Введение в анализ данных

Введение
в компьютерное зрение

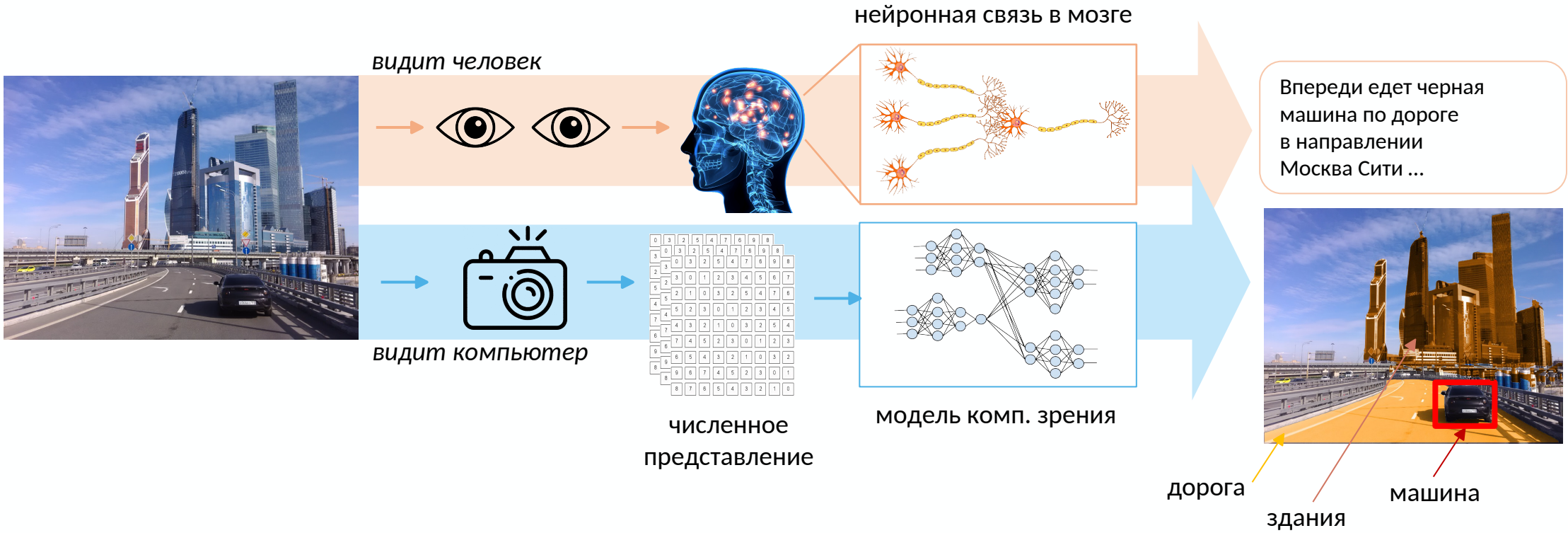
Введение в компьютерное зрение

- **Компьютерное зрение**
- Классификация изображений
- Свертка и Pooling
- Классификация с помощью CNN
- Обзор задач компьютерного зрения

Компьютерное зрение

Компьютерное зрение — это область ИИ, связ. с анализом и генерацией изображений и видео.

Оно **позволяет компьютеру "увидеть" окружающий мир** так же, как мы видим его глазами.



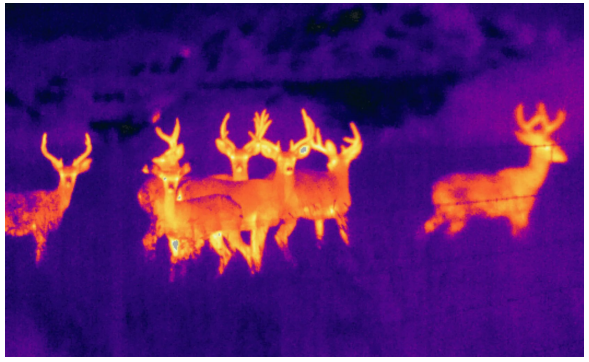
Что служит глазами компьютеру?

Различные сенсоры позволяют компьютеру получать информацию из окружающего мира:

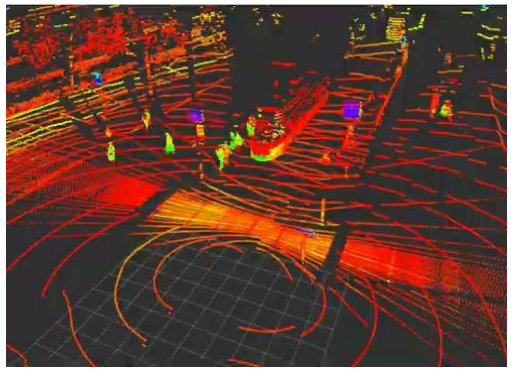
камеры, лидары и радары, тепловизоры и сонары, МРТ, КТ, и рентгеновские аппараты, и т.д.

КАМЕРА

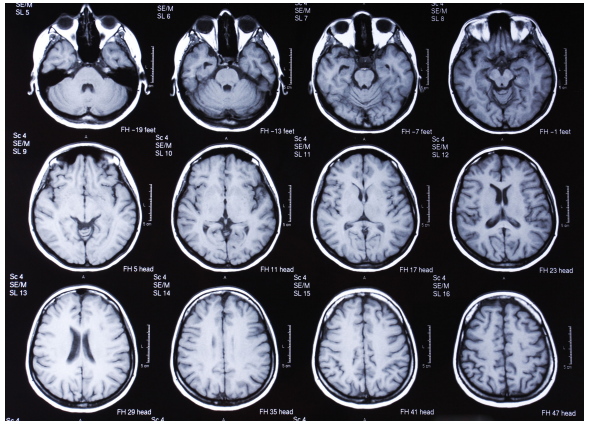
ТЕПЛОВИЗОР



ЛИДАР



МРТ



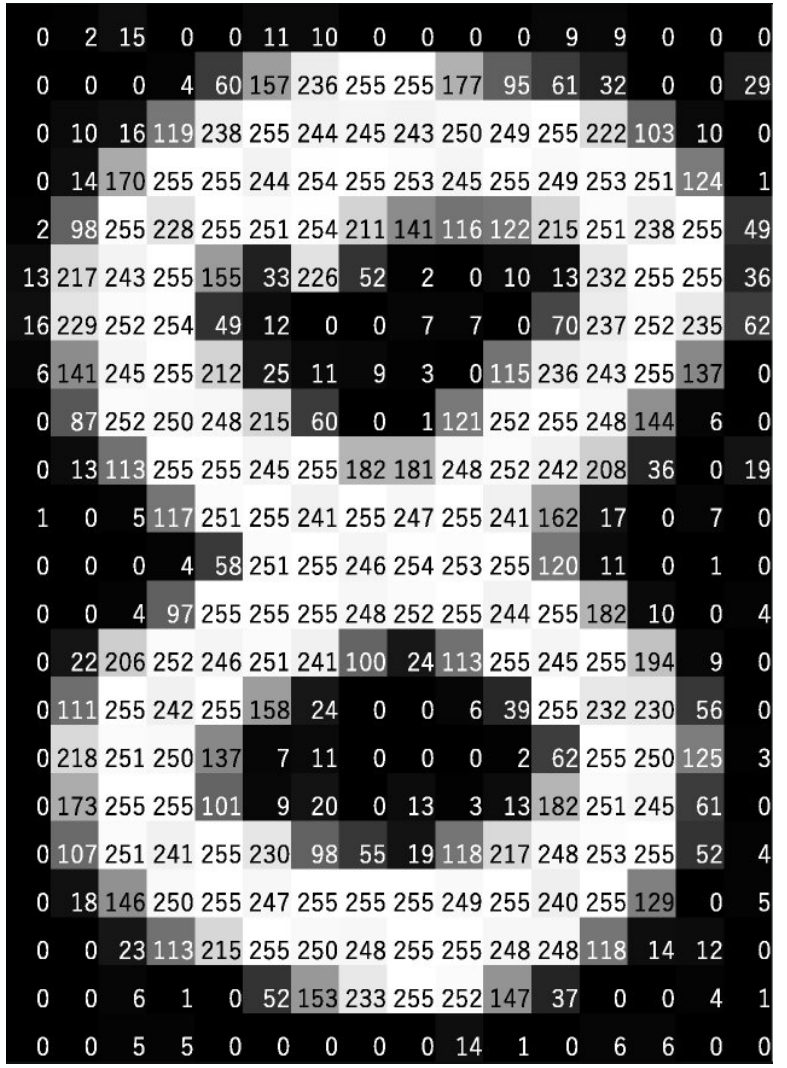
Стандартное представление изображения

- **Черно-белая картинка** — матрица из пикселей.

  H — длина в пикселях,

  W — ширина в пикселях.

- **Пиксель** — число, означ. интенсивность цвета.
- **Интенсивность** — число от 0 до 1.
  Обычно ее кодируют числом от 0 до 255 (1 байт).



Черно-белая картинка —
тензор размера (*H*, *W*), состоящий из `uint8` чисел.

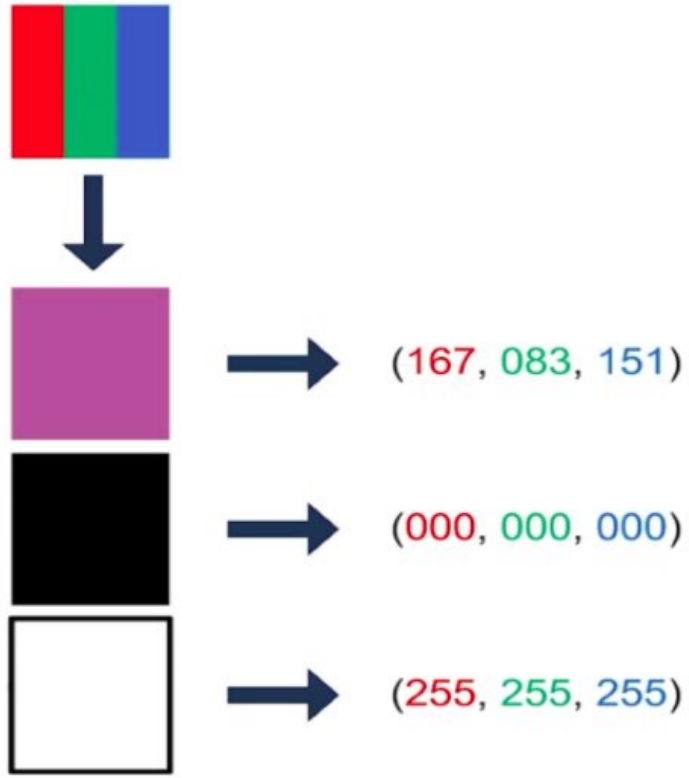
Стандартное представление изображения

- **Цветная картинка** — матрица из пикселей,

  H — высота в пикселях,

  W — ширина в пикселях.

- **Пиксель** *обычно* представляют в RGB формате:
  массив интенсивностей красного, зеленого и синего.
- **Интенсивность** — число от 0 до 1.
  Обычно ее кодируют числом от 0 до 255 (1 байт).



Цветная картинка —
тензор размера (*H*, *W*, 3), состоящий из `uint8` чисел.

Стандартное представление изображения

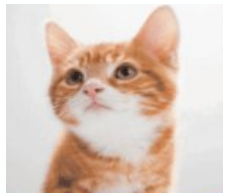
Цветная картинка — тензор размера (*H*, *W*, 3), состоящий из чисел `uint8`.

Изображение можно разложить на **3 канала** по размерности пикселя.

Что значит, что компьютер видит?

Это значит, что он может решать задачи компьютерного зрения (не обязательно все сразу).

