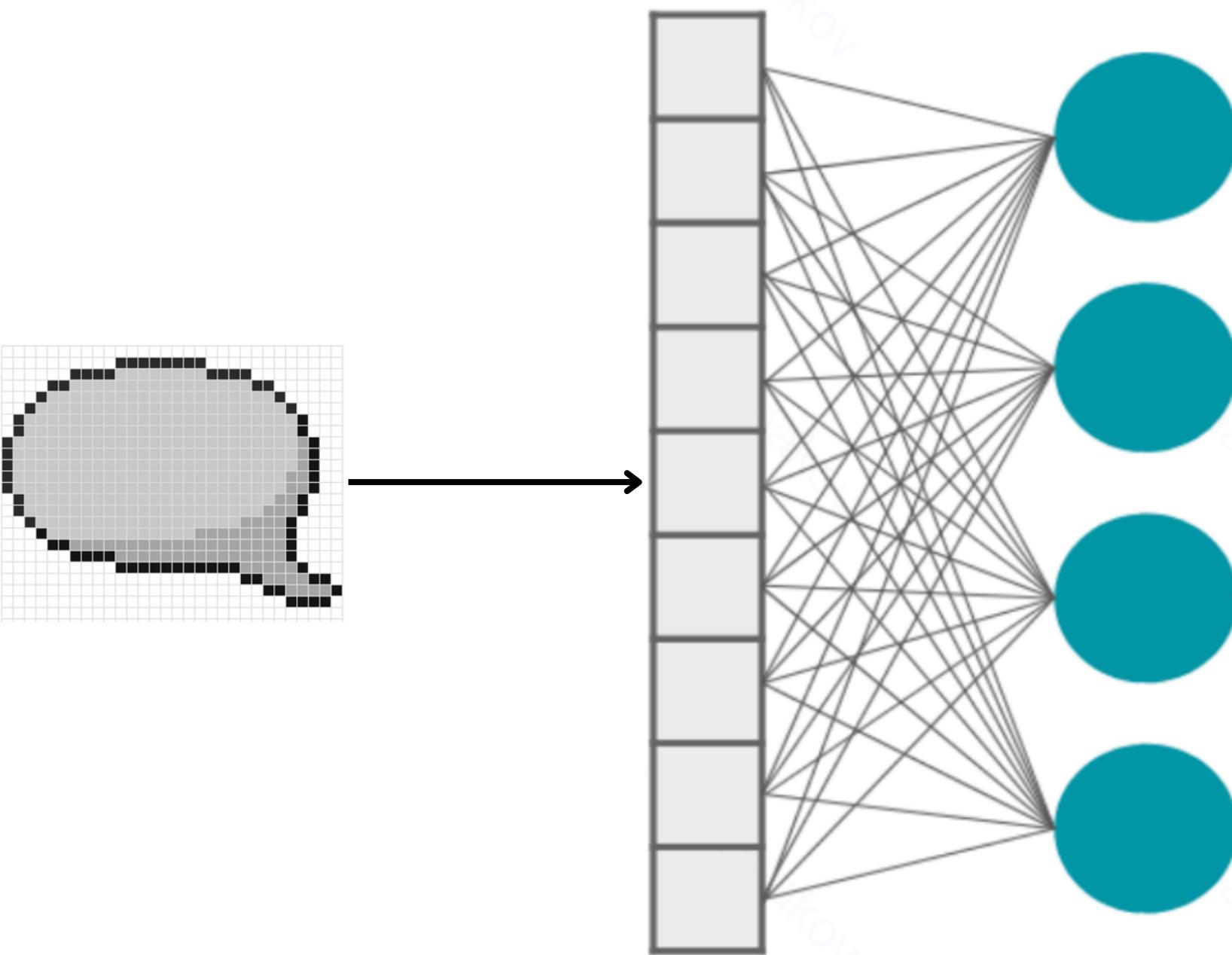
Как обработать изображение?

Первая идея - использовать
полносвязную нейронную
сеть:

- развернем изображение в
один вектор
- подадим на вход сети

Проблемы:

- Огромное количество
параметров: $3 \times 1920 \times$
1080 = 6 220 800. А это
только первый слой



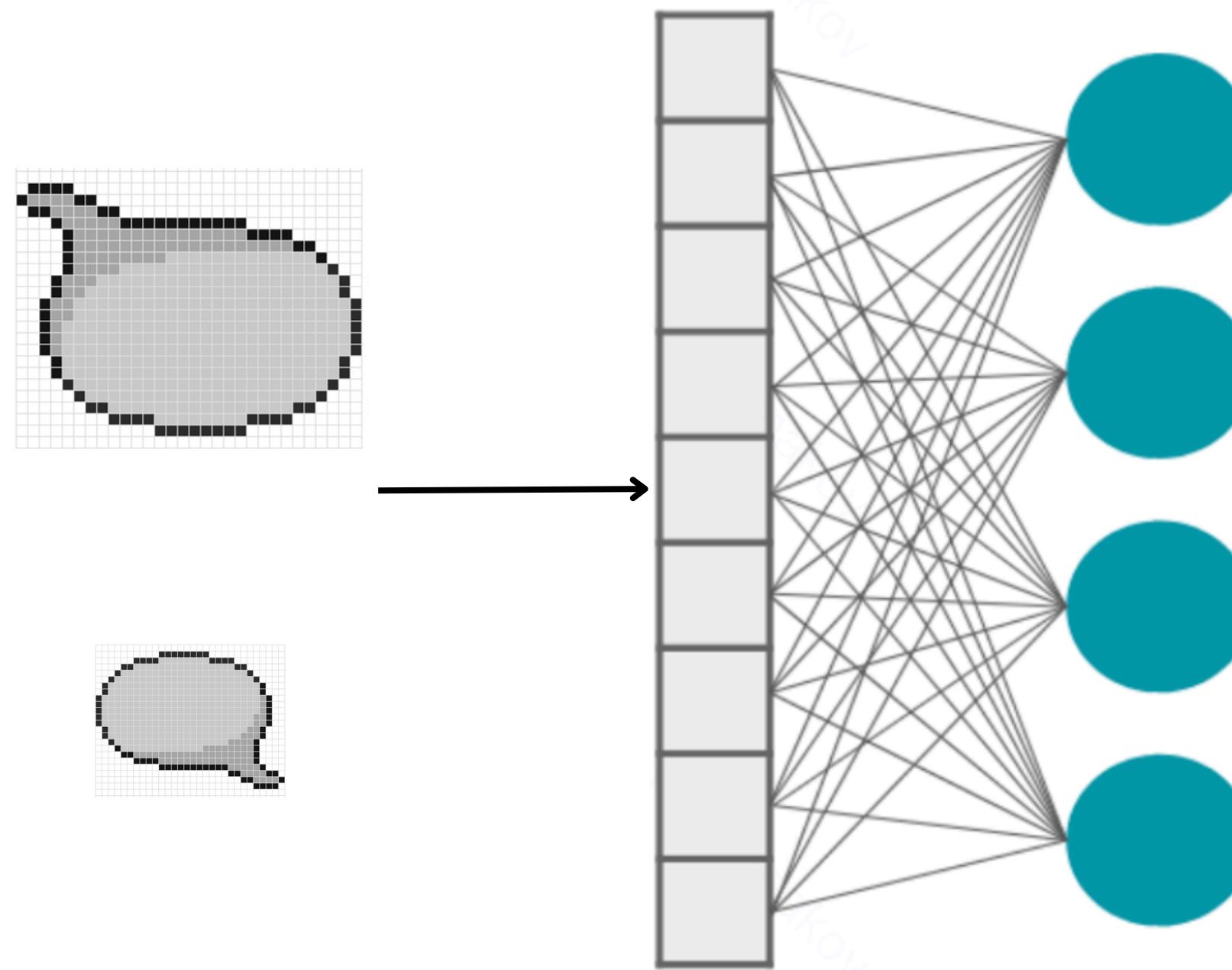
Как обработать изображение?

Первая идея - использовать
полносвязную нейронную
сеть:

- развернем изображение в
один вектор
- подадим на вход сети

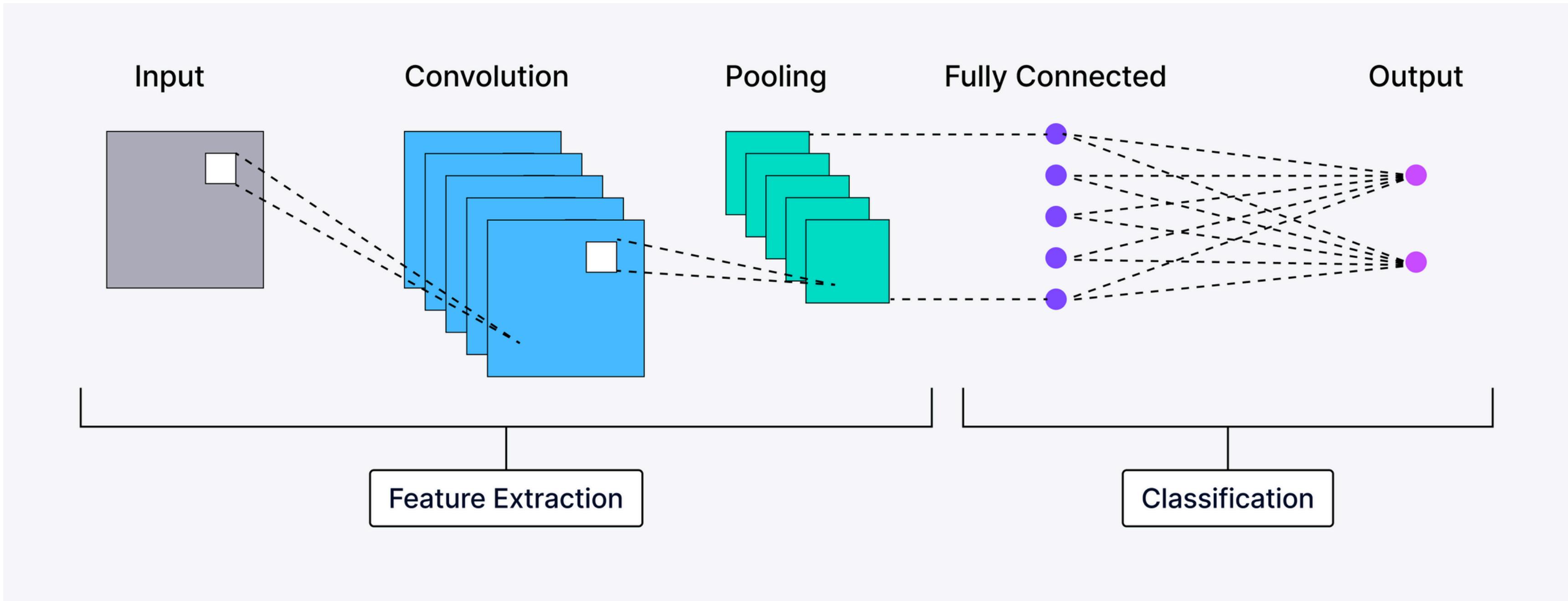
Проблемы:

- Огромное количество
параметров: $3 \times 1920 \times 1080 = 6\ 220\ 800$. А это
только первый слой
- Не учитывается структура
данных

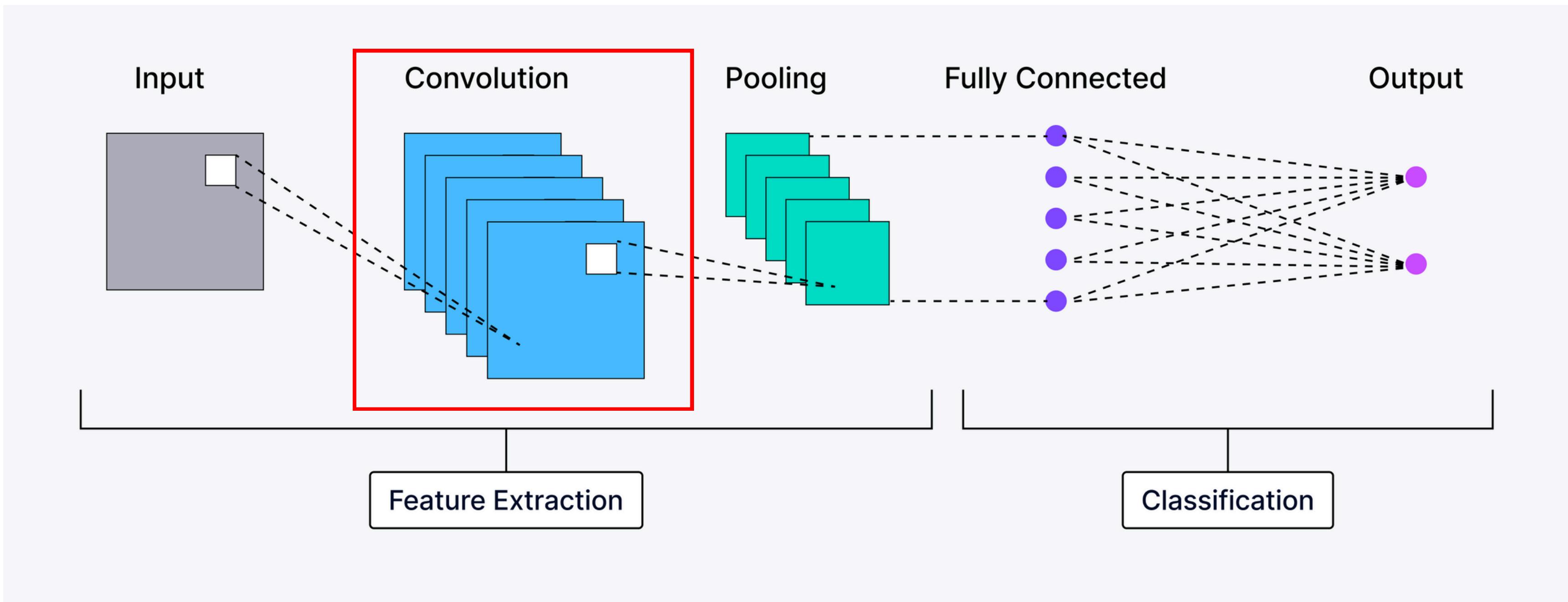


Сверточные нейронные сети

Структура сверточной нейронной сети



Свертка



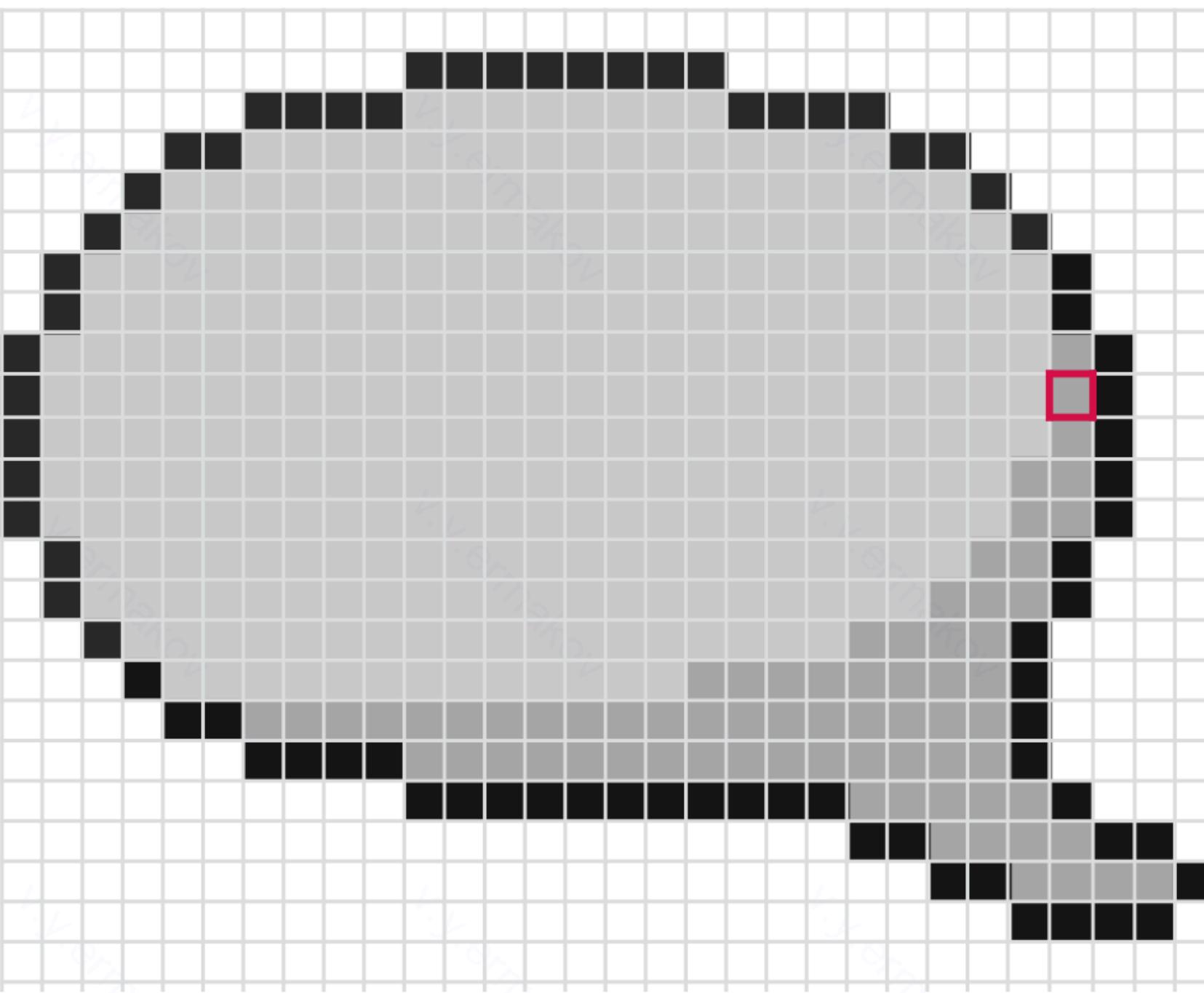
Свертка

Свертка - наложение на картинку фильтров (ядер свертки) для получения разных вариаций изображения.

Ядро свертки (фильтр) - матрица, значения которой определяют характер преобразования

Ядра свертки могут:

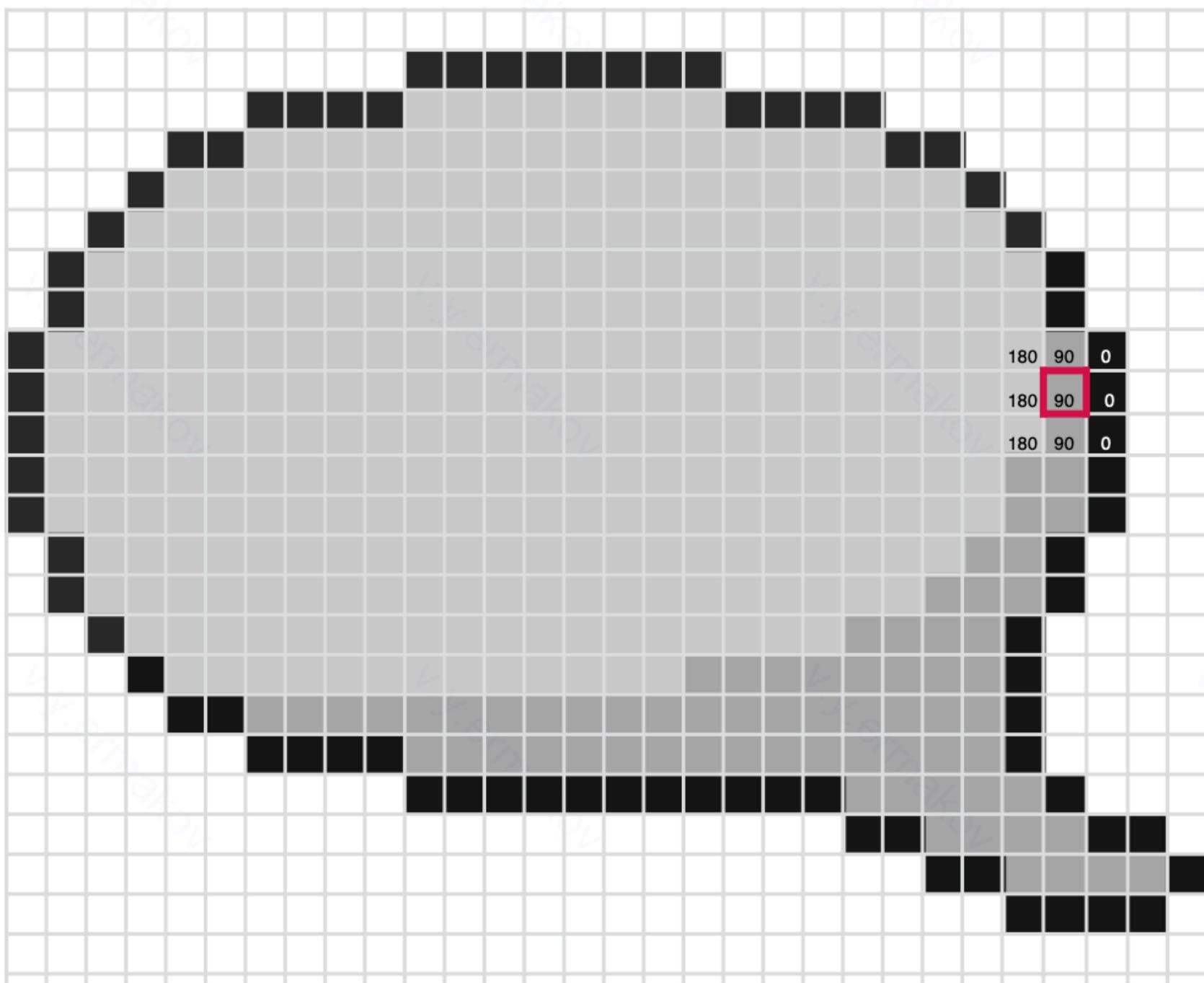
- выделить вертикальные и горизонтальные границы
- размыть изображения
- увеличить контраст



