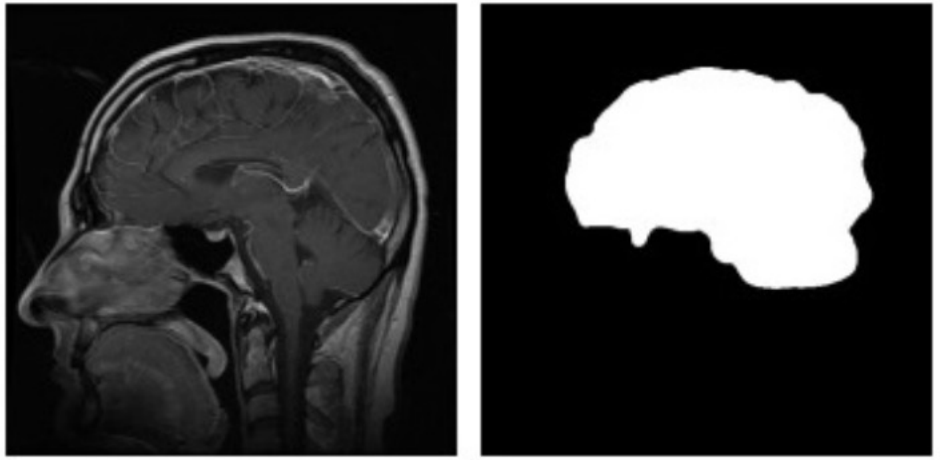
Лекция 6
Сегментация и LiDAR

Разработка нейросетевых систем

Канев Антон Игоревич

Сегментация

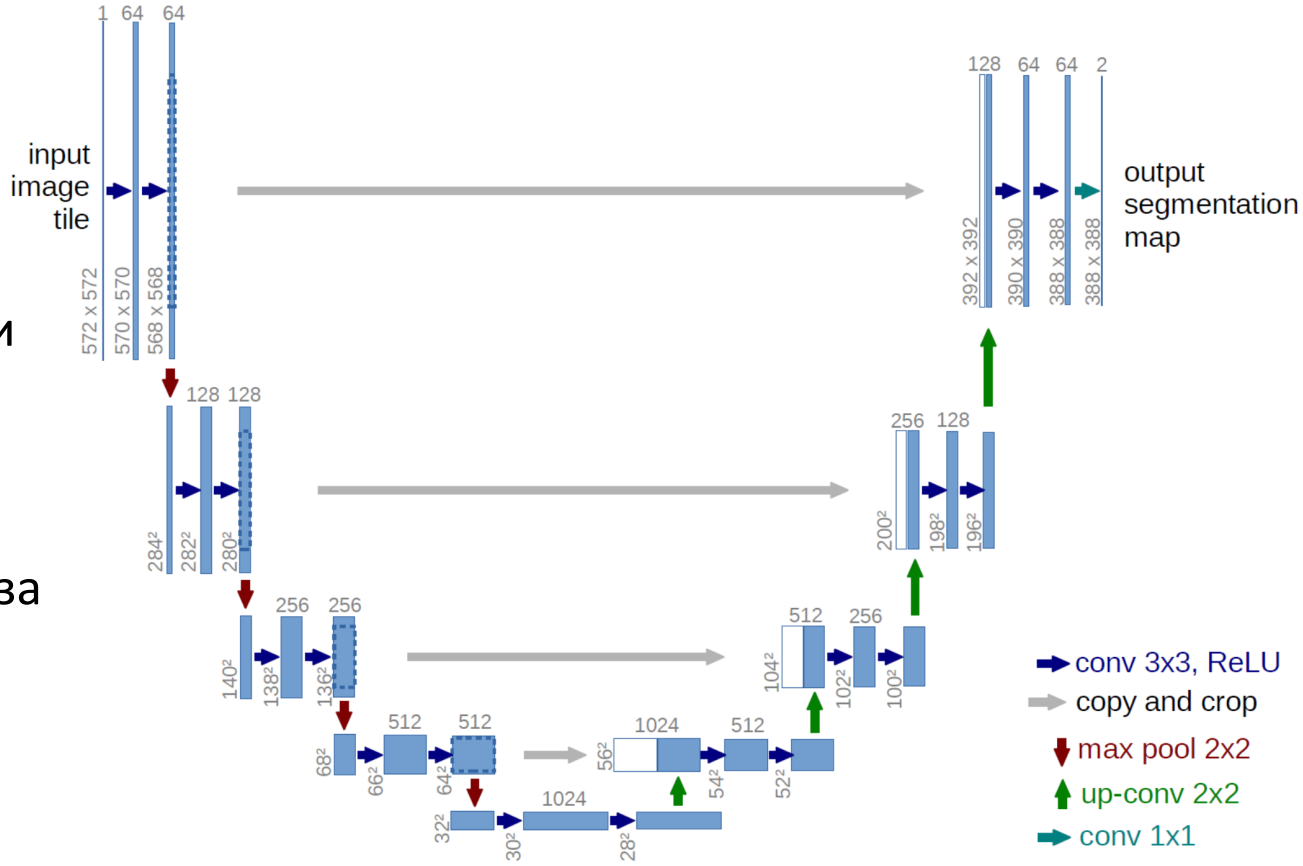
- Сегментация изображений является задачей разбиения цифрового изображения на одну или несколько областей, представляющих интерес.



- В модели для сегментации нет полносвязных слоёв, как в сетях для классификации
- Мы используем её для сегментации и в результате хотим получить картинку-маску, размер которой зависит от размера входного изображения.

U-Net

- Архитектура состоит из сужающегося пути и расширяющегося пути.
- Сужающийся путь — типичная архитектура сверточной нейронной сети. Он состоит из повторного применения двух сверток 3×3, за которыми следуют ReLU и max pool 2×2.
- Размер слоев становится все меньше и меньше для изучения более частных признаков, известный как «понижающая выборка».
- Сеть включает 4 соединения с пропуском — после каждой транспонированной свертки (или «up-conv») на пути повышающей дискретизации результирующая карта объектов объединяется с одной из пути понижающей дискретизации.



- pool/свертка 2×2 уменьшает/увеличивает количество каналов свойств;
- объединение с соответствующей обрезанной картой свойств из стягивающегося пути;
- две 3×3 свертки, за которыми следует ReLU.
- свертка 1×1 для сопоставления 64-мерного вектора с нужным количеством классов

Dice

- Dice коэффициент показывает попиксельное соотношение между прогнозируемой маской и соответствующей ей истиной
- Коэффициент Dice — это двойная площадь их пересечения, деленная на общее количество пикселей в обоих изображениях.
- Dice принимает значения от 0 до 1 (полное совпадение)

$$DC = \frac{2TP}{2TP + FP + FN} = \frac{2|X * Y|}{|X| + |Y|}$$

Ансамбли моделей

Усреднение прогнозов моделей

- Самый простой способ объединения предсказания подмоделей - это рассчитать их среднее значение. Также можно добавить веса, чтобы сделать более ощутимым вклад удачных моделей.