Классификация

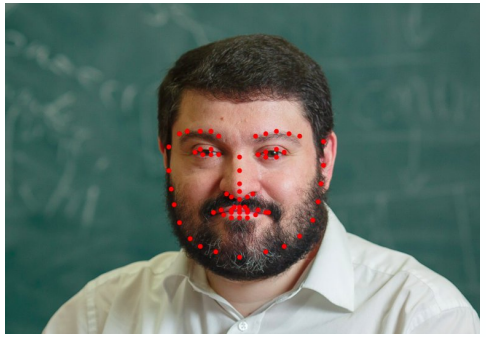


КОШКА

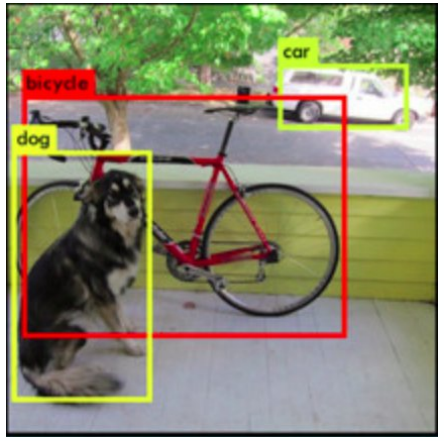
СОБАКА

Сегментация

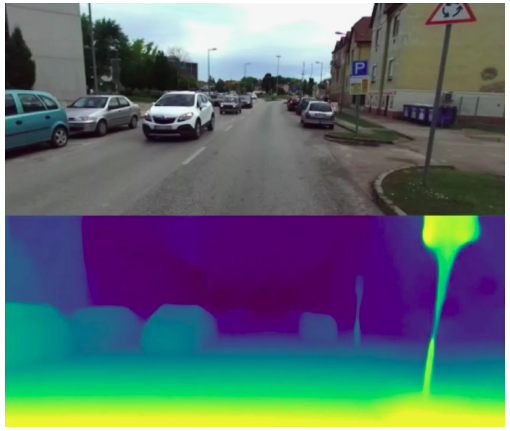
Определение ключевых точек



Детекция



Определение глубины



Генерация

и другие задачи...

Введение в компьютерное зрение

- Компьютерное зрение
- **Классификация изображений**
- Свертка и Pooling
- Классификация с помощью CNN
- Обзор задач компьютерного зрения

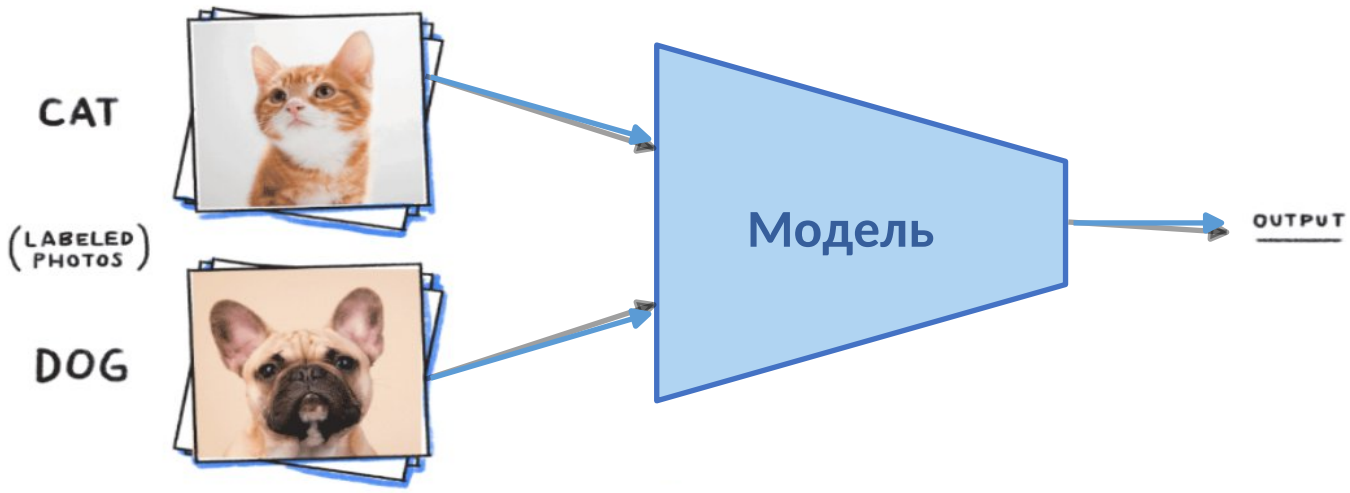
Классификация изображений

X— пространство картинок,

Y — набор классов, например {кошка, собака}.

Требуется построить модель $y(x)$: $X \to Y$,

определяющую к какому классу относится объект на картинке.



Проблемы классификации изображений

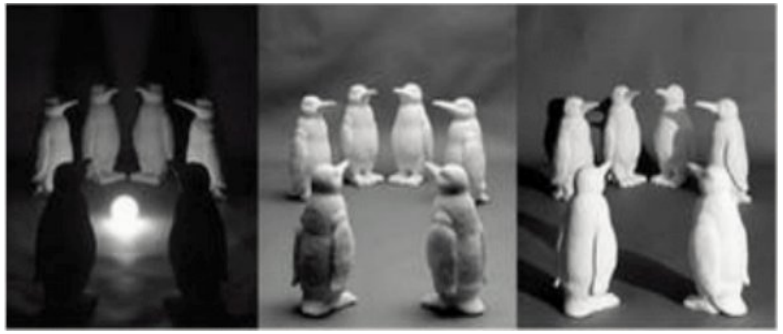
Разные углы обзора

Разный размер

Деформация

Перекрытие

Разная освещенность



Фоновые помехи

Разная форма

Классификация изображений до нейросетей

Необходима генерация признаков изображений вручную :(

Генерация признаков

Полезные признаки изображения для упрощения классификации

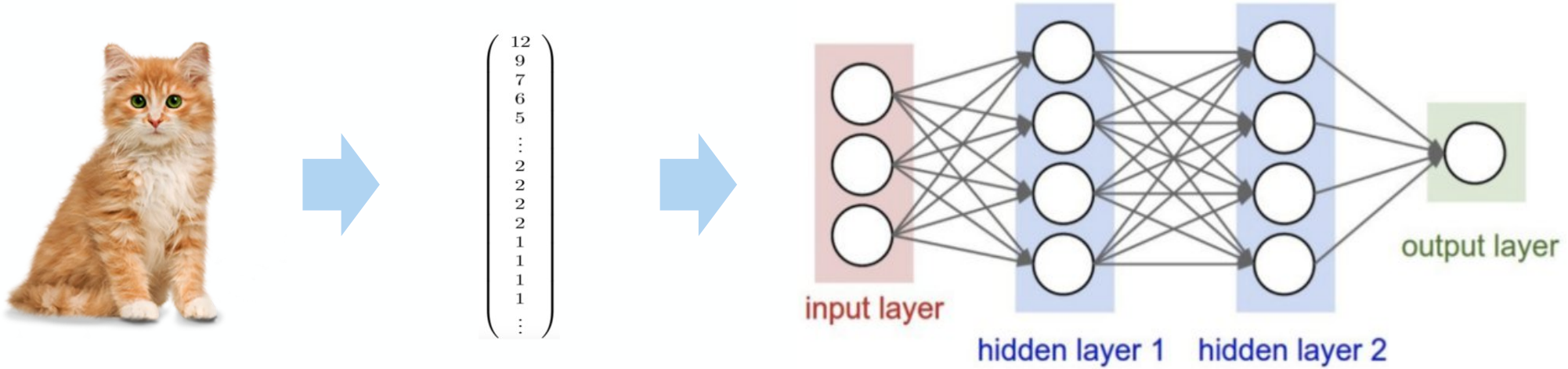
Обучение модели

KNN,
Лин. регрессия,
Лог. регрессия,
...

Предсказание

Классификация изображений с помощью нейросетей

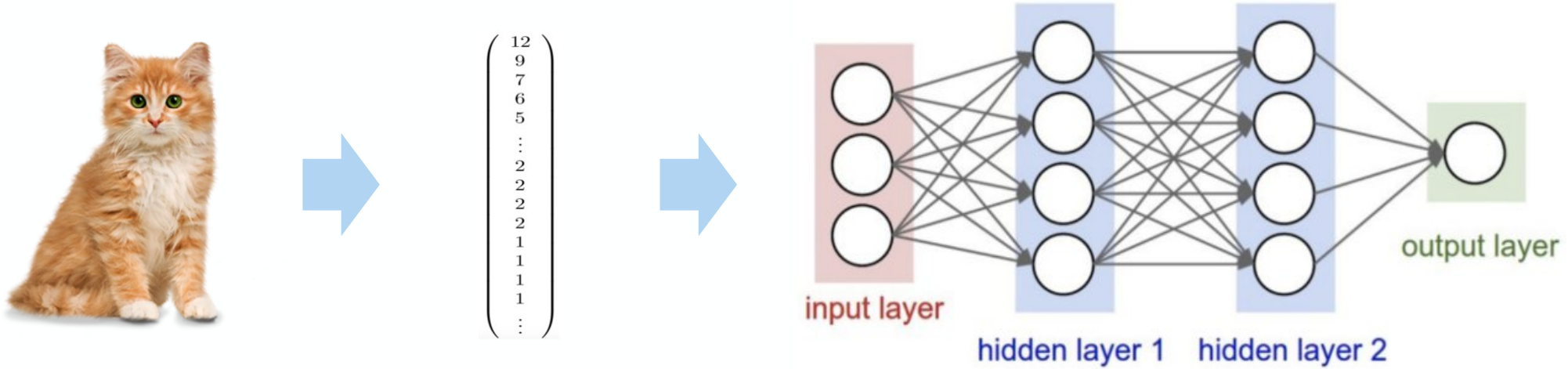
Преобразуем картинку в вектор и подаем результат нейросети.



Насколько хорошо Linear-слои подходят для решения задачи классификации картинок?

Классификация изображений с помощью нейросетей

Преобразуем картинку в вектор и подаем результат нейросети.



Проблемы линейных слоев:

- имеют очень много параметров — напр., кол-во пар-в в Linear(256x256, 256)=16M;
- изначально не понимают связи между близко расположенными пикселями, что делает нахождение паттернов сложнее.

Введение в компьютерное зрение

- Компьютерное зрение
- Классификация изображений
- **Свертка и Pooling**
- Классификация с помощью CNN
- Обзор задач компьютерного зрения

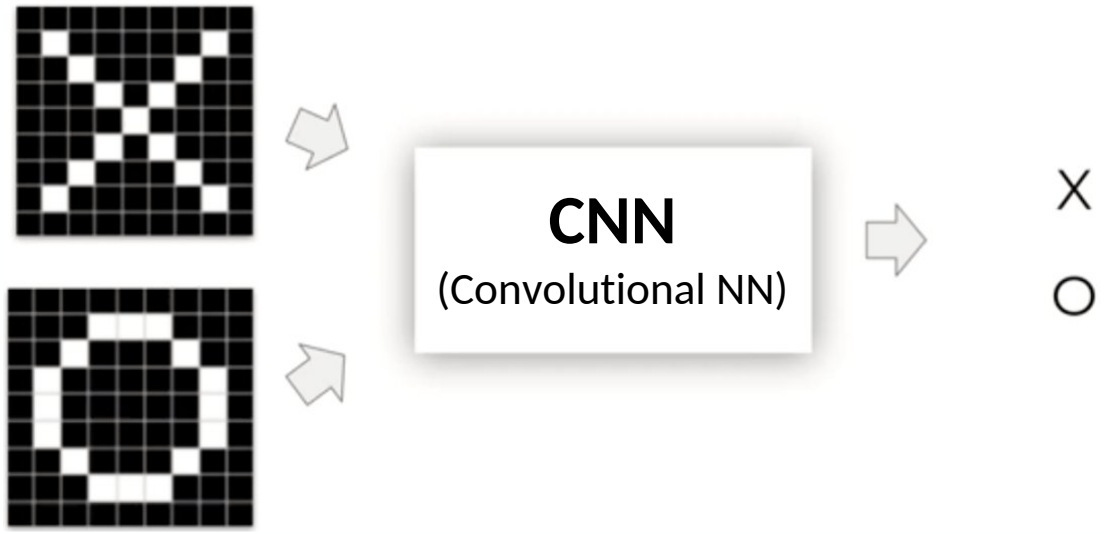
Интуиция свёртки

Поиграем в классификацию!

Требуется уметь находить **крестики и нолики** на картинке.

Причем *объекты могут находиться в разных местах* картинки,

быть *не в точности равны искомым паттернам*.