0	0	0
0	0	1
0	0	0

Фильтр

Применение одного фильтра - скалярное произведение области в картинке на значения в фильтре.
Фильтр в примере сдвигает пиксель влево

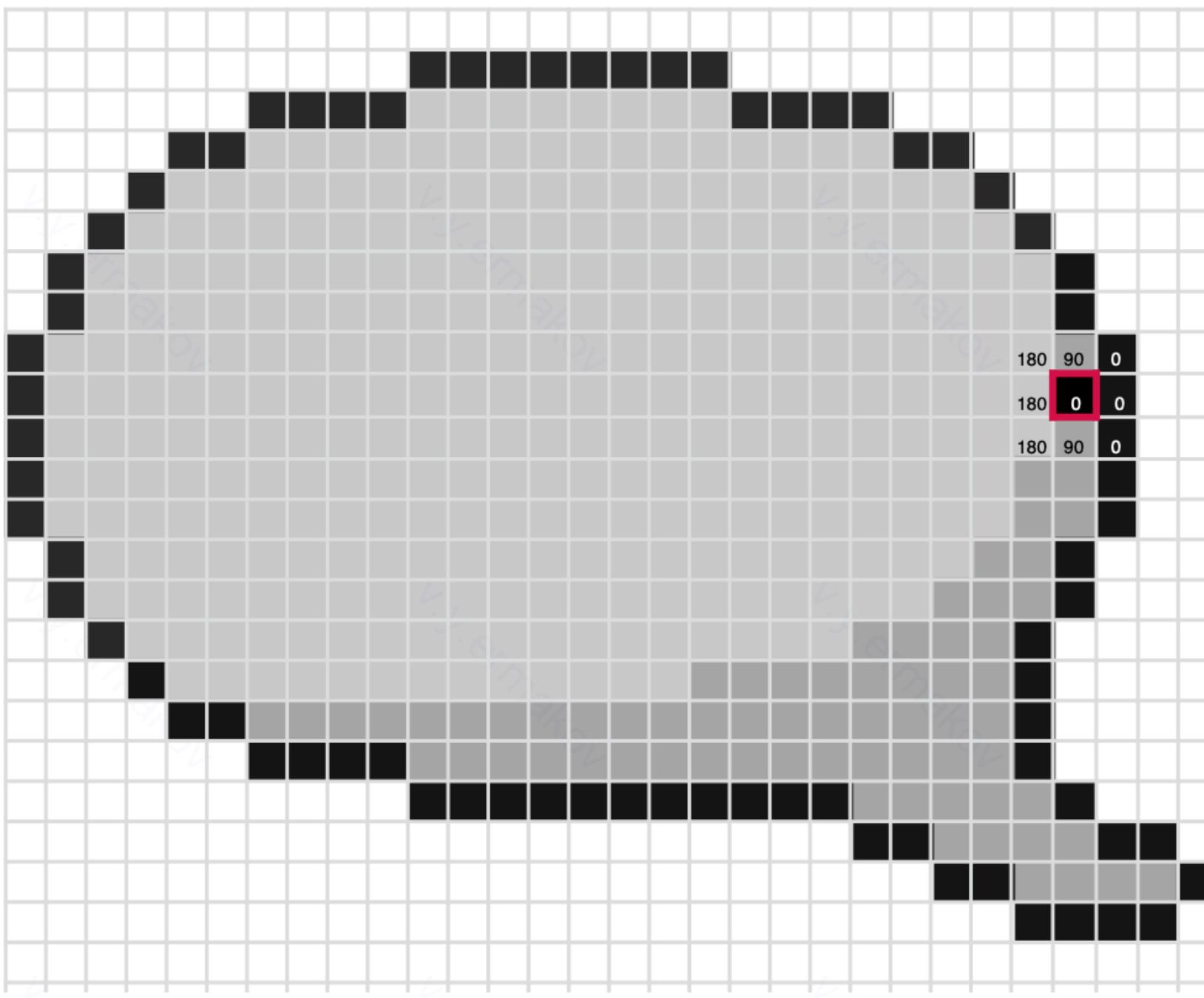


0	0	0
0	0	1
0	0	0

$$\begin{aligned} & 0 \times 180 + 0 \times 90 + 0 \times 0 \\ & + 0 \times 180 + 0 \times 90 + 1 \times 0 \\ & + 0 \times 180 + 0 \times 90 + 0 \times 0 \\ & = 0 \end{aligned}$$

Фильтр

Применение одного фильтра - скалярное произведение области в картинке на значения в фильтре.
Фильтр в примере сдвигает пиксель влево

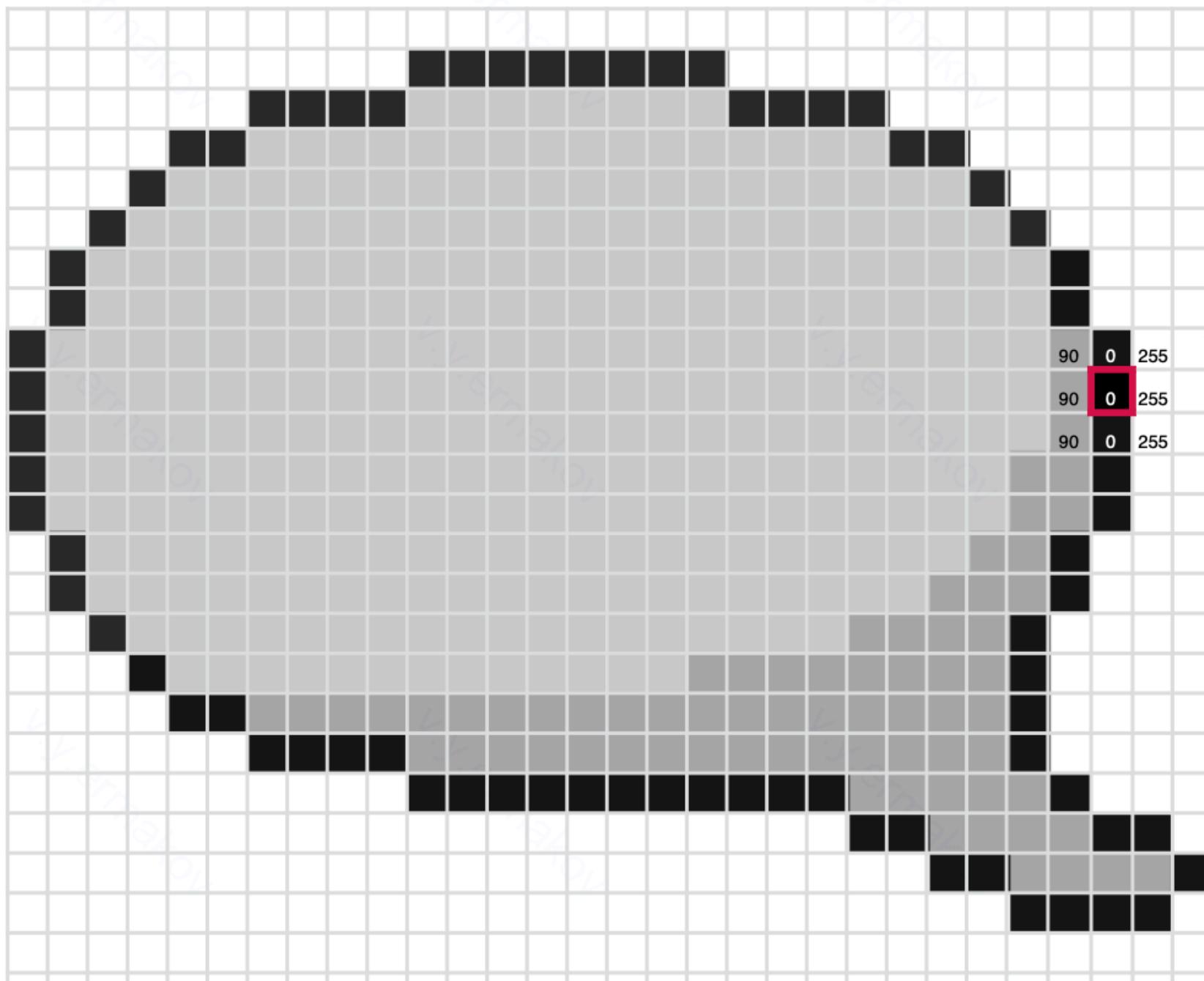


0	0	0
0	0	1
0	0	0

$$\begin{aligned} 0 \times 180 + 0 \times 90 + 0 \times 0 \\ + 0 \times 180 + 0 \times 90 + 1 \times 0 \\ + 0 \times 180 + 0 \times 90 + 0 \times 0 \\ = 0 \end{aligned}$$