Stacking Ensemble

- Мы можем обучить мета-ученика, который будет сочетать прогнозы из подмоделей и в идеале делать предсказания точнее, чем любая отдельная подмодель.
- В качестве мета-ученика можно использовать линейную регрессию или нейронную сеть. Во втором случае можно рассматривать стековый ансамбль как единую большую модель (multi-headed model)
- Подмодели могут быть встроены в более крупную многоголовую нейронную сеть, которая затем учится, как лучше всего комбинировать прогнозы из каждой входной подмодели.

Keras. Сегментация

- Входному слою каждой подмодели нужно предоставить свои входные данные
- Затем скрытый слой будет подбирать веса для интерпретации этого «входа»
- Выходной слой (сверточный+активация sigmoid) делать свой собственный вероятностный прогноз.

```
members = [model1, model2]
```

```
from keras.layers import concatenate

ensemble_visible = [model.input for model in members]
ensemble_outputs = [model.output for model in members]
x = concatenate(ensemble_outputs)

x = Conv2D(16, (3, 3), padding="same")(x)
#x = Dropout(0.5)(x)
x = BatchNormalization()(x)
x = Activation("relu")(x)

x = Conv2D(1, (1, 1), padding="same")(x)
x = Activation("sigmoid")(x)
```

```
model1 = tf.keras.models.load_model(local_download_path + 'files/model186_dice.h5', compile=False)
model2 = tf.keras.models.load_model(local_download_path + 'files/model230_bc.h5', compile=False)
model3 = tf.keras.models.load_model(local_download_path + 'files/model230_tversky.h5', compile=False)
model4 = tf.keras.models.load_model(local_download_path + 'files/model230_dice_bc.h5', compile=False)
model5 = tf.keras.models.load_model(local_download_path + 'files/model_dice_wind.h5', compile=False)
```

Keras. Обучение ансамблей

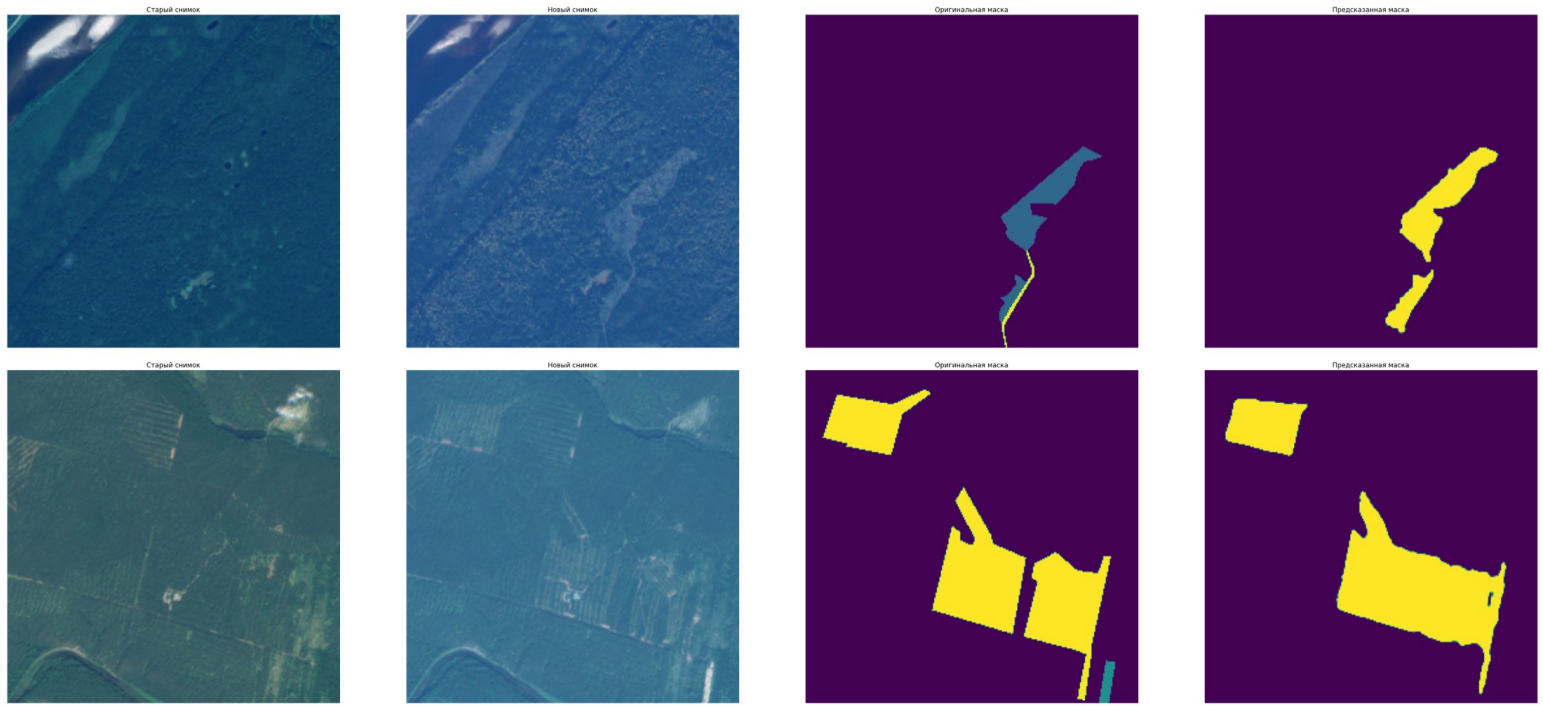
- Keras – надстройка над tensorflow для обучения нейронных сетей
- Обучение на Keras схоже с PyTorch
- Сами циклы обучения теперь скрыты в функции fit()

```
lr = 1e-3

opt = tf.keras.optimizers.Adam(learning_rate=lr)
metrics = [dice_coefficient]

ensemble_model.compile(loss=dice_loss, optimizer=opt, metrics=metrics)
```

```
batch = 8
epochs = 20

validSize = int(len(trainTiles)*0.15)

train_generator = DataGenerator(trainTiles[:-validSize],batch_size=batch,num_models=2)
valid_generator = DataGenerator(trainTiles[-validSize:],batch_size=batch,num_models=2)

hist = ensemble_model.fit(train_generator,
                          steps_per_epoch=len(trainTiles[:-validSize])//batch,
                          validation_data=valid_generator,
                          epochs=epochs,
                          validation_steps=len(trainTiles[-validSize:])//batch,
                          # callbacks=callbacks
                         )
```

