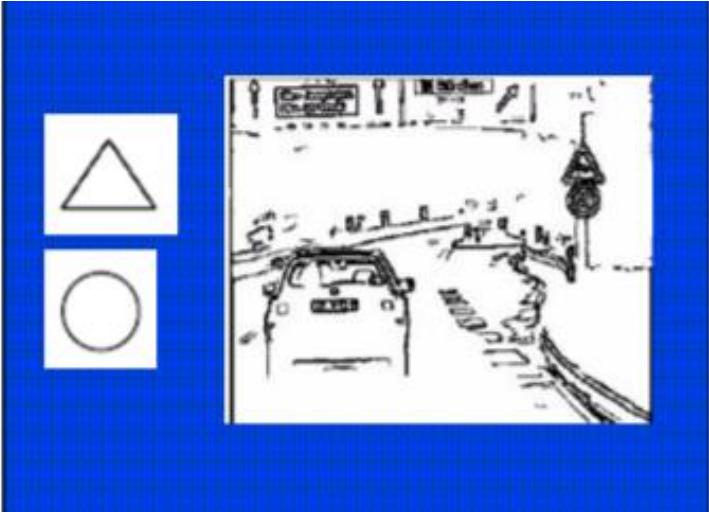
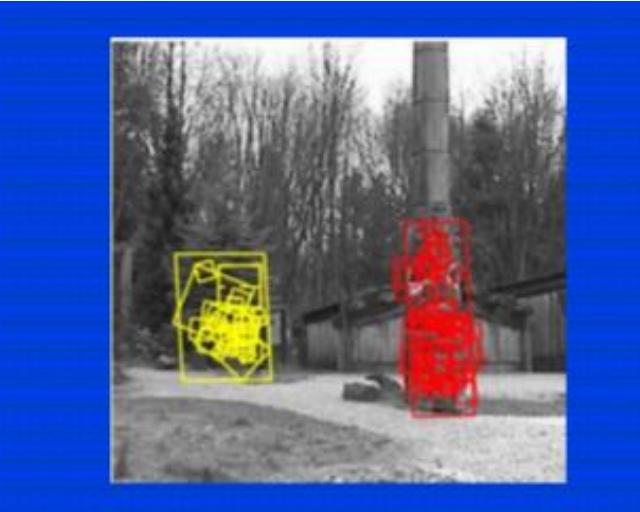


# Первые подходы к распознаванию объектов

Поиск шаблонов



Локальные  
особенности



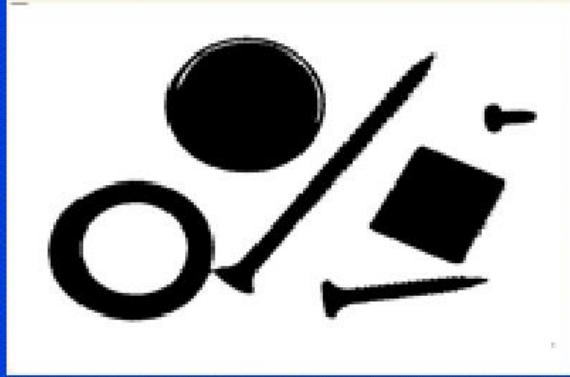
Инвариантные  
признаки для  
выделенных  
сегментов



Все эти методы нашли свое применение, но решить задачу распознавания полностью не получилось

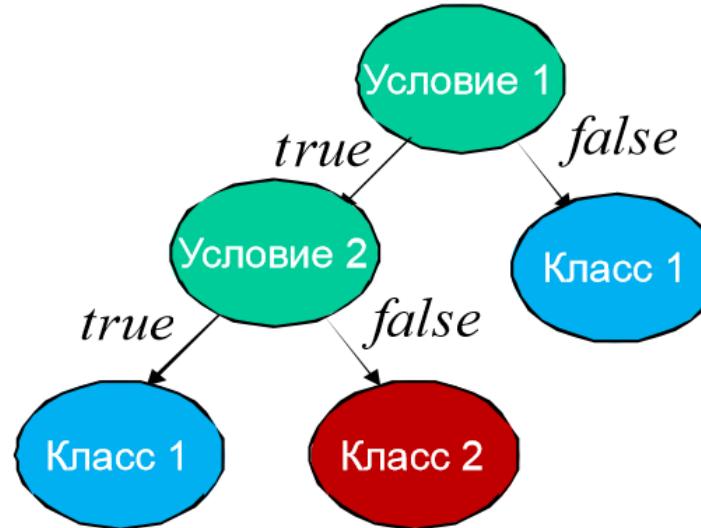
# Проблемы распознавания

---



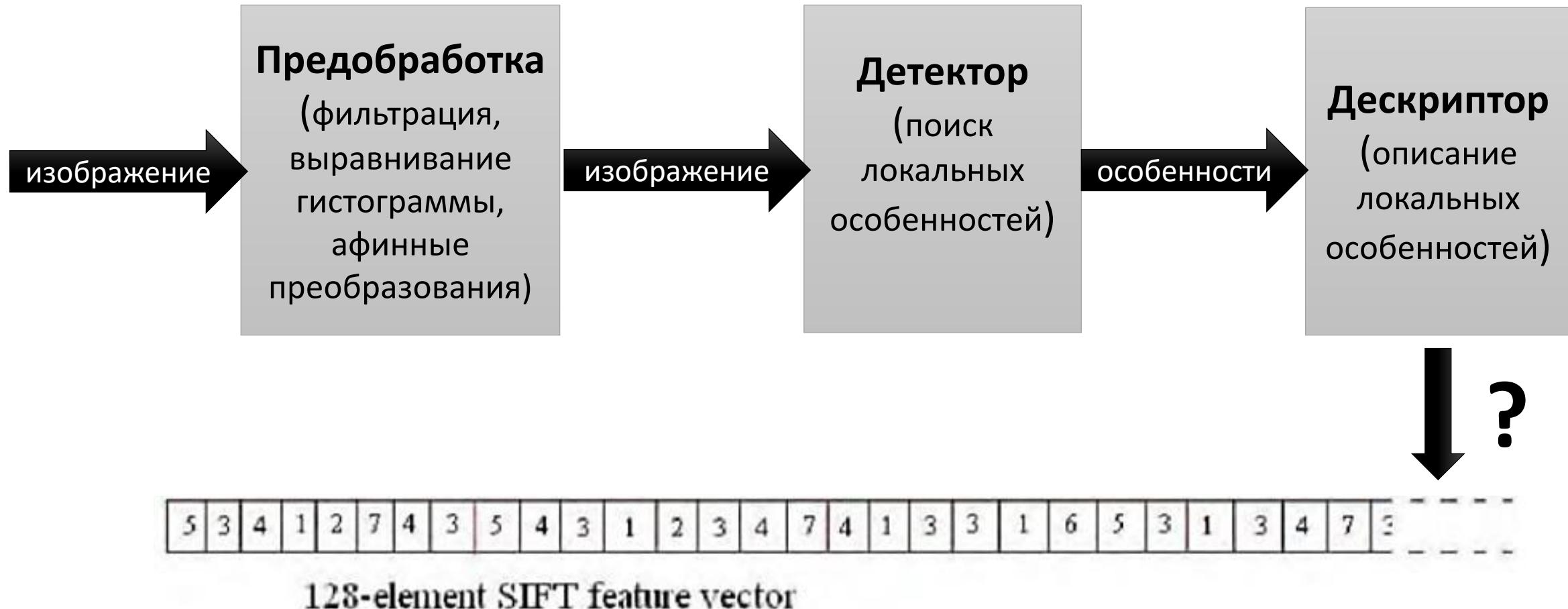
Геометрические  
признаки для  
выделенных  
сегментов

В чём причины  
ограниченности успехов  
этого подхода?

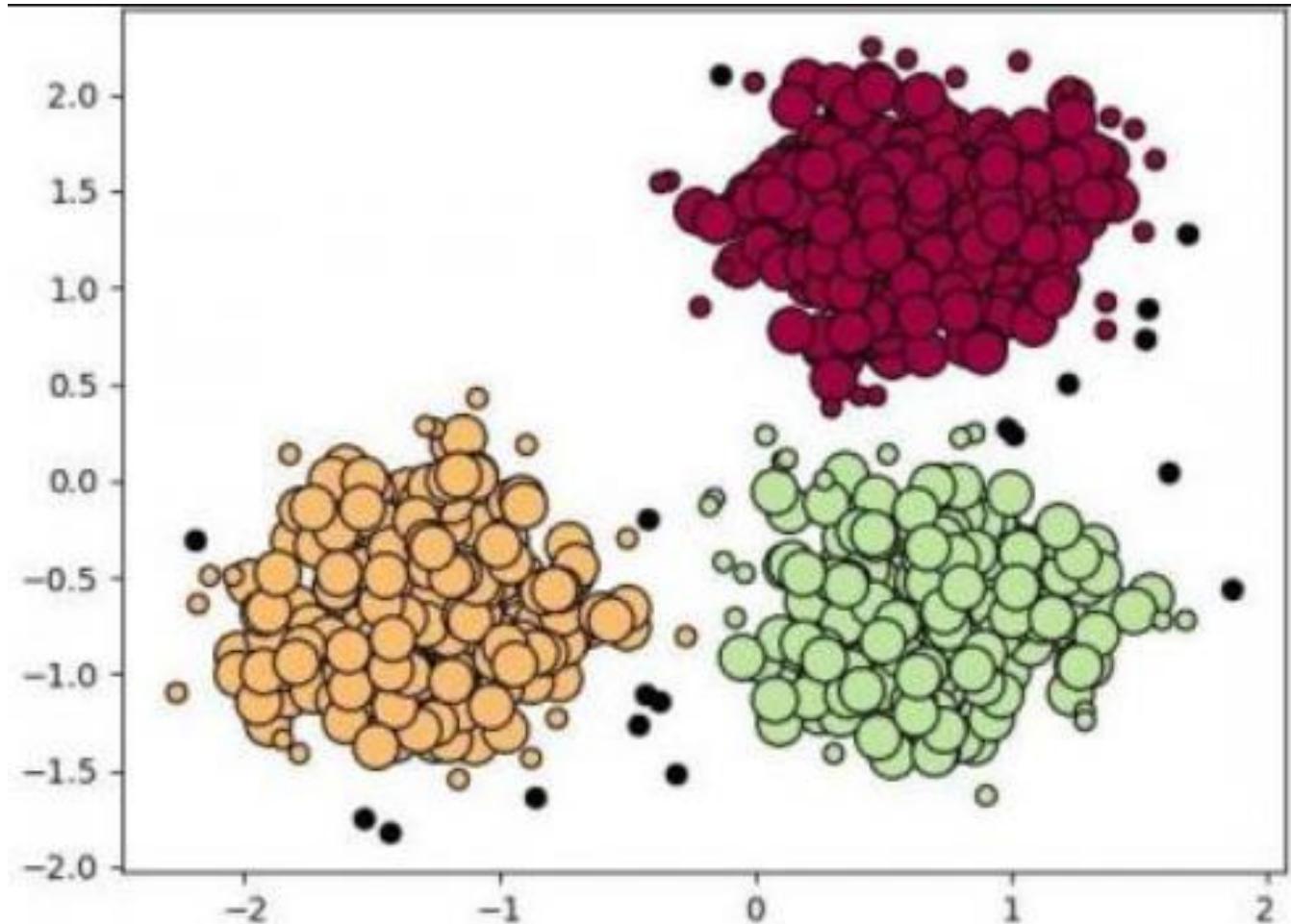


- Правила приходится подбирать вручную
- Признаки нужно использовать «осмыслиенные» и очень информативные
- Таких признаков мало, и сложные комбинации признаков обработать уже нельзя

Что мы уже знаем (изучили) и можем применить для классификации объектов на изображении?



# Классификация по признакам



Допустим есть два признака, отложим их значение по осям и увидим локализацию объектов по классам

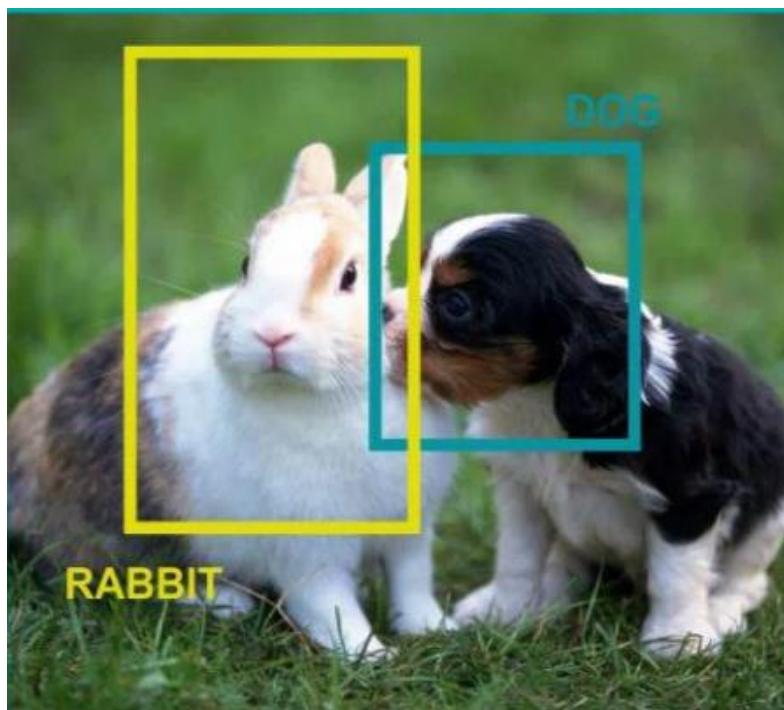
В случае в дескриптором SIFT пространство будет 128-мерное !

# Машинное обучение

# Обучение с учителем. Классификация

В машинном обучении задача классификации относится к разделу обучения с учителем.

Обучение с учителем подразумевает, что данные в обучающей выборке должны быть размечены (в случае распознавания изображений, символов) или каждому набору признаков должны соответствовать метки (классы).

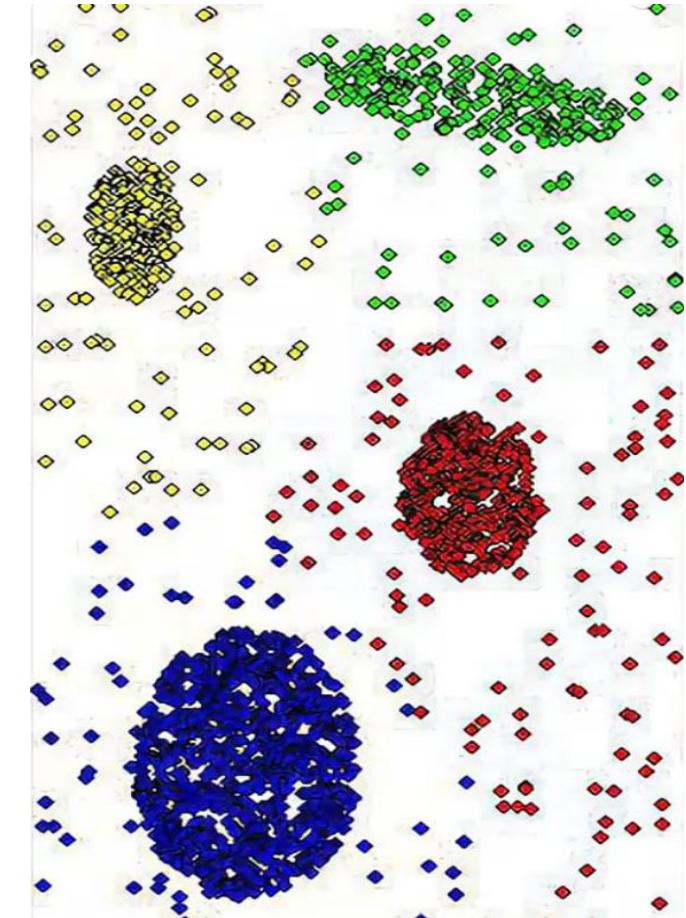


PassengerId	Survived	Pclass	Name	Sex	Age
1	0	3	Braund, Mr. Owen Harris	male	22.0
2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th... Heikkinen, Miss. Laina	female	38.0
3	1	3	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	26.0
4	1	1	Allen, Mr. William Henry	male	35.0
5	0	3			

# Обучение без учителя. Кластеризация

К задачам обучения без учителя относят задачи **кластеризации**

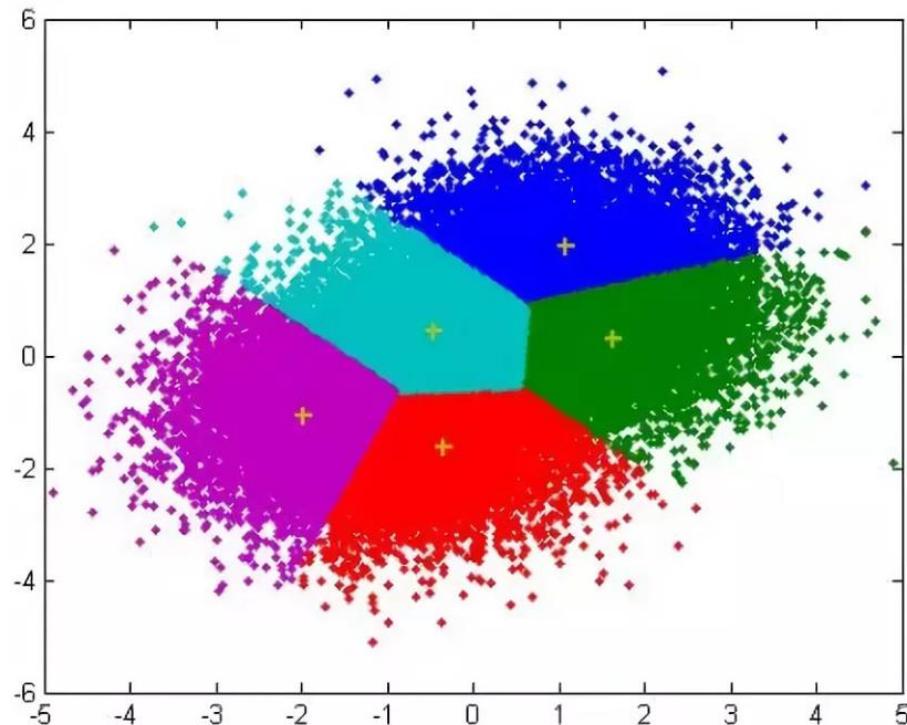
Задача кластеризации – это задача разбиения заданного набора объектов на непересекающиеся подмножества (кластеры), т. е. группы близких по своему признаковому описанию объектов. «Похожие» друг на друга объекты должны входить в один кластер, «не похожие» объекты должны попасть в разные кластеры.