Фильтр

Применение одного фильтра - скалярное произведение области в картинке на значения в фильтре.
Фильтр в примере сдвигает пиксель влево

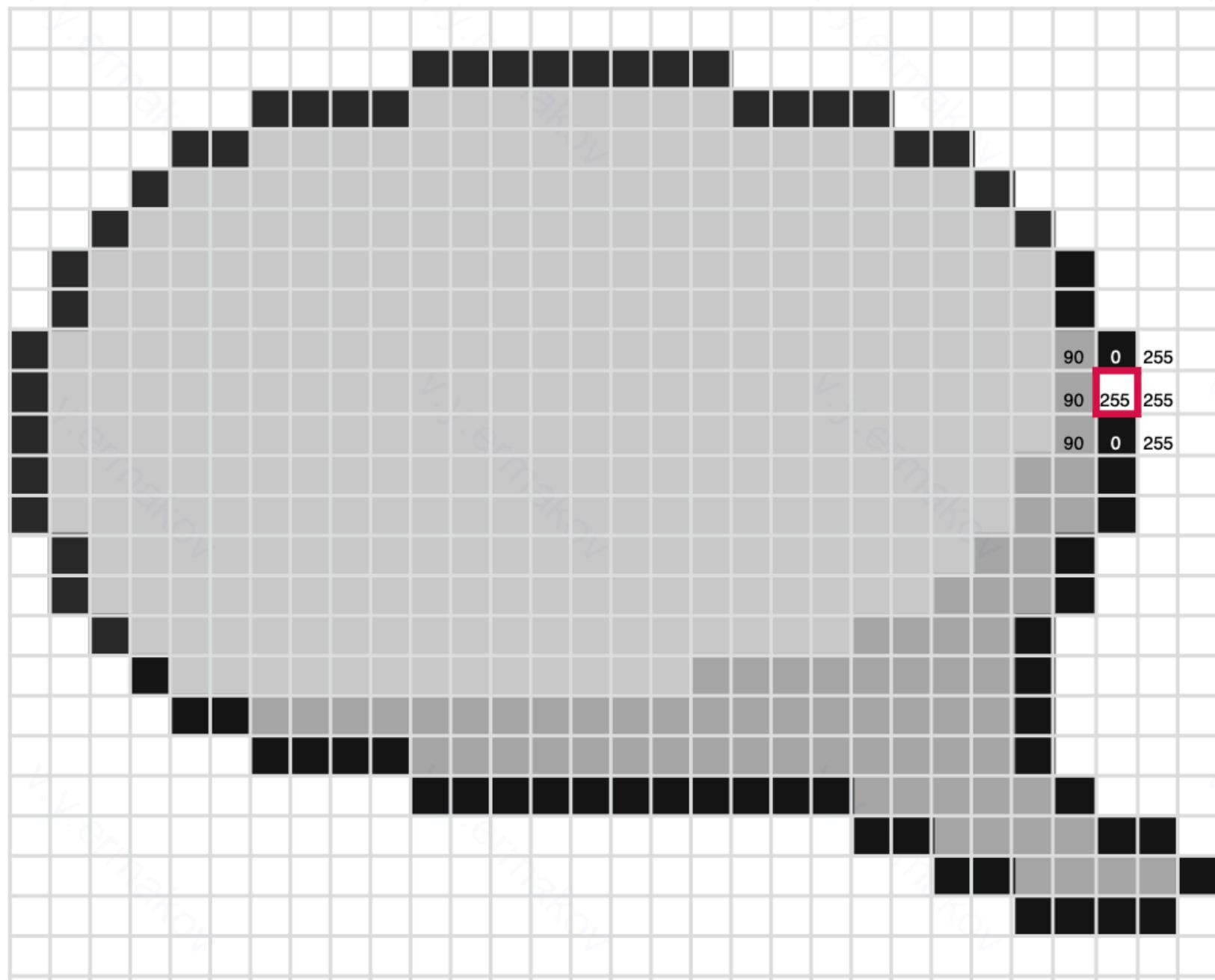


0	0	0
90	0	255
0	0	1

$$\begin{aligned} & 0 \times 90 + 0 \times 0 + 0 \times 255 \\ & + 0 \times 90 + 0 \times 0 + 1 \times 255 \\ & + 0 \times 90 + 0 \times 0 + 0 \times 255 \\ & = 255 \end{aligned}$$

Фильтр

Применение одного фильтра - скалярное произведение области в картинке на значения в фильтре.
Фильтр в примере сдвигает пиксель влево



0	0	0
90	255	255
0	0	1
0	0	0

$$\begin{aligned} & 0 \times 90 + 0 \times 0 + 0 \times 255 \\ & + 0 \times 90 + 0 \times 0 + 1 \times 255 \\ & + 0 \times 90 + 0 \times 0 + 0 \times 255 \\ & = 255 \end{aligned}$$

Свертка

Фильтр применяется для
каждой области изображения
Результат - матрица

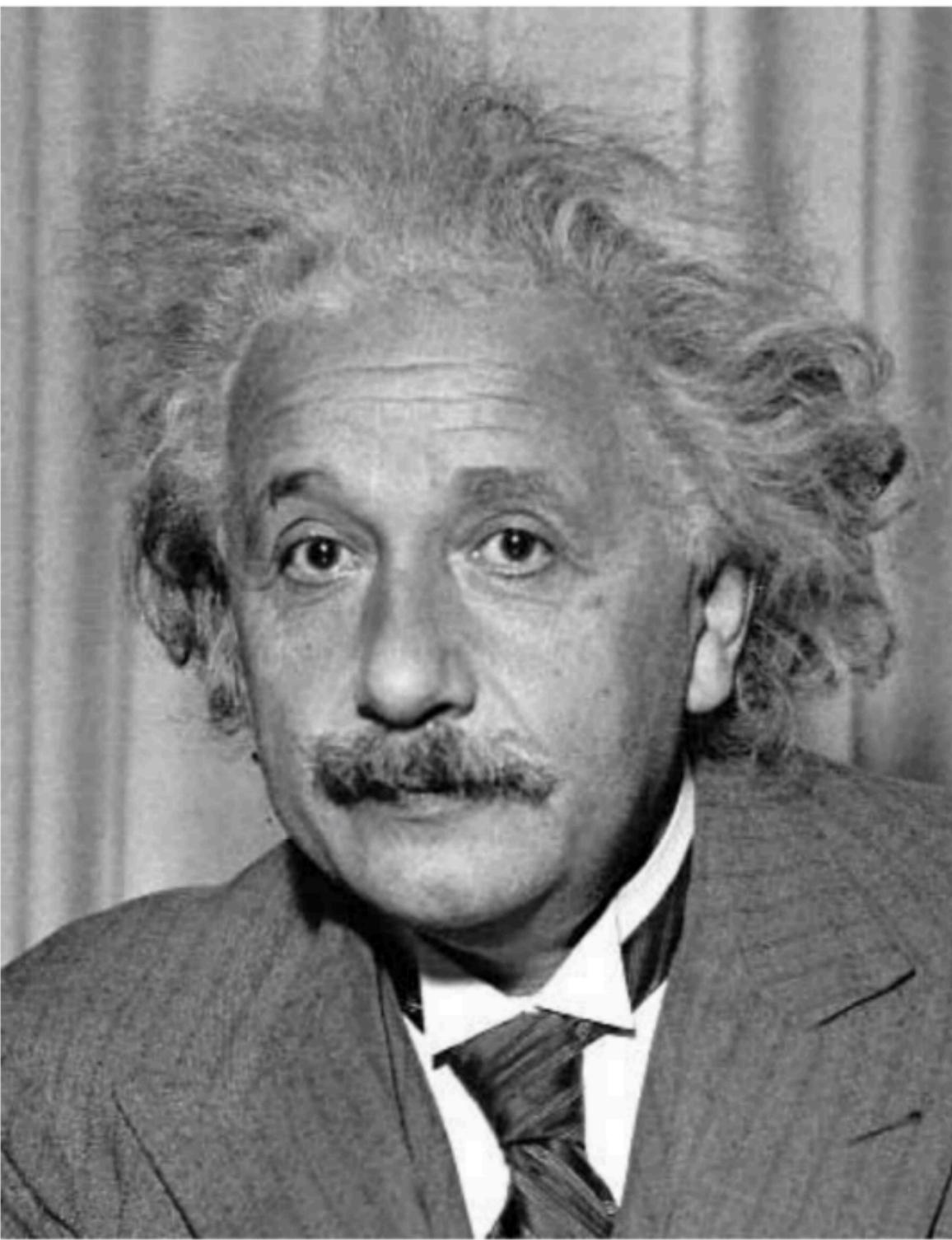
1	1	1	0	0
0	1 _{×1}	1 _{×0}	1 _{×1}	0
0	0 _{×0}	1 _{×1}	1 _{×0}	1
0	0 _{×1}	1 _{×0}	1 _{×1}	0
0	1	1	0	0