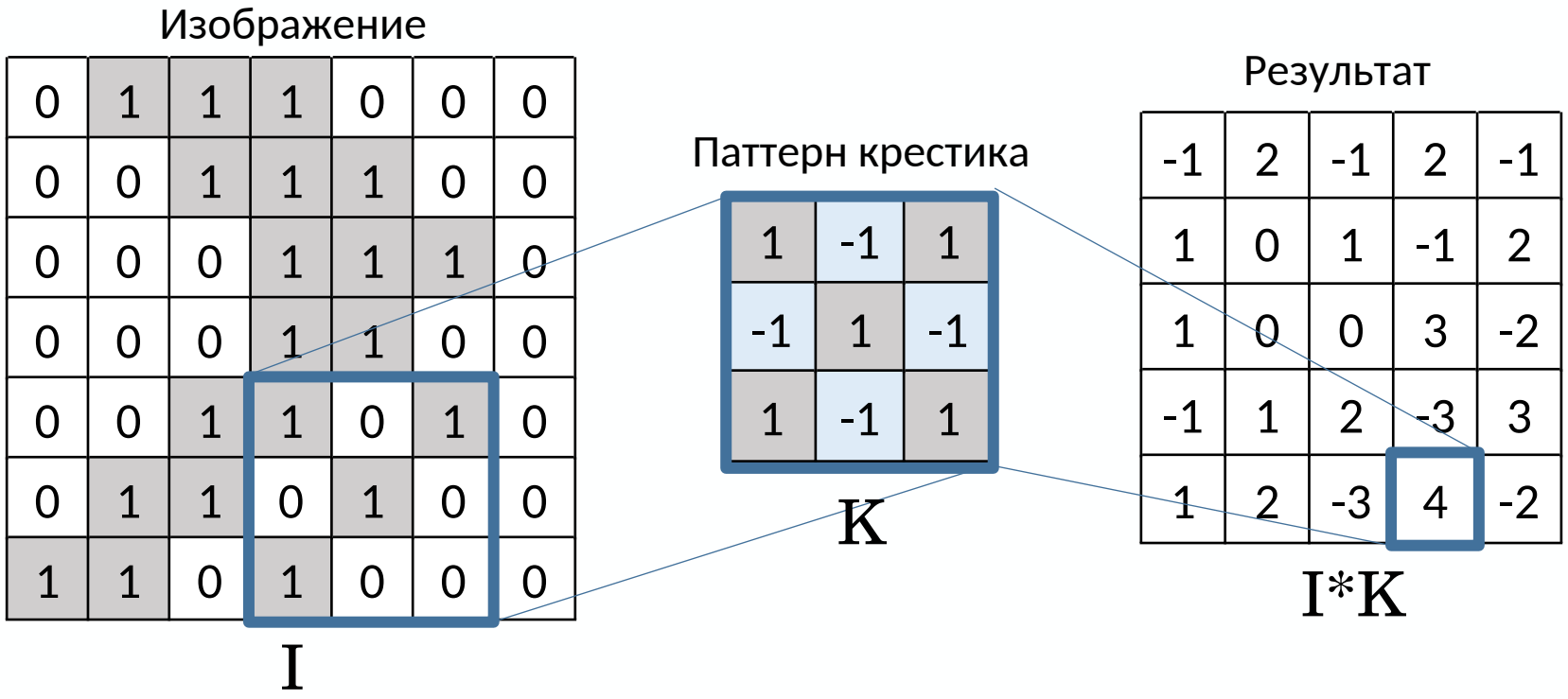
Интуиция свёртки

Идея: сделаем локальный поиск паттернов.

Берем паттерн крестика и ищем похожий паттерн на картинке.

Алгоритм

- Проходимся окнами по картинке.
- Считаем схожесть этой части картинки и паттерна.
- Записываем результат в соотв. место в матрице.



Как считать схожесть?

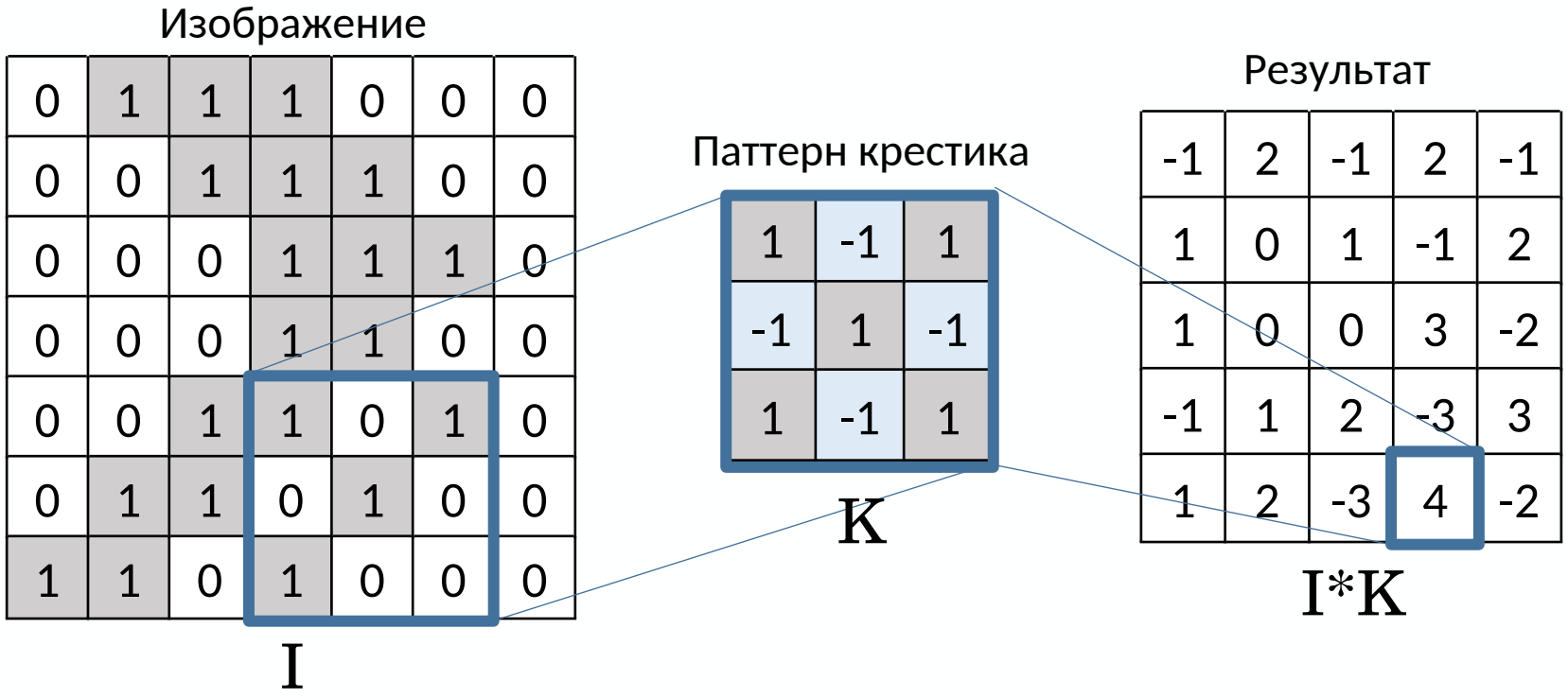
Интуиция свёртки

Как считать схожесть?

Чтобы оценить схожесть двух паттерна и области картинки, перемножим их скалярно:

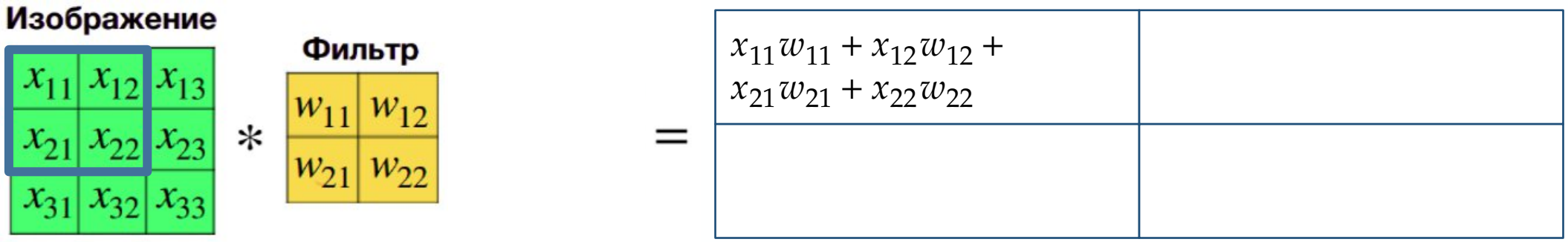
- умножим матрицы поэлементно;
- сложим числа получ. матрицы.

Чем больше значение в матрице, тем больше похожа область картинки на паттерн.



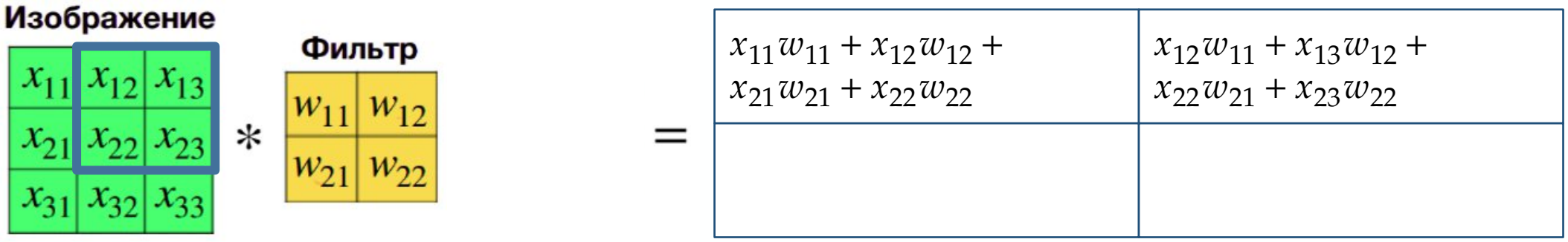
2D-свёртка

Фильтр / ядро, представляющий из себя матрицу весов,
пробегает по исходным данным и вычисляет скалярные произведения
с той частью изображения, над которой он сейчас находится.



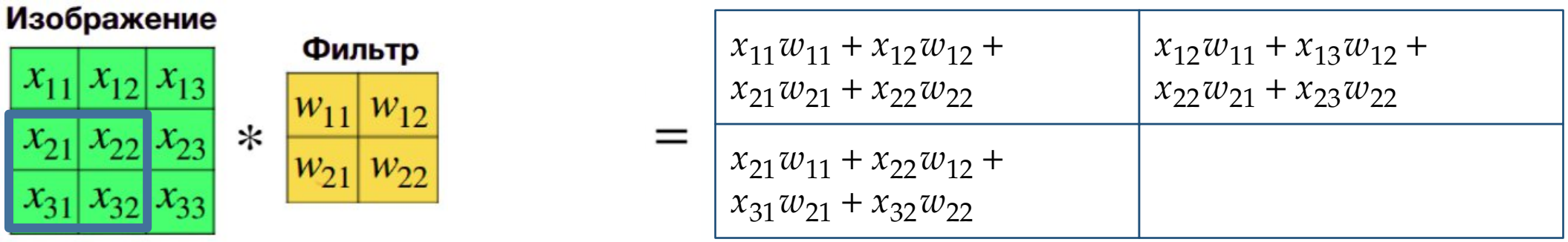
2D-свёртка

Фильтр / ядро, представляющий из себя матрицу весов,
пробегает по исходным данным и вычисляет скалярные произведения
с той частью изображения, над которой он сейчас находится.