Сегментация результаты

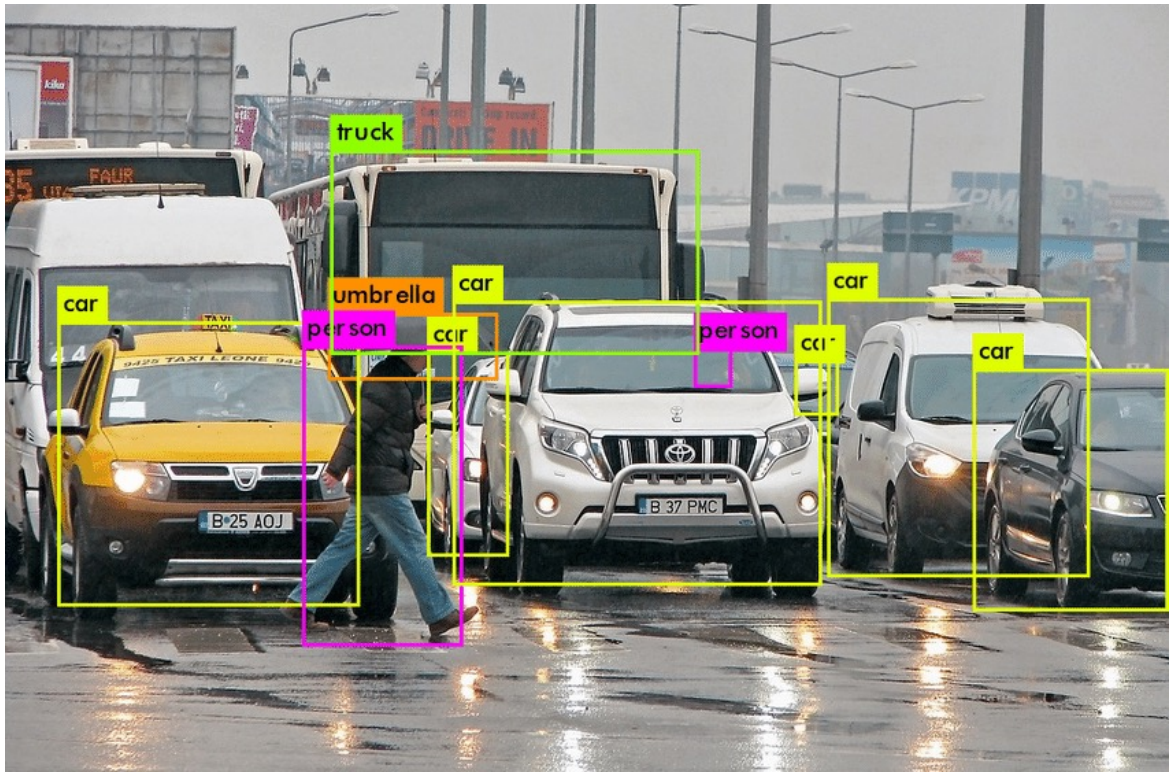
- Сравним маску, полученную моделью (справа), и оригинальную маску (слева)
- Для наглядности выведем старый и новый снимки



Обнаружение объектов

- **Object Detection** (обнаружение объектов) – определение объекта на изображении или в видео потоке

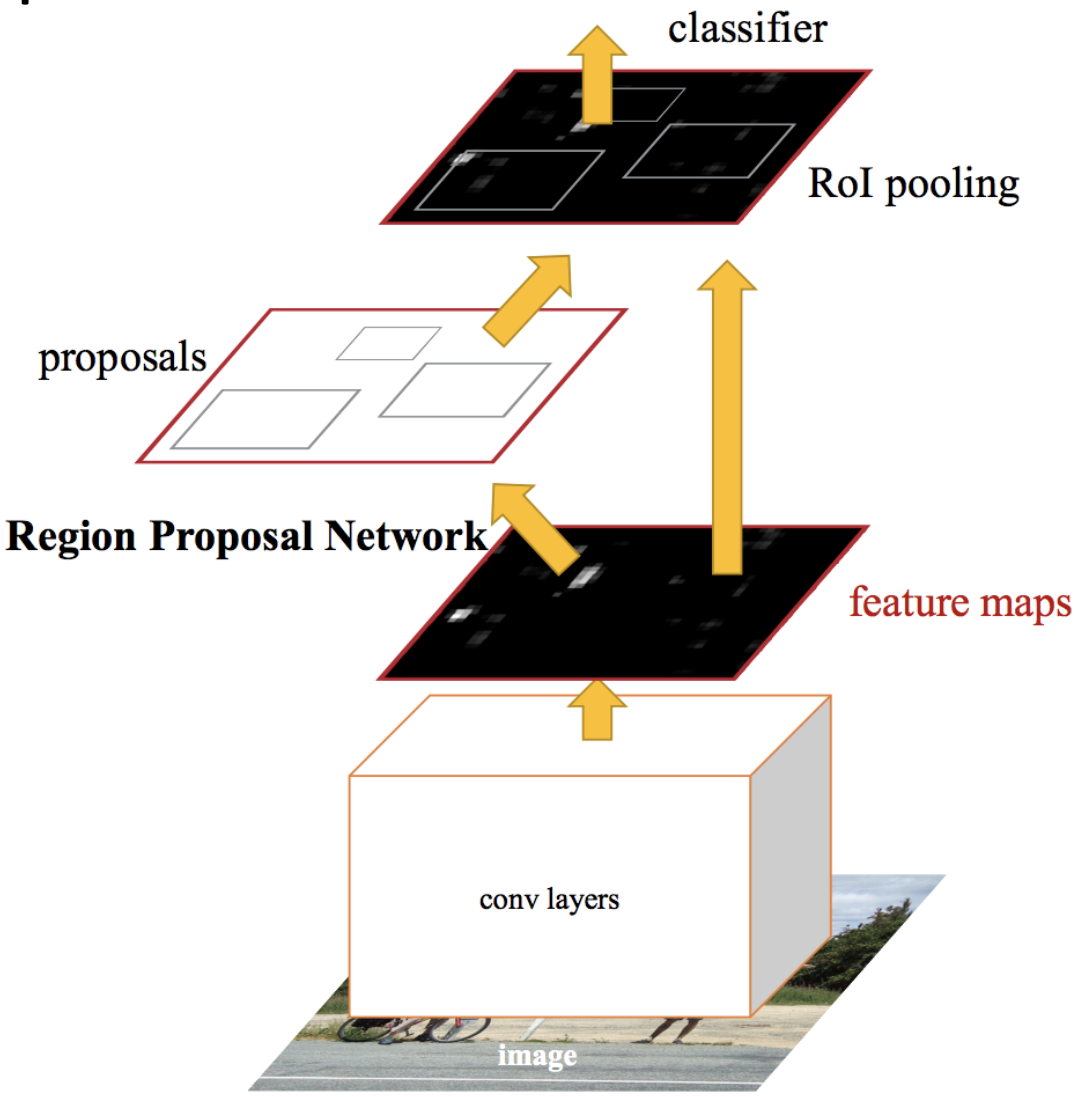
- Для этого используются различные модели, которые разделяются на «двухуровневые», такие как R-CNN, fast R-CNN и faster R-CNN, и «одноуровневые», такие как YOLO



Модели Object Detection

Faster R-CNN:

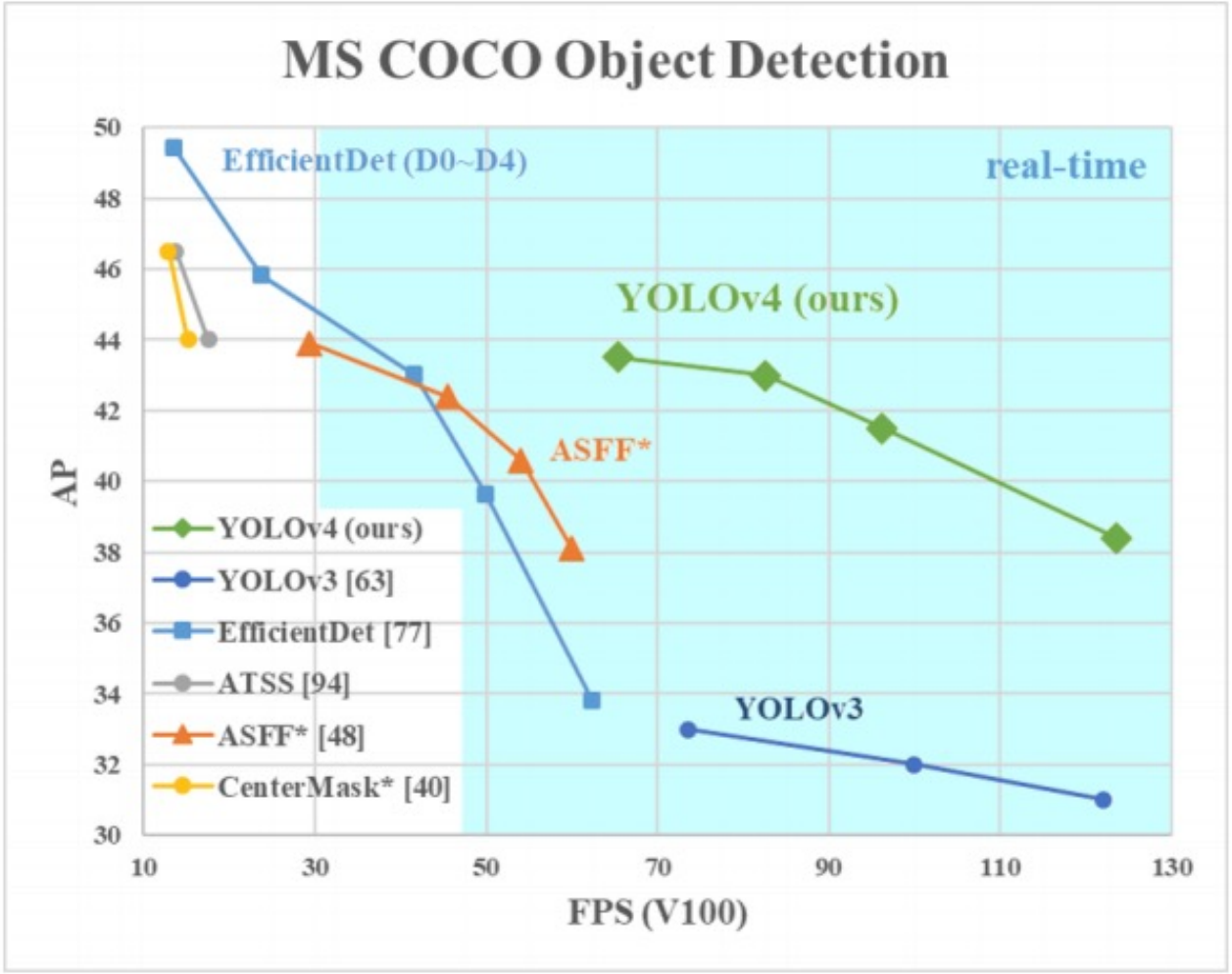
- Подается картинка/кадр на вход
- Кадр прогоняется через CNN для формирования feature maps
- Отдельной нейронной сетью определяются регионы с высокой вероятностью нахождения в них объектов
- Дальше эти регионы с помощью RoI pooling сжимаются и подаются в нейронную сеть, определяющую класс объекта в регионах



https://arxiv.org/pdf/1311.2524v4.pdf
R-CNN

YOLO

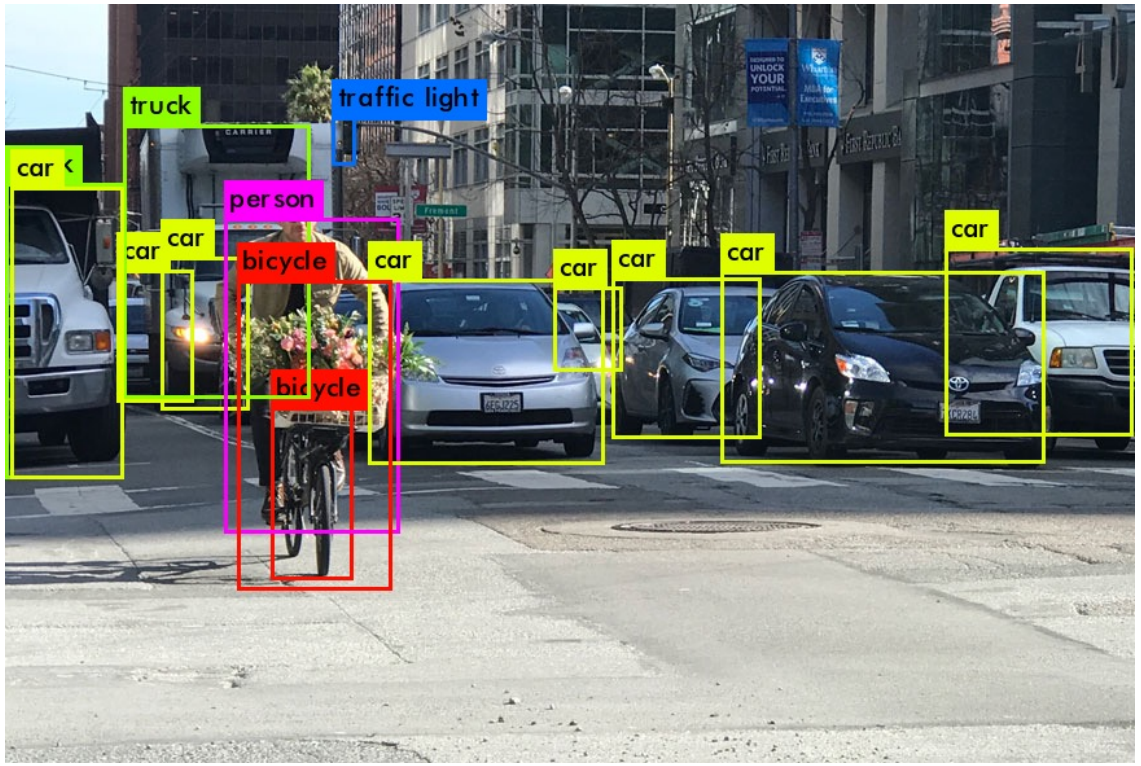
- YOLO (You Only Look Once) смотрит на картинку один раз, и за этот один прогон картинки через одну нейронную сеть делает все необходимые определения объектов
- За счет этого повышается производительность модели
- Существуют разные версии модели (v3, v4, v7 и тд)