Image

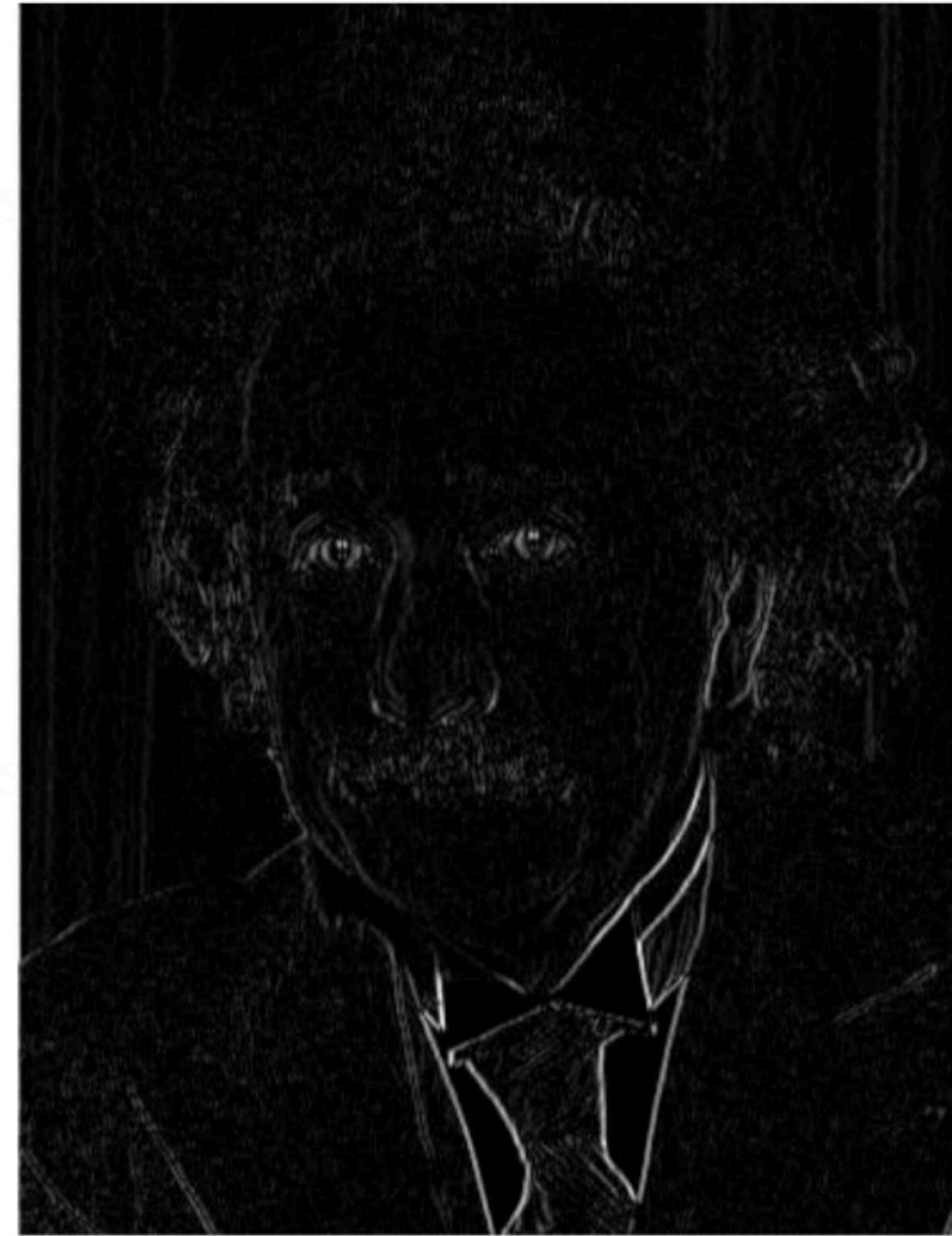
4	3	4
2	4	

Convolved
Feature

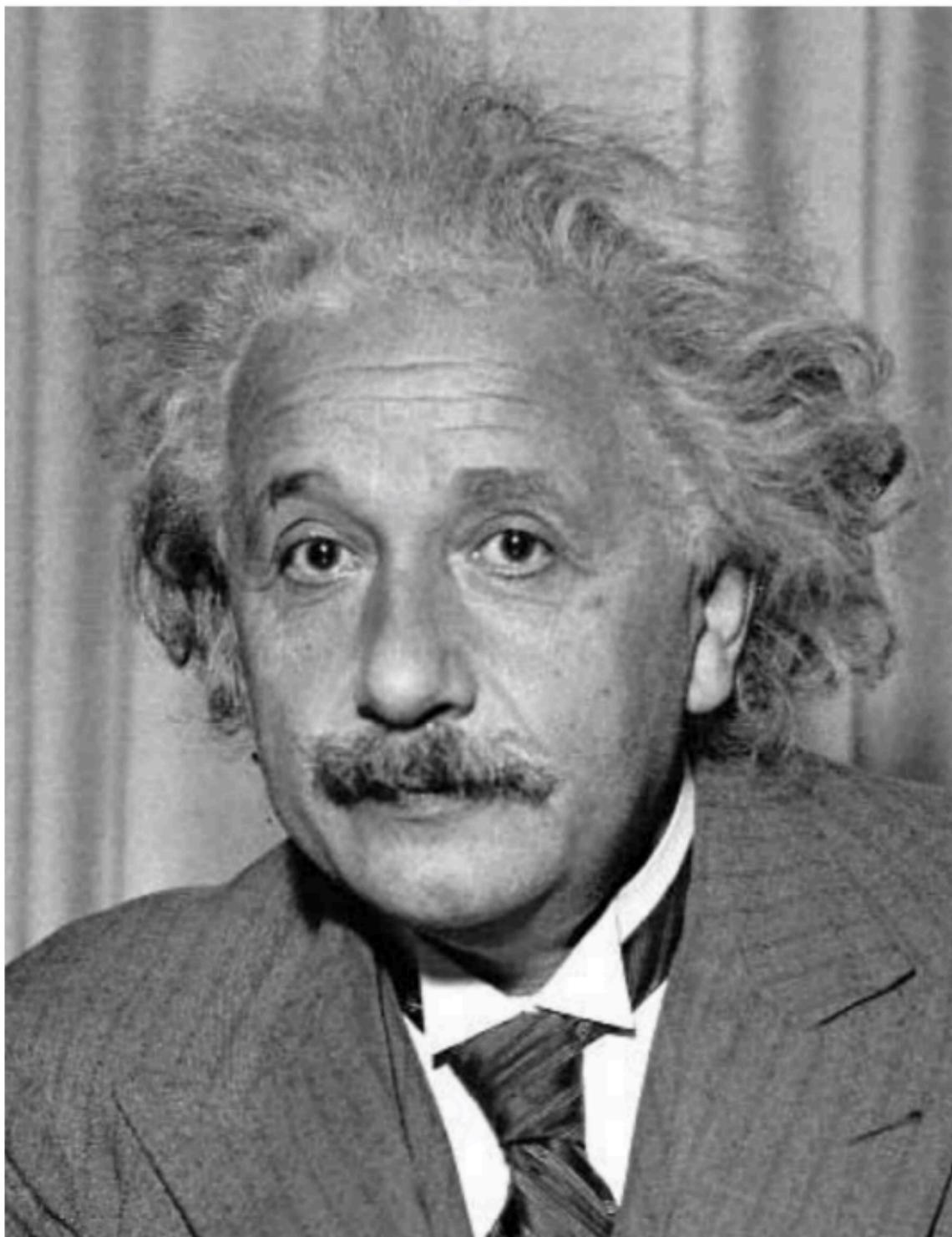
Фильтр для выделения вертикальных границ



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



Фильтр для выделения горизонтальных границ

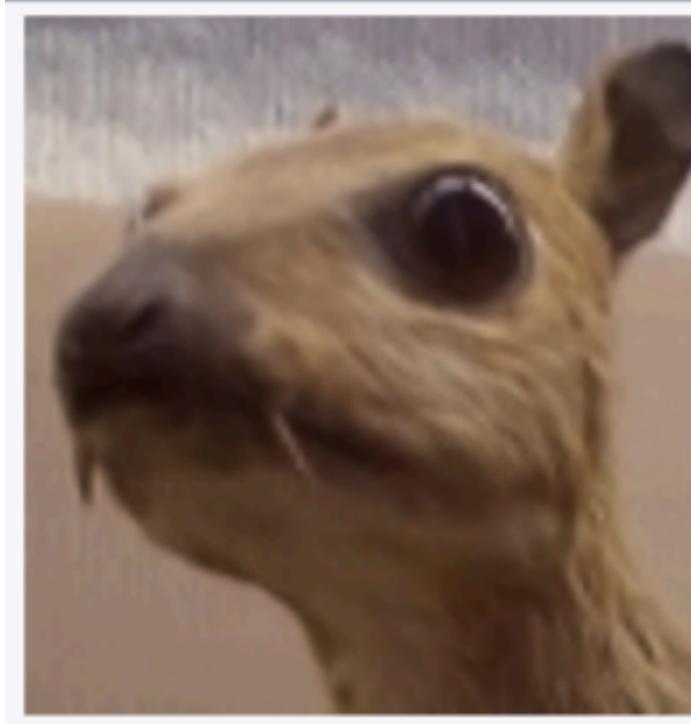


1	2	1
0	0	0
-1	-2	-1



Вопрос

Какое изображение получится
после применения следующих
фильтров?



$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

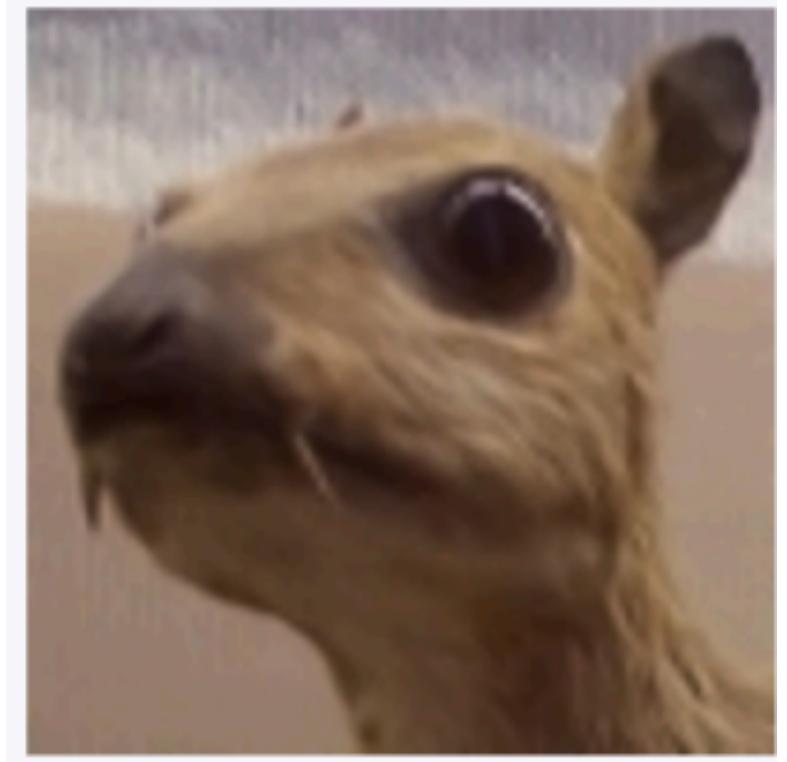
$$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Вопрос