# Метод центров тяжести (k-means)

Для решения задачи кластеризации в библиотеки OpenCV реализован метод центров тяжести (k-means).

Метод центров тяжести разбивает выборку на заданное количество кластеров путем выбора их центров. Поиск центров кластеров производится из соображений минимизации суммарного расстояния от каждой точки до ближайшего центра с помощью метода локальной оптимизации.



Пример сегментации изображения



Все рассмотренные методы машинного обучения получают на вход числовые признаки объектов и классифицируют объекты на изображении.

**Вопрос:** Какие методы машинного обучения для классификации вы знаете?

**Вопрос:** Откуда взять числовые признаки?

**Ответ:** детекторы + дискрипторы

# Мешок визуальных слов (Bag of Visual Words)

---

**Мешок визуальных слов (BOVW)** обычно используется в классификации изображений.

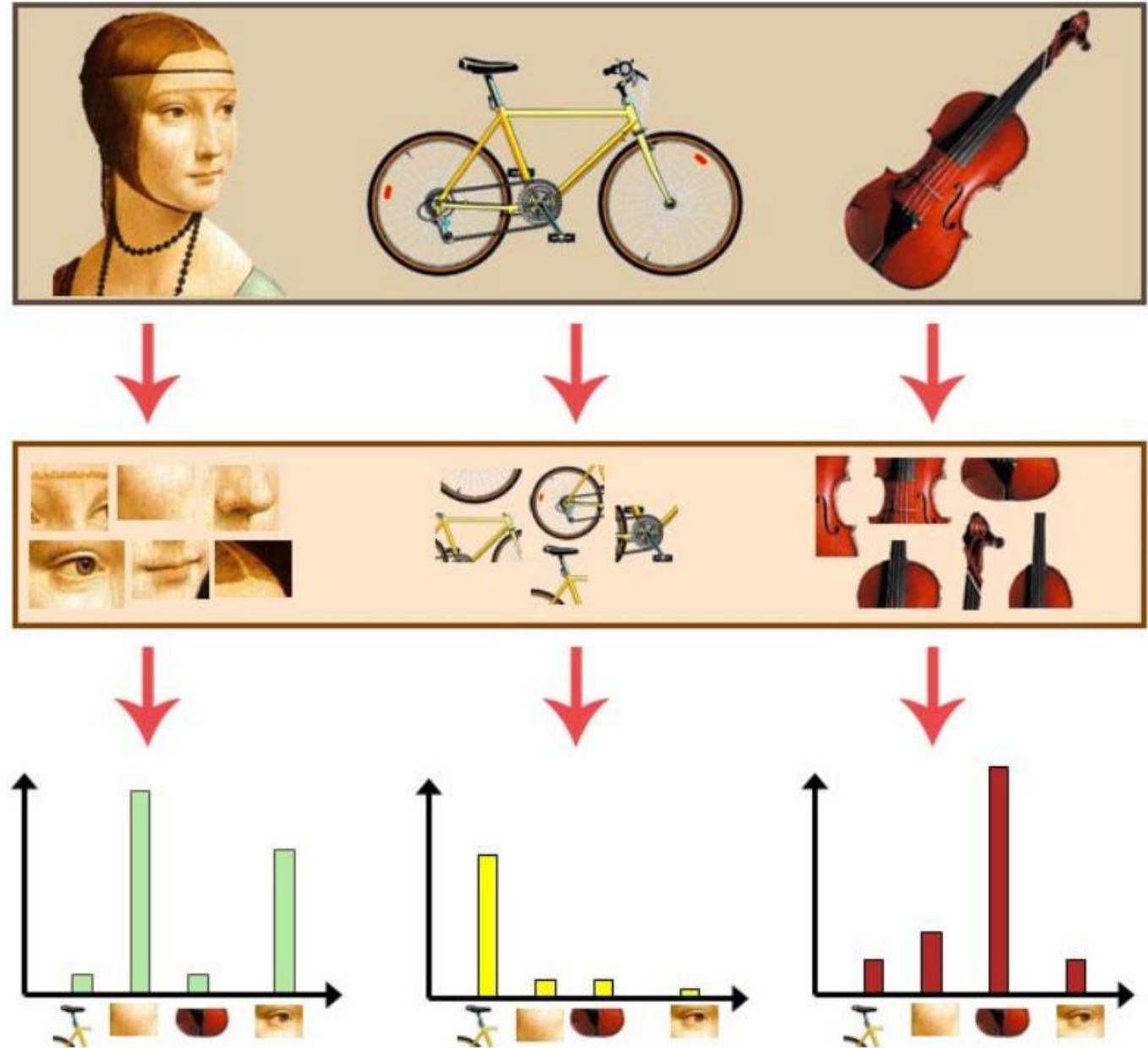
Его концепция адаптирована из информационного поиска и пакета слов НЛП (BOW). В мешке слов (*BOW*) мы подсчитываем количество каждого слова, которое появляется в документе, используем частоту каждого слова, чтобы узнать ключевые слова документа, и делаем из него гистограмму частоты. Мы рассматриваем документ как пакет слов (*BOW*).

Та же концепция в наборе визуальных слов (BOVW), но вместо слов мы используем функции изображения в качестве «слов». Функции состоят из ключевых точек и дескрипторов.

Ключевые точки — это «особые» точки изображения, поэтому независимо от того, поворачивается ли изображение, сжимается или расширяется, его ключевые точки всегда будут одинаковыми. А дескриптор — это описание ключевой точки.

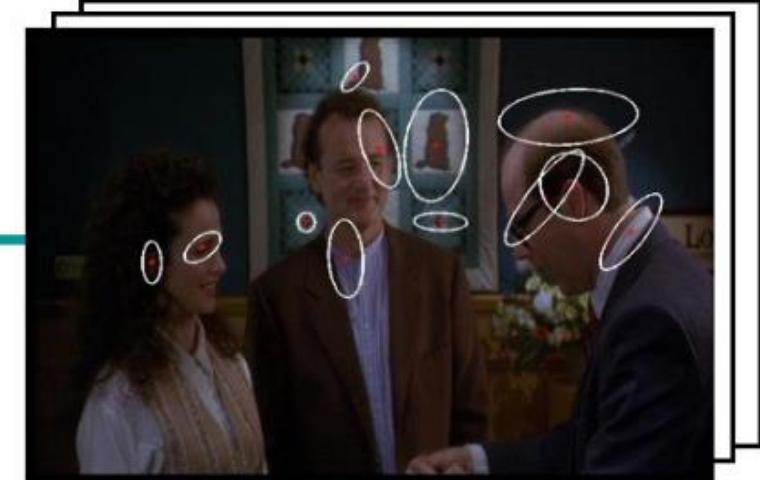
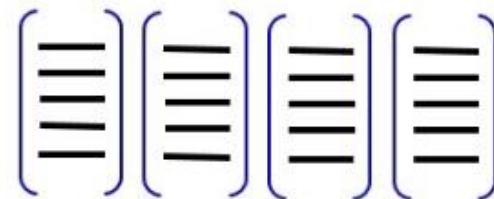
Особенности могут состоять из локальных особенностей (угловых точек, блобов, областей) и их дескрипторов.

Особенности и дескрипторы используются для создания словарей и представления каждого изображения в виде частотной гистограммы объектов, которые находятся в изображении.

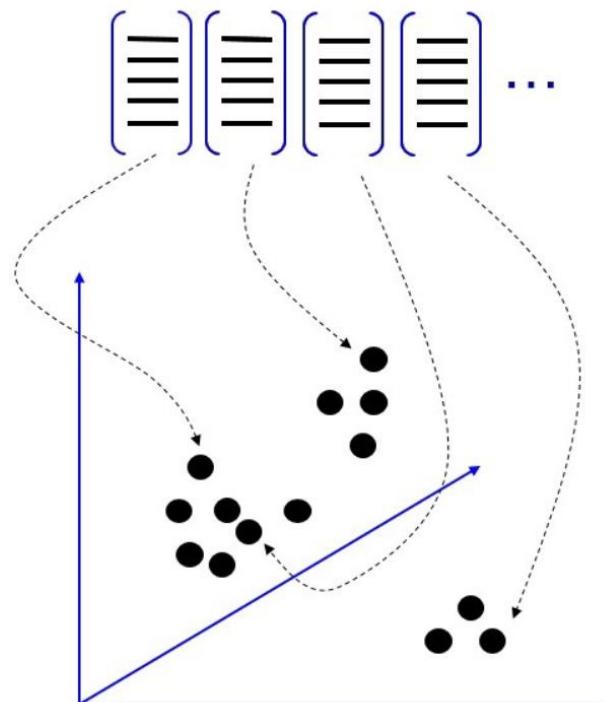


# Алгоритм метода визуальных слов (BOVW)

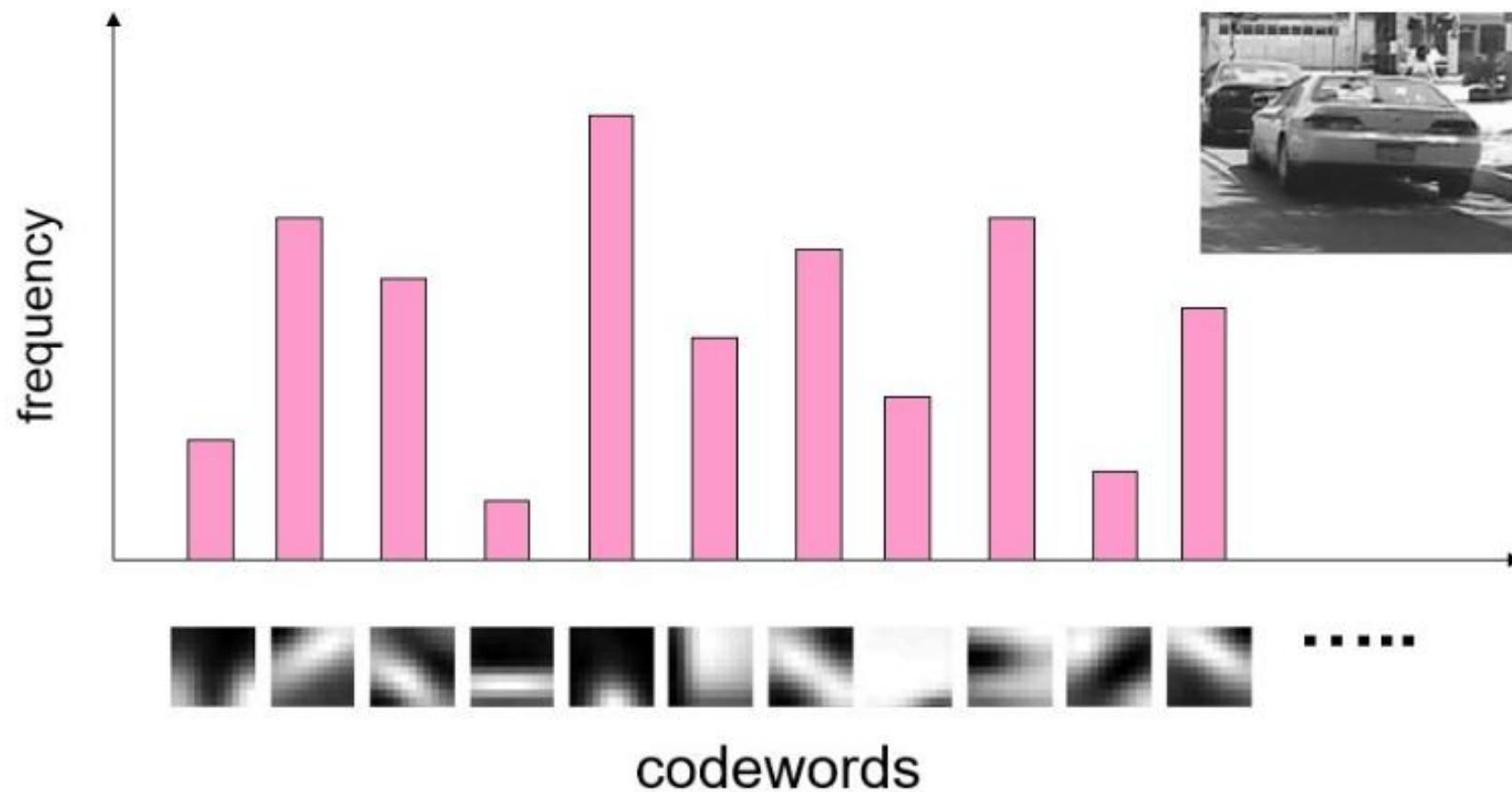
1. Обнаруживаем особенности, извлекаем дескрипторы



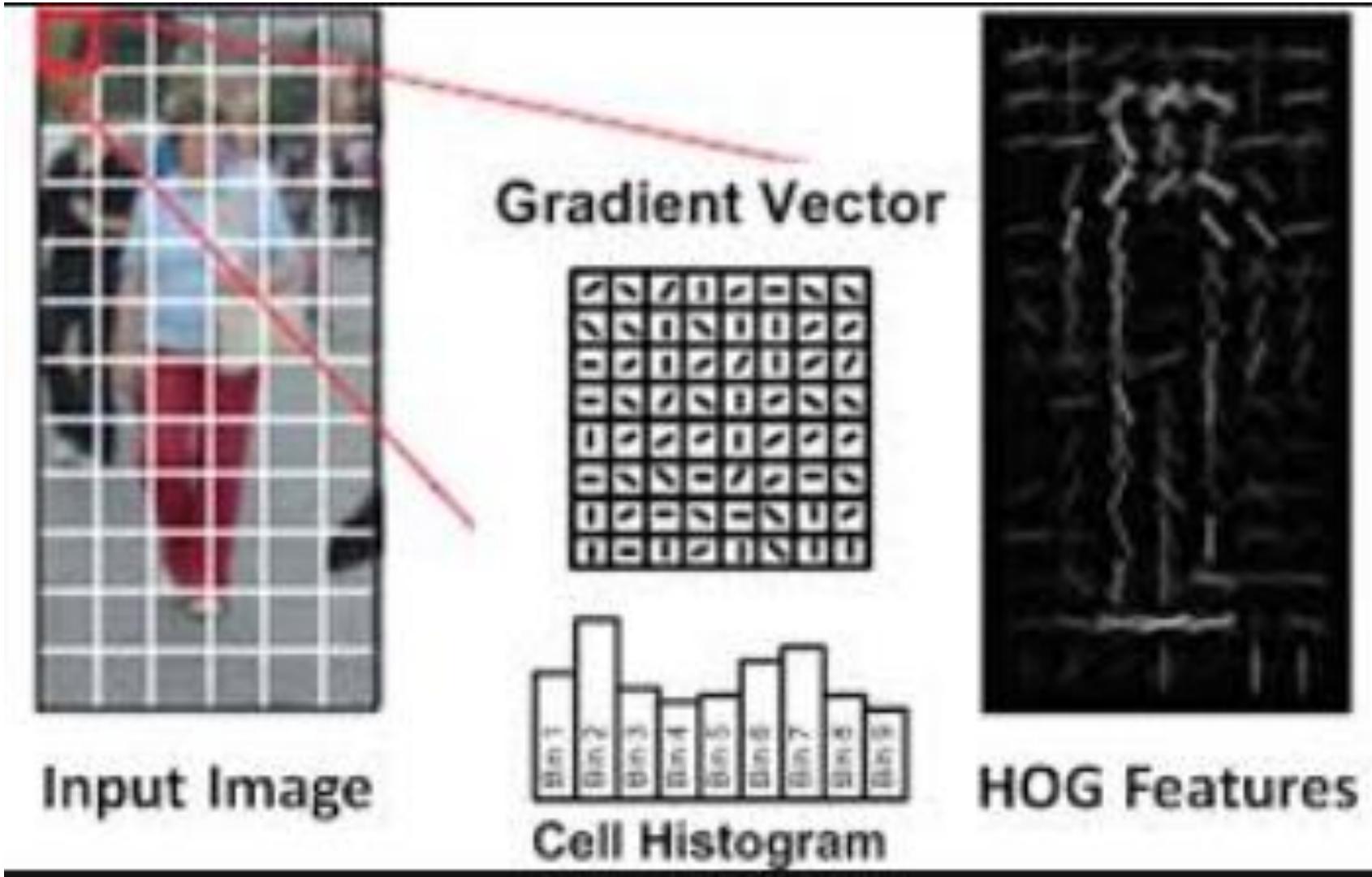
2. Делаем кластеры из дескрипторов (можно использовать K-Means, DBSCAN или другой алгоритм кластеризации).



