Глубокое обучение

Коломейцева Катерина

Лекция 6: Обзор задач компьютерного зрения.

Детекторы (object detectors) – алгоритмы, выделения объектов на изображении

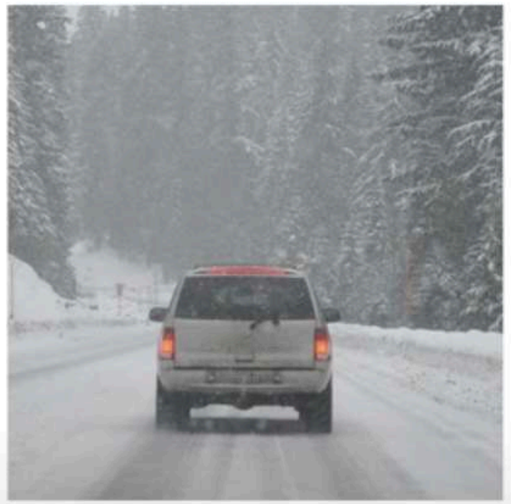
- Как мы можем описать локализацию объекта?

Локализация объекта

- Ограничивающий прямоугольник
- Эллипс/окружность в зависимости от формы объекта
- Граница/область объекта

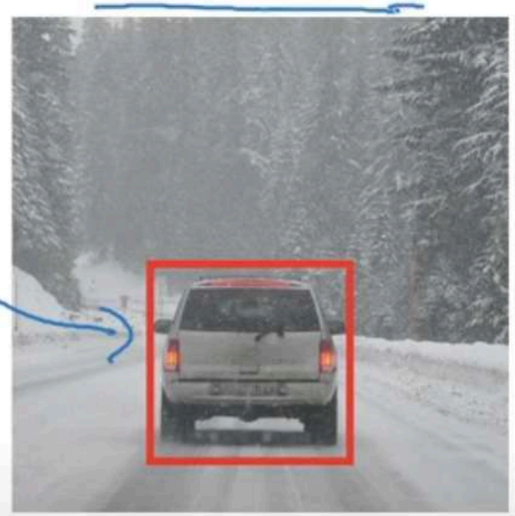
What are localization and detection?

Image classification



"Car"

Classification with localization



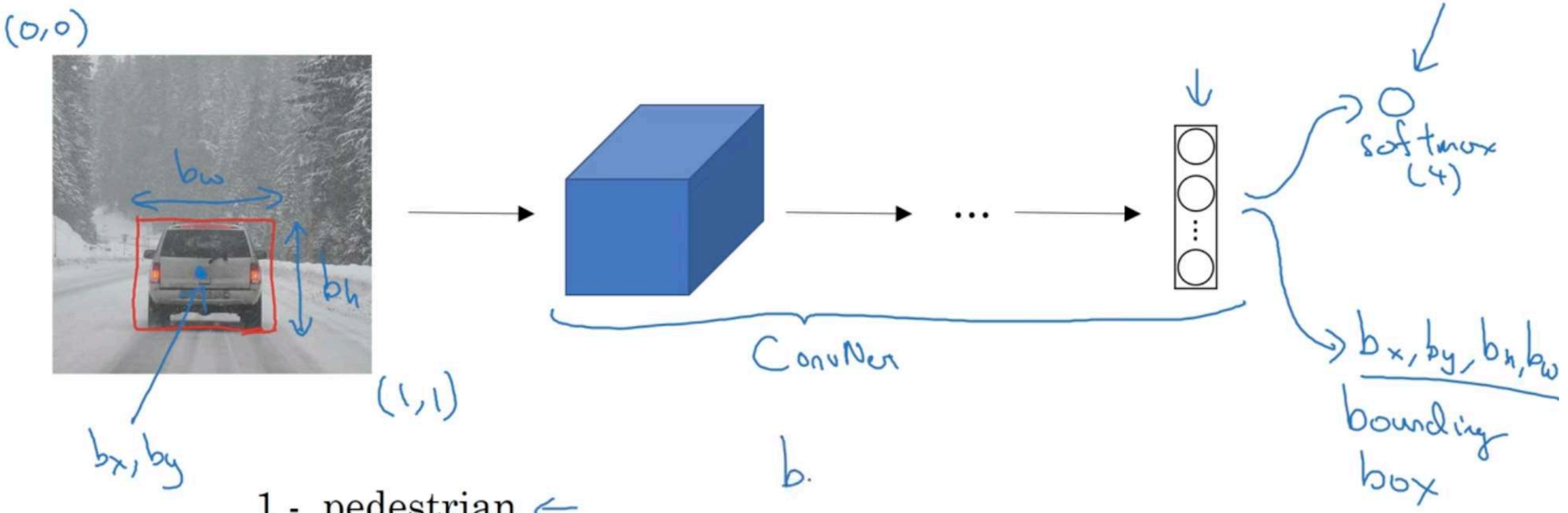
"Car"