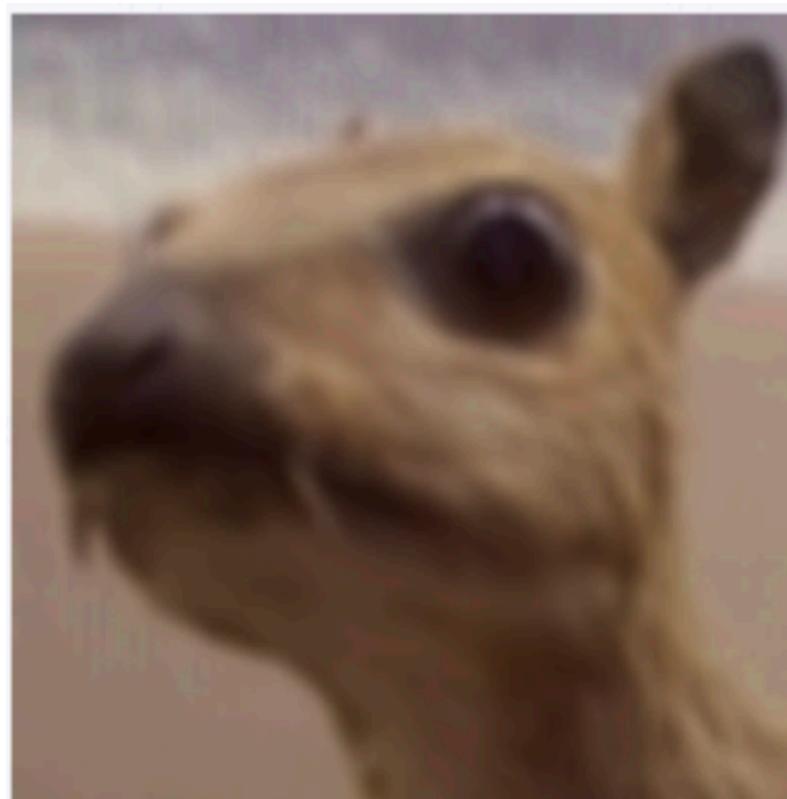
Какое изображение получится
после применения следующих
фильтров?



$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$



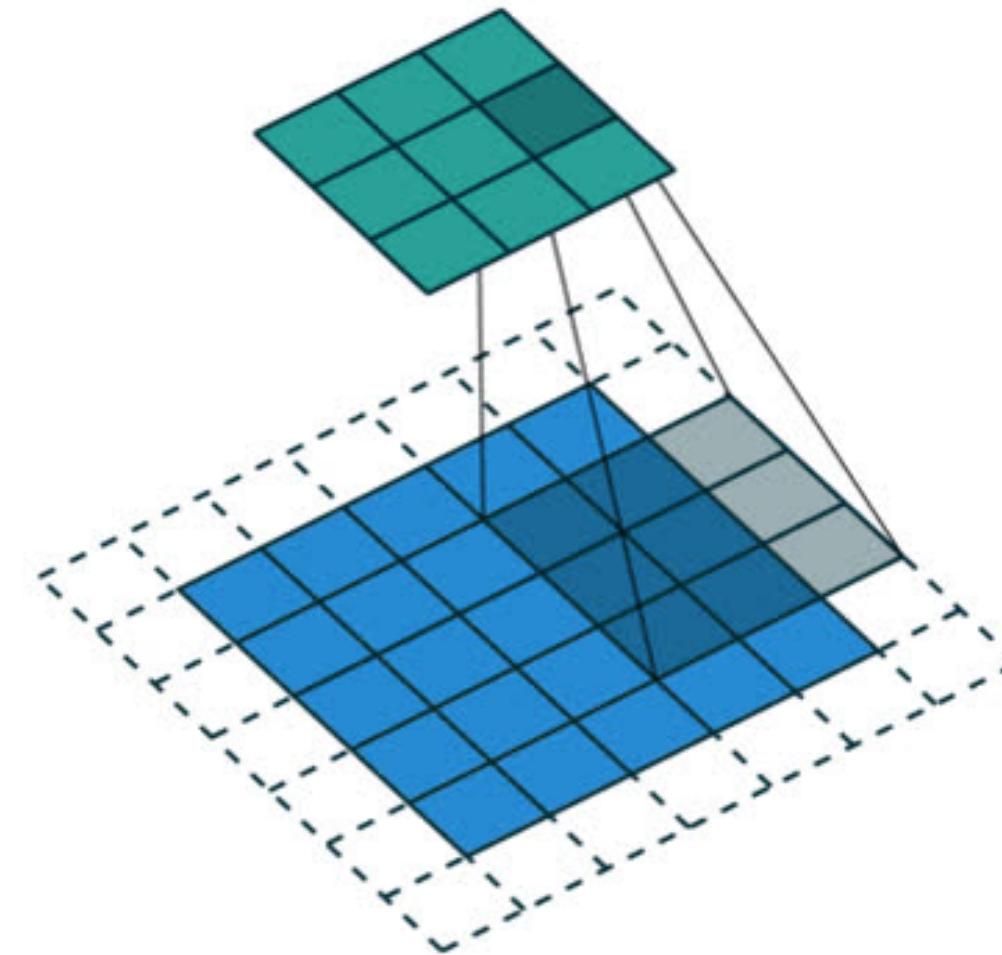
$$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$



Stride и padding

Шаг свертки (stride) -
значение сдвига ядра свертки.
Чем больше stride, тем
меньше размерность на
выходе

Padding - добавление
дополнительных элементов
вокруг изображения. Как
правило, нулей.

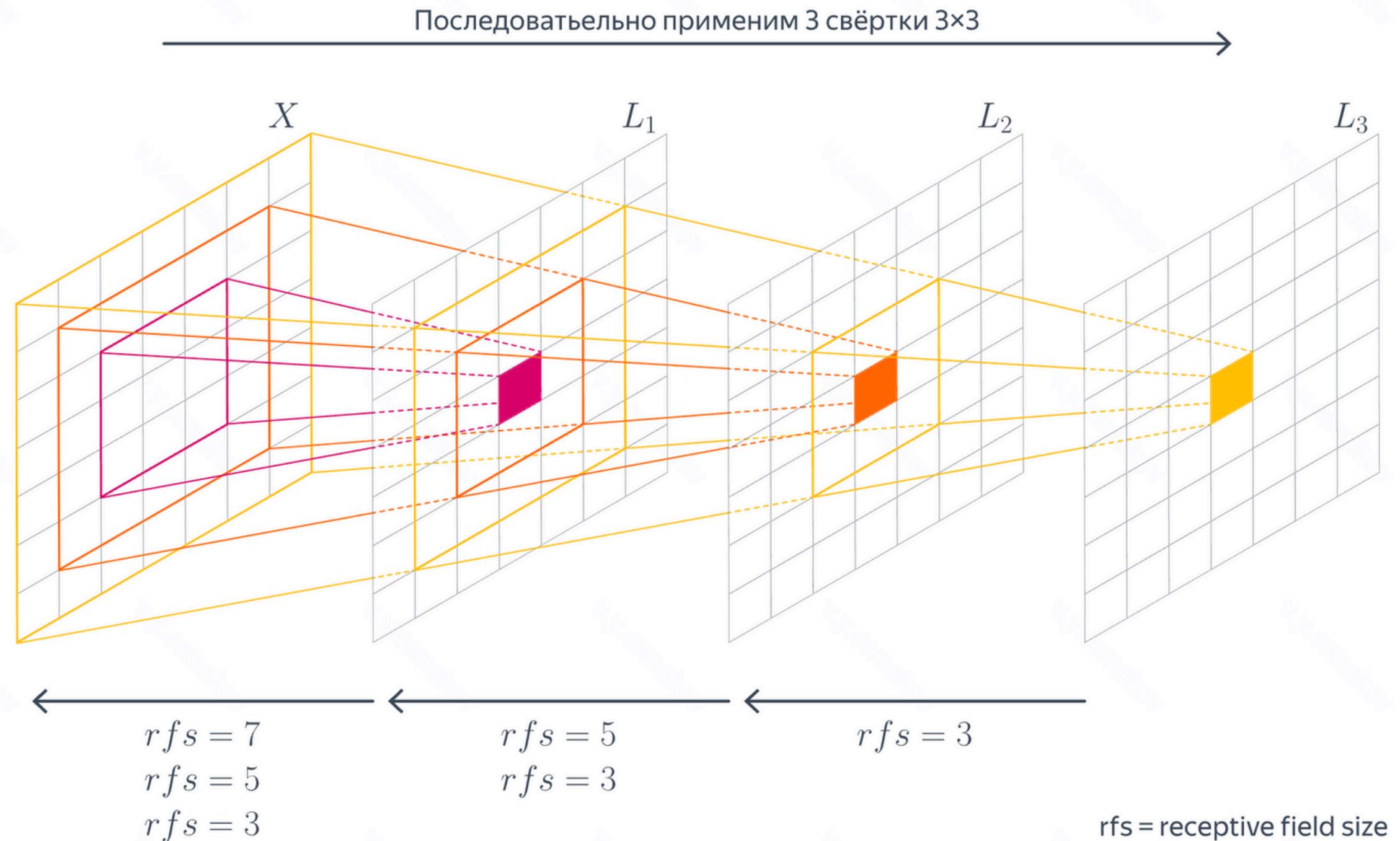


kernel=3
stride=2
padding=1

Receptive field

Receptive field - область картинки, на которую «сматрят» нейрон на определенном слое сети.