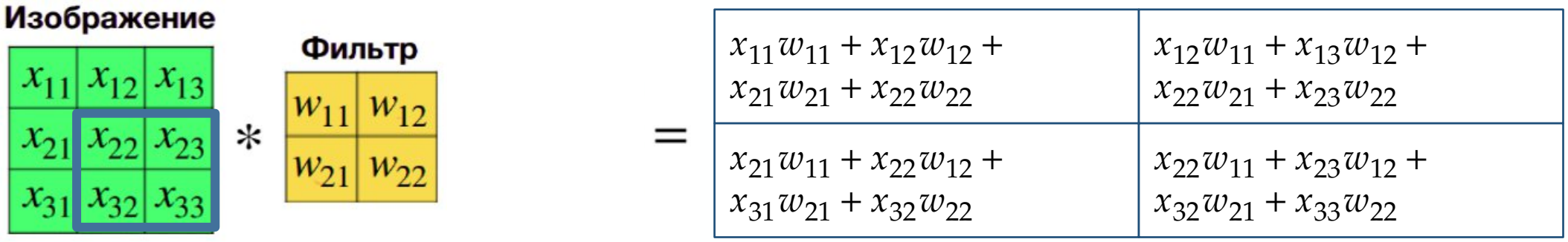
2D-свёртка

Фильтр / ядро, представляющий из себя матрицу весов,
пробегает по исходным данным и вычисляет скалярные произведения
с той частью изображения, над которой он сейчас находится.



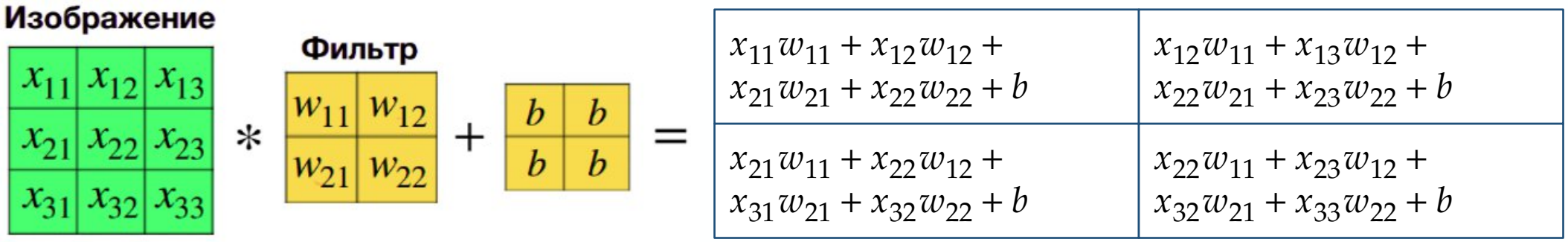
2D-свёртка

Фильтр / ядро, представляющий из себя матрицу весов,
пробегает по исходным данным и вычисляет скалярные произведения
с той частью изображения, над которой он сейчас находится.



2D-свёртка

Фильтр / ядро, представляющий из себя матрицу весов,
пробегает по исходным данным и вычисляет скалярные произведения
с той частью изображения, над которой он сейчас находится.



После чего к каждому элементу также добавляется смещение b.

2D-свёртка

В глубоком обучении 2D-свёртка— операция над матрицей (изображением) $X \in \mathbb{R}^{H \times W}$, фильтром / ядром $W \in \mathbb{R}^{K_1 \times K_2}$ и смещением $b \in \mathbb{R}$, результатом которой является матрица $Y \in \mathbb{R}^{(H-K_1+1)\times(W-K_2+1)}$, определяемая формулой

$$Y_{l,m} = (X * W)_{l,m} + b = \sum_{i=1}^{K_1} \sum_{j=1}^{K_2} w_{ij} \cdot x_{l+i-1,m+j-1} + b,$$

где веса w_{ij} фильтра W и смещение b — обучаемые параметры,

K_1, K_2— гиперпараметры (часто используют $K = K_1 = K_2$).

Изображение

x_{11}	x_{12}	x_{13}
x_{21}	x_{22}	x_{23}
x_{31}	x_{32}	x_{33}

$*$

Фильтр

w_{11}	w_{12}
w_{21}	w_{22}

$+$

b	b
b	b

$=$

$x_{11}w_{11} + x_{12}w_{12} + x_{21}w_{21} + x_{22}w_{22} + b$	$x_{12}w_{11} + x_{13}w_{12} + x_{22}w_{21} + x_{23}w_{22} + b$
$x_{21}w_{11} + x_{22}w_{12} + x_{31}w_{21} + x_{32}w_{22} + b$	$x_{22}w_{11} + x_{23}w_{12} + x_{32}w_{21} + x_{33}w_{22} + b$

2D-свёртка: а как обучать?

Несложно увидеть, что эта операция *линейна* по отношению к w_{ij} и b.

А это значит, что градиенты операции свертки выражается аналитически,
то есть Back Propagation работает!

Изображение

x_{11}	x_{12}	x_{13}
x_{21}	x_{22}	x_{23}
x_{31}	x_{32}	x_{33}

$*$

Фильтр

w_{11}	w_{12}
w_{21}	w_{22}

$+$

b	b
b	b

$=$

$x_{11}w_{11} + x_{12}w_{12} + x_{21}w_{21} + x_{22}w_{22} + b$	$x_{12}w_{11} + x_{13}w_{12} + x_{22}w_{21} + x_{23}w_{22} + b$
$x_{21}w_{11} + x_{22}w_{12} + x_{31}w_{21} + x_{32}w_{22} + b$	$x_{22}w_{11} + x_{23}w_{12} + x_{32}w_{21} + x_{33}w_{22} + b$

$$\frac{\partial L}{\partial F} =$$

$$\frac{\partial L}{\partial X} =$$

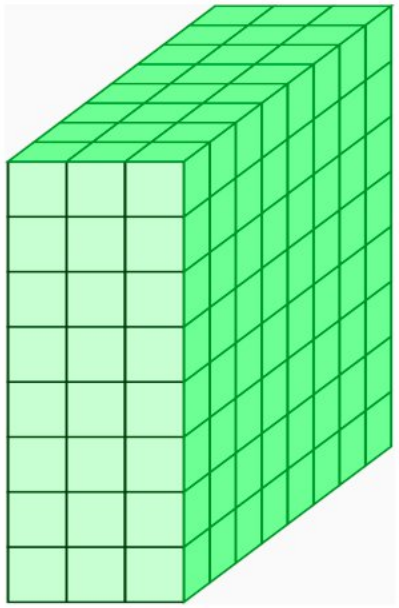
Fun fact: градиент свертки –
тоже своего рода свертка.

2D-свёртка: многоканальный вход

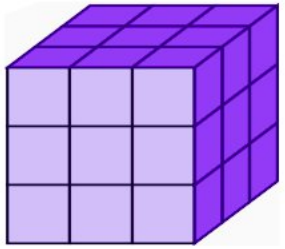
Вход — трехмерный тензор (изображение) $X \in \mathbb{R}^{C \times H \times W}$, где C — кол-во каналов.

Фильтр / ядро — трехмерная матрица размера $W \in \mathbb{R}^{C \times K_1 \times K_2}$.

$$Y_{l,m} = \sum_{c=1}^{C} (X_c * W_c)_{l,m} + b = \sum_{c=1}^{C} \sum_{i=1}^{K_1} \sum_{j=1}^{K_2} w_{cij} \cdot x_{c,\, l+i-1, m+j-1} + b$$

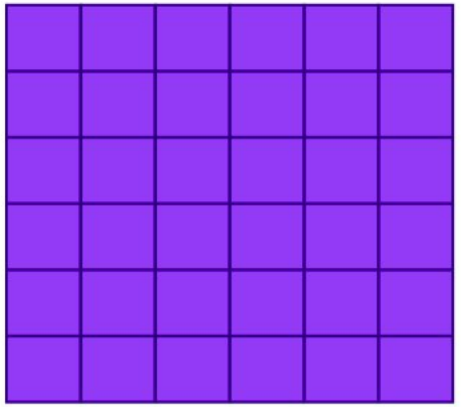


изображение 8 x 8 x 3



3 x 3 x 3

смещение

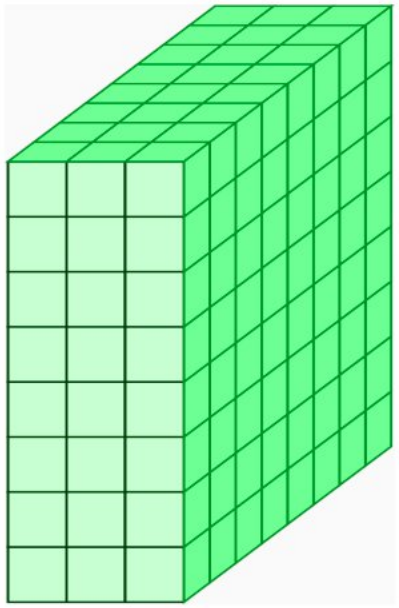


результат — карта 6 x 6

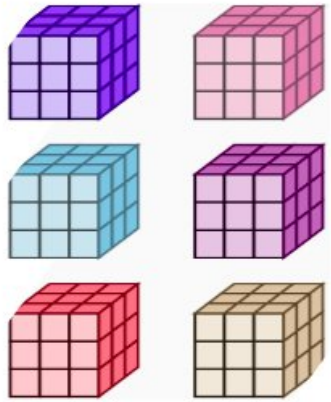
2D-свёртка: многоканальный вход

Один фильтр — одна карта, соответствующая одному паттерну.
Возьмем по D *разных фильтров и смещений*, применим свертку к ним.
Получим D карт на выходе:

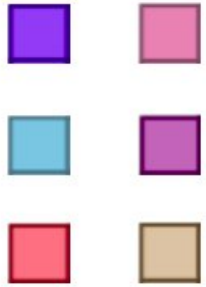
$$Y_{d,l,m} = \sum_{k=1}^{C} (X_k * W_{dk})_{l,m} + b_d = \sum_{k=1}^{C} \sum_{i=1}^{K_1} \sum_{j=1}^{K_2} w_{dkij} \cdot x_{k,\, l+i-1,m+j-1} + b_d$$



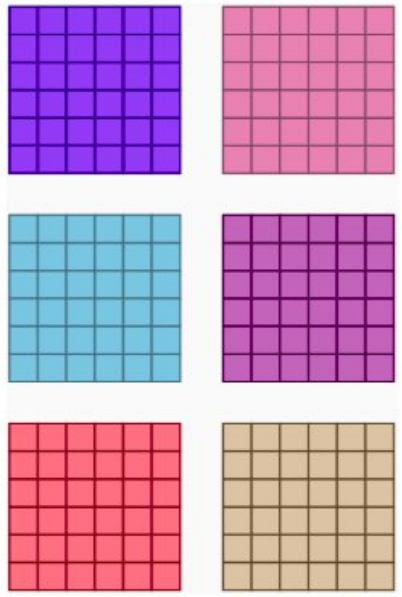
изображение 8 x 8 x 3



6 фильтров 3 x 3 x 3



смещения

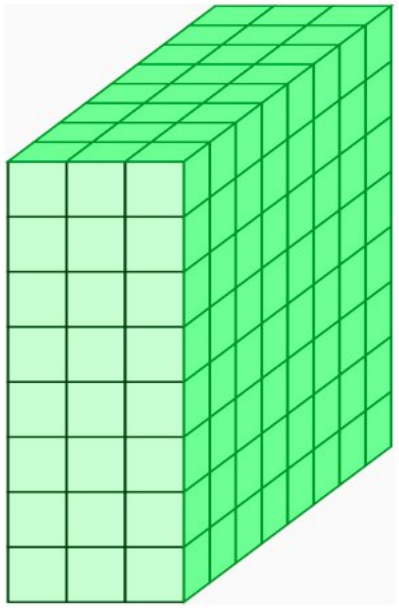


6 карт 6 x 6

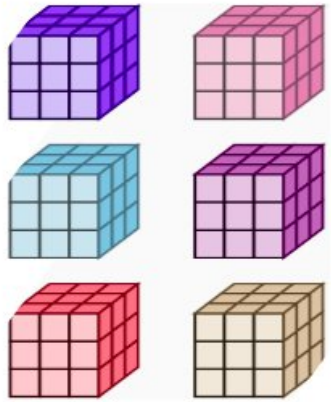
2D-свёртка: многоканальный вход

$$Y_{d,l,m} = \sum_{k=1}^{C} (X_k * W_{dk})_{l,m} + b_d = \sum_{k=1}^{C} \sum_{i=1}^{K_1} \sum_{j=1}^{K_2} w_{dkij} \cdot x_{k,\, l+i-1, m+j-1} + b_d, \quad Y \in \mathbb{R}^{D \times (H-K_1+1) \times (W-K_2+1)}$$

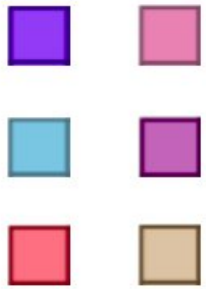
Выходной тензор Y можно использовать как вход для следующего сверточного слоя.