1 object

Detection

multiple objects

Classification with localization



1 - pedestrian ←

2 - car ←

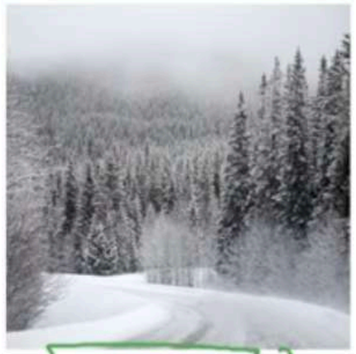
3 - motorcycle ←

4 - background

Defining the target label y

1 - pedestrian
2 - car
3 - motorcycle
4 - background

Need to output b_x, b_y, b_h, b_w, class label (1-4)



$$\mathcal{L}(\hat{y}, y) = \begin{cases} (\hat{y}_1 - y_1)^2 + (\hat{y}_2 - y_2)^2 + \cdots + (\hat{y}_8 - y_8)^2 & \text{if } y_1 = 1 \\ (\hat{y}_1 - y_1)^2 & \text{if } y_1 = 0 \end{cases}$$

$$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$ p_c is there an object?

$$\begin{bmatrix} 1 \\ b_x \\ b_y \\ b_h \\ b_w \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} 0 \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \end{bmatrix}$$ p_c ← "don't care"

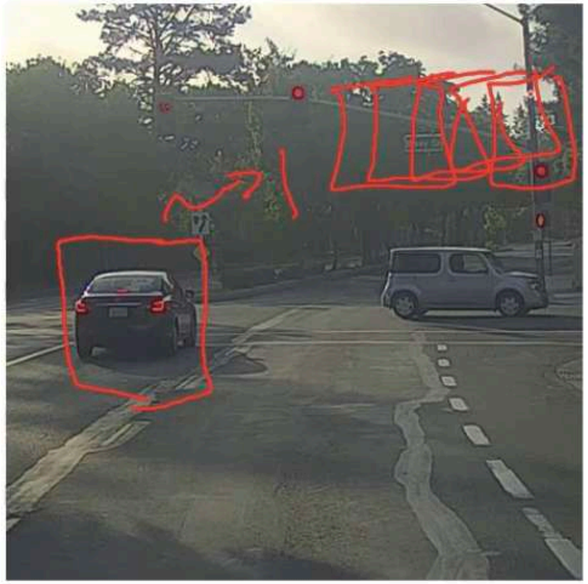
Car detection example

Training set:

x	y
	1
	1
	1
	0
	0

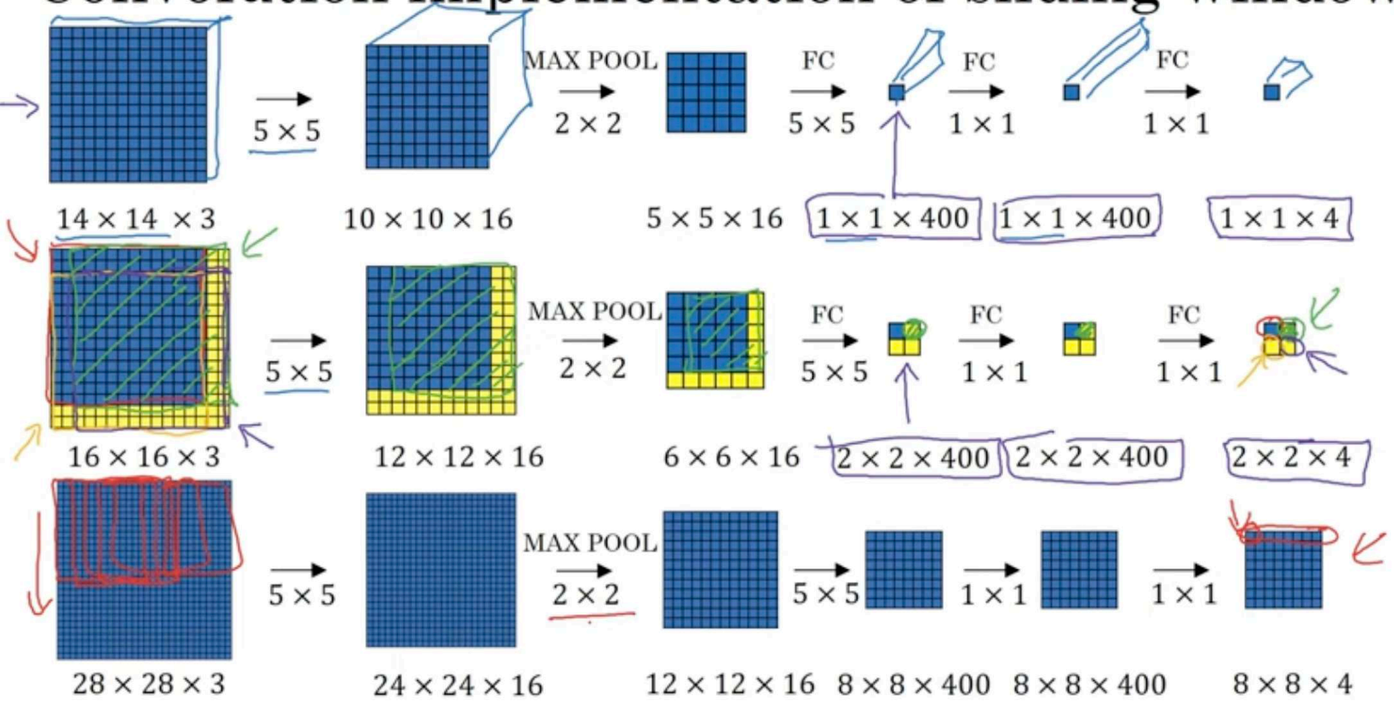
→ ConvNet → y

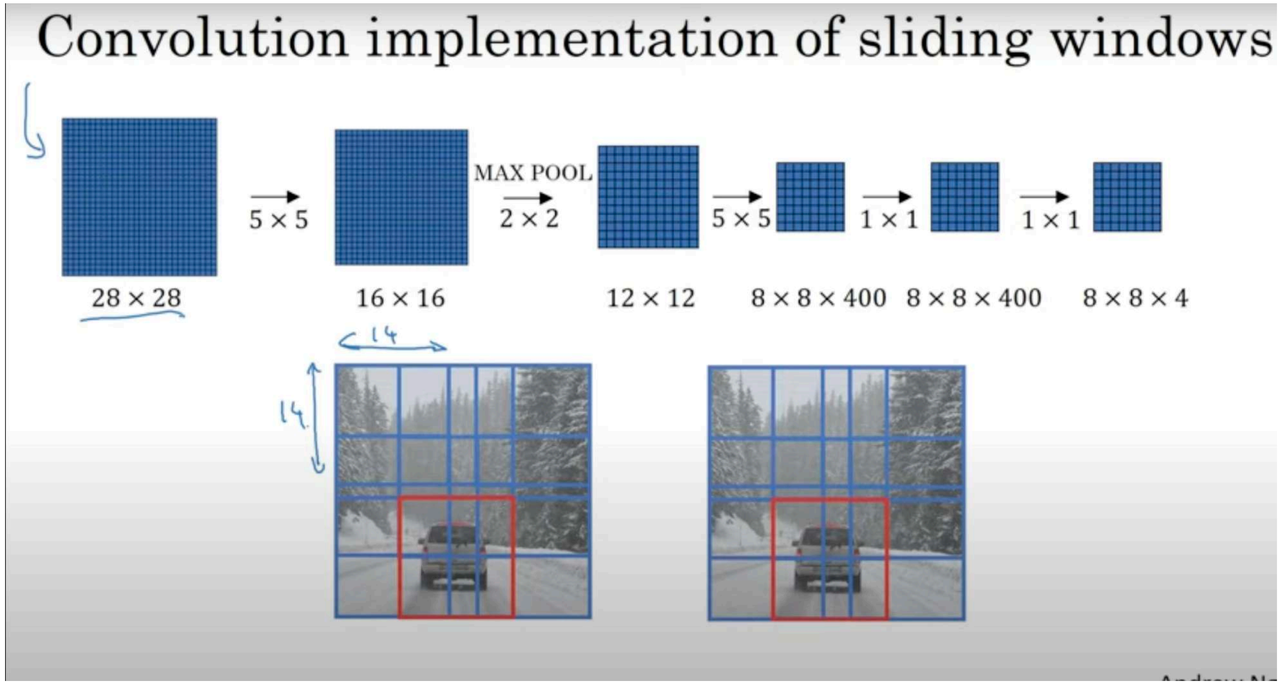
Sliding windows detection



Computation cost

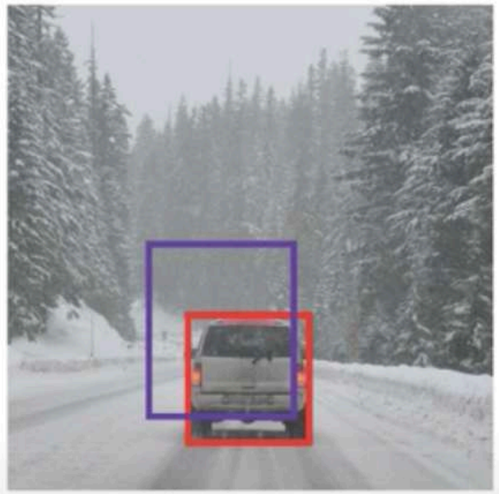
Convolution implementation of sliding windows





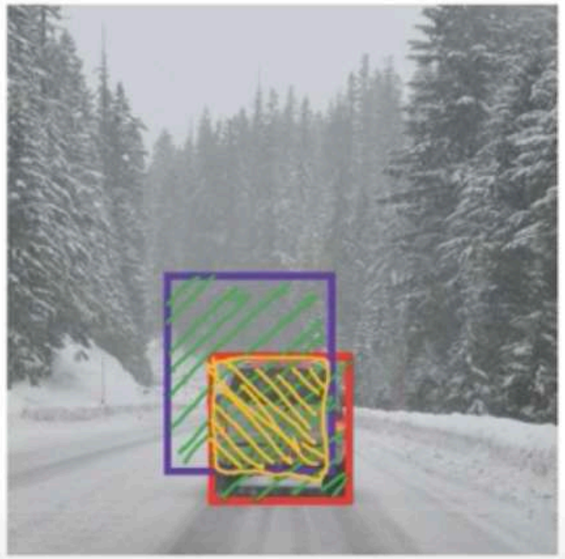
Как оценить точность?

Evaluating object localization



Intersection over Union (IoU)