Применяя последовательно несколько сверток, мы увеличиваем **receptive field** для нейронов следующего слоя

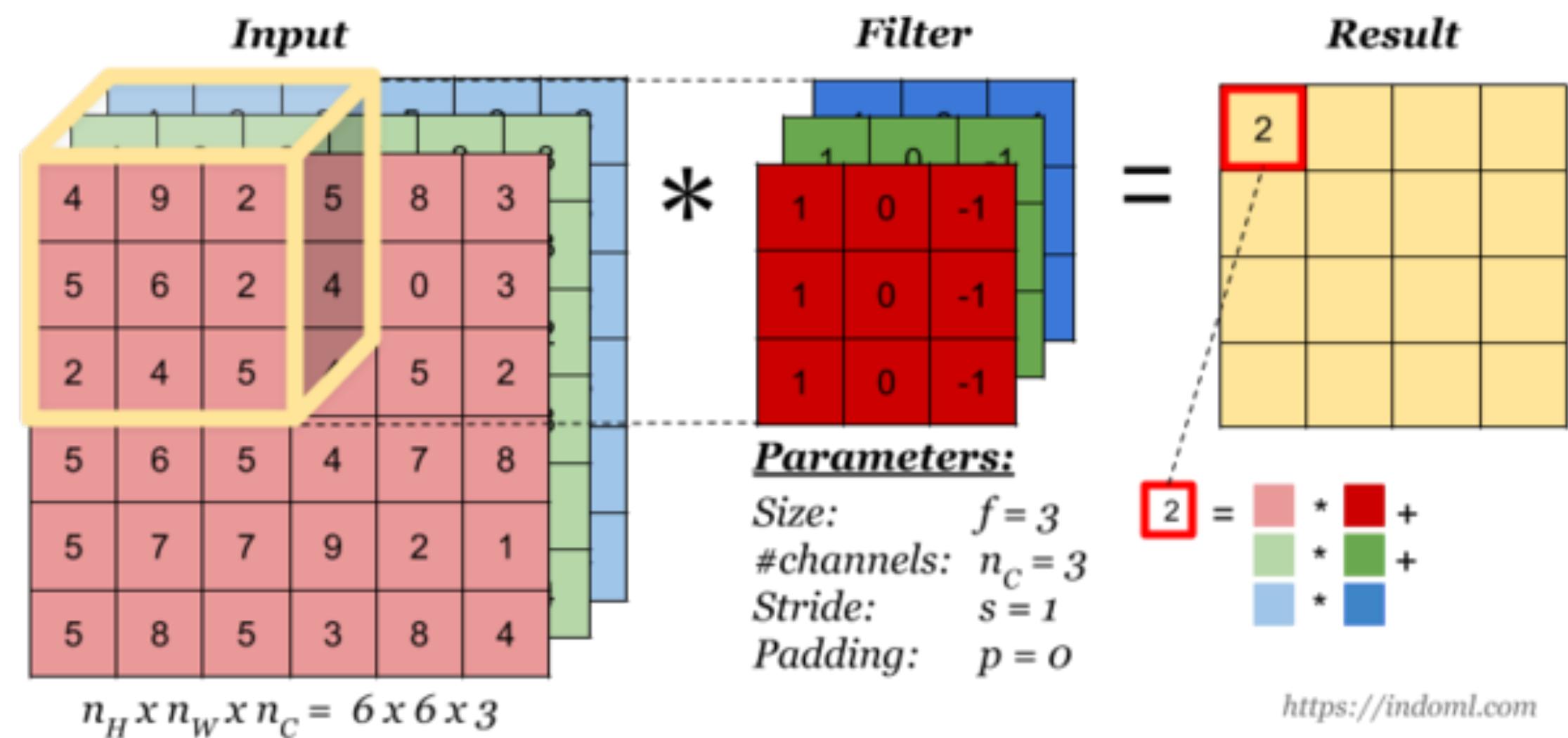


Свертка для цветного изображения

Применяем свертку для каждого канала, затем суммируем

Делаем аналогично на каждом сверточном слое

После свертки также добавляем **нелинейность** (функции активации) для охвата сложных зависимостей