3. Составляем частотную гистограмму из словарей и частоту словарей в изображении. Эти гистограммы - наш мешок визуальных слов (BOVW).



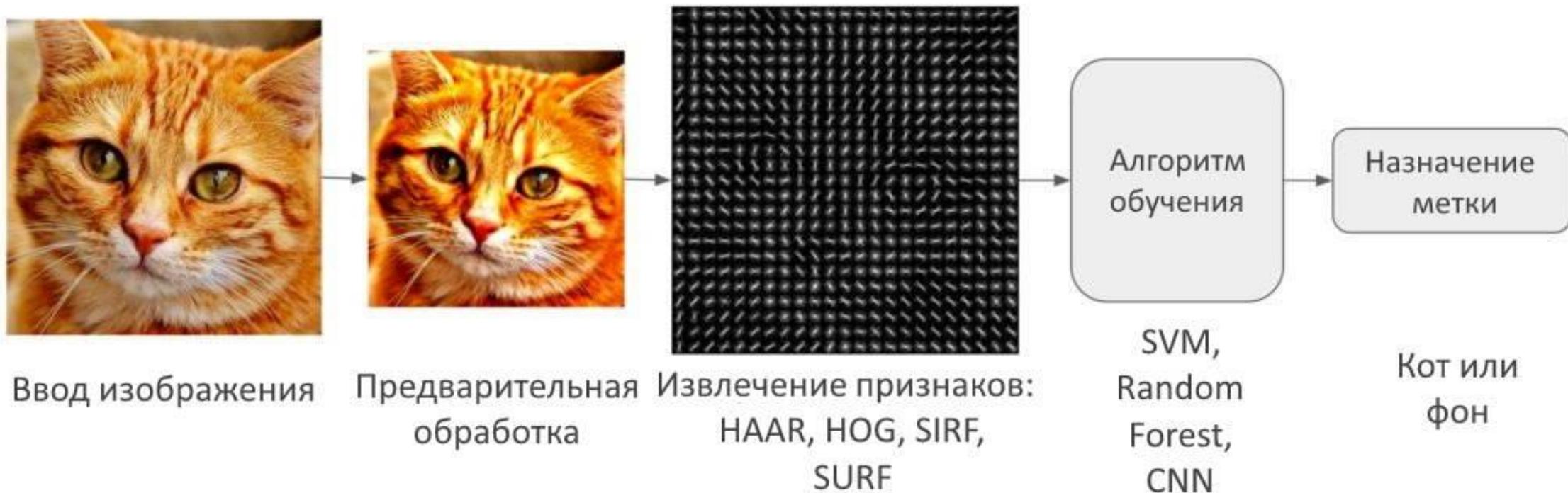
# Гистограмма направленных градиентов (Histogram of Oriented Gradients, HOG)



Данная техника основана на подсчете количества направлений градиента в локальных областях изображения.

В случае дескриптора признака HOG входное изображение имеет размер  $64 \times 128 \times 3$ , а выходной вектор признака имеет длину 3 780.

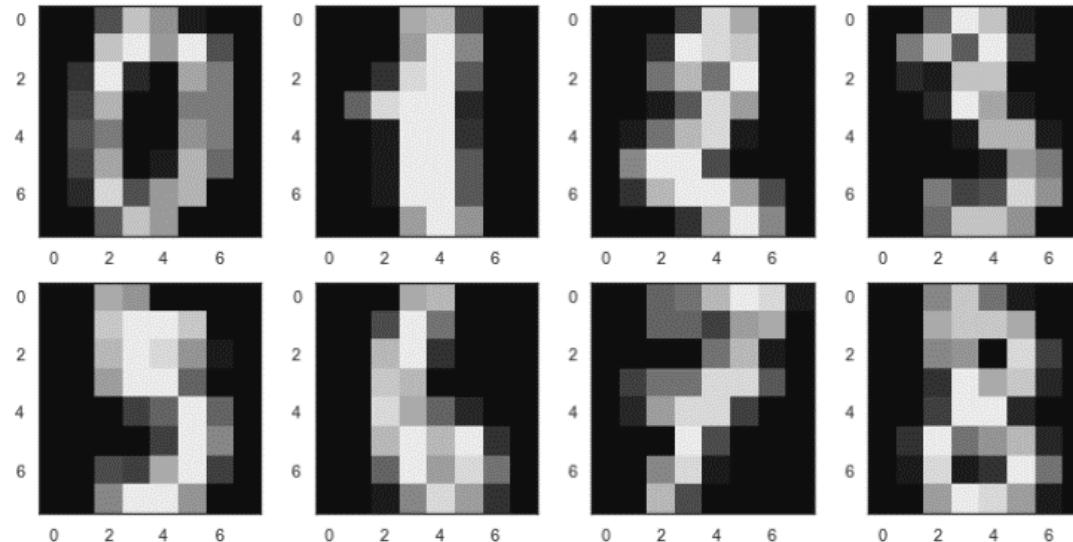
# Этапы работы традиционного классификатора изображений.



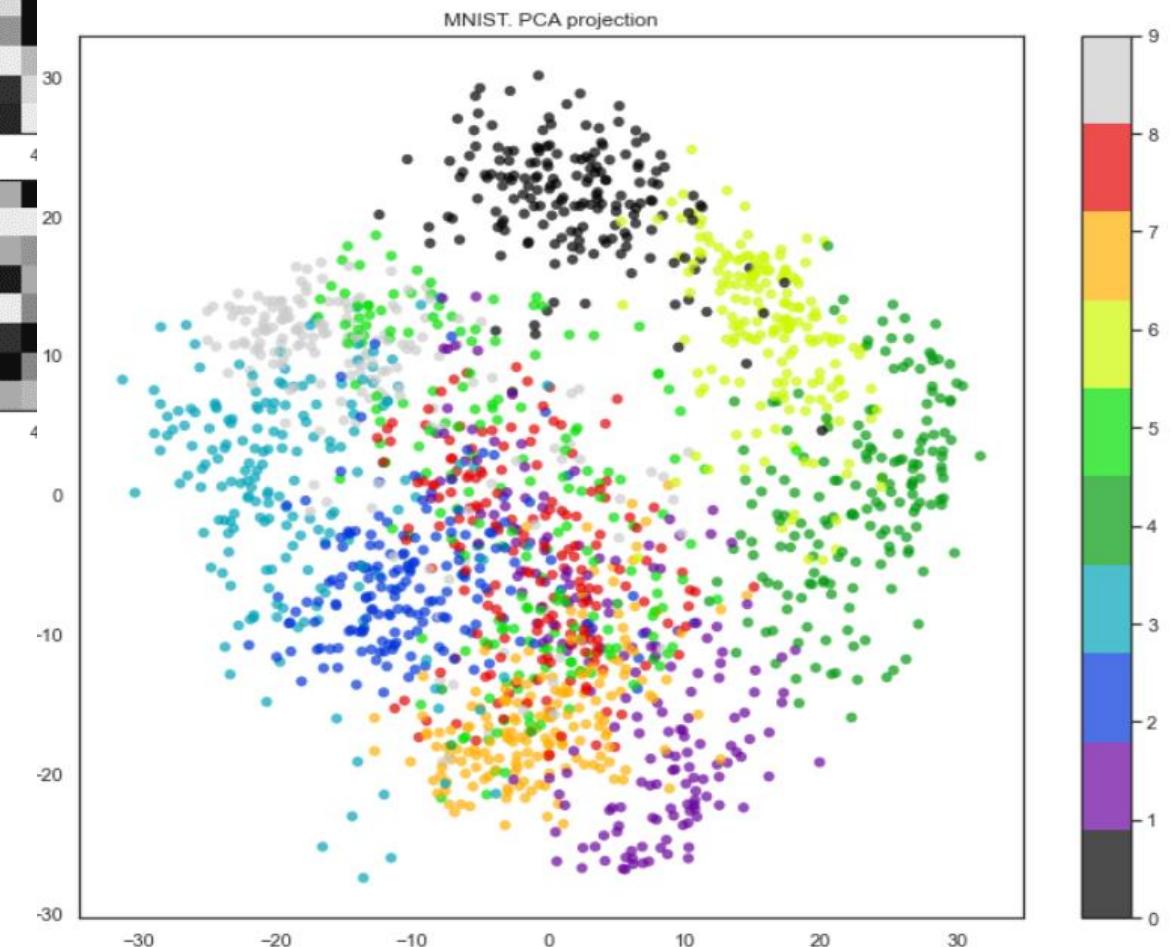
# Понижение размерности

## Метод главных компонент PCA

### Пример с цифрами



Получается, размерность  
признакового пространства  
здесь – 64. Снизим  
размерность всего до 2 при  
помощи метода РСА.



Bill Gates



Arnold Schwarzenegger



Gwyneth Paltrow



Angelina Jolie



Queen Elizabeth II



David Beckham



LeBron James



Michael Jordan



Michael Jackson



Dwayne Johnson



Marilyn Monroe



Azra Akin



Hillary Clinton



Oprah Winfrey



George W Bush



Daniel Radcliffe

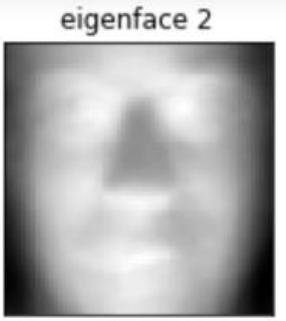


## Работа с изображениями

Есть набор данных из 1000 изображений размером 64x64 пикселя. Размер нашей выборки (1000, 4096)

Визуализируем 16 первых:

<https://towardsdatascience.com/eigenfaces-recovering-humans-from-ghosts-17606c328184>



## Визуализация компонент

Мы не сможем понять весь содержательный смысл этих компонент, мы можем догадаться, какие характеристики изображений лиц были выделены некоторыми компонентами. Похоже, что первая компонента главным образом кодирует контраст между лицом и фоном, а вторая компонента кодирует различия в освещенности между правой и левой половинами лица и т.д. Такое представление данных весьма далеко от того, как человек привык воспринимать лицо. Важно помнить, что, как правило, алгоритмы в отличие от человека интерпретируют визуальные данные совершенно по-другому.



Рис. 3.11 Реконструкция трех изображений лица с помощью постепенного увеличения числа главных компонент

Здесь визуализируем результаты реконструкции некоторых лиц, используя 10, 50, 100 и 500 компонент.

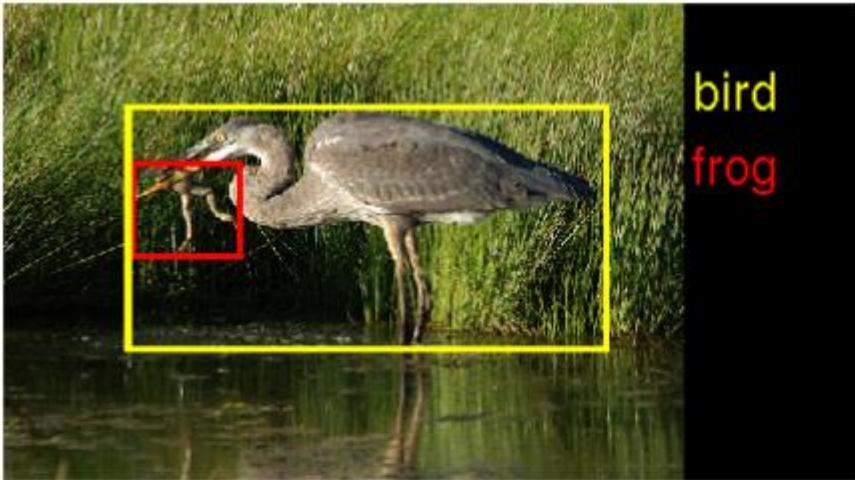
Таким образом можно сказать, что из первоначальных **4096** признаков изображения достаточно описать **500** признаками. Уменьшили размерность в 8! раз.

# Классическое Обучение

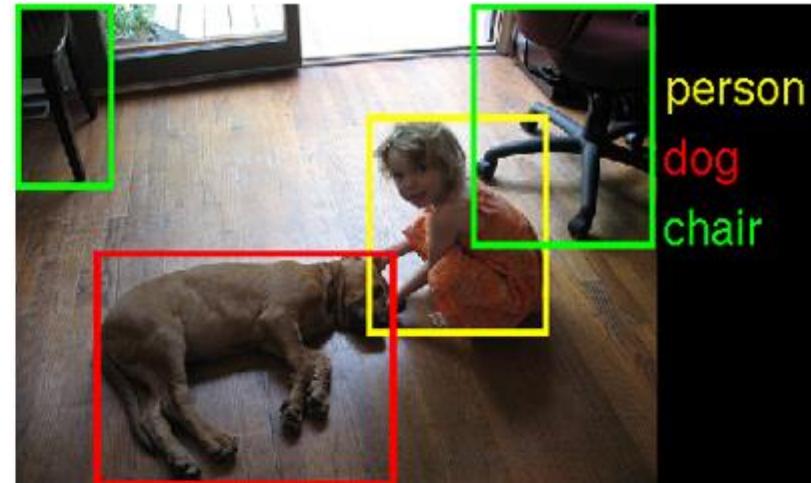


# Нейронные сети

---



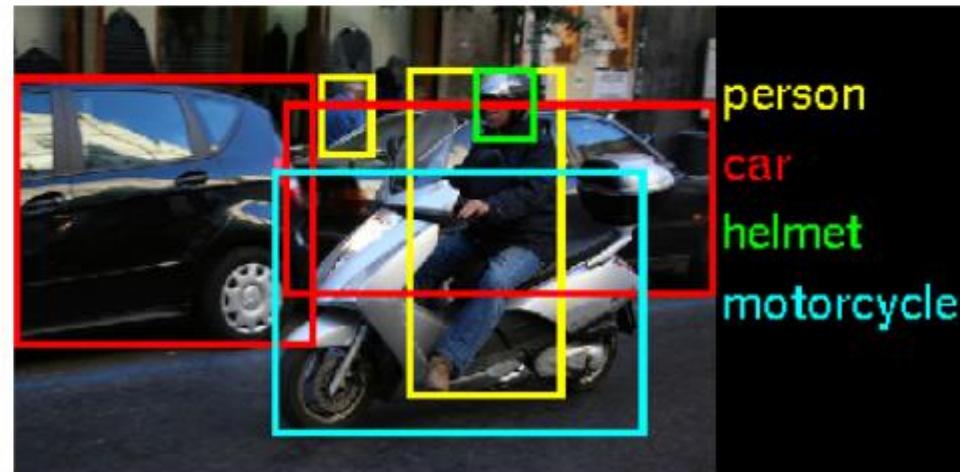
bird  
frog



person  
dog  
chair



person  
hammer  
flower pot  
power drill



person  
car  
helmet  
motorcycle

Большинство решаемых задач по распознаванию образов в компьютерном зрении сводится к следующим 4 задачам :

**Классификация.** Нейронные сети можно обучить, например, идентифицировать собак или кошек с высокой степенью точности при наличии таких изображений.

**Локализация** необходима для определения местоположения объектов. Обычно найденные объекты ограничиваются в прямоугольники.

**Детектирование**, когда алгоритмы Machine Learning не только находят объект на изображении, но и проводят классификацию.

**Сегментация** — это отнесение пикселей к определенной категории объекта. В отличие от детектирования, подсвечиваются лишь те пиксели, которые принадлежат объекту.