Evaluating object localization

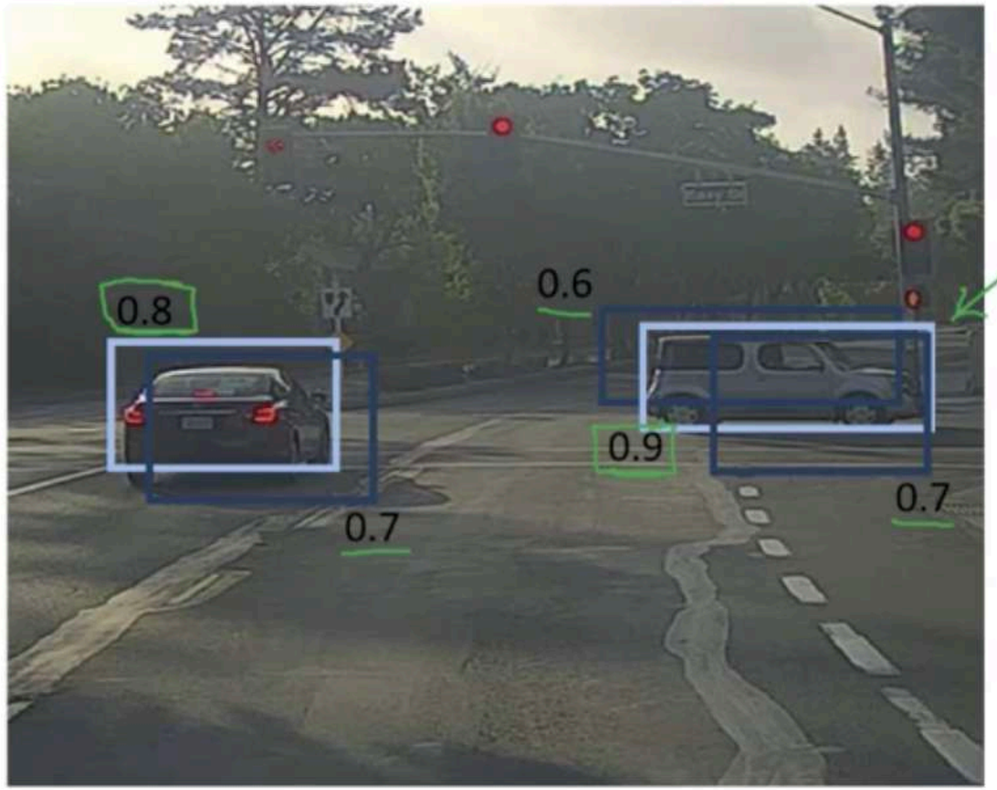


Intersection over Union (IoU)

$$= \frac{\text{size of } \square}{\text{size of } \square}$$

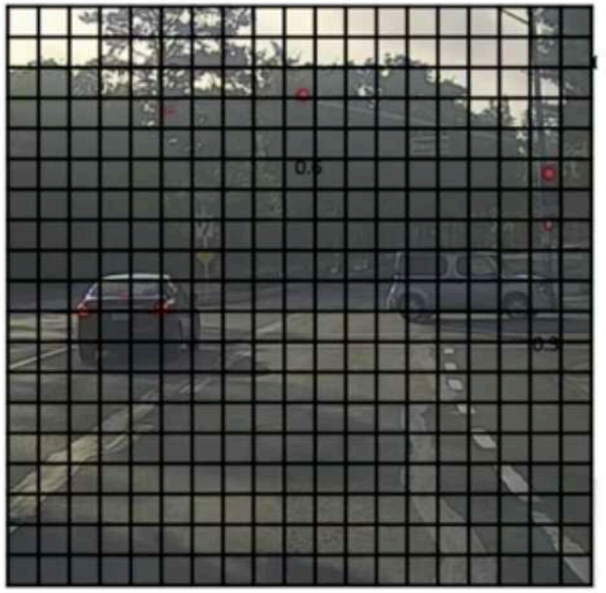
"Correct" if IoU ≥ 0.5

Non-max suppression example



p_c

Non-max suppression algorithm



19× 19

Each output prediction is:

$$\begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \end{bmatrix}$$

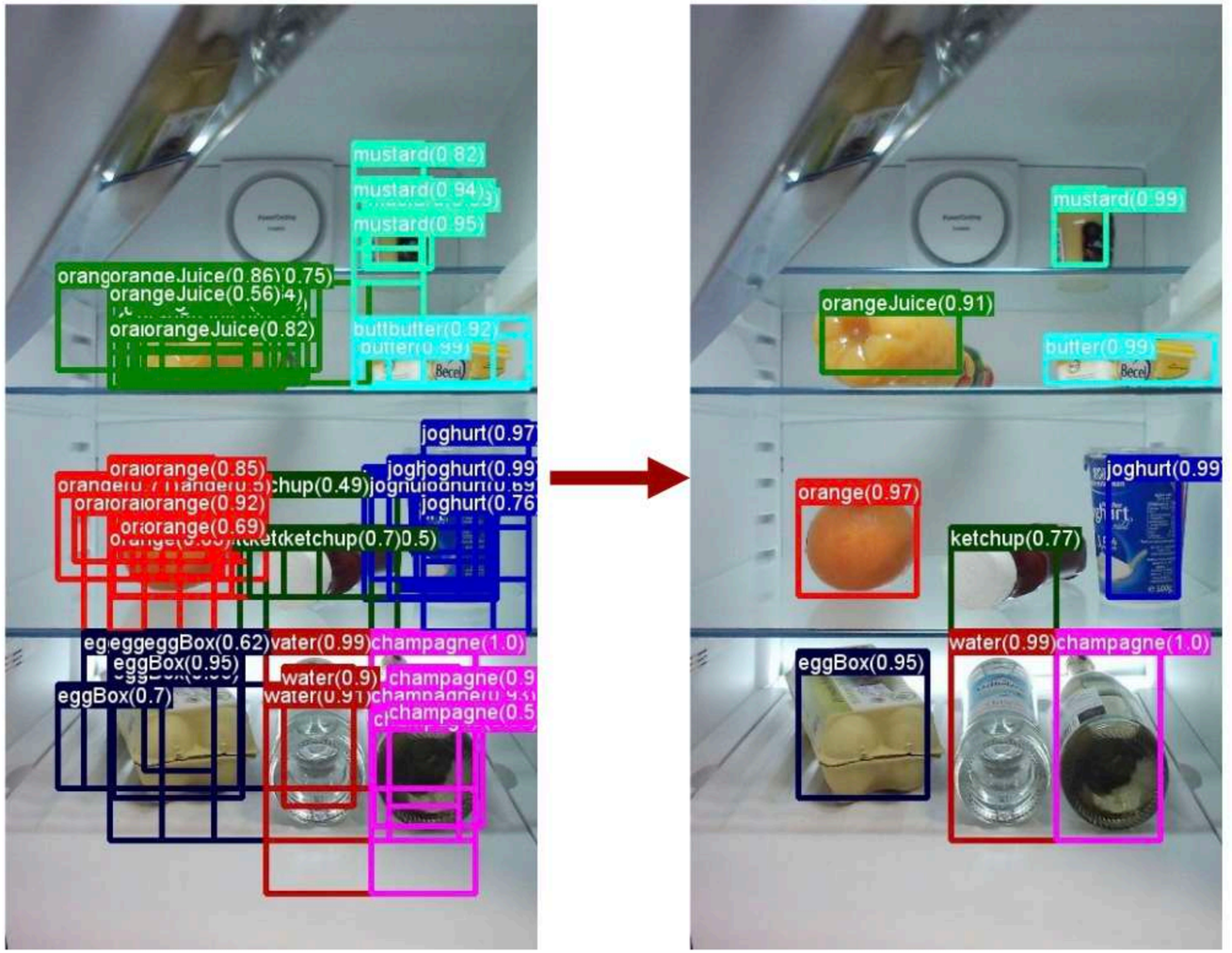
Discard all boxes with $p_c \leq 0.6$

While there are any remaining boxes:

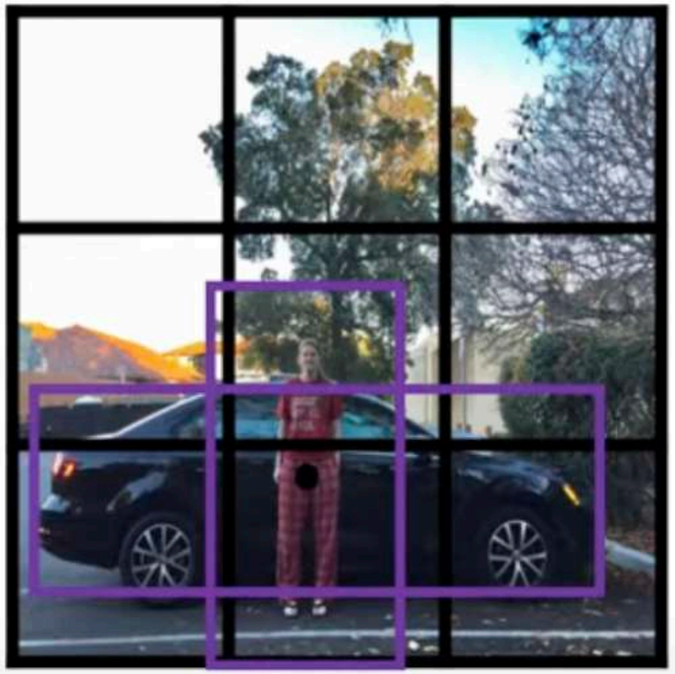
- Pick the box with the largest p_c Output that as a prediction.
- Discard any remaining box with IoU ≥ 0.5 with the box output in the previous step