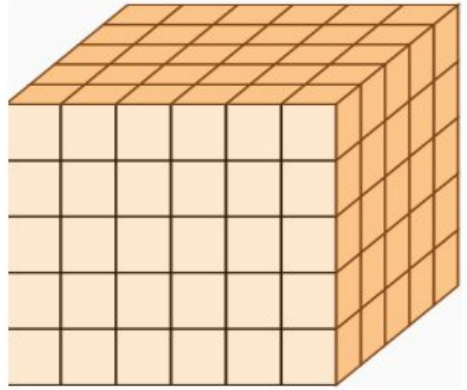
изображение 8 x 8 x 3



6 фильтров 3 x 3 x 3



смещения



6 карт 6 x 6

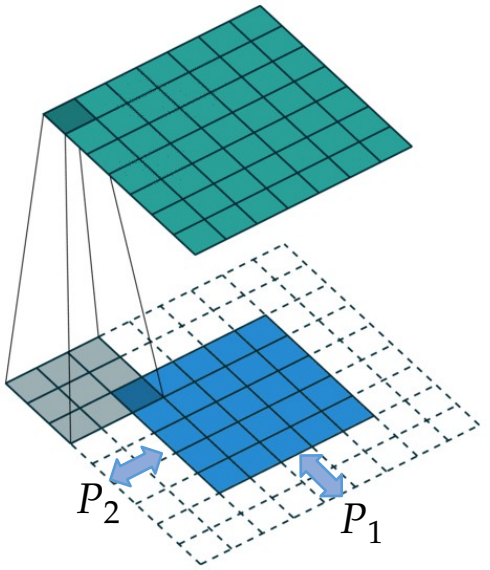
2D-свёртка: padding

Проблемы стандартной свертки

- Выходной размер получается меньше входного.
- Крайние пиксели никогда не оказываются в центре ядра.

Параметр свертки padding добавляет по краям мнимые пиксели.

- Выходной размер: $D \times (H - K_1 + \mathbf{2P_1} + 1) \times (W - K_2 + \mathbf{2P_2} + 1)$
- При $P_1 > \frac{K_1}{2}$, $P_2 > \frac{K_2}{2}$ крайние пиксели могут оказаться в центре ядра.



$P_1 = P_2 = 2$

Какими значениями заполнять?

Константой, напр. нулем

Самый популярный вариант.
Сеть учиться понимать,
что окно находится
на границе картинки.

Зеркально

Подходит для случаев,
когда нужно
минимизировать
артефакты на границах
изображения.

Циклично

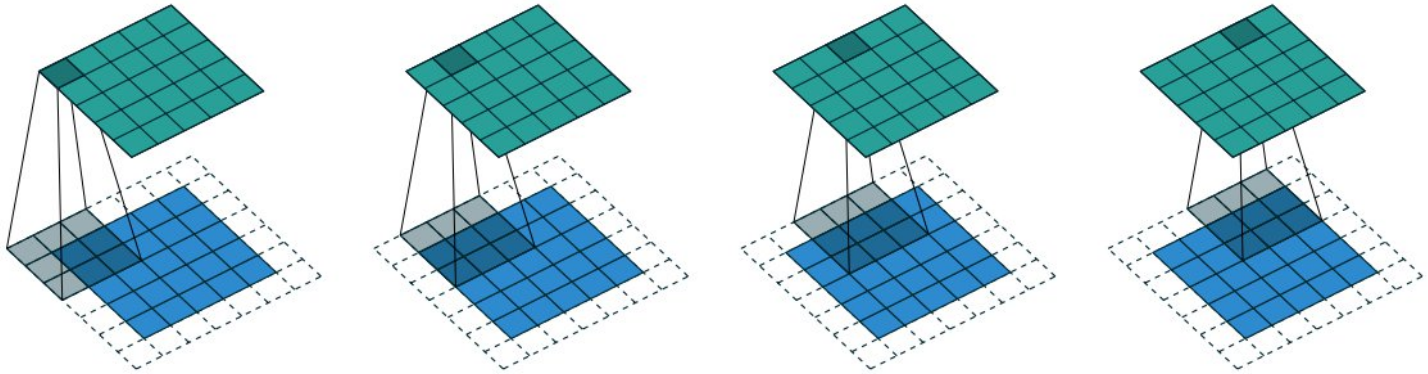
Подходит для случаев,
когда данные обладают
циклической структурой.

2D-свёртка: stride

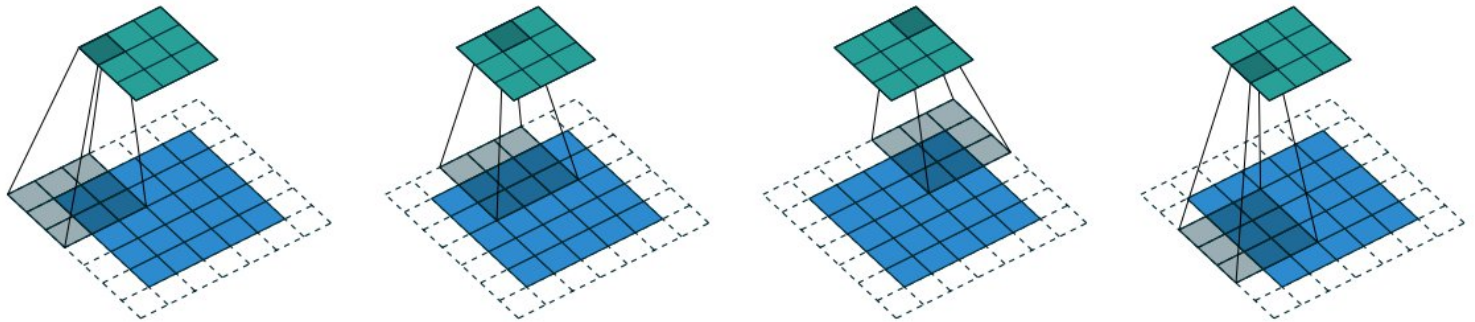
Параметр свертки stride соответствует шагу, с котором перемещаем ядро свертки.

Выходной размер: $D \times \left(\frac{\lfloor H - K_1 + 2P_1 \rfloor}{S_1} + 1 \right) \times \left(\frac{\lfloor W - K_2 + 2P_2 \rfloor}{S_2} + 1 \right)$ — уменьшается с ростом S_1, S_2.

- $S_1 = S_2 = 1$ (стандартный) ядро сдвигается на 1 пиксель.



- $S_1 = S_2 = 2$ ядро сдвигается на 2 пикселя.



Больше визуализаций: онлайн визуализатор свертки, гитхаб с подробным объяснением сверток

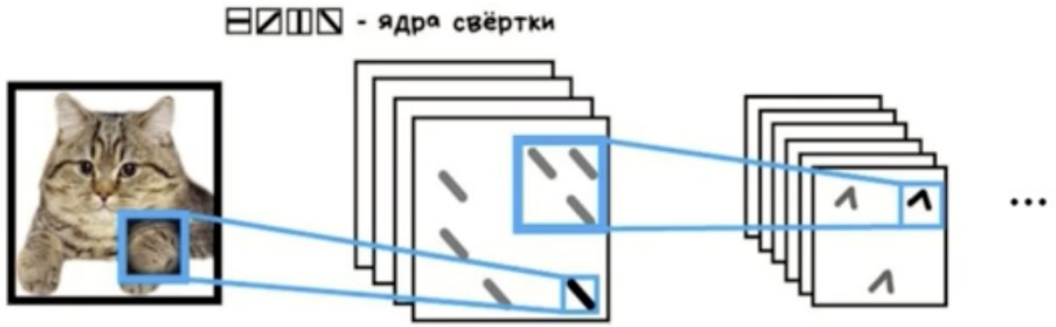
Больше свёрток!

Одной свёрткой

- можем узнать только наличие простых паттернов на картинке
- не можем найти сложные паттерны
  даже если фильтр будет изображать морду кота, то мы вряд ли найдем что-то похожее на картинке с котом, ведь коты бывают разные.

Сделаем несколько свёрток подряд.

- Первая свёртка будет искать простые паттерны на исходной картинке.
- Вторая будет искать простые паттерны уже на картах после первой свёртки. Простые паттерны из простых паттернов —уже более сложные паттерны.
- …

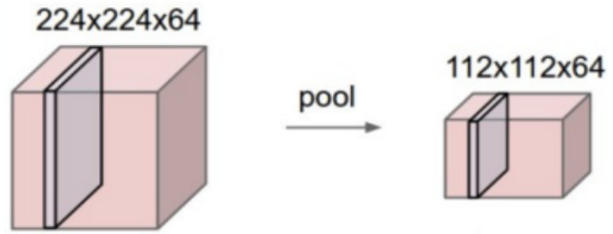


2D Pooling

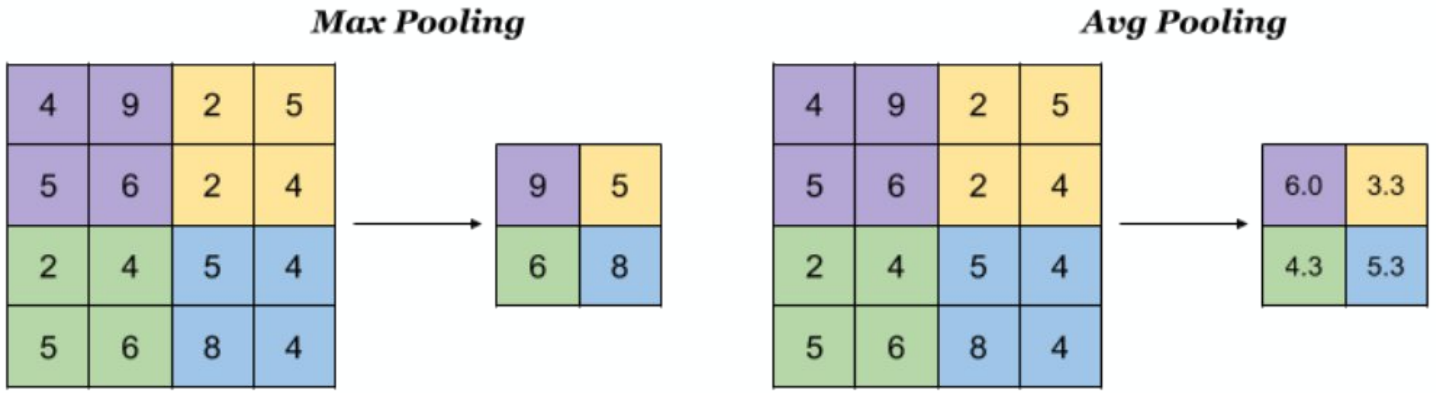
2d Pooling — такая операция, где мы скользим окном по входным данным и вычисляем некоторую функцию от его элементов.

Цель — уменьшение размерности.
Применяется обычно после сверточного слоя.

Хотим выделить наиболее важные признаки среди огромного числа признаков изображения, полученных после свертки.



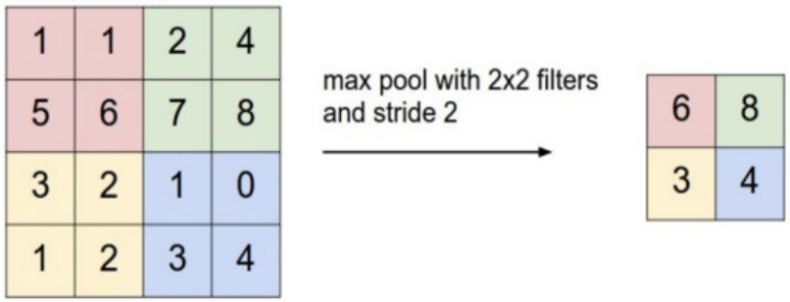
Используются такие функции как максимум и среднее, которые соответствуют **Max Pooling** и **Avg Pooling**.