**Классификация**

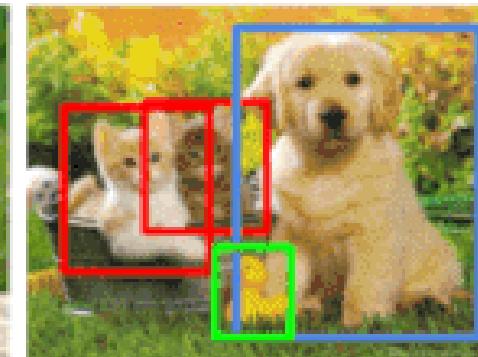


**Локализация**



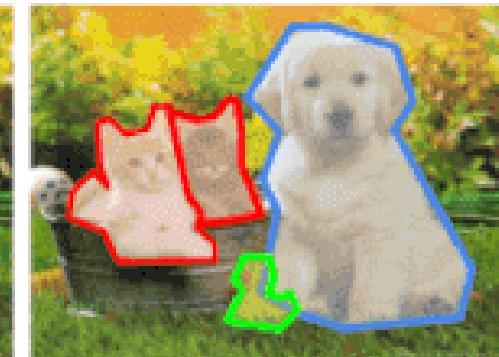
CAT

**Детектирование**



CAT, DOG, DUCK

**Сегментация**



CAT, DOG, DUCK

Свёрточная нейронная сеть, СНС  
convolutional neural network, CNN

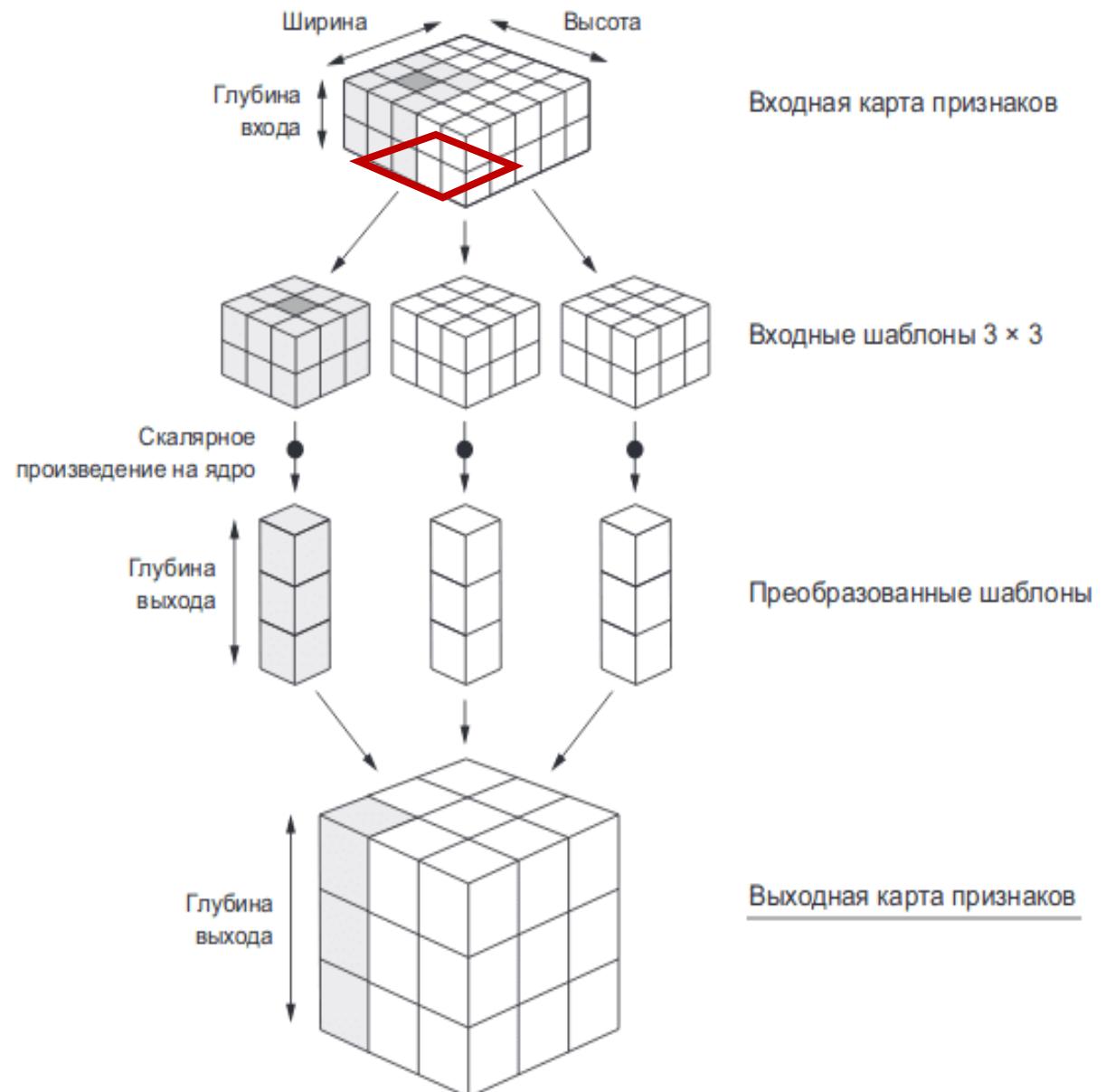
**Свёрточная нейронная сеть** (англ.convolutional neural network, CNN) — специальная архитектура искусственных нейронных сетей, предложенная Яном Лекуном в 1988 году и нацеленная на эффективное распознавание образов. Использует некоторые особенности зрительной коры, в которой были открыты так называемые простые клетки, реагирующие на прямые линии под разными углами, и сложные клетки, реакция которых связана с активацией определённого набора простых клеток.

**Сверточная нейронная сеть (СНС)** является одной из основных категорий для распознавания и классификации изображений. Своё название она получила из за операции свёртки, которая является основой всей сети.

**В каком-то смысле СНС — это прототип зрительной коры мозга.** Зрительная кора имеет небольшие участки клеток, которые чувствительны к конкретным областям поля зрения. Эту идею детально рассмотрели с помощью потрясающего эксперимента Хьюбел и Визель в 1962г, в котором показали, что отдельные мозговые нервные клетки активировались только при визуальном восприятии границ определенной ориентации. Например, некоторые нейроны активировались, когда воспринимали вертикальные границы, а некоторые — горизонтальные или диагональные. Хьюбел и Визель выяснили, что все эти нейроны сосредоточены в виде стержневой архитектуры и вместе формируют визуальное восприятие. Эту идею специализированных компонентов внутри системы, которые решают конкретные задачи (как клетки зрительной коры, которые ищут специфические характеристики) и используют машины, и эта идея — основа СНС.

## Операцию свёртки можно представить следующим алгоритмом:

- Скользящее окно, называемое фильтром, с размером  $(n,n)$  двигается по входному признаку. Количество таких операций определяется заданным количеством фильтров.
- Каждый полученный шаблон имеет форму  $(n,n,d)$ , где  $d$  — глубина входного признака.
- Каждый шаблон умножается на своё ядро свёртки, в результате, формируется выходная карта признаков. Полученная выходная карта признаков имеет форму  $(h,w,N)$ , где  $h$  и  $w$  — длина и ширина, полученные в результате отсечения, а  $N$  — количество фильтров.
- Количество фильтров — гиперпараметр**, поэтому выбирается самостоятельно. Обычно его подбирают как степень двойки с увеличением количества фильтров по мере увеличения глубины архитектуры. А **ядра свёртки являются обучаемыми параметрами**.



Рассмотрим процесс свёртки на примере изображения в оттенках серого с размером  $(28,28)$ . Глубина изображения в оттенках серого равна 1, если бы это было RGB, то глубина входа равнялась бы 3. Пусть размер фильтра равняется  $(3,3)$ , а всего их 32.

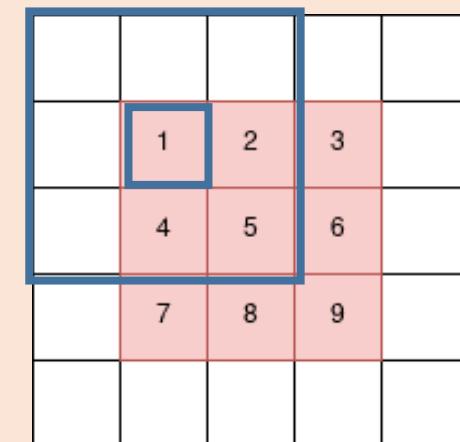
- На первом этапе сформируются 32 шаблона размером  $(3,3,1)$ , где 1 — глубина изображения.
- Полученные шаблоны умножаются на ядра свёртки. Каждый преобразованный в результате умножения шаблон формирует вектор с длиной равной количеству фильтров, т.е. 32.
- Все преобразованные шаблоны объединяются в выходную карту признаков. Она имеет размер  $(26,26,32)$

### Почему уменьшается размерность после операции свёртки

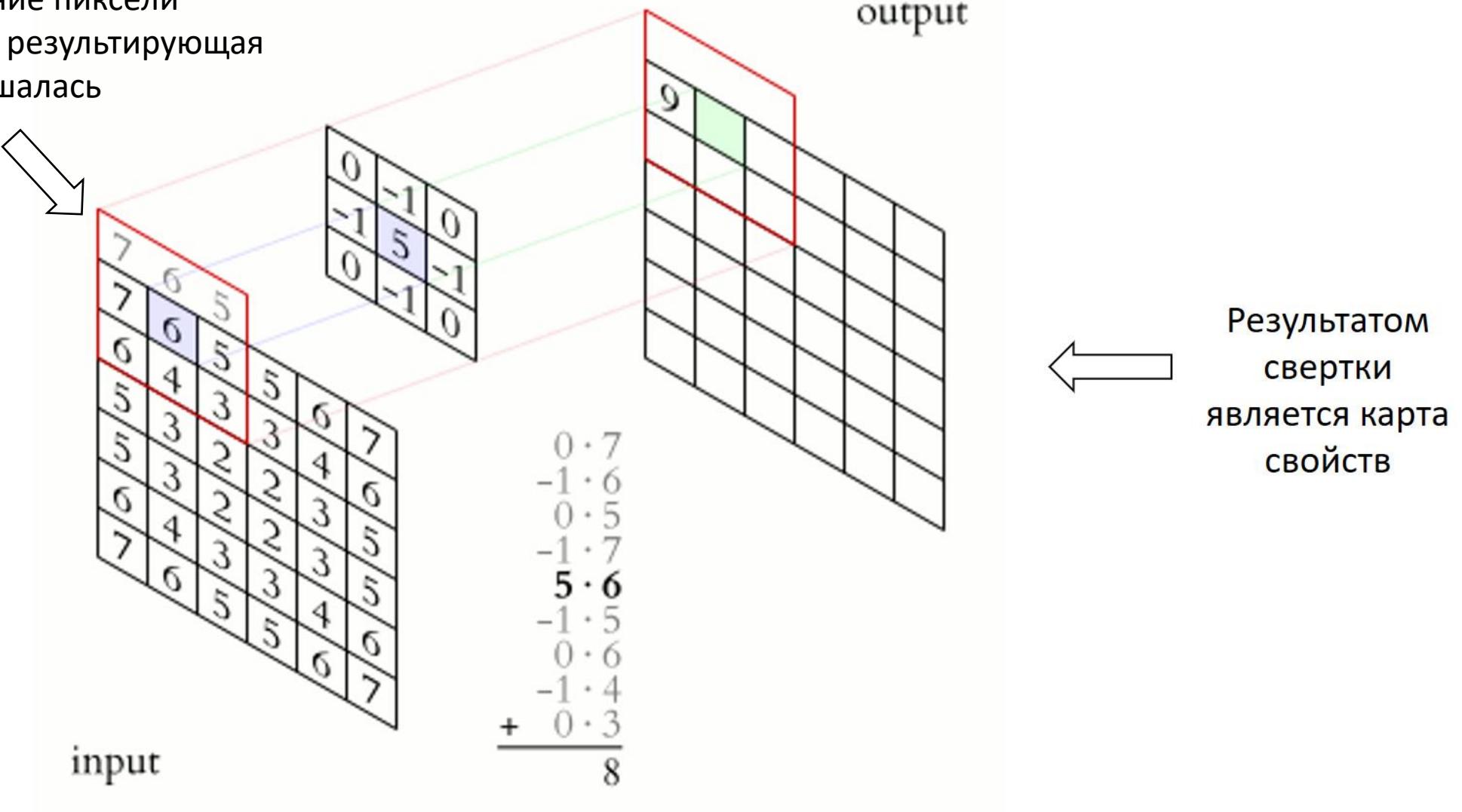
В рассмотренном выше примере выходная карта признаков имеет размерность  $(26,26,32)$ , в то время как исходное изображение имело размерность  $(28,28,1)$ . Если 32 — количество фильтров, а 1 — глубина входа, тогда почему исходный размер 28 уменьшился до 26?

Рассмотрим матрицу  $(5,5)$  и фильтр  $(3,3)$ . Дело в том, что центр скользящего окна может встать только в 9 клеток матрицы  $(5,5)$ , как это показано на рисунке ниже. Следовательно, после умножения на ядра свёртки сформируется выходная карта признаков с высотой и шириной  $(3,3)$ .

Такого обрезания можно избежать путём эффекта дополнения (padding). Он заключается в добавлении строк и столбцов.



В этом примере использован метод  
дополнения, крайние пиксели  
дополняют, что бы результирующая  
матрица не уменьшалась



# Слой Pooling

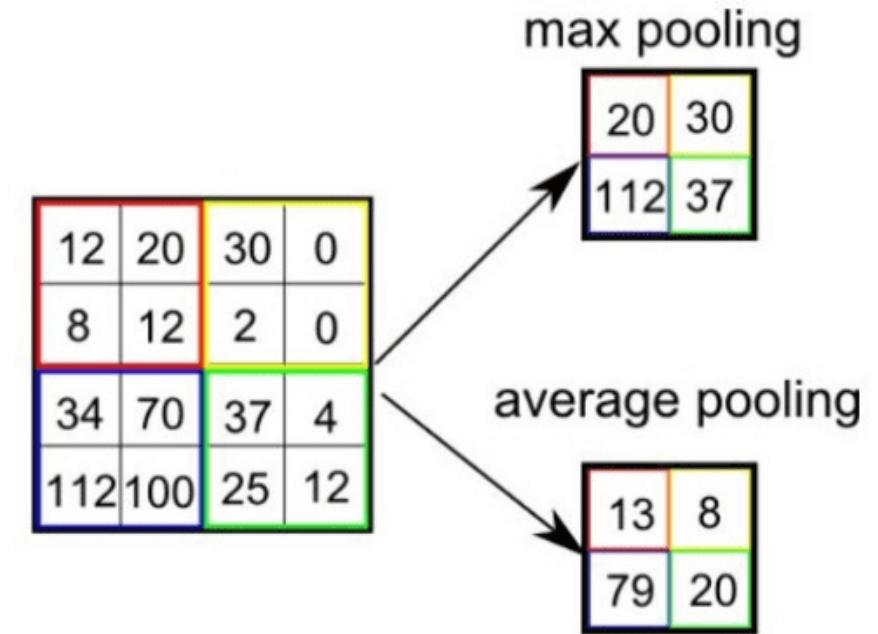
В сверточных нейронных сетях применяется ещё один слой, называемый слоем Pooling. Суть этого слоя заключается в уменьшении размерности карты признаков.

Pooling имеет две разновидности: **max-pooling** и **average-pooling**. Чаще применяется max-pooling.

Операция Pooling схожа с операцией свертки:

- Скользящее окно, обычно это окно (2,2), двигается по карте признаков.
- Из выбранного шаблона выбирается максимальное (max-pooling) или среднее (average-pooling) значение.
- Формируется уменьшенная в размере карта признаков.

На рисунке показано, как из матрицы (4,4) получается выходная карта (2,2) после операции max-pooling и average-pooling.



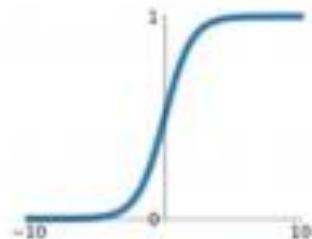
## Зачем нужно уменьшать размерность с помощью Pooling?

- Для поддержания иерархичности. Архитектура сверточных нейронных сетей похожа на воронку, где все начинается с большой картины с последующим углублением в отдельные детали. Человеческий мозг устроен также: сначала он видит на улице кошку, а затем начинает разглядывать ее цвет, пятна, уши, глаза и т.д. Это является основой Deep learning — обучение на представлениях.
- Уменьшение размерности приводит к уменьшению количества обучаемых коэффициентов, поэтому это ещё и выигрыш в вычислительных ресурсах.

**Функция активации** — это один из самых мощных инструментов, который влияет на силу, приписываемую нейронным сетям. Отчасти, она определяет, какие нейроны будут активированы и какая информация будет передаваться последующим слоям.