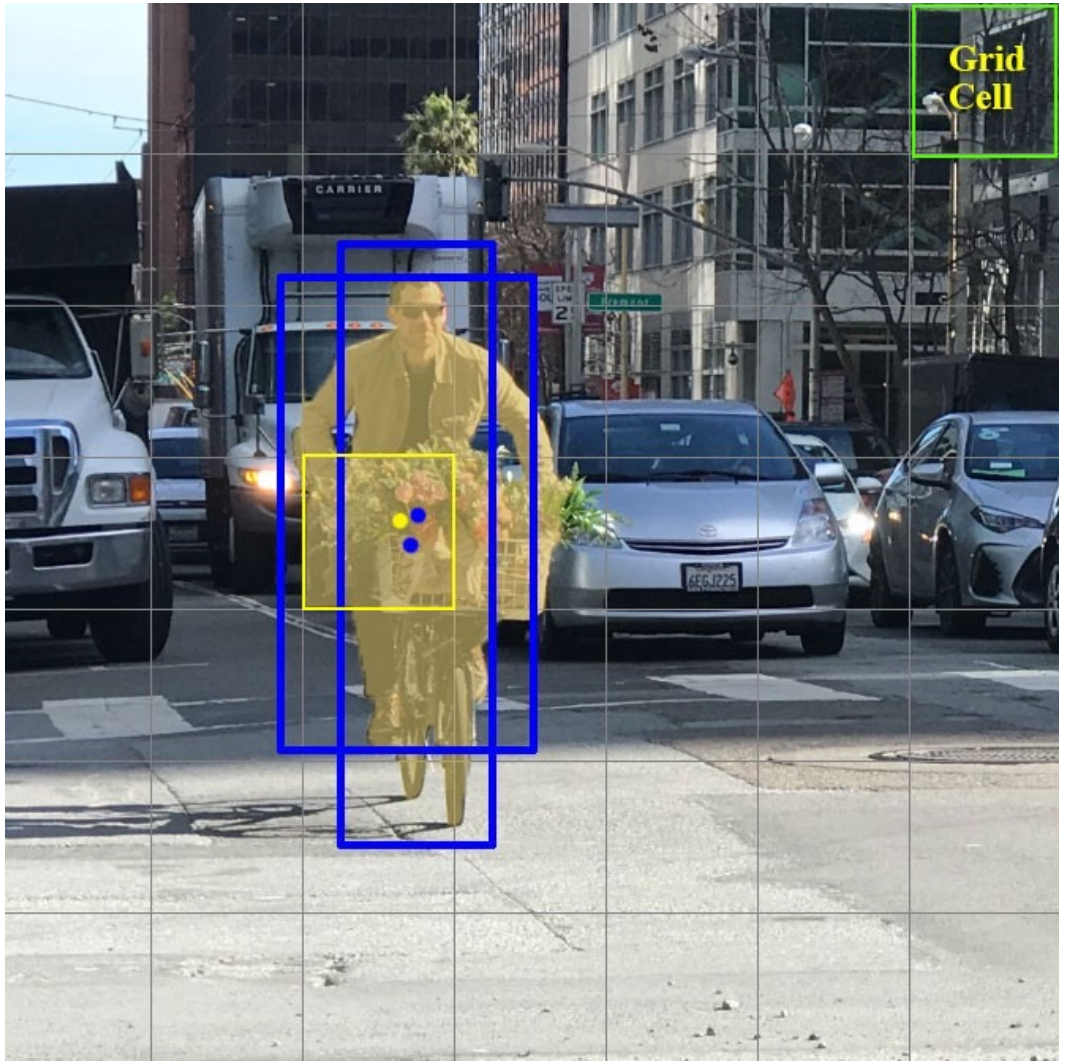


YOLOv3
https://pjreddie.com/media/files/papers/YOLOv3.pdf

YOLO

- Каждая клетка grid cell является «якорем», к которому прикрепляются bounding boxes. Это основа идеи YOLO
- Вокруг клетки рисуются несколько прямоугольников для определения объекта (непонятно, какой будет наиболее подходящим), и их позиции, ширина и высота вычисляются относительно центра этой клетки.