2D Pooling

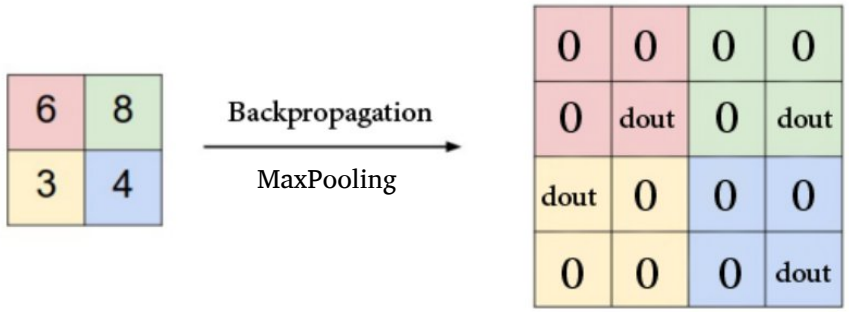
Гиперпараметры

- **Размер ядра** (K_1, K_2): часто берут $K_1 = K_2 = 2$
- **Шаг (stride)** (S_1, S_2), с которым будем перемещать окно: часто берут $S_1 = S_2 = 2$.



Backpropagation

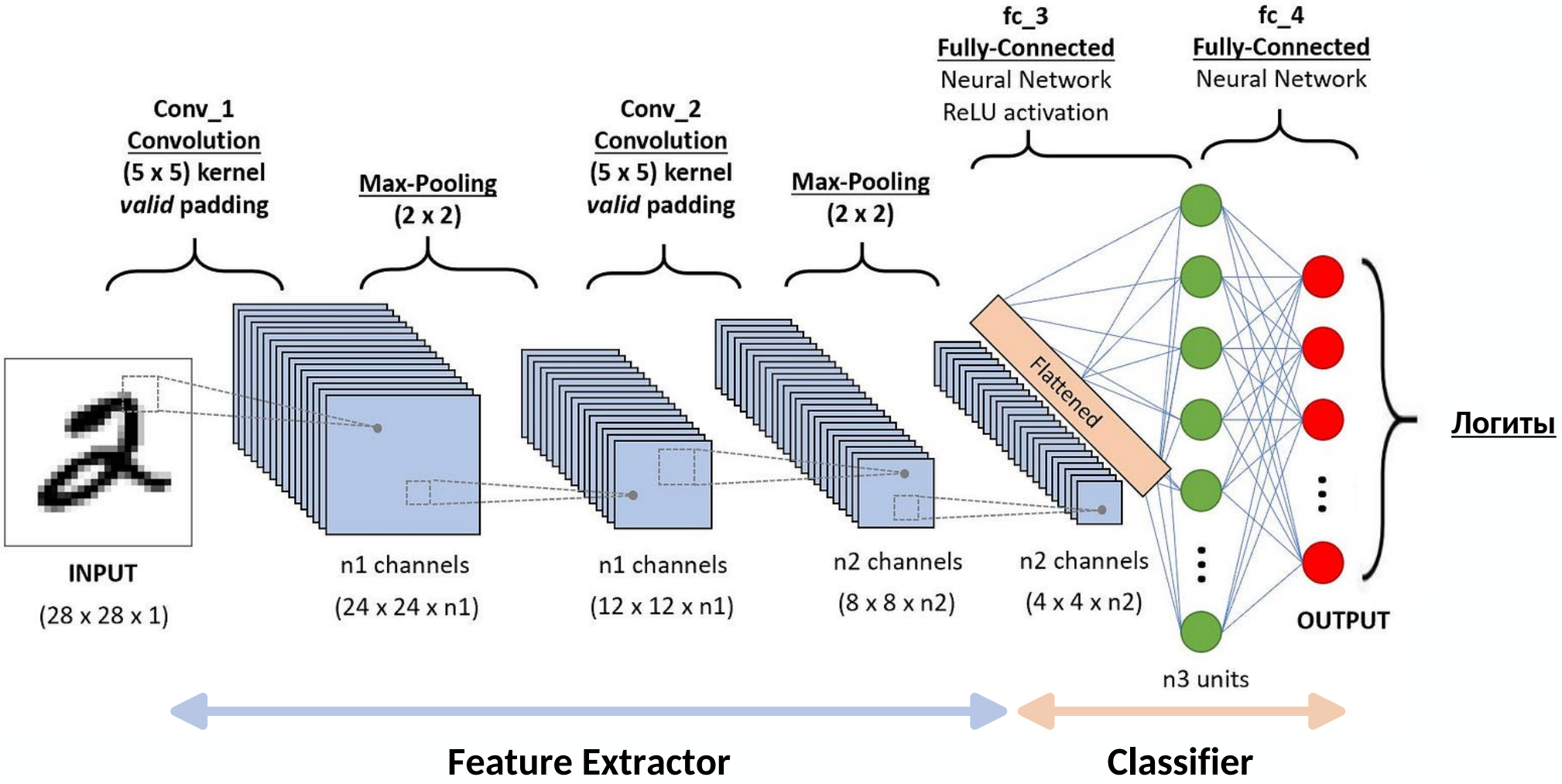
- для **MaxPooling** градиент потечет назад только через значения, выбранные в качестве максимумов.
- для **AvgPooling** градиент потечет назад равномерно по значениям.



Введение в компьютерное зрение

- Компьютерное зрение
- Классификация изображений
- Свертка и Pooling
- **Классификация с помощью CNN**
- Обзор задач компьютерного зрения

Классификация с помощью CNN

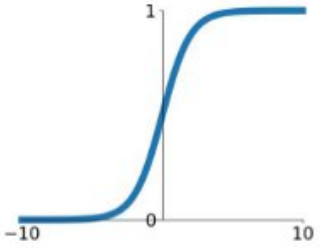


Предсказание вероятности

Бинарная классификация

Нужно нормализовать выход сети на отрезок $[0, 1]$, для этого используется **сигмоида.**

$$\sigma(z) = \frac{1}{1+e^{-z}}$$



Мультиклассовая классификация

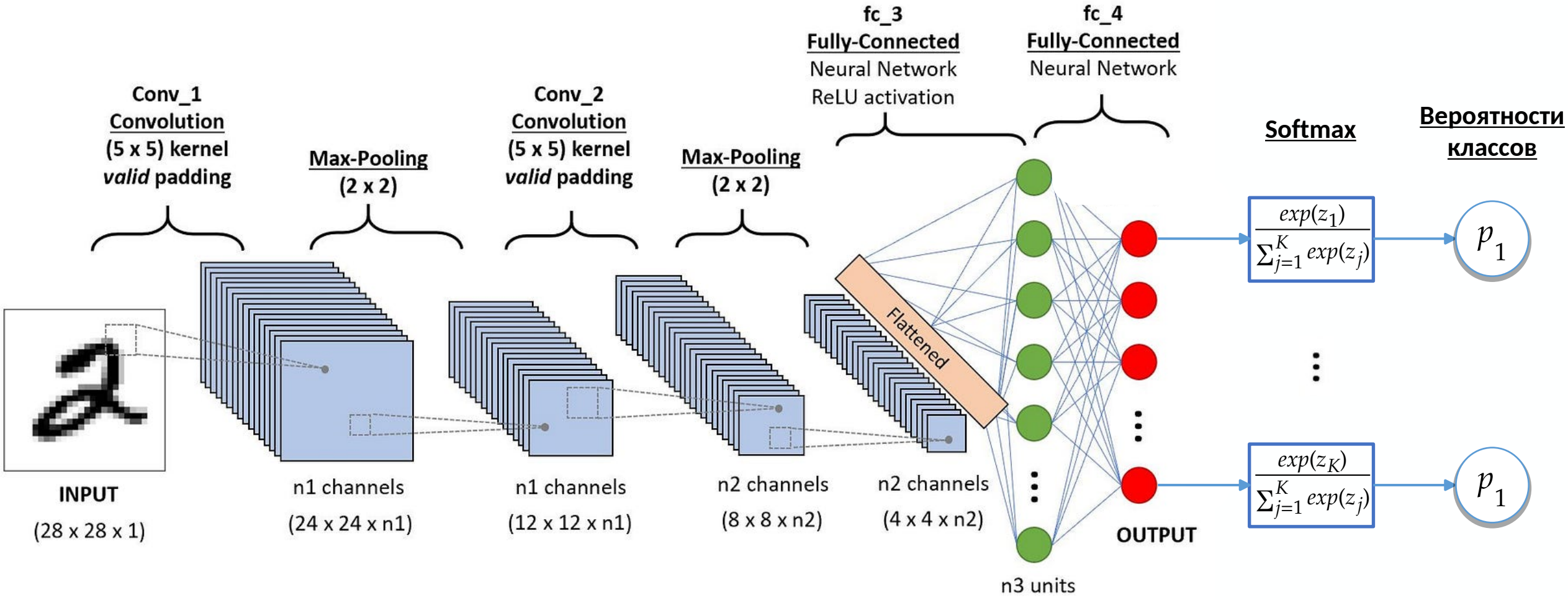
Классы не зависят друг от друга, по сути имеем бинарную классификацию для каждого класса в отдельности. Чтобы отнормировать значения выходов модели на отрезок применяется функция **SoftMax**.

$$f(z)_k = \frac{exp(z_k)}{\sum_{j=1}^{K} exp(z_j)},\ k \in \{1, \ldots, K\},\ K$$ – число классов

Почему бы просто не взять argmax из значений как метку класса?

Градиент будет нулевым практически везде, что не позволит сети обучаться.

Вероятности классов из логитов



Алгоритм обучения

1. **Прямой проход**:
   - вход X — батч картинок (n, H, W, C), подается в сеть с параметрами θ;
   - выход $\hat{y}_\theta(X)$ — вероятности размерности (n, K);
   - Y — one-hot представление истинных меток класса (n, K).

2. **Функция ошибки**: для многоклассовой классификации используется Cross Entropy Loss:

$$L(Y, \hat{y}_\theta) = -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{K} Y_{ij}\log(\hat{y}_\theta(X_i))_j$$

3. **Обратный проход**: считаем градиенты $\nabla_\theta L(Y, \hat{y}_\theta)$
   в порядке от последних к первым слоям.

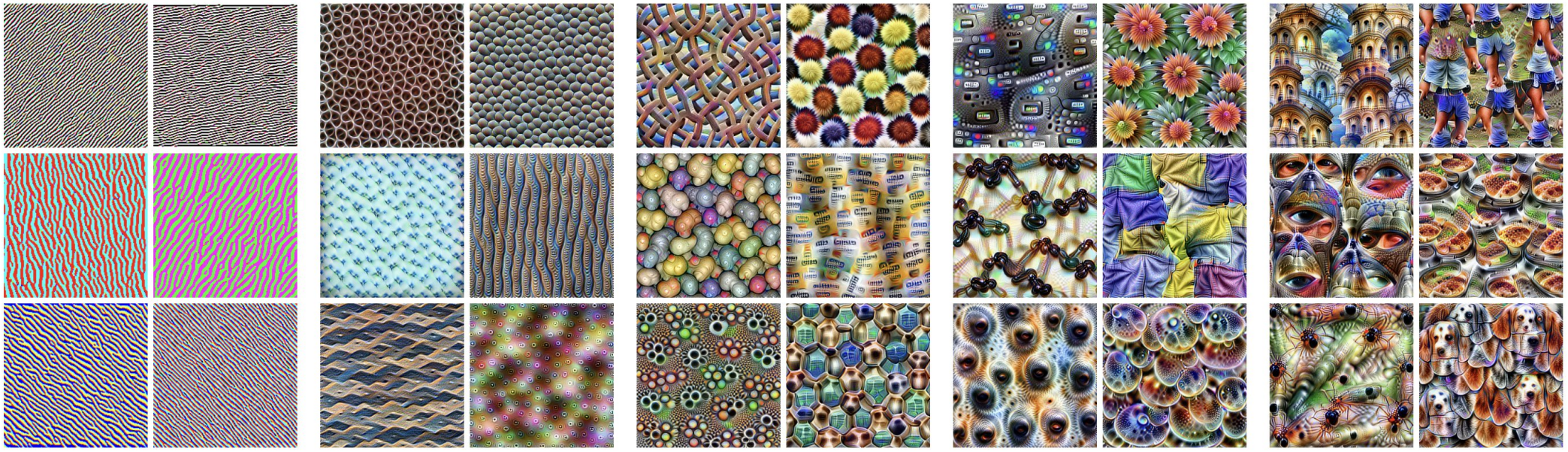
4. **Шаг оптимизации**: $\theta_{t+1} = \theta_t - \eta\nabla_\theta L(Y, \hat{y}_{\theta_t})$

Чему учатся CNN?