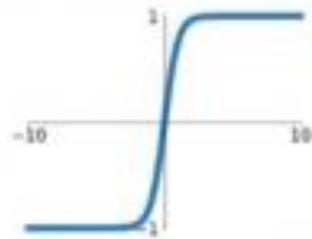
### Sigmoid

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



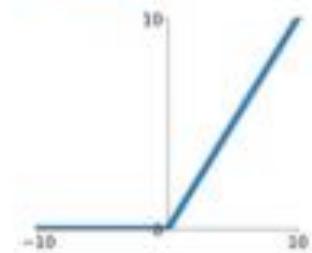
### tanh

$$\tanh(x)$$



### ReLU

$$\max(0, x)$$



### Leaky ReLU

$$\max(0.1x, x)$$

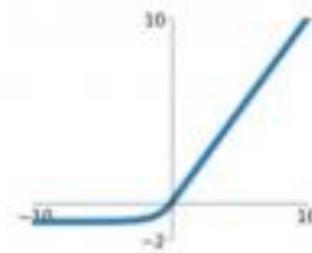


### Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

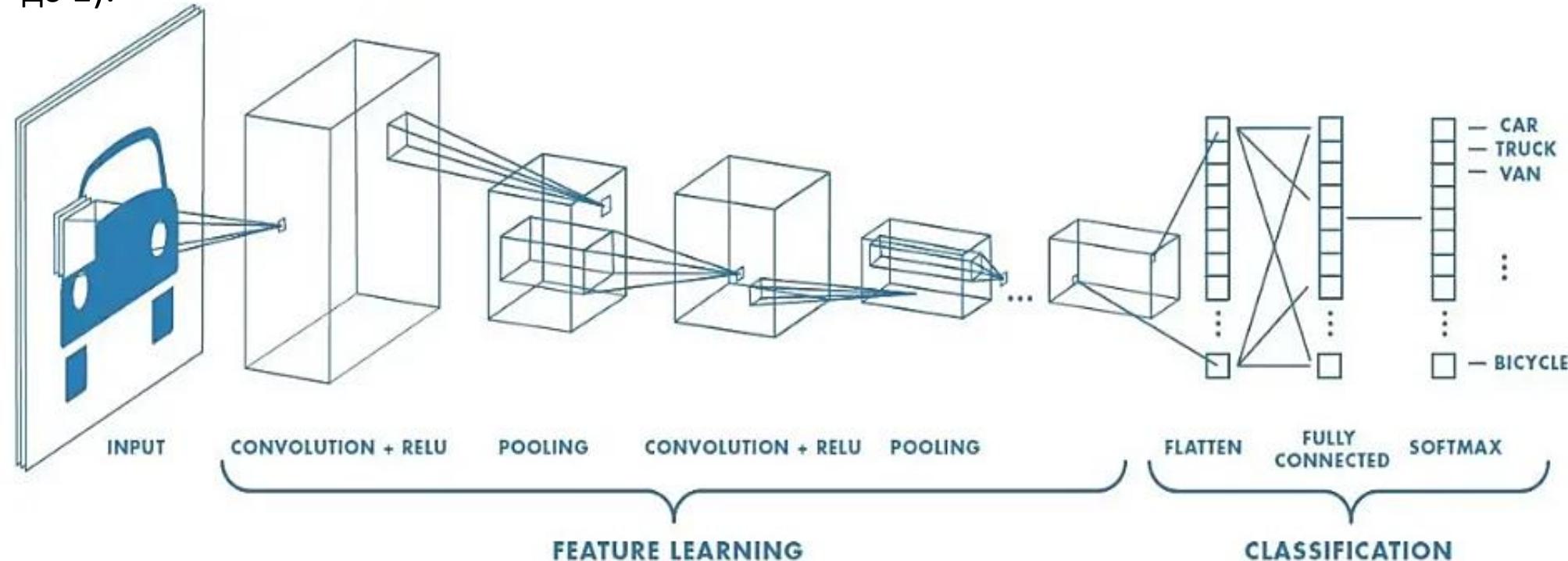
### ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



Простую архитектуру CNN для классификации изображений по k классам можно разделить на две части:

1. цепочка чередующихся слоев свертки/подвыборки (Conv/Pool) (иногда с несколькими слоями свертки подряд)
2. и несколько полно связных слоев (принимающих каждый пиксель как независимое значение) с слоем softmax в качестве завершающего. Функция softmax превращает вектор действительных чисел в вектор вероятностей (неотрицательные действительные числа, от 0 до 1).



Нейронная сеть с множеством сверточных слоев.

<https://habr.com/ru/post/309508/>

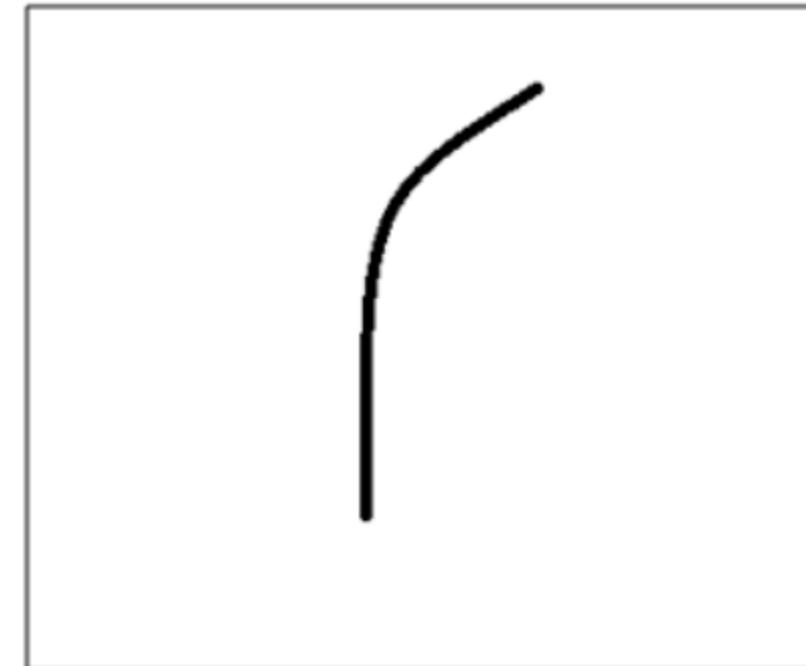
В СНС применяют несколько фильтров. Фильтр можно рассматривать как идентификатор свойства. Под свойством подразумеваются прямые границы, простые цвета и кривые .

Рассмотрим упрощенный пример.

Например, фильтр  $7 \times 7 \times 3$ , и он будет детектором кривых. У фильтра пиксельная структура, в которой численные значения выше вдоль области, определяющей форму кривой.

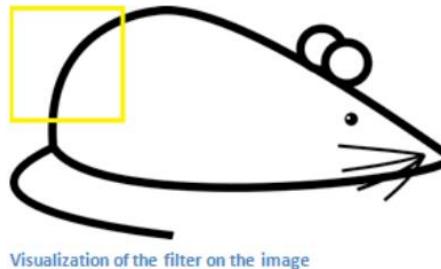
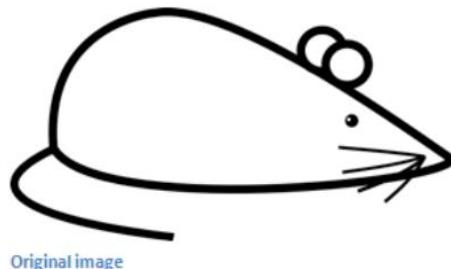
0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter



Visualization of a curve detector filter

Применим фильтр к верхней левой части изображения:



Если на изображении есть форма, в общих чертах похожая на кривую, которую представляет этот фильтр, и все умноженные значения суммируются, то результатом будет большое значение!



Visualization of the receptive field

0	0	0	0	0	0	0	30
0	0	0	0	50	50	50	50
0	0	0	20	50	0	0	0
0	0	0	50	50	0	0	0
0	0	0	50	50	0	0	0
0	0	0	50	50	0	0	0
0	0	0	50	50	0	0	0

Pixel representation of the receptive field

\*

0	0	0	0	0	0	30	0
0	0	0	0	0	30	0	0
0	0	0	30	0	0	0	0
0	0	0	30	0	0	0	0
0	0	0	30	0	0	0	0
0	0	0	30	0	0	0	0
0	0	0	30	0	0	0	0
0	0	0	0	0	0	0	0

Pixel representation of filter

$$\text{Multiplication and Summation} = (50*30)+(50*30)+(50*30)+(20*30)+(50*30) = 6600 \text{ (A large number!)}$$

А если применить фильтр к другой части изображения, где таких кривых нет, то результат будет намного меньше.



Visualization of the filter on the image

0	0	0	0	0	0	0	0
0	40	0	0	0	0	0	0
40	0	40	0	0	0	0	0
40	20	0	0	0	0	0	0
0	50	0	0	0	0	0	0
0	0	50	0	0	0	0	0
25	25	0	50	0	0	0	0

Pixel representation of receptive field

\*

0	0	0	0	0	30	0	0
0	0	0	0	30	0	0	0
0	0	0	30	0	0	0	0
0	0	0	30	0	0	0	0
0	0	0	30	0	0	0	0
0	0	0	30	0	0	0	0
0	0	0	30	0	0	0	0
0	0	0	0	0	0	0	0

Pixel representation of filter

Multiplication and Summation = 0

**Вывод свёрточного слоя — карта свойств.** В нашем простом случае, при наличии одного фильтра свертки (детектора кривой), карта свойств покажет области, в которых больше вероятности наличия кривых. В этом примере в левом верхнем углу значение нашей  $28 \times 28 \times 1$  карты свойств будет 6600. Это высокое значение показывает, что, возможно, что-то похожее на кривую присутствует на изображении, и такая вероятность активировала фильтр. В правом верхнем углу значение у карты свойств будет 0, потому что на картинке не было ничего, что могло активировать фильтр (проще говоря, в этой области не было кривой). И это только для одного фильтра. Это фильтр, который обнаруживает линии с изгибом наружу.

Могут быть другие фильтры для линий, изогнутых внутрь или просто прямых. Чем больше фильтров, тем больше глубина карты свойств, и тем больше информации мы имеем о вводной картинке.

Рассмотренный ранее фильтр упрощён для простоты понимания математики свёртывания.

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



1	0	1
0	1	0
1	0	1

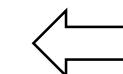
3 x 3 – Filter Matrix

5 x 5 – Image Matrix

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

Image

4		



Результатом  
свертки  
является  
карта свойств

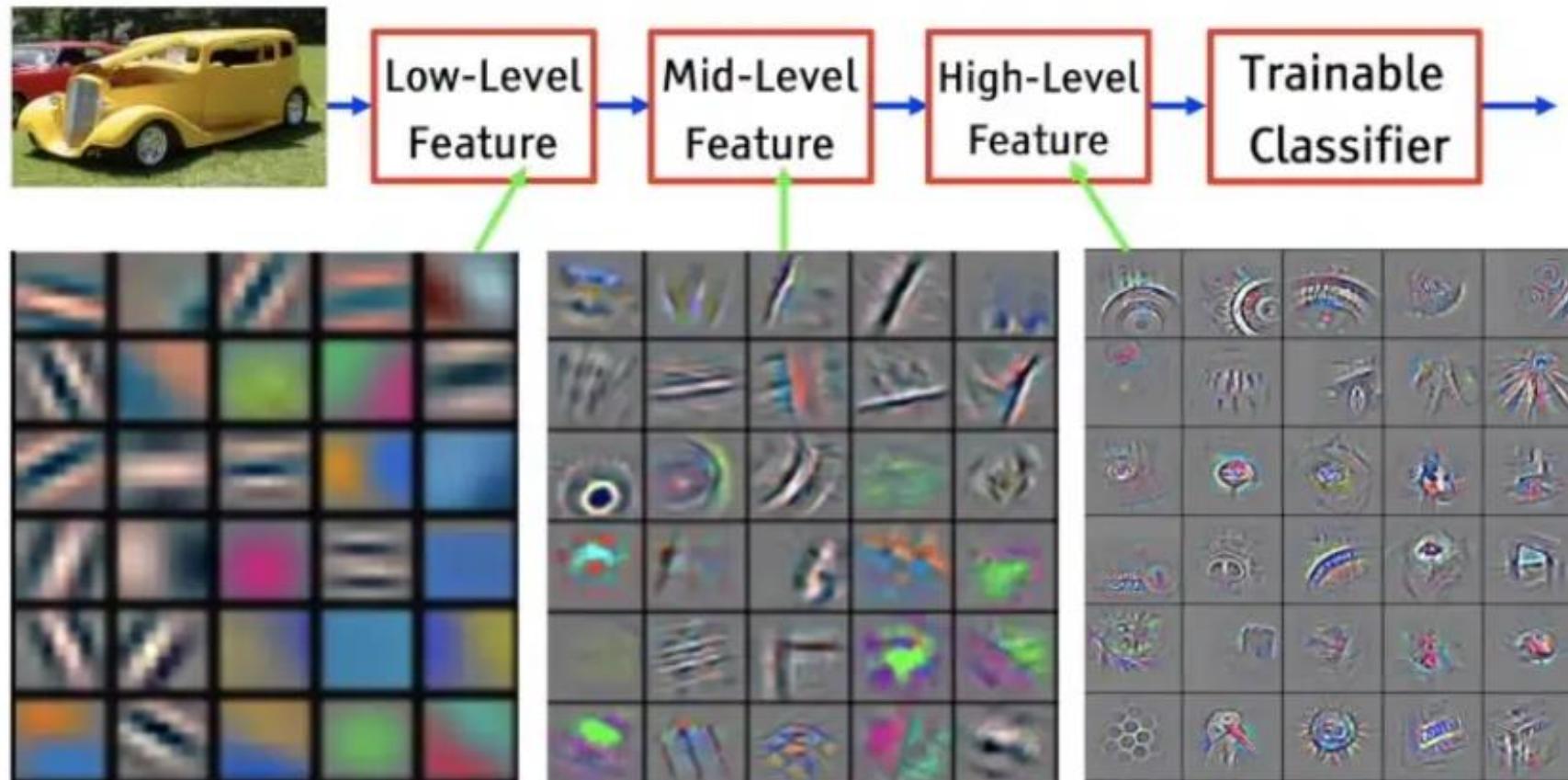
Convolved  
Feature

Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	

Свертка изображения с различными фильтрами может выполнять такие операции, как обнаружение краев, размытие и повышение резкости путем применения фильтров.

<https://medium.com/analytics-vidhya/the-world-through-the-eyes-of-cnn-5a52c034dbef>

Фильтры на первых нескольких начальных слоях **извлекают направление и цвет**, эти примитивные (края и цвет) карты объектов объединяются в **базовые сетки, текстуры и узоры**. Они, в свою очередь, объединяются для извлечения все более **сложных функций**, которые **напоминают части объектов**. По мере продвижения вниз по сети извлеченные признаки становятся более сложными для интерпретации.



На рисунке показаны карты функций на разных этапах сети. Карты признаков в начальных (нижних) слоях кодируют **края и направление**, то есть **горизонтальные или вертикальные линии**, карты признаков, полученные в середине сети, визуализируют **текстуры и узоры**, карты признаков в верхних слоях отображают **части объектов и объекты на изображении**.

## Низкоуровневые/примитивные функции в начальных слоях СНС

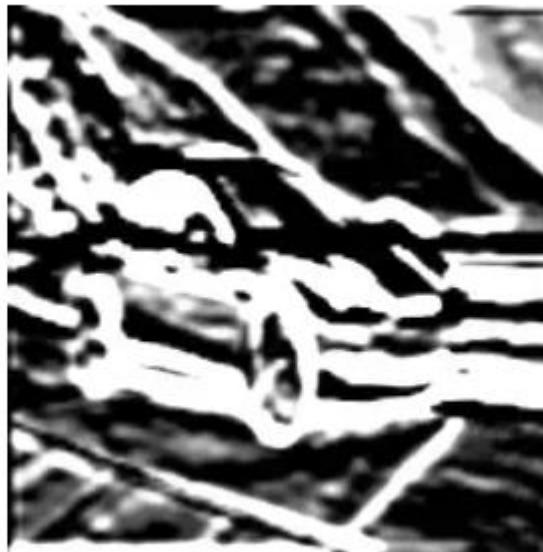
Карты объектов нижнего уровня извлекают края и линии изображений. Это примитивные экстракторы признаков (фильтры), которые улавливают мелкие детали изображения, такие как линии, кривые и точки.

Карты выходных объектов взяты из начальных слоев сети VGG16, обученных на наборе данных ImageNet. Горизонтальные, вертикальные линии и линии под углом 45° помогают изобразить форму кузова автомобиля.