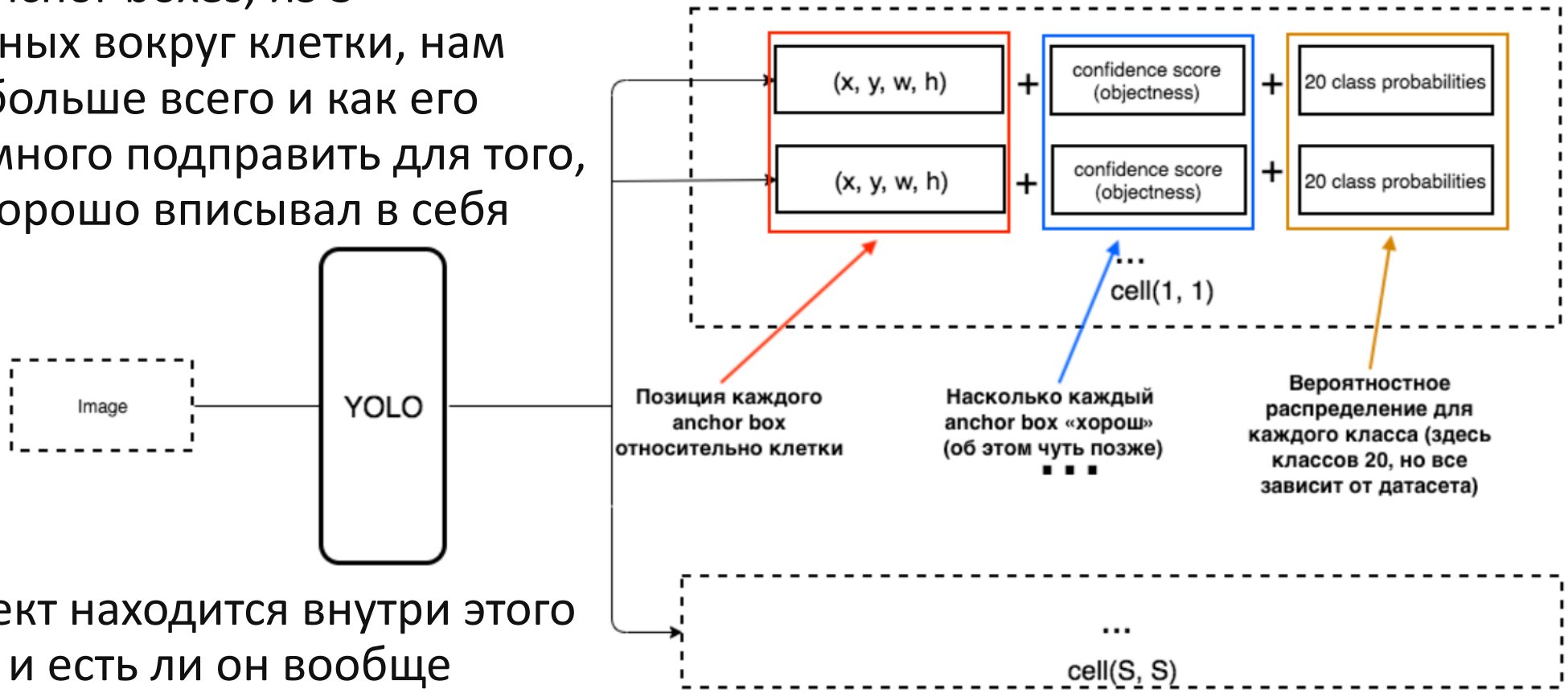




Выход YOLO

Нам нужно понять две принципиальные вещи:

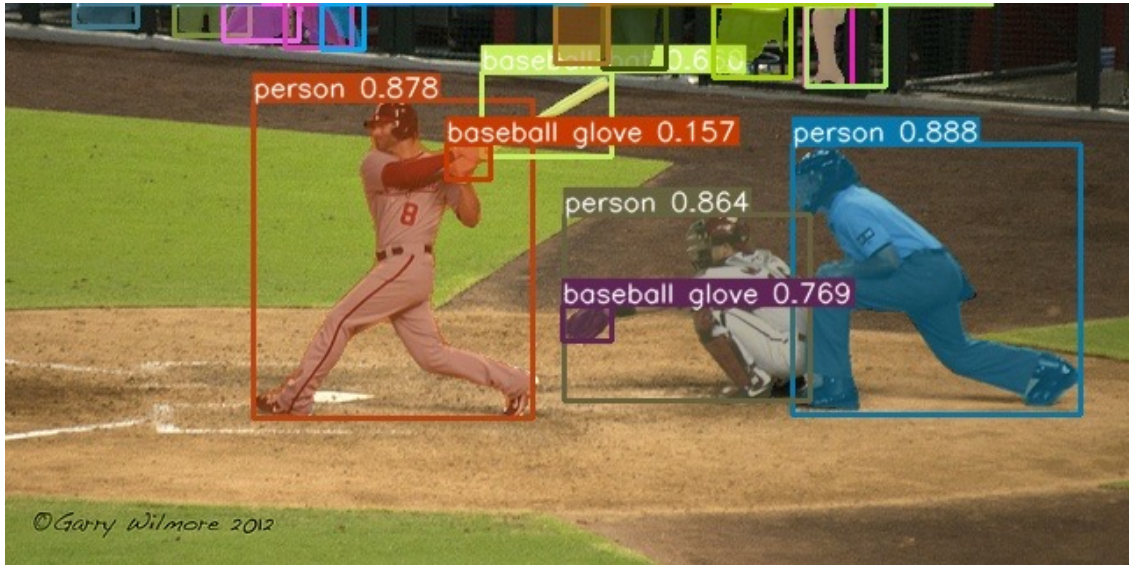
- Какой из anchor boxes, из 3 нарисованных вокруг клетки, нам подходит больше всего и как его можно немного подправить для того, чтобы он хорошо вписывал в себя объект

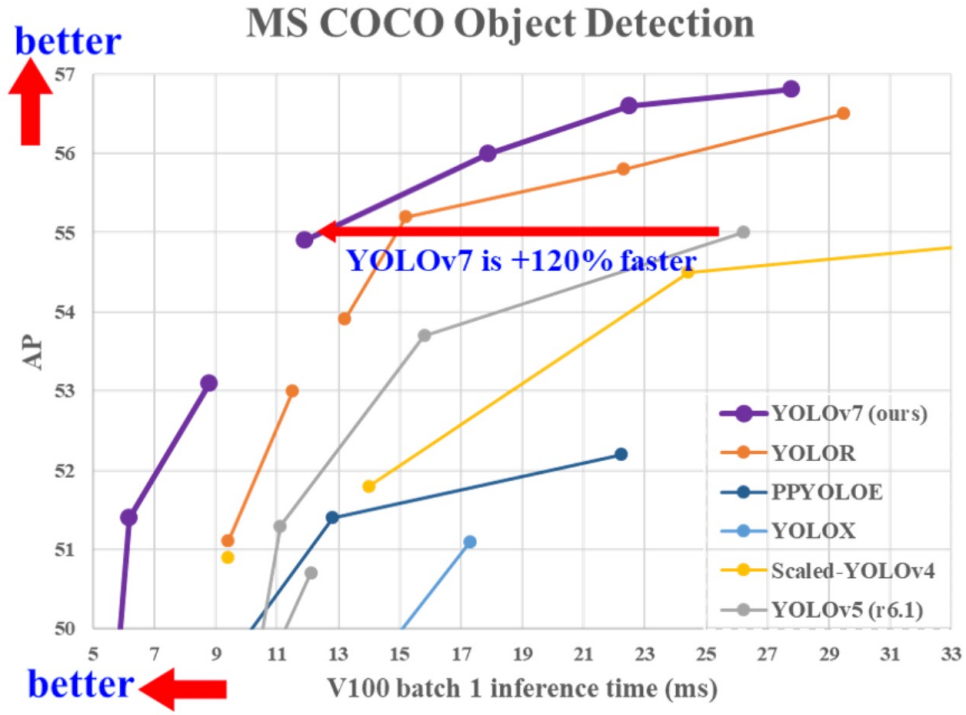


- Какой объект находится внутри этого anchor box и есть ли он вообще

YOLOv7

- Помимо object detection модель YOLO используется для оценки положения или сегментации