Пусть X — вход, Z_{kl} — выход нейрона l на слое k после функции активации.

Так как CNN дифференцируема по своим входам, можно решать задачу $Z_{kl} \longrightarrow \max_X$.

Результат — картинка, больше всего активирующая данный нейрон.

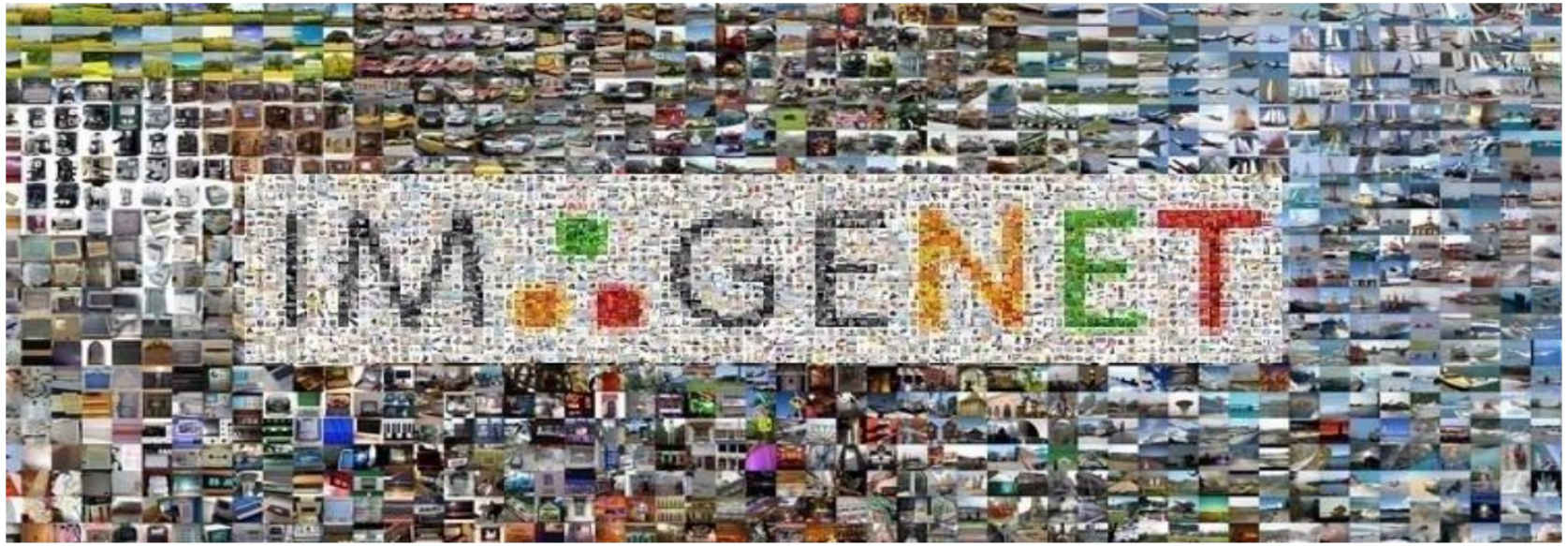


Edges (layer conv2d0) **Textures** (layer mixed3a) **Patterns** (layer mixed4a) **Parts** (layers mixed4b & mixed4c) **Objects** (layers mixed4d & mixed4e)

Историческая справка

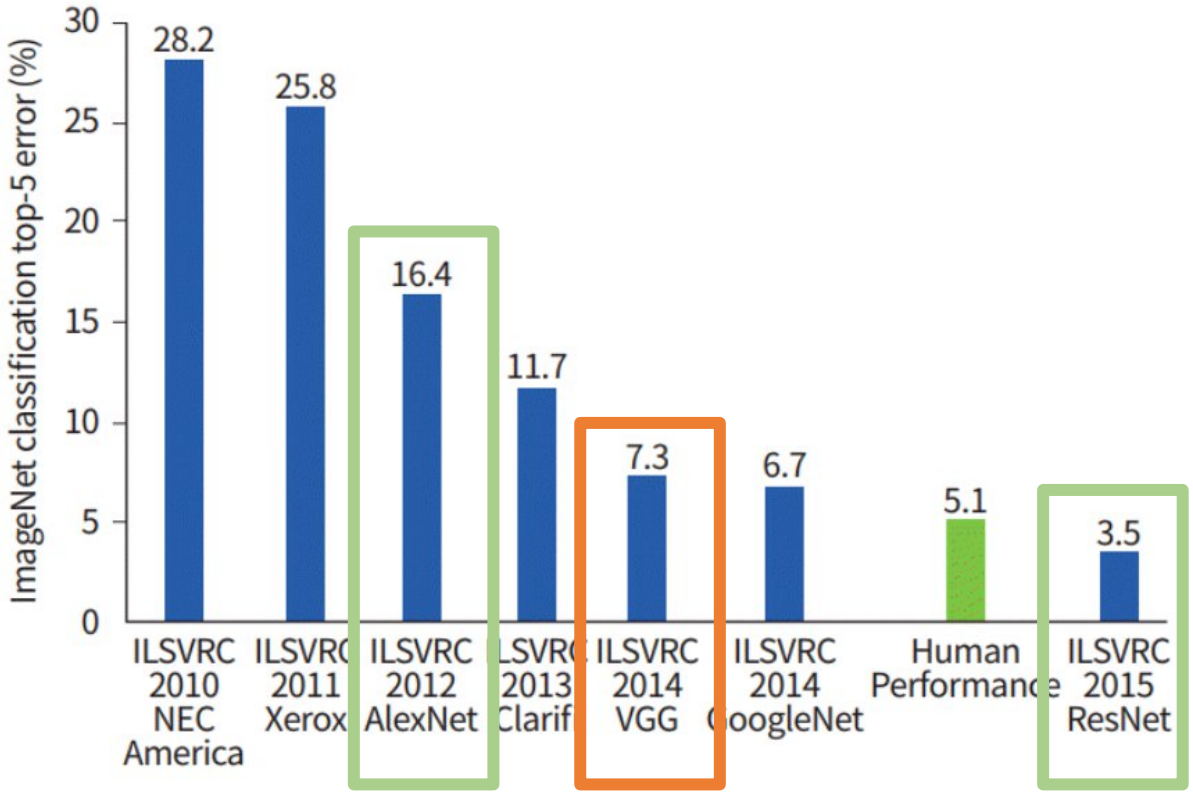
ImageNet — база данных из 14 млн. изображений, размеч. на 20 тысяч классов.

ImageNet Large Scale Visual Recognition Challenge (ILSVRC) — конкурс, использующий 1 млн. изображений с 1000 классов базы ImageNet. Соревнование проводилось с 2010 по 2017 года.



Историческая справка

- 1998 год — Ян ЛеКун предложил архитектуру первой CNN — **LeNet**.
- ~2010 год — появилась **имплементация LeNet**.
- 2012 год — сверточная нейронная сеть **AlexNet** победила в конкурсе ILSVRC.
- 2014 год — сверточная нейронная сеть **VGG** в 2 раза улучшила результат AlexNet в конкурсе ILSVRC.
- 2015 год — сверточная нейронная сеть **ResNet** обогнала по качеству человека в конкурсе ILSVRC.

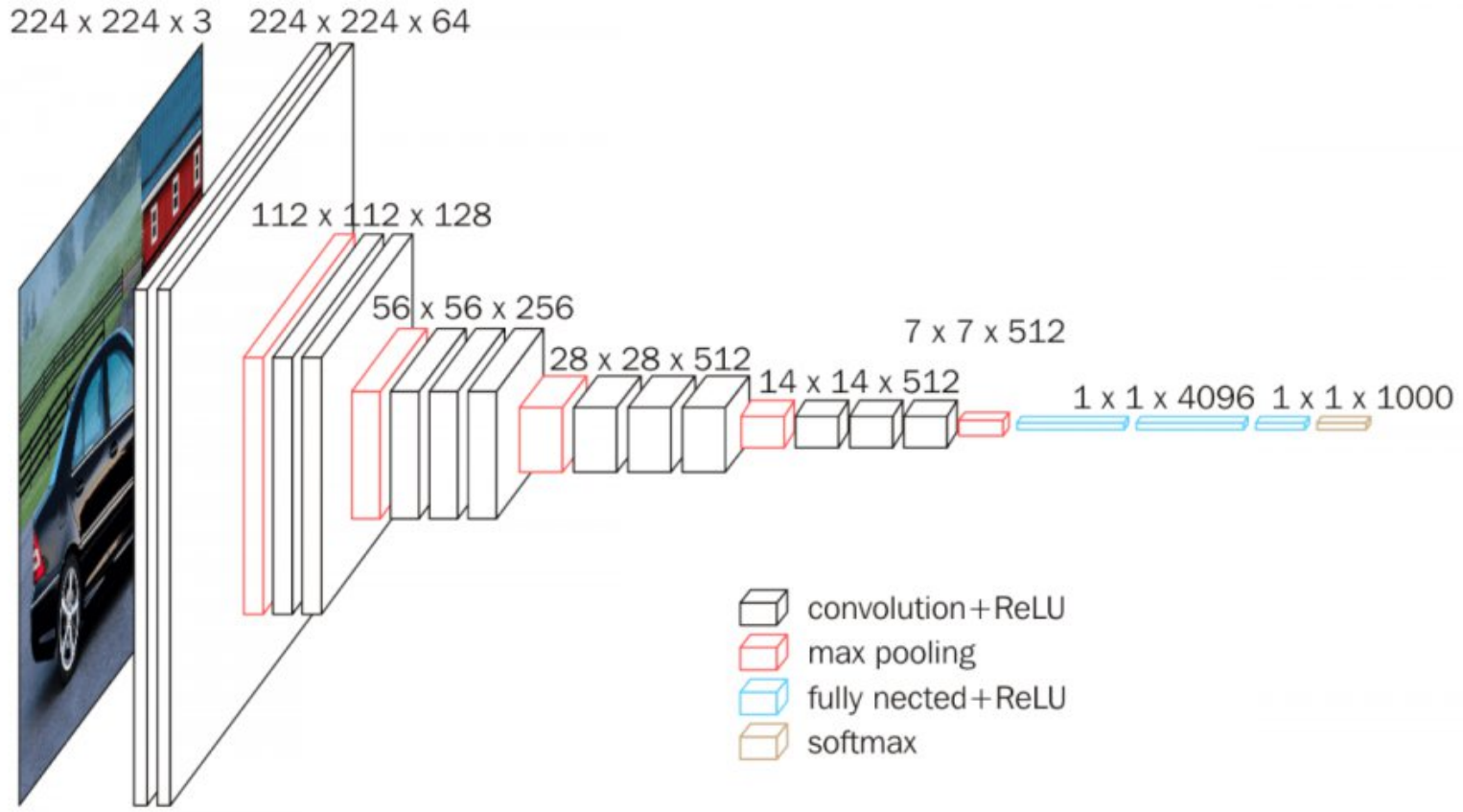


Больше про разные сверточные сети узнаем на 3 курсе DS-потока.

VGG16

Актуальная сверточная архитектура.

- Состоит из сверточных, maxpooling и полносвязных слоев.
- Также есть версия VGG19 и доп. версии VGG11 и VGG13.



138 млн параметров:
14 млн — сверточная часть,
124 млн — полносвязная часть.