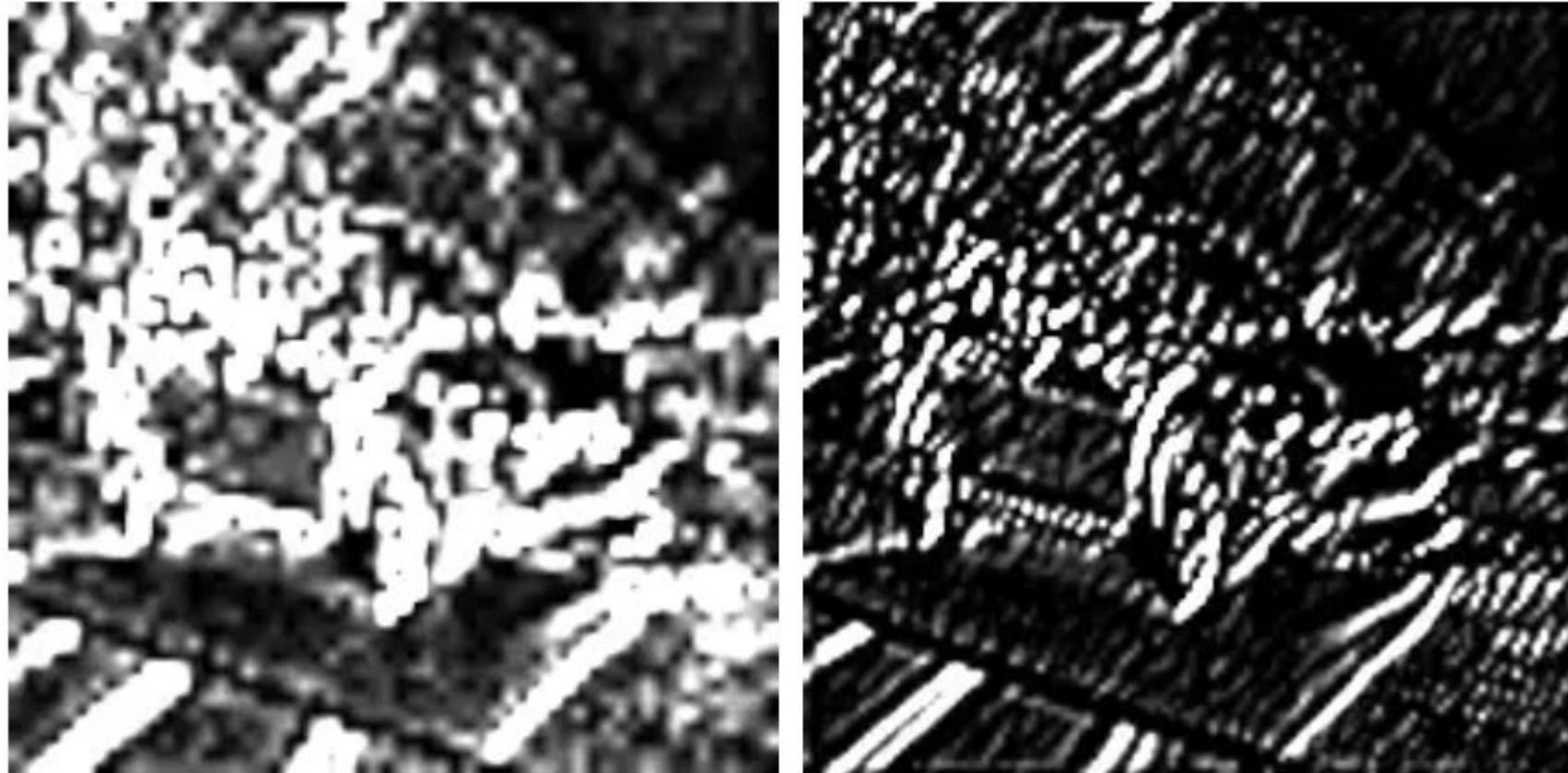
## Особенности среднего уровня СНС

Карты примитивных объектов объединяются для создания простых форм, таких как круги, квадраты, эллипсы и т. д. **Функции среднего уровня** предоставляют больше информации, чем первый, например, фильтры учатся обнаруживать углы, воспроизводить основные узоры, текстуры и формы. Признаки среднего уровня напоминают круглую форму шины, двери, бампера и т. д. Мы можем обобщить, что признаки среднего уровня изображают текстуры, узоры на изображениях.



## Функции более высокого уровня СНС

Карты характеристик автомобиля на более высоких уровнях СНС показаны ниже. Изображения могут быть трудными для понимания, но, присмотревшись к изображению справа, мы заметим такие детали, как окна, шины и т. д.



карта характеристик автомобиля tesla на более высоких уровнях VGG16.

## Функции более высокого уровня СНС

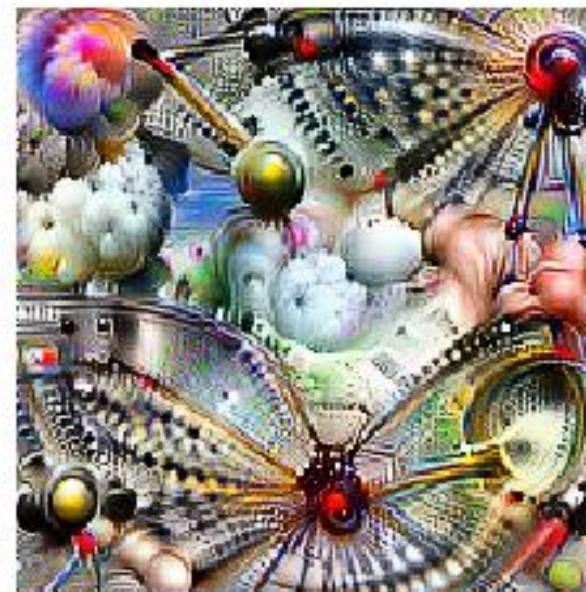
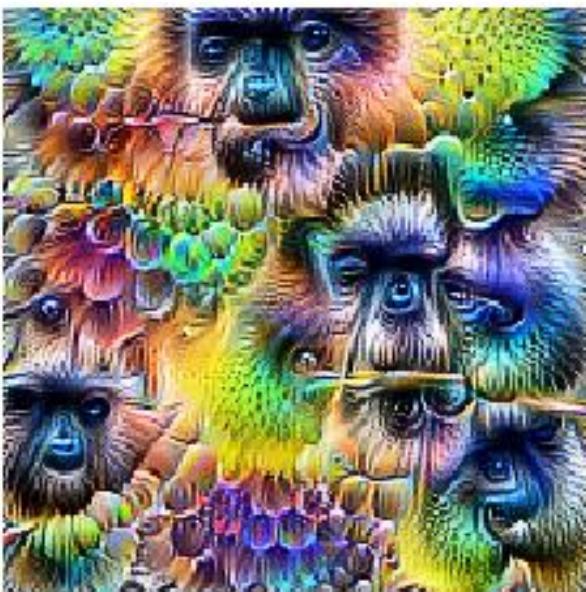
**Функции более высокого уровня изображают сложные шаблоны**, которые напоминают части объектов на изображениях, которые, как правило, являются **уникальной чертой для этого класса** (объекта), что также способствует оценке классификации для этого класса.

По мере того, как мы углубляемся, CNN создает сложные узоры, такие как глаза, нос и т. д. На приведенных картах признаков изображены различные части собаки, ее нос, глаза, мех, висячие уши, хвост и многое другое.



# Как нейронные сети формируют понимание образов

<https://distill.pub/2017/feature-visualization/>



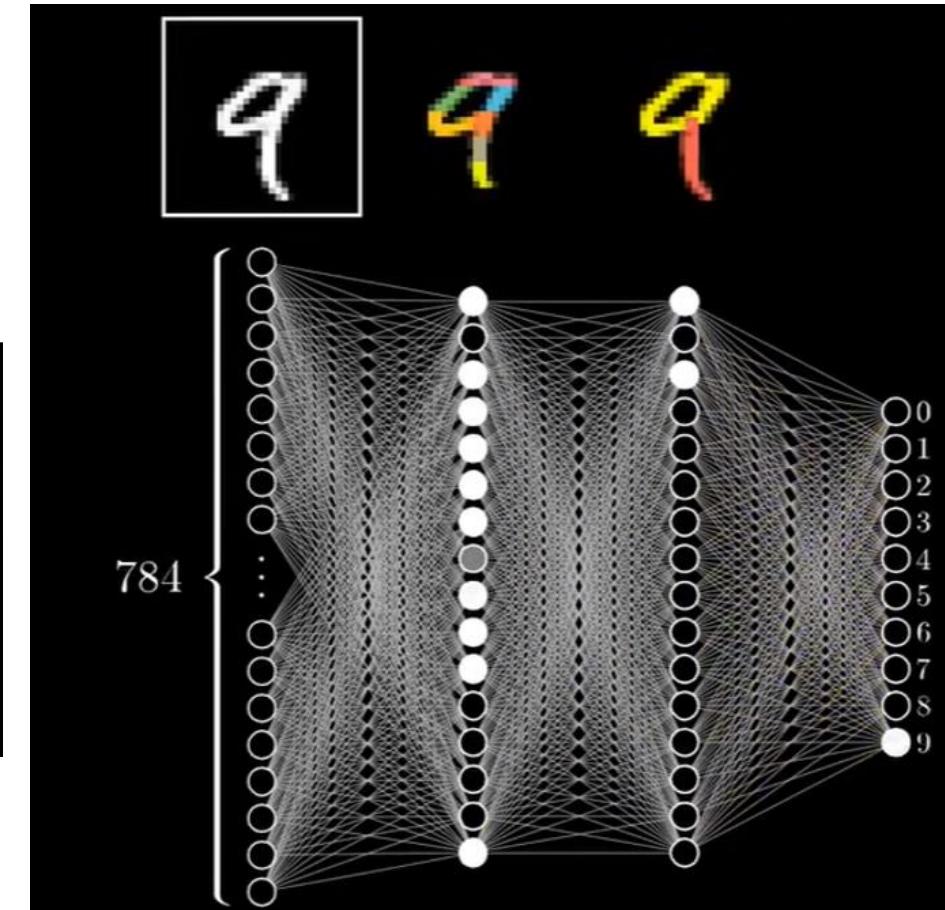
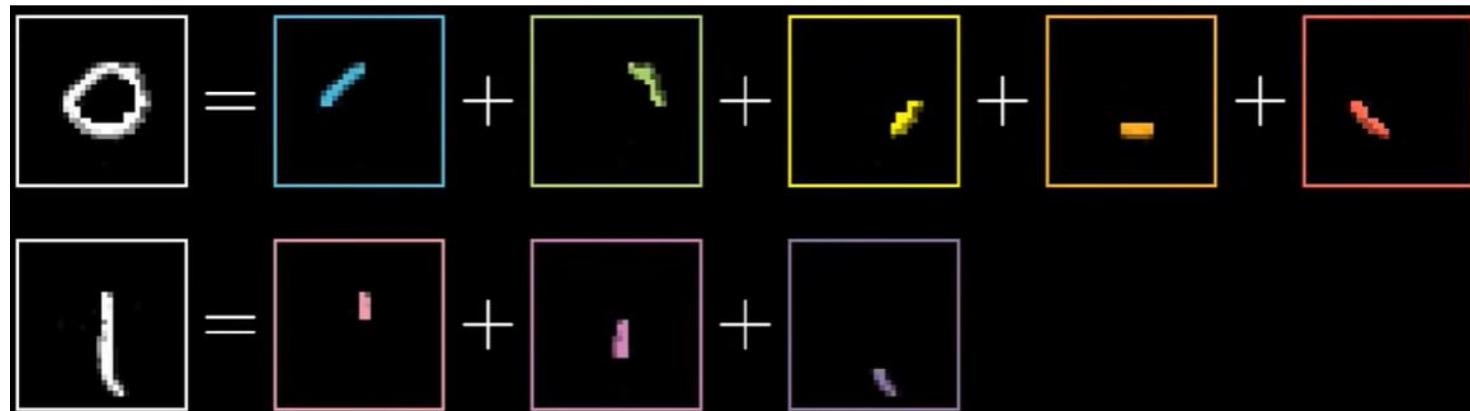
Совместно  
оптимизируя два  
нейрона, мы можем  
понять, как они  
взаимодействуют.



Хорошее и понятное видео о работе сверточной нейронной сети на примере распознавания рукописных цифр.

<https://www.youtube.com/watch?v=RJCIYBAAiEI&list=PLZjXXN70PH5itkSPe6LTS-yPyl5soOovc&index=2>

Советую  
посмотреть!  
Всего 18мин



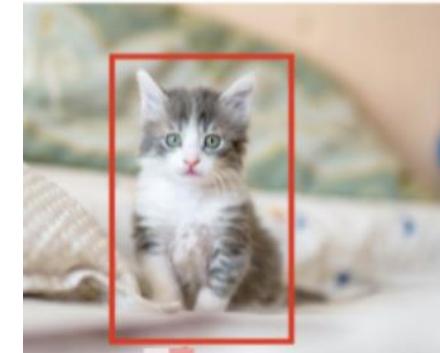
# Меры измерения точности при детектировании объектов на изображении

Детектирование состоит из локализации и классификации. Локализация определяет местоположение экземпляра (например, координаты ограничивающей рамки), а классификация сообщает вам, что это такое (например, собака или кошка).

Основные меры измерения точности детектирования **Precision**, **Recall**, и **F1-Score** связаны с понятиями **IoU**, **map**, **map50**.

Системы детектирования объектов делают прогнозы в ограничивающей рамке. Однако ограничивающая рамка не всегда точна, и для определения ее точности используется мера **IoU** *Intersection over Union*

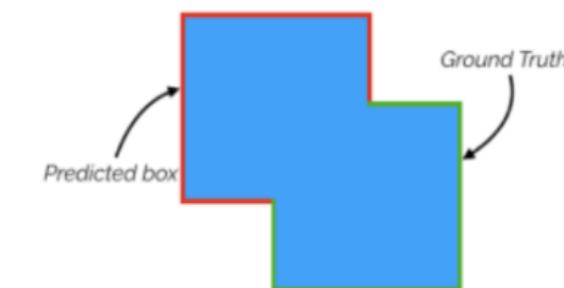
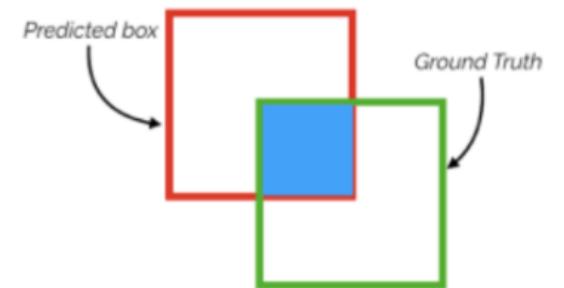
Локализация



Классификация



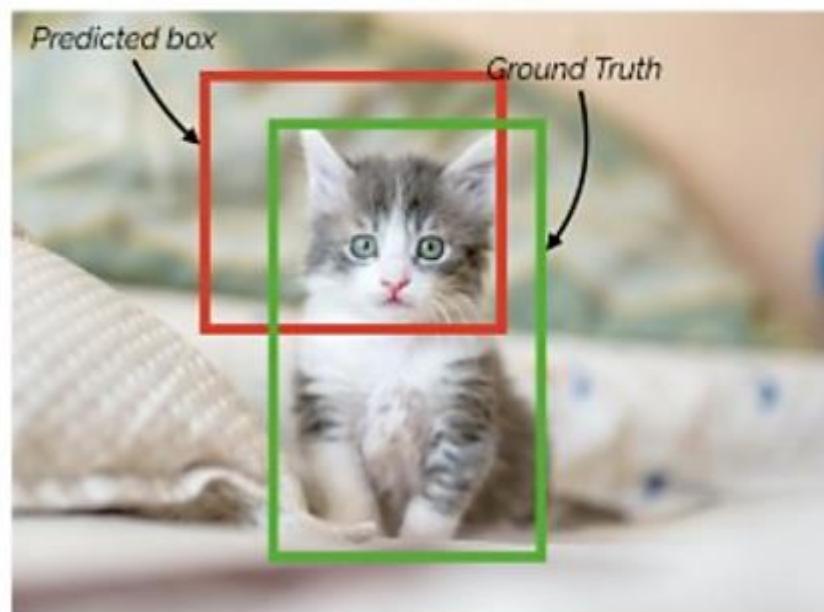
$$\text{IoU} = \frac{\text{Область пересечения}}{\text{Общая область}} =$$



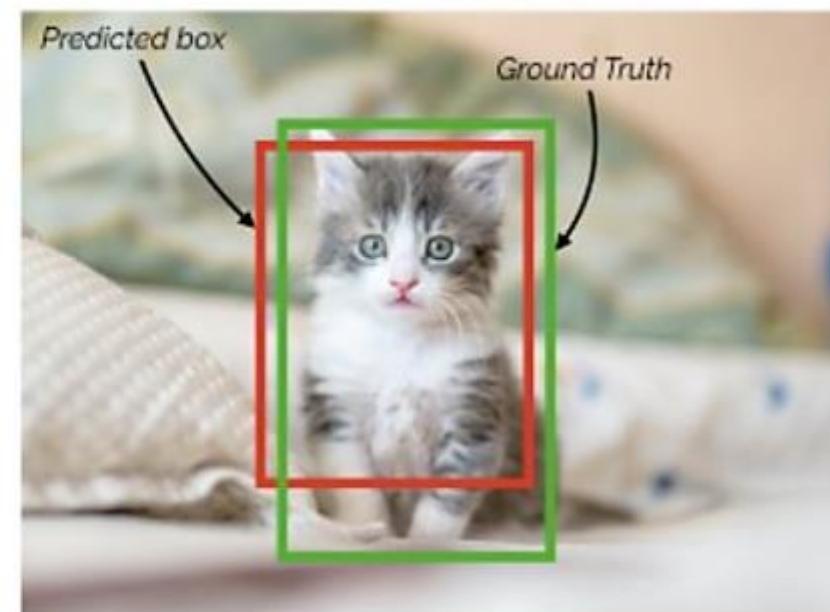
Для задач обнаружения объектов рассчитываем Precision и Recall, используя значение IoU для заданного порогового значения IoU. Например, если порог IoU равен 0,5, а значение IoU для прогноза составляет 0,7, то классифицируем прогноз как истинно положительный (TF). С другой стороны, если IoU равно 0,3, классифицируем его как ложноположительный (FP).