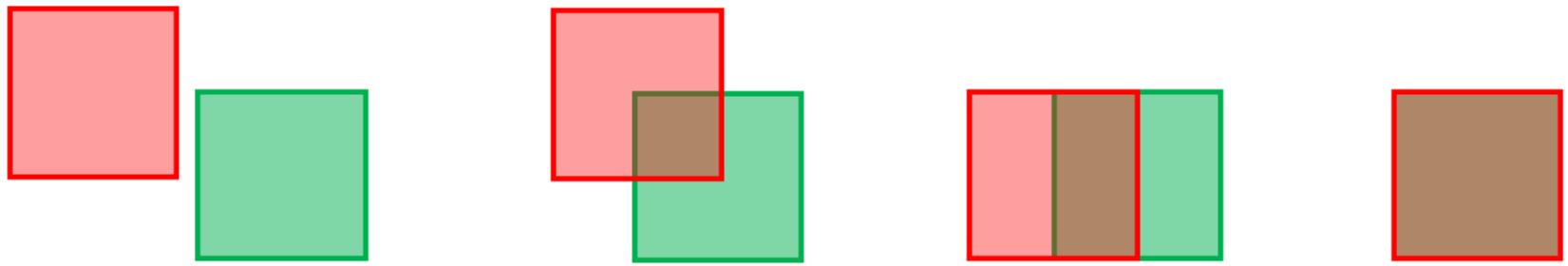




IoU

- Метрика Intersection over Union (IoU), также известная как Jaccard index
- Число от 0 до 1, показывающее, насколько у двух объектов (эталонного и текущего) совпадает общая часть

$$IoU = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$

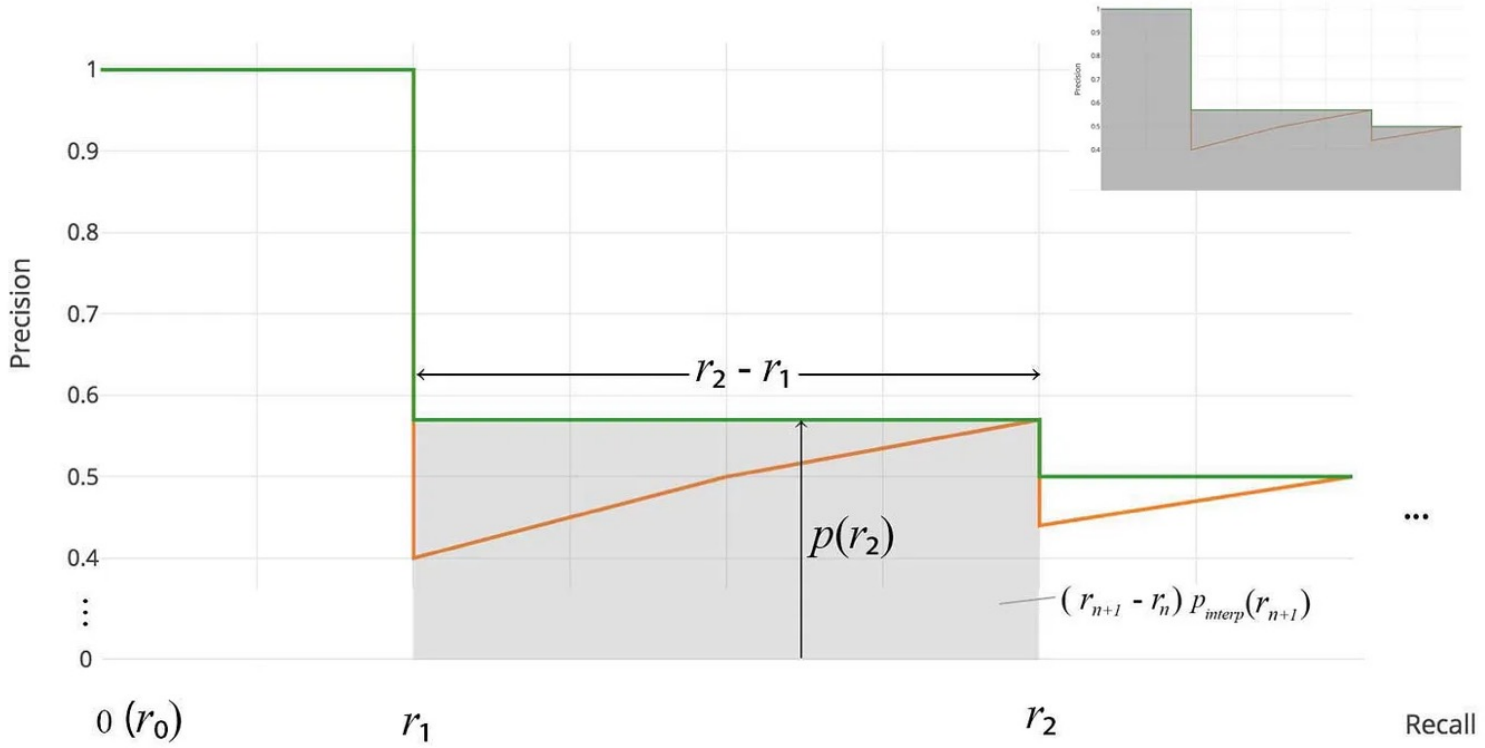


IoU = 0 IoU = 0.142 IoU = 0.333 IoU = 1

AP

- AP (Average precision) – популярная метрика для оценки точности моделей обнаружения объектов

Rank	Correct?	Precision	Recall
1	True	1.0 ↑	0.2 ↑
2	True	1.0 –	0.4 ↑
3	False	0.67 ↓	0.4 –
4	False	0.5 ↓	0.4 –
5	False	0.4 ↓	0.4 –
6	True	0.5 ↑	0.6 ↑
7	True	0.57 ↑	0.8 ↑



- Precision и Recall связаны друг с другом: чем больше одна, тем обычно меньше другая метрика

Разметка изображений

- Cvat.ai – можно развернуть локально и работать совместно
- ImageJ – устанавливается локально

- Roboflow – поддерживается аугментация, разметка
- ij.imjoy.io
- VGG Image Annotator