Введение в компьютерное зрение

- Компьютерное зрение
- Классификация изображений
- Свертка и Pooling
- Классификация с помощью CNN
- **Обзор задач компьютерного зрения**

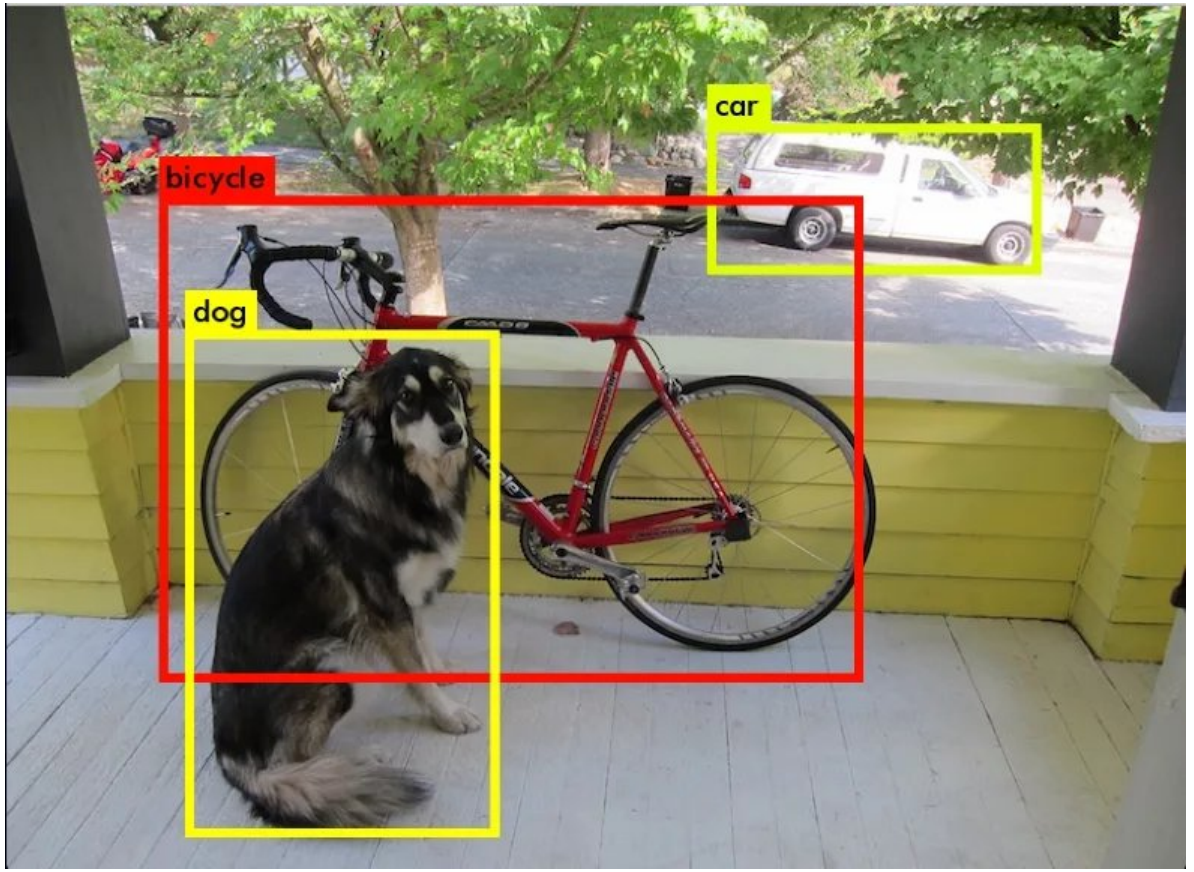
Детекция

Задача

Построить модель, определяющую координаты всех объектов на изображении.

Идея

Предсказывать координаты ограничивающих прямоугольников, для разных объектов, решая задачу регрессии и классификации одновременно.

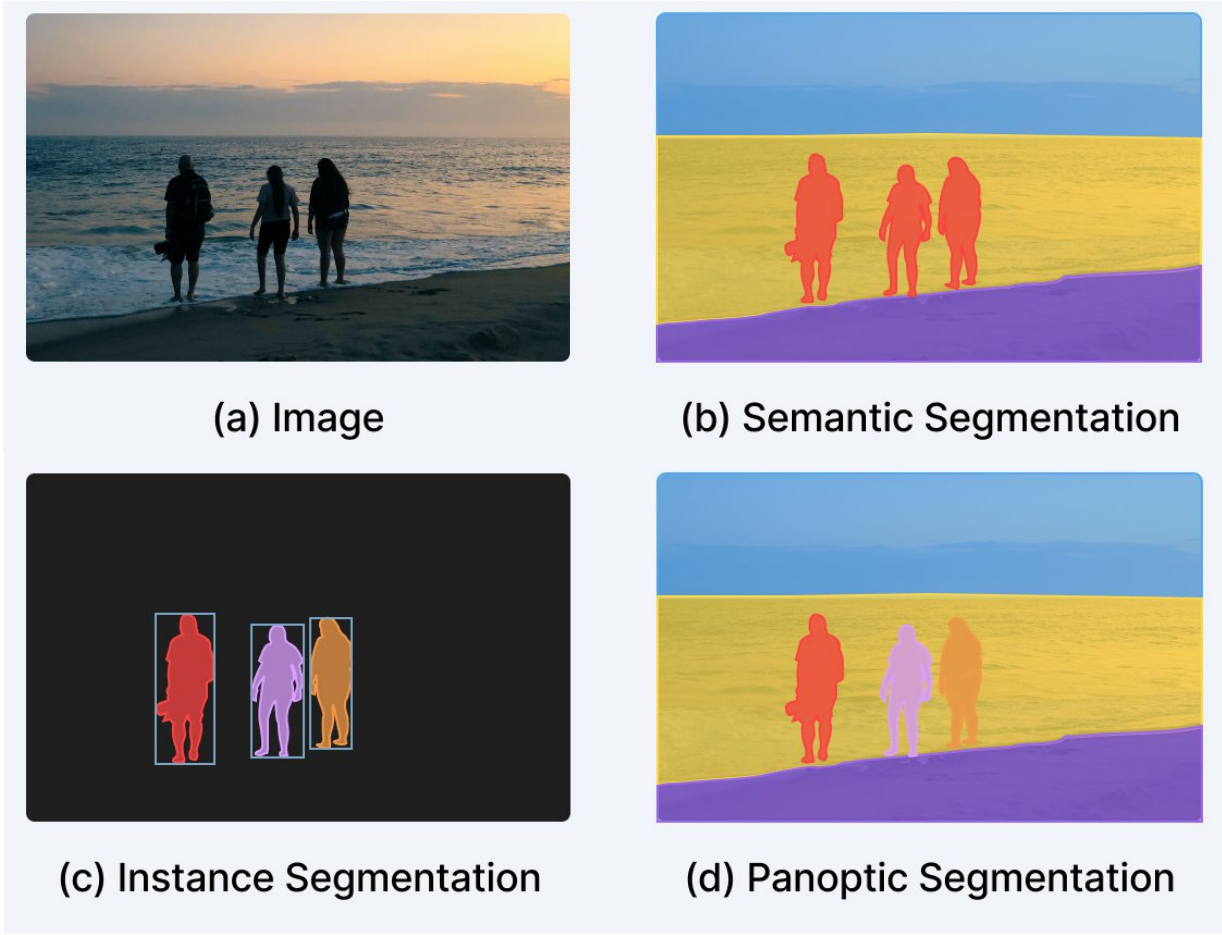


Больше про детекцию узнаем на 3 курсе DS-потока.

Сегментация

Задача

- **Semantic**: классифицировать каждый пиксель изображения.
- **Instance**: вывести bounding box и маску пикселей для каждого объекта, классифицировать объект.
- **Panoptic**: классифицировать каждый пиксель изображения, при этом разделяя объекты одного класса.



Больше про сегментацию узнаем на 3 курсе DS-потока.

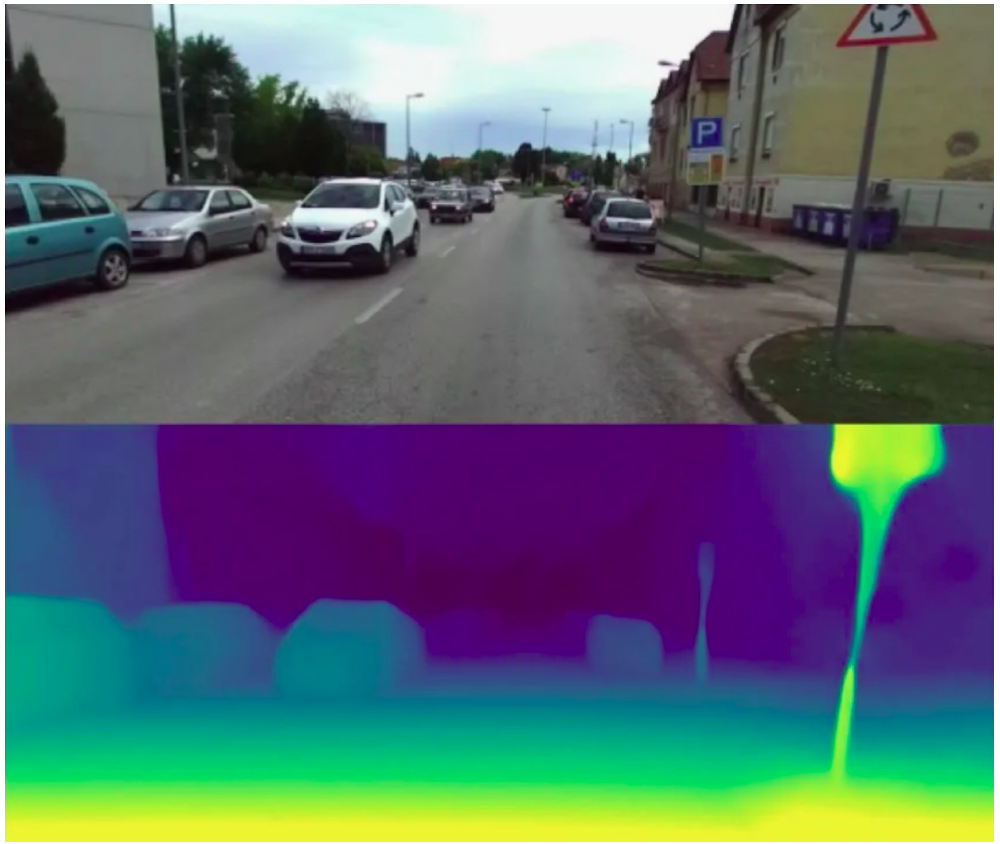
Оценка глубины

Задача

Построить модель, определяющую расстояние до всех объектов на изображении.

Идея

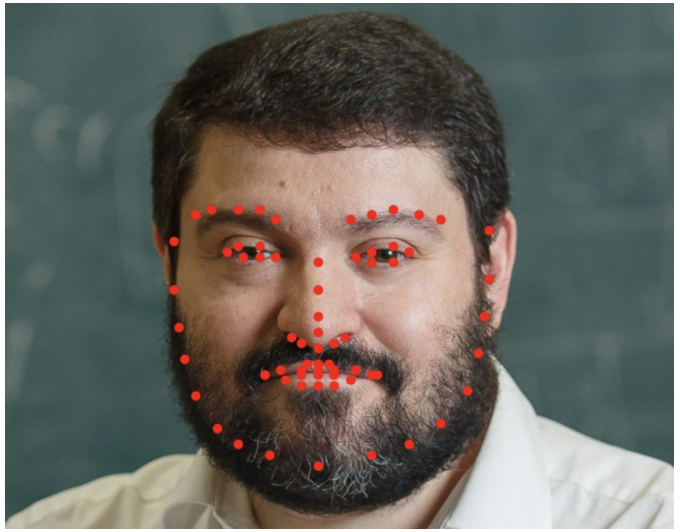
Предсказывать относительную глубину для каждого пикселя сцены.



Keypoint Estimation

Задача. Построить модель, определяющую координаты заданных ключевых точек лица или тела.

Идея. Сначала решить задачу детекции объектов, а затем для каждого из них предсказывать координаты точек.



Больше про Keypoint Estimation узнаем на 3 курсе DS-потока.

Генерация изображений

Подробнее в следующей презентации!