*If IoU threshold = 0.5*

*False Positive (FP)*



*True Positive (TP)*



# Ансамбли в детектировании объектов

<http://recog.ru/ensemble-object-detection/>

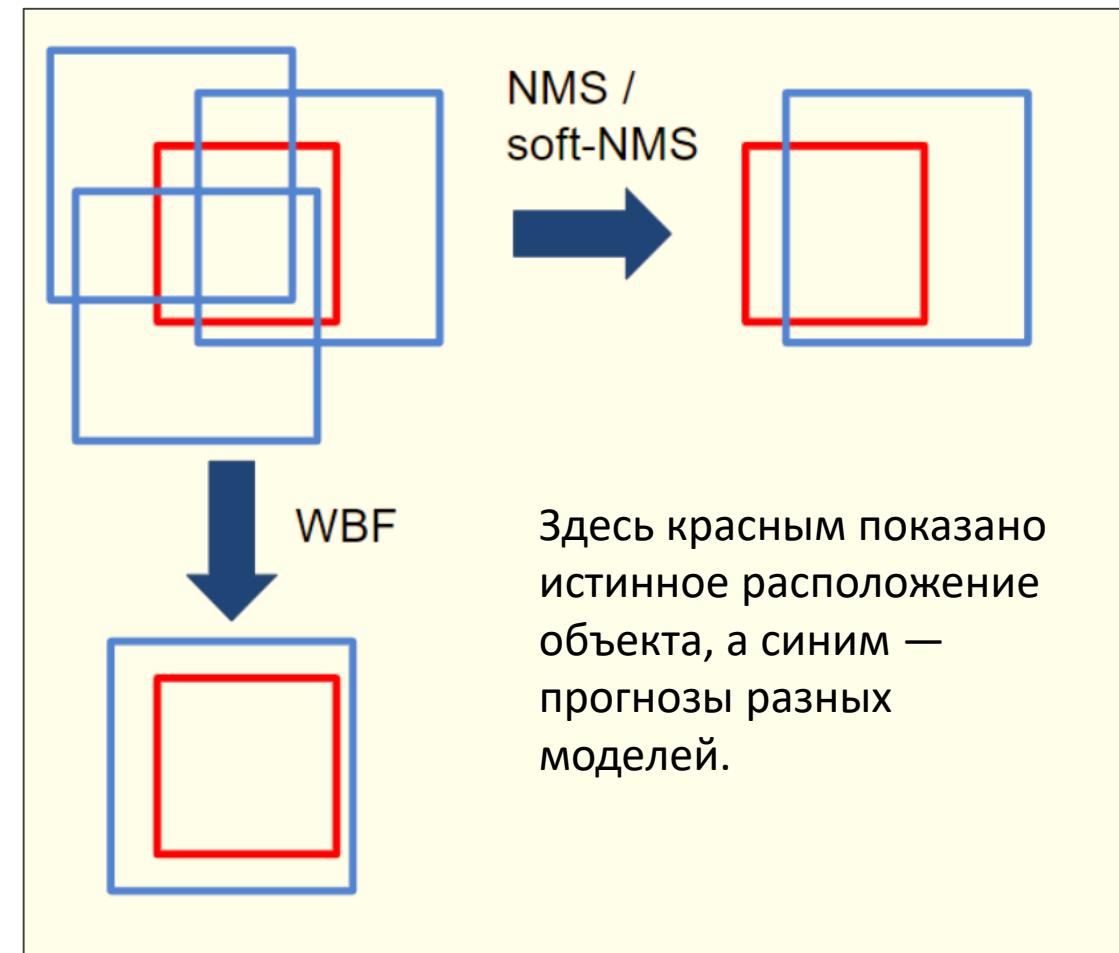
Часто для повышения качества детектирования используются несколько моделей, объединяя их в ансамбли. **Основной проблемой ансамблей является объединение результатов для детектированного одного объекта.**

Существуют несколько известных методов объединения результатов:

- NMS (non-maximum suppression)
- soft-NMS
- NMW (non-maximum weighted)
- WBF (Weighted boxes fusion)

Например, **NMS** сортирует все области по их доверительной вероятности. Затем выбирается область с максимальной оценкой достоверности. В то же время все другие области, которые значительно перекрывают эту область, отфильтровываются, а методы **NMW** и **WBF** работают с IoU более сложным образом

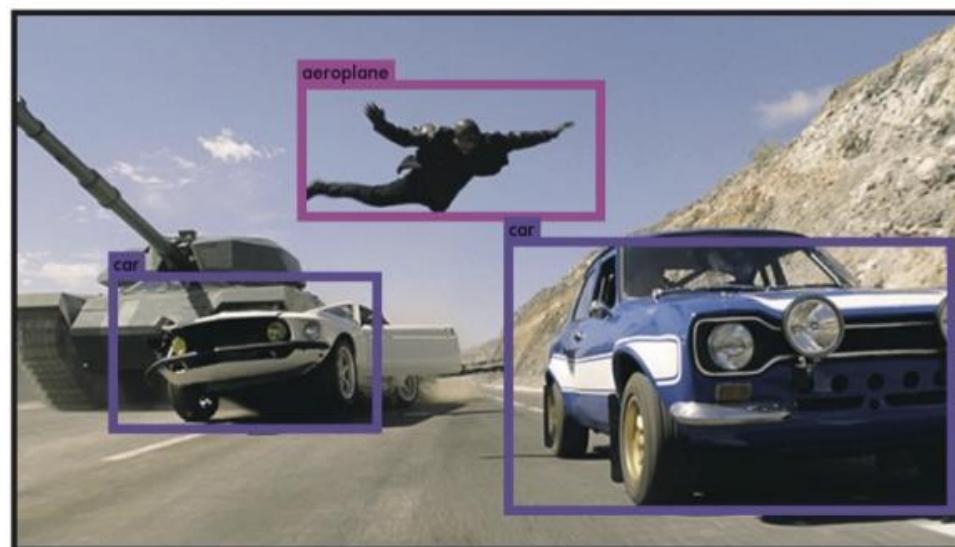
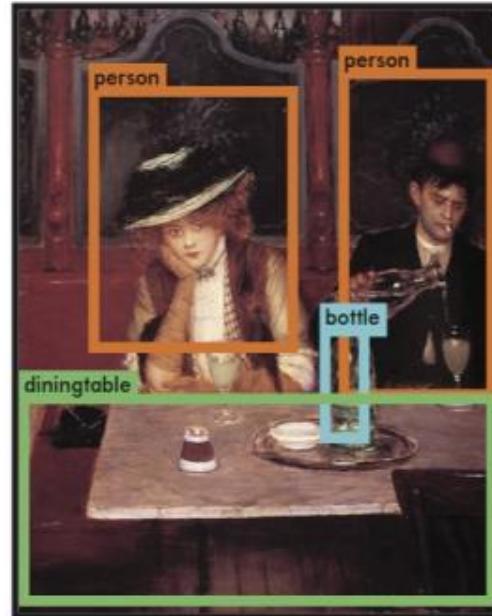
Отличия работы методов объединения результатов:



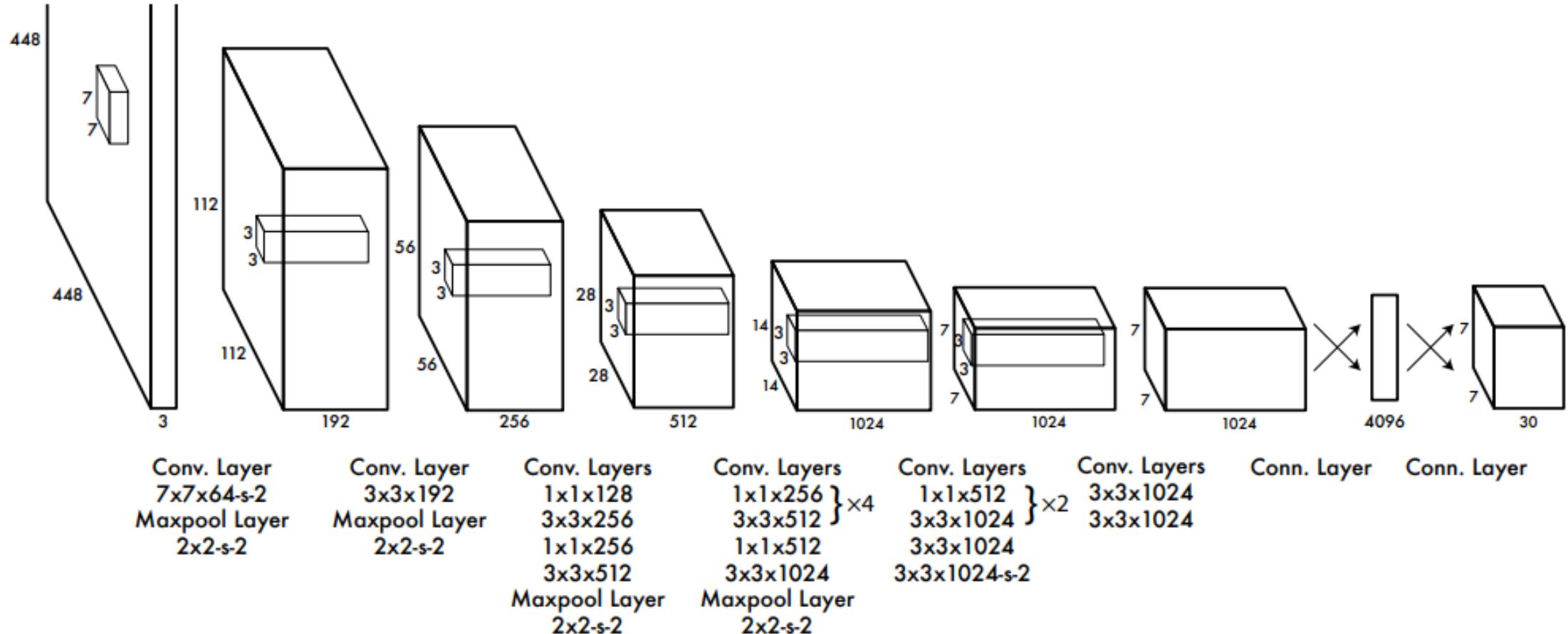
## Примеры нейронных сетей для работы с изображениями

### You Only Look Once (YOLO)

Модель YOLO напрямую предсказывает ограничивающие рамки и вероятности классов с помощью одной сети в одной оценке. Простота модели YOLO позволяет делать прогнозы в реальном времени.

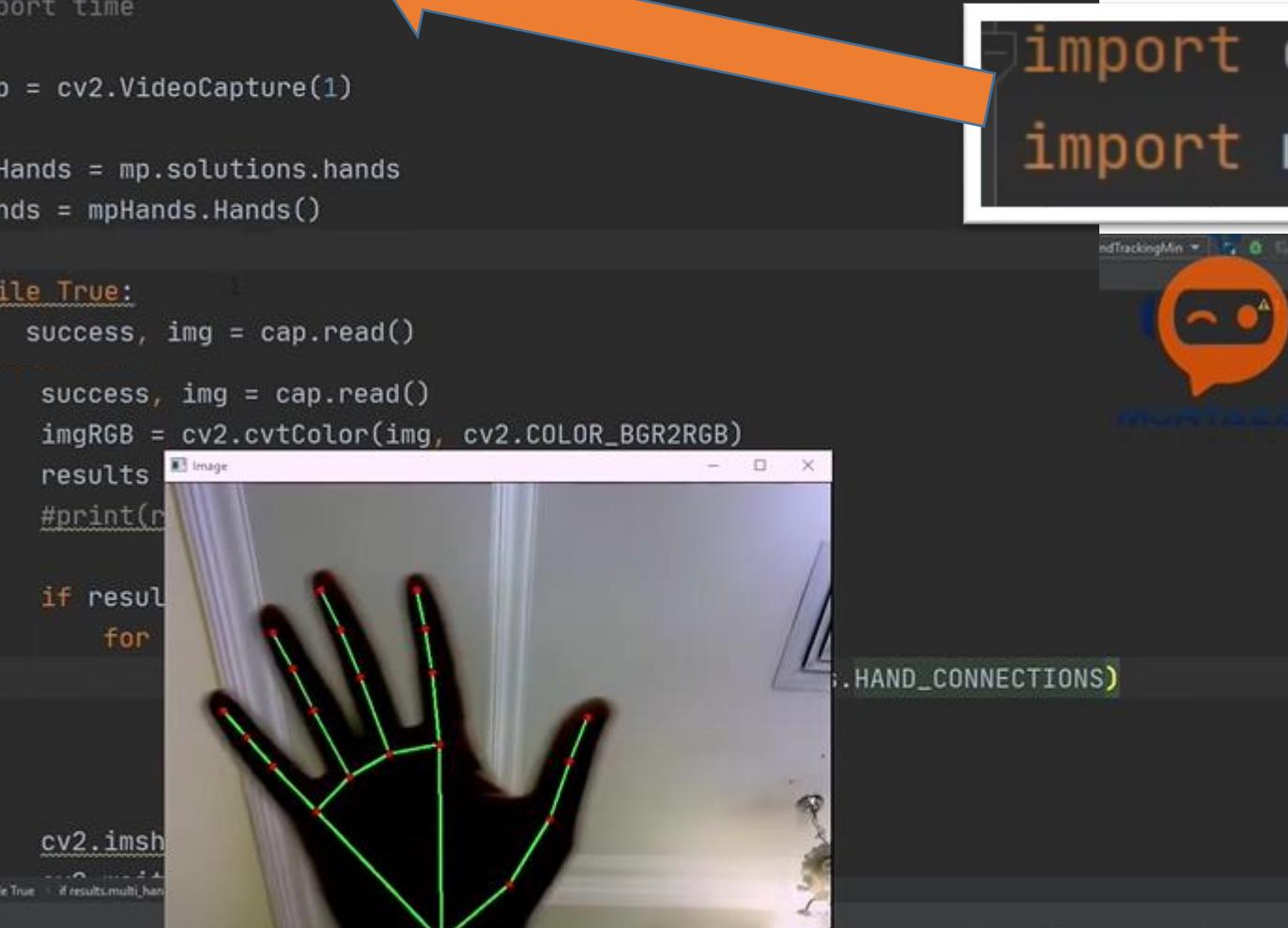


Архитектура YOLO: она состоит из 24 сверточных слоев и 2 полносвязных слоев.



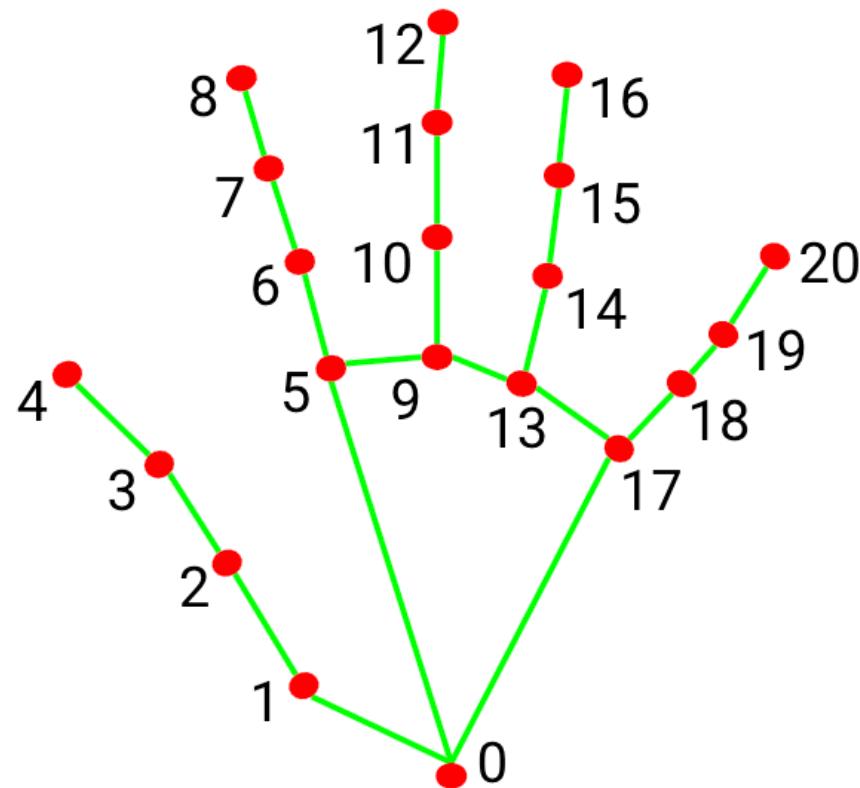


## ML solutions in MediaPipe

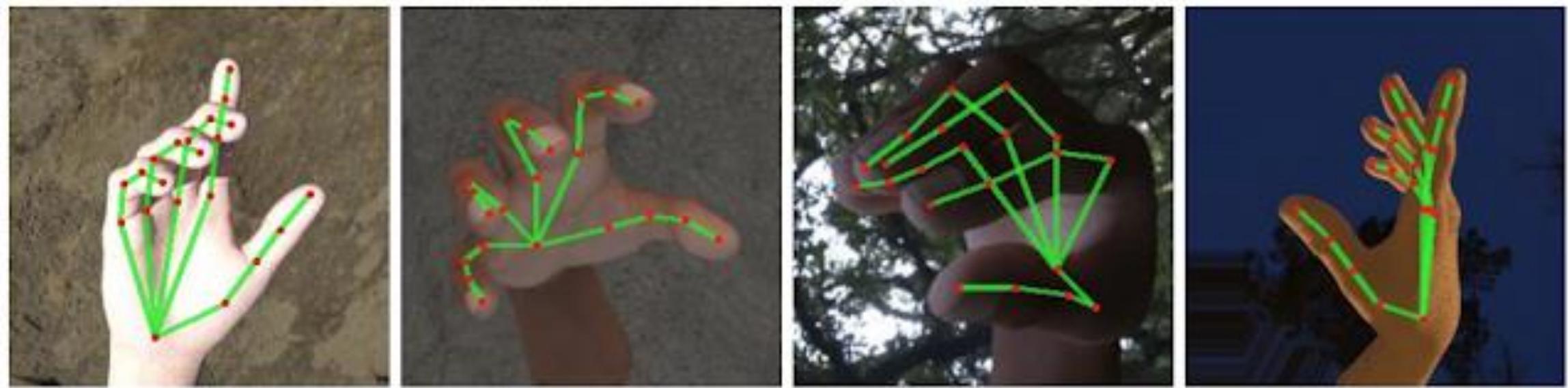
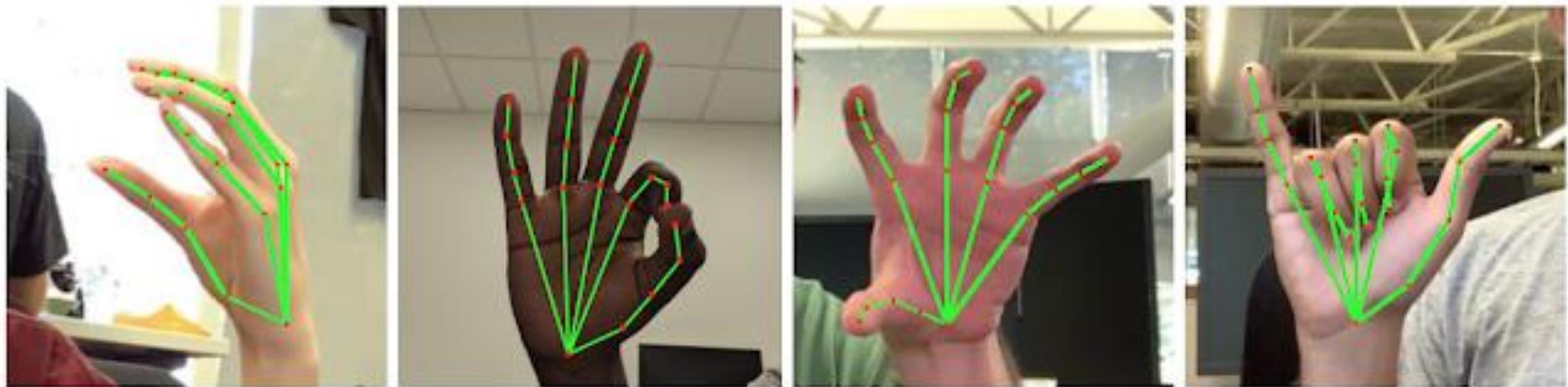