https://www.cvat.ai

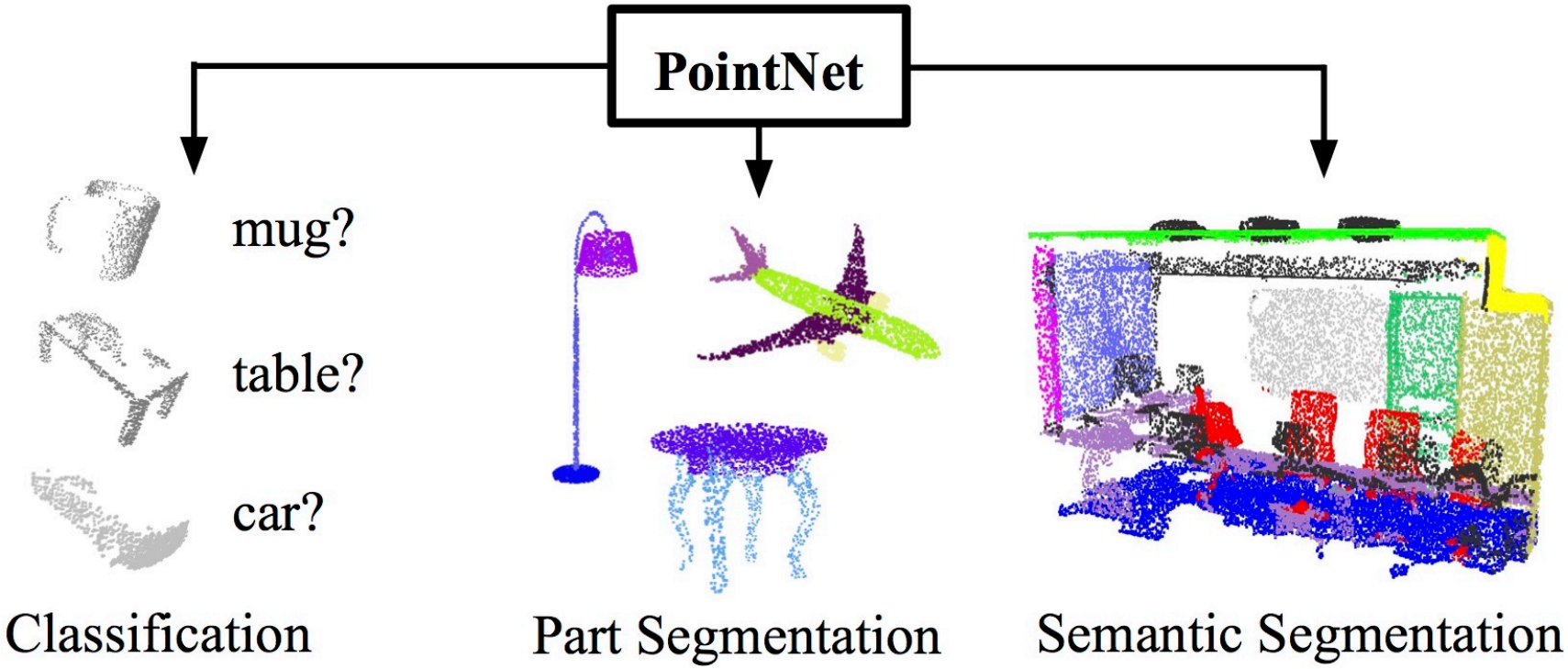
LiDAR

- Облако точек - это самый простой способ представления различных объектов в виде неупорядоченного набора точек в трехмерной плоскости. Такие данные можно получить с помощью сканирования предметов или их структуры с помощью 3D-датчиков, например LiDAR.

- Качественные облака точек с высокой точностью измерения позволяют представить цифровую версию реального мира.
- Основная проблема работы с облаком точек заключается в том, что типичная сверточная архитектура требует упорядоченный формат входных данных (например, изображение).
- Поскольку облако точек не является таким, общепринятые подходы заключаются в преобразовании данных в обычную 3D-воксельную сетку или проекцию.

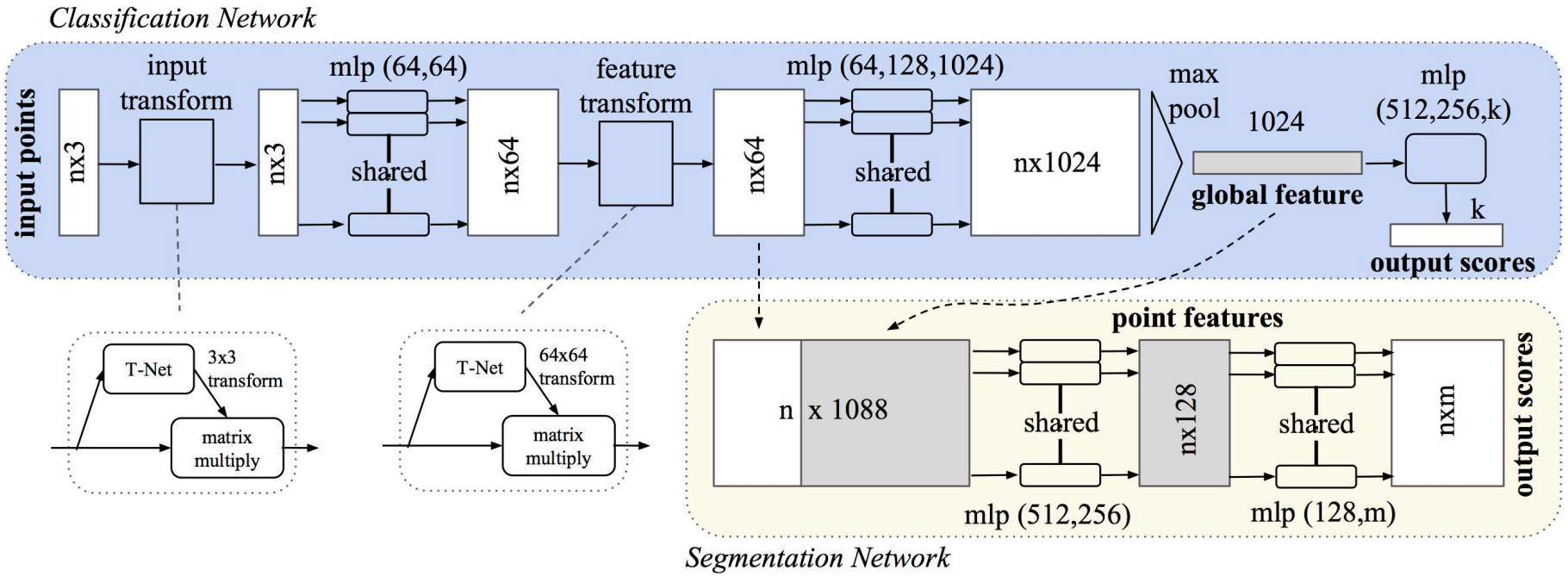
PointNet. Применение

- PointNet использует неупорядоченные облака точек и может выполнять классификацию и сегментацию объектов, а также семантический анализ сцены.



PointNet. Архитектура

- В сети есть 3 ключевых модуля: слой Max Pooling, принимающий n векторов входных данных и выводящий новый вектор, две сети трансформации с многослойным персептроном (MLP) с размерами (64,64) и (64,128,1024) и две сети для предсказания с обученной матрицей T-Net.
- PointNet изучает характеристики каждой точки с помощью MLP и объединяет все характеристики с помощью симметричной функции для классификации объектов и их сегментации на части.



Лабораторная LiDAR

```
inputs = keras.Input(shape=(NUM_POINTS, 3))

x = tnet(inputs, 3)
x = conv_bn(x, 32)
x = conv_bn(x, 32)
x = tnet(x, 32)
x = conv_bn(x, 32)
x = conv_bn(x, 64)
x = conv_bn(x, 512)
x = layers.GlobalMaxPooling1D()(x)
x = dense_bn(x, 256)
x = layers.Dropout(0.3)(x)
x = dense_bn(x, 128)
x = layers.Dropout(0.3)(x)

outputs = layers.Dense(NUM_CLASSES, activation="softmax")(x)
```

```
model.compile(
    loss="sparse_categorical_crossentropy",
    optimizer=keras.optimizers.SGD(learning_rate=0.001),
    metrics=["sparse_categorical_accuracy"],
)

history = model.fit(train_dataset, epochs=10, validation_data=test_dataset)
```

- Каждый сверточный и полносвязный слой, кроме выходного состоит из Convolution/Dense, также Batch Normalization и ReLU
- PointNet состоит из двух основных компонентов: основная сеть MLP (многослойный перцептрон) и трансформаторная сеть T-net.