Интуиция сверток

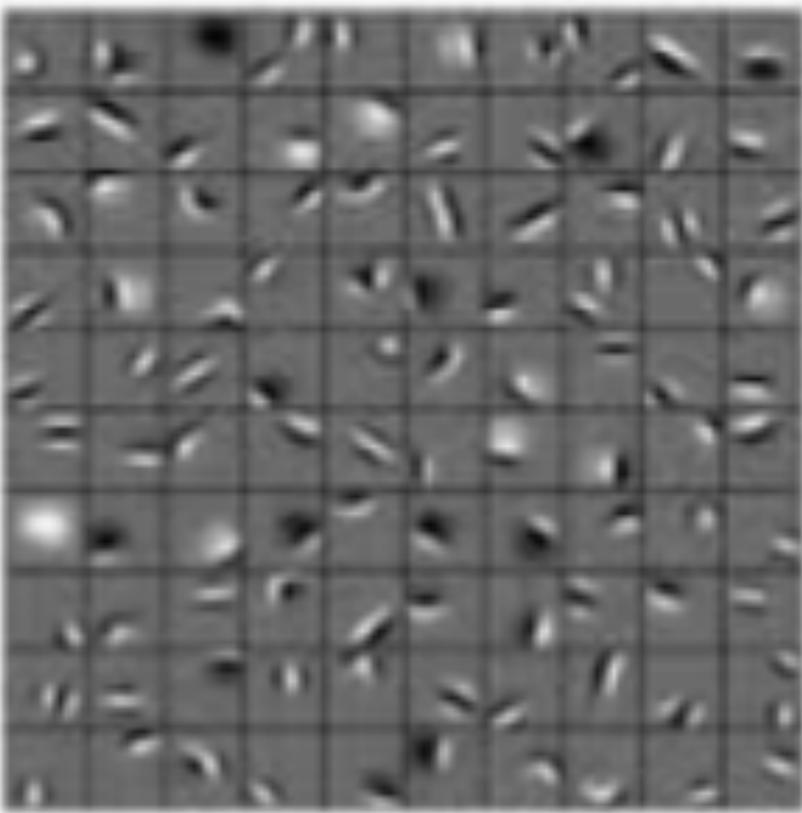
Low-Level
Feature

Overlap

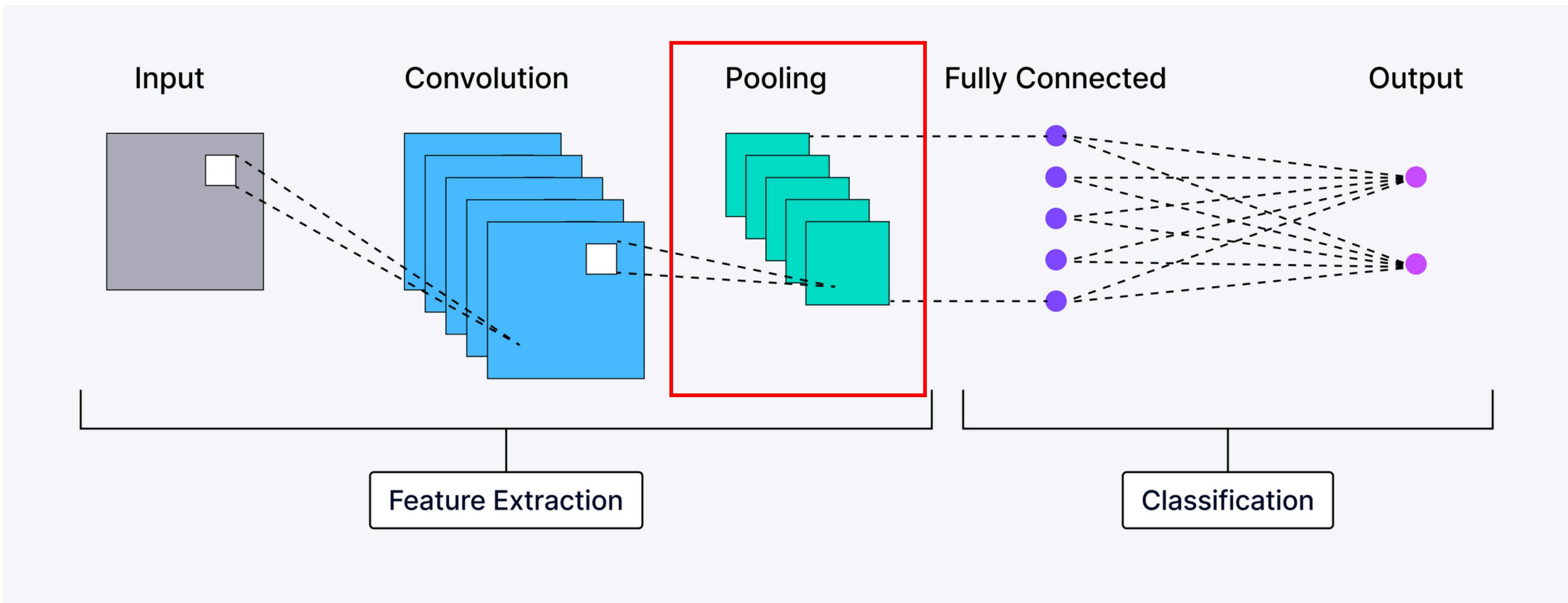
Mid-Level
Feature

Overlap

High-Level
Feature



Pooling



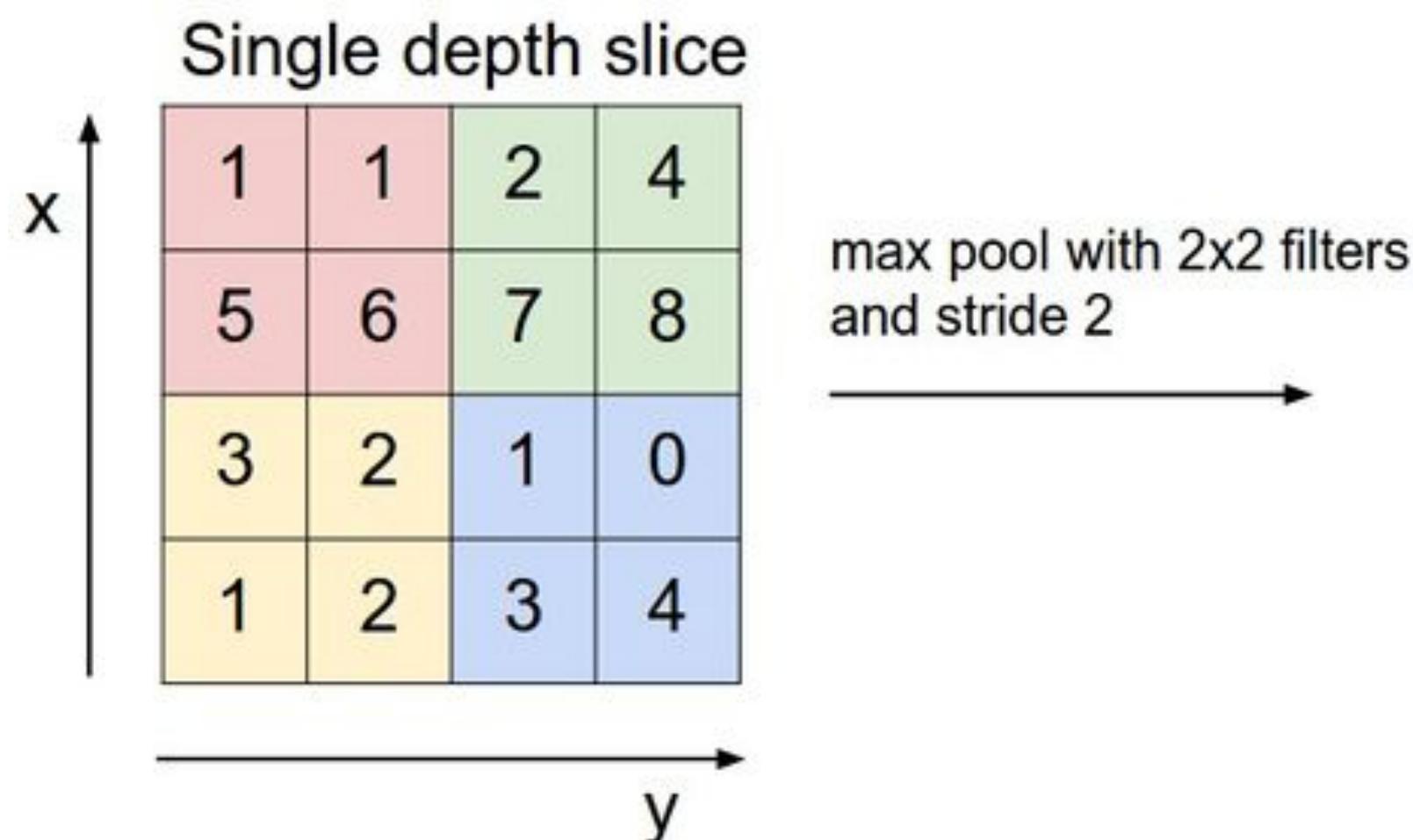
Pooling

Больше сверток - больше признаков изображения. Однако это приводит к росту параметров.

Используем **Pooling**, который агрегирует информацию об области в одно значение. Например, **Max pooling** берет максимум, **Average pooling** - среднее.

Помимо снижения количества параметров:

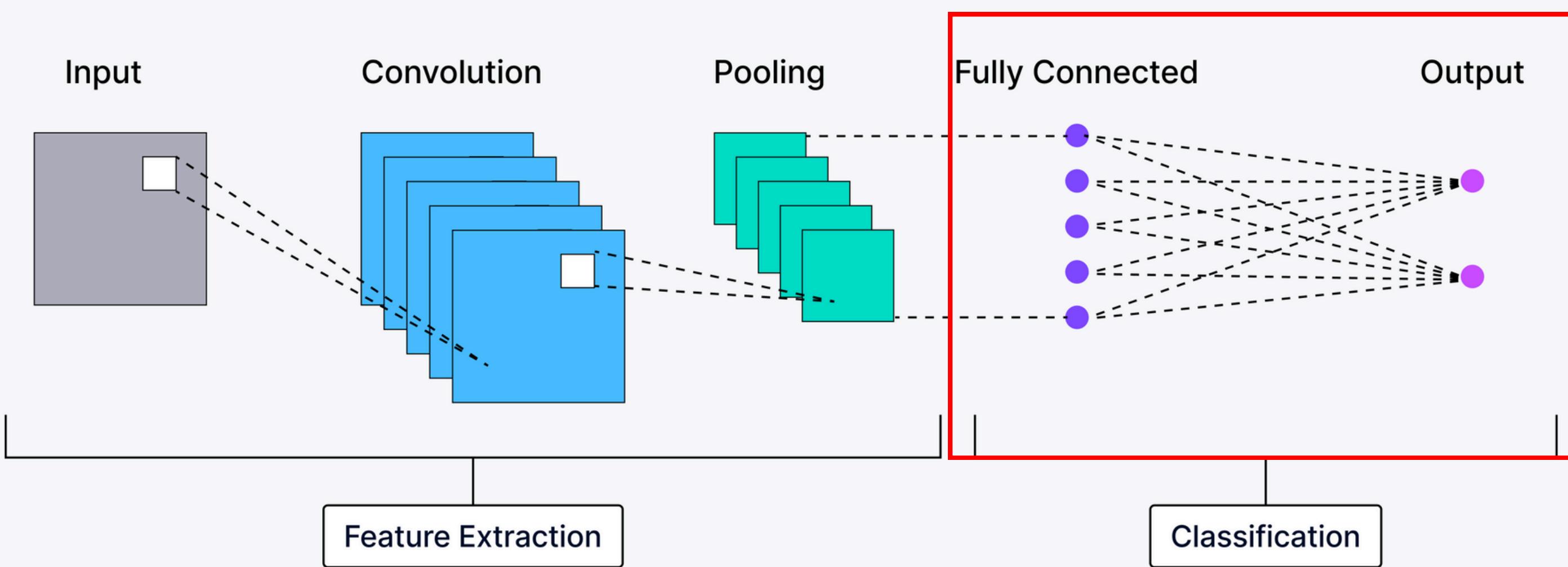
- Увеличивается receptive field
- Увеличивается инвариантность



6	8
3	4

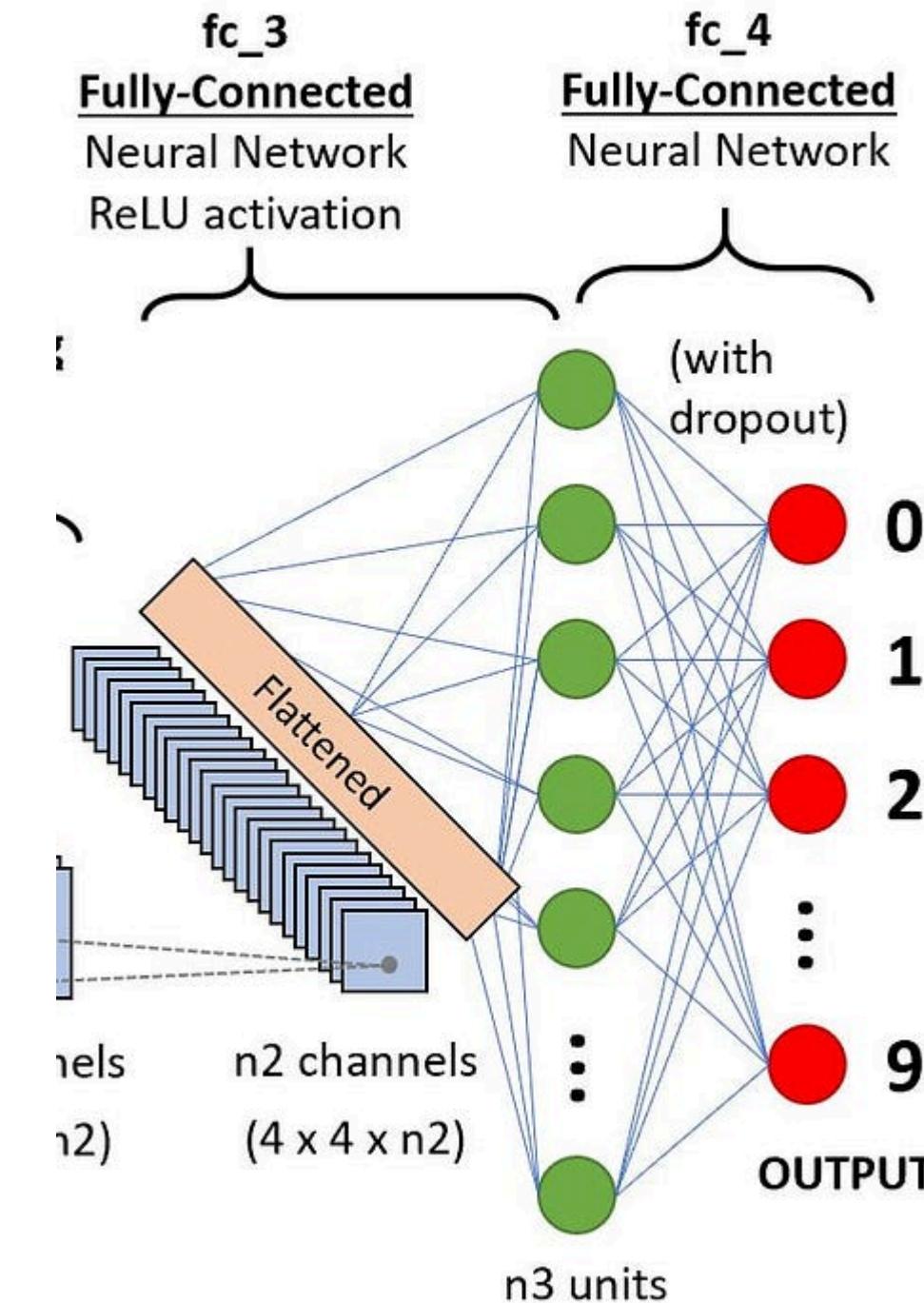
Полносвязный слой

The Architecture of Convolutional Neural Networks



Полносвязная нейронная сеть

- Перед подачей на вход к сети, “разворачиваем” собранные признаки в вектор.
- Одного-двух скрытых слоев достаточно
- Последний слой определяет решаемую задачу. Например:
 - Два значения на выходе - бинарная классификация
 - Десять значений - мультиклассовая классификация с десятью классами



Финальный слайд и вопросы

Реализуем в
Pytorch