The screenshot shows a Python code editor with a script named `HandTrackingMin.py`. The code imports cv2 and mediapipe, initializes a video capture, and sets up a hands solution. It then enters a loop where it reads frames, processes them with mediapipe, and displays the results in a window titled "Image". The window shows a person's hands with green lines connecting the detected landmarks. An orange arrow points from the code editor towards the top right corner of the screen, which contains a snippet of code and a small orange icon.

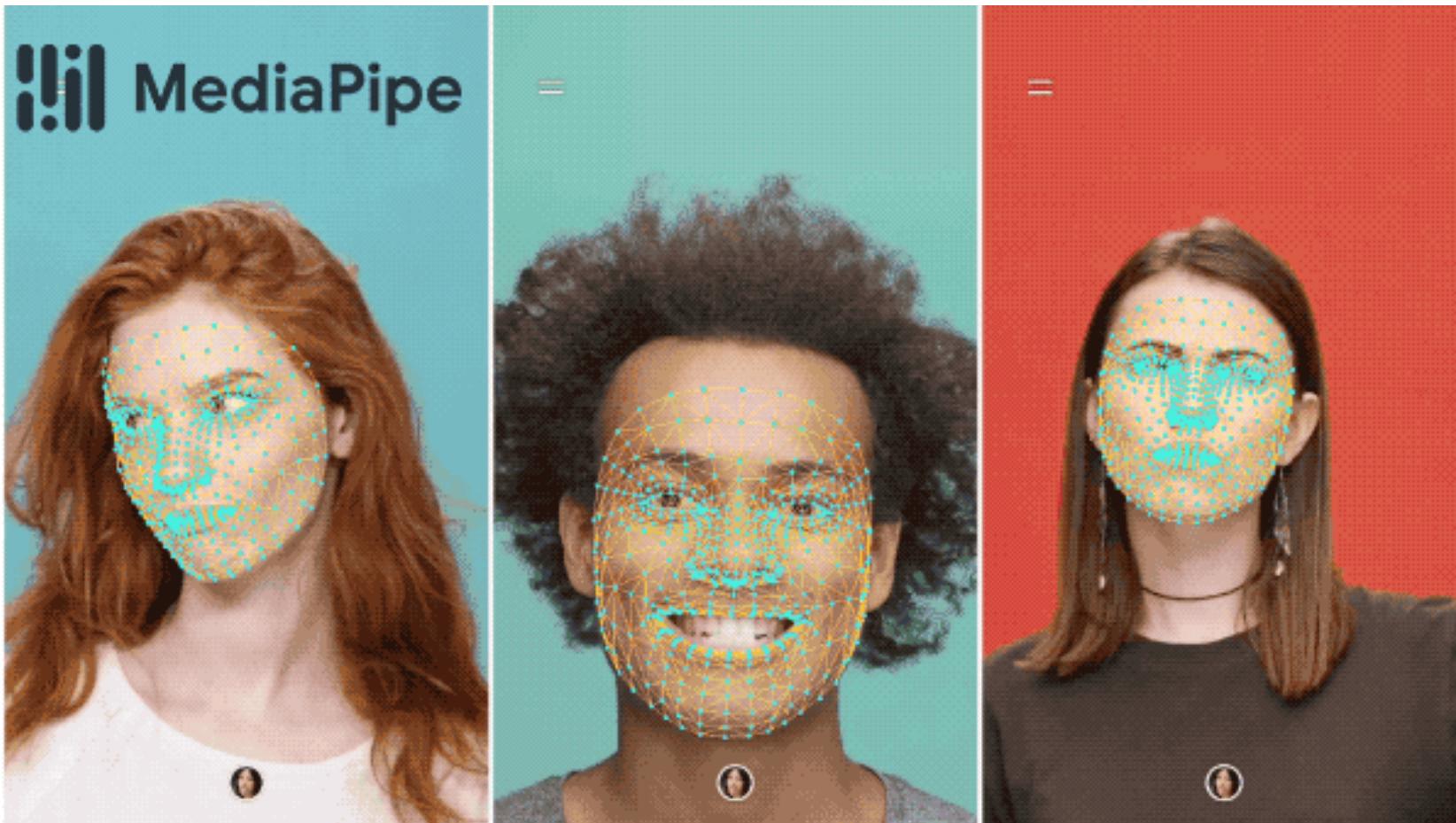
```
1 import cv2
2 import mediapipe as mp
3 import time
4
5 cap = cv2.VideoCapture(1)
6
7 mpHands = mp.solutions.hands
8 hands = mpHands.Hands()
9
10 while True:
11     success, img = cap.read()
12
13     success, img = cap.read()
14     imgRGB = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
15     results = hands.process(imgRGB)
16
17     if results.multi_hand_landmarks:
18         for handLms in results.multi_hand_landmarks:
19             mp.solutions.drawing_utils.draw_landmarks(
20                 img,
21                 handLms,
22                 mp.solutions.hands.HAND_CONNECTIONS)
23
24     cv2.imshow("Image", img)
25
26     if cv2.waitKey(1) == 27:
27         break
28
29 cap.release()
30 cv2.destroyAllWindows()
```



- 0. WRIST
- 1. THUMB\_CMC
- 2. THUMB\_MCP
- 3. THUMB\_IP
- 4. THUMB\_TIP
- 5. INDEX\_FINGER\_MCP
- 6. INDEX\_FINGER\_PIP
- 7. INDEX\_FINGER\_DIP
- 8. INDEX\_FINGER\_TIP
- 9. MIDDLE\_FINGER\_MCP
- 10. MIDDLE\_FINGER\_PIP
- 11. MIDDLE\_FINGER\_DIP
- 12. MIDDLE\_FINGER\_TIP
- 13. RING\_FINGER\_MCP
- 14. RING\_FINGER\_PIP
- 15. RING\_FINGER\_DIP
- 16. RING\_FINGER\_TIP
- 17. PINKY\_MCP
- 18. PINKY\_PIP
- 19. PINKY\_DIP
- 20. PINKY\_TIP



**MediaPipe Face Mesh** - это решение для геометрии лица, которое оценивает 468 трехмерных ориентиров лица в реальном времени даже на мобильных устройствах. Он использует машинное обучение (ML) для определения геометрии трехмерной поверхности, требуя только входа одной камеры без необходимости в специальном датчике глубины. Используя упрощенные архитектуры моделей вместе с ускорением графического процессора на всем конвейере, решение обеспечивает производительность в реальном времени,



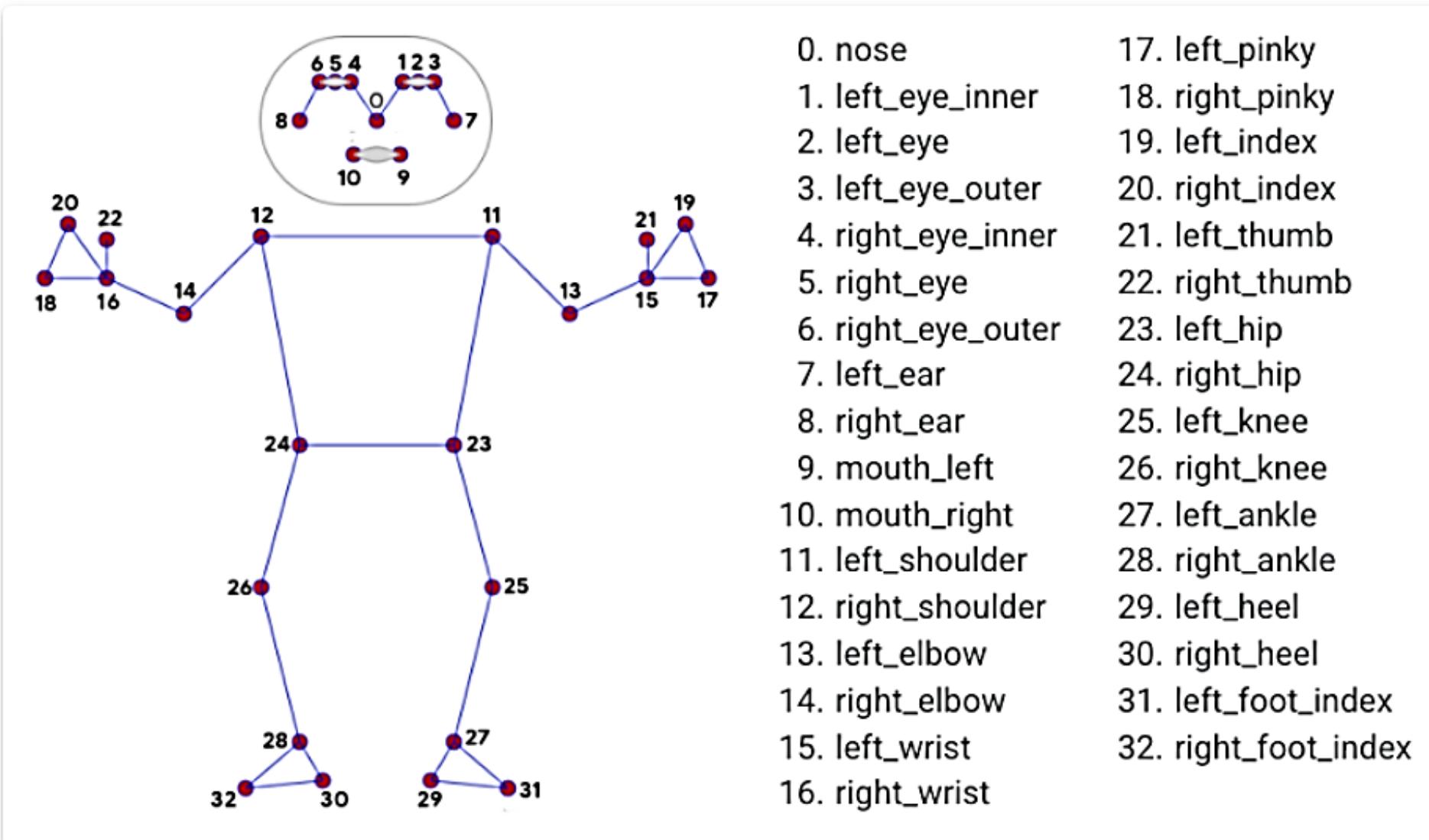


**MediaPipe Pose** - это решение машинного обучения для точного отслеживания позы тела, выведения 33 трехмерных ориентиров и маски сегментации фона на всем теле из видеокадров RGB с использованием нашего исследования [BlazePose](#), которое также [поддерживает API определения позы ML Kit](#).

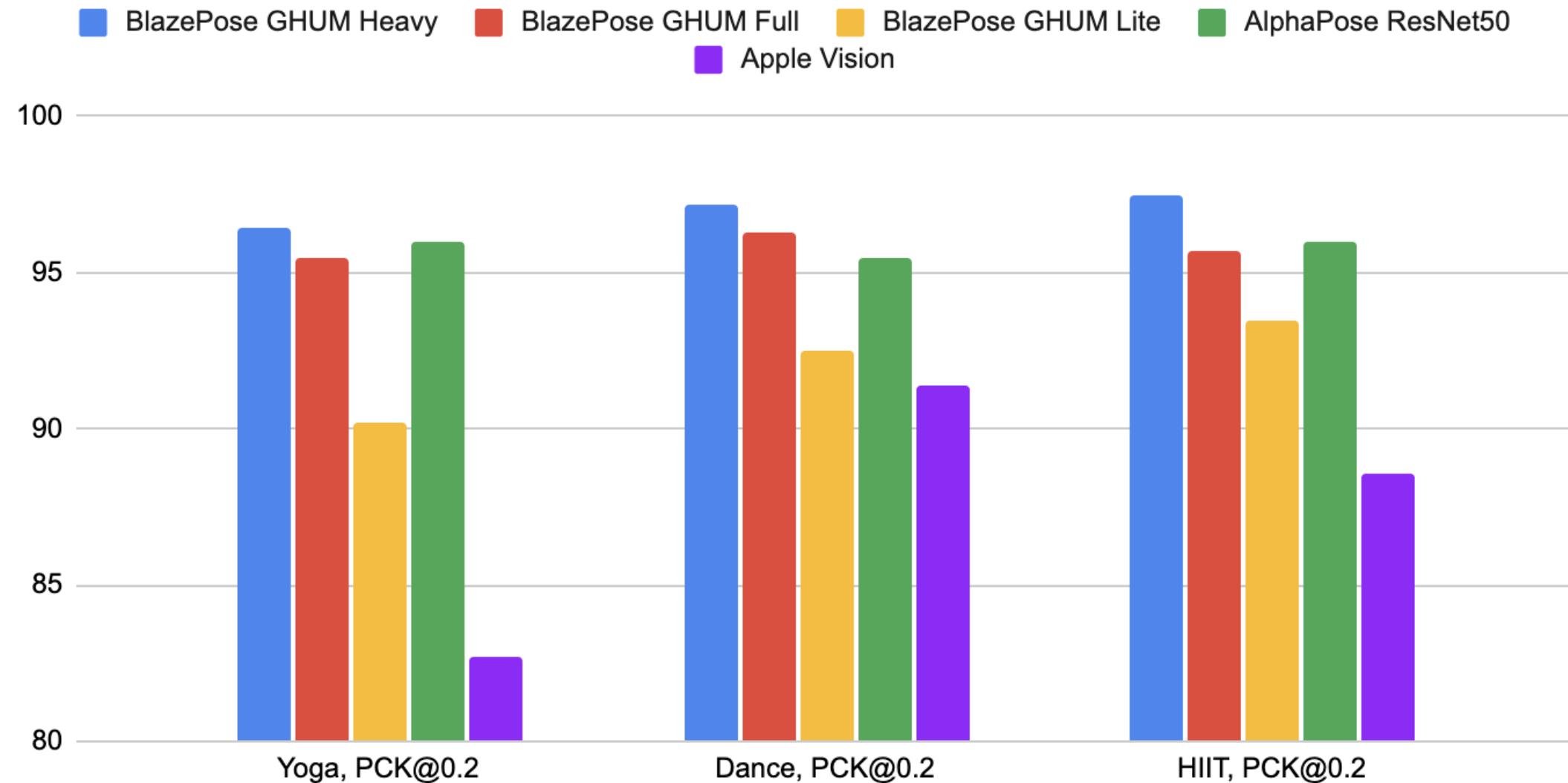
Метод обеспечивает производительность в реальном времени на большинстве современных [мобильных телефонов](#) , [настольных компьютеров / ноутбуков](#) , на [Python](#) и даже в [Интернете](#) .

## Модель Pose Landmark (BlazePose GHUM 3D)

Модель ориентира в MediaPipe Pose предсказывает расположение 33 ориентиров (см. Рисунок ниже).



В MediaPipe есть три модели для оценки позы: BlazePose GHUM Heavy, BlazePose GHUM Full и BlazePose GHUM Lite



Много интересных примеров в статье

## YOLOv7 pose vs MediaPipe при оценке позы человека

<https://habr.com/ru/articles/701268/>

Обработка окклюзии YOLOv7 vs MediaPipe