Non Maximum Supression (NMS)

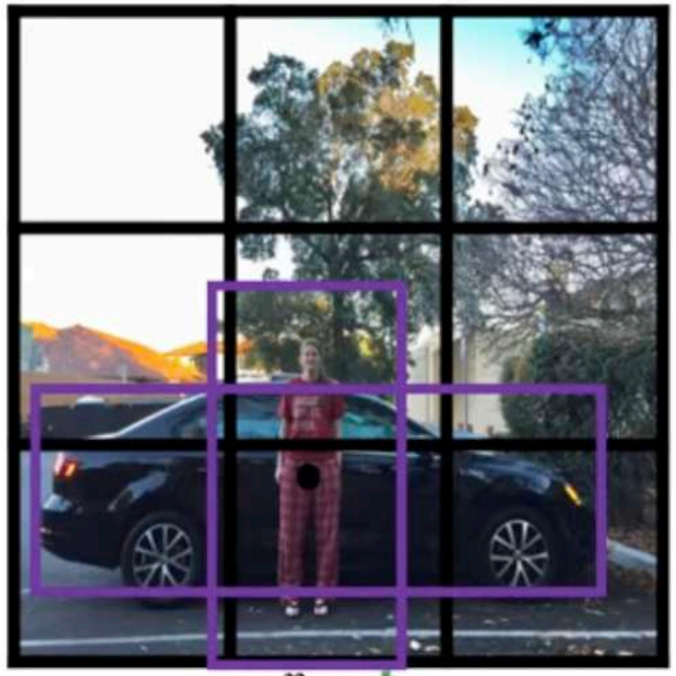
Detection = Предсказываем много боксов, а потом фильтруем



Overlapping objects:

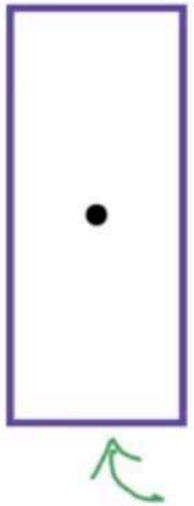


Overlapping objects:

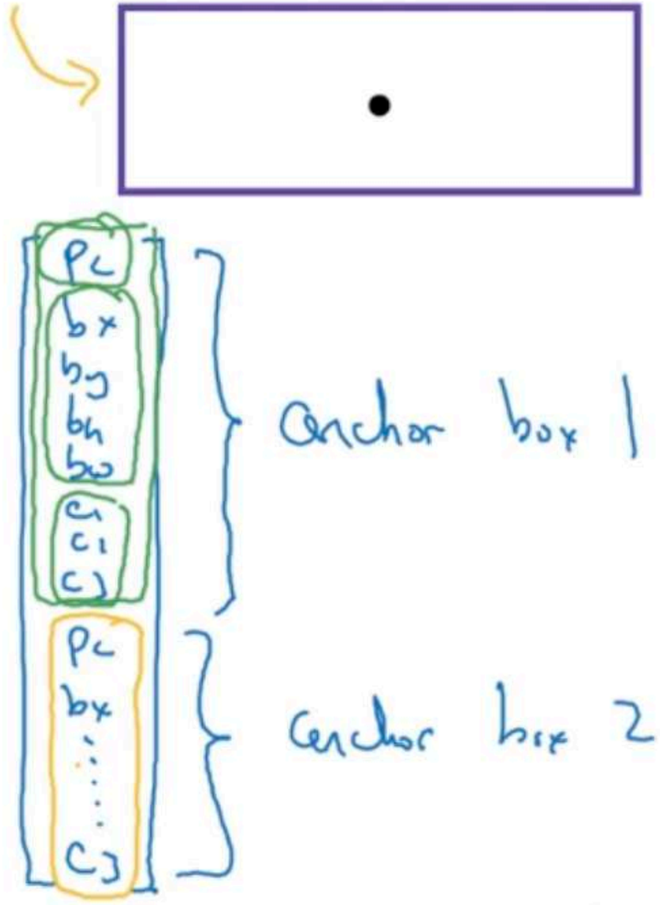


$$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

Anchor box 1:



Anchor box 2:



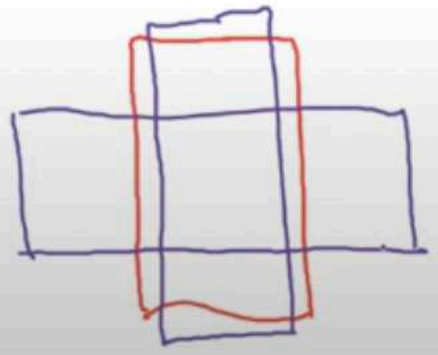
Anchor box algorithm

Previously:

Each object in training image is assigned to grid cell that contains that object's midpoint.

Output y:

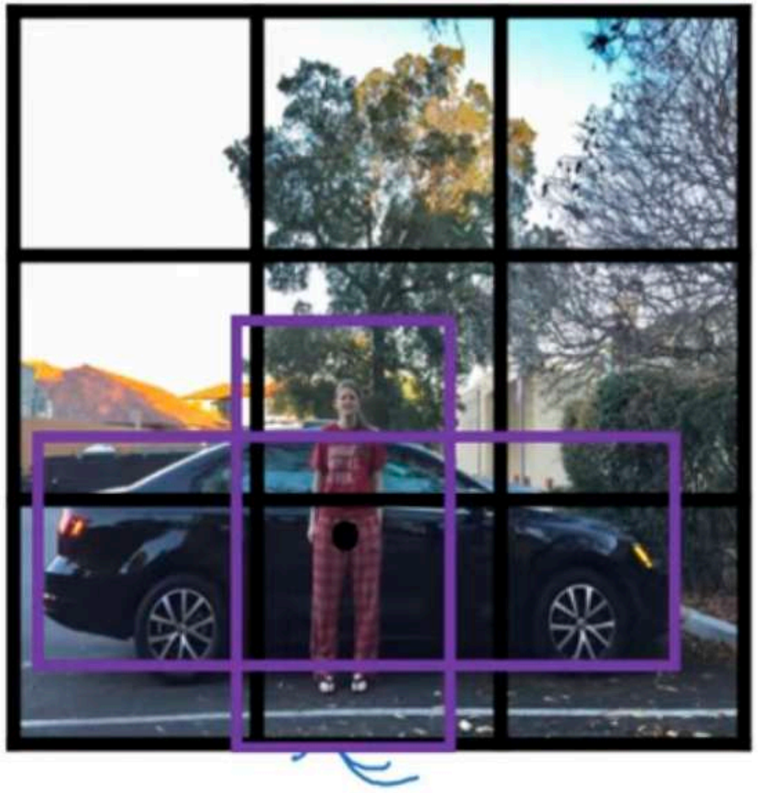
$3 \times 3 \times 8$



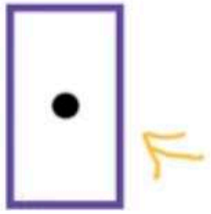
With two anchor boxes:

Each object in training image is assigned to grid cell that contains object's midpoint and anchor box for the grid cell with highest IoU.

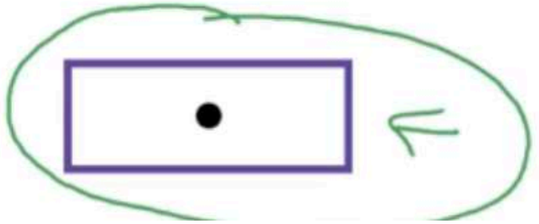
Anchor box example



Anchor box 1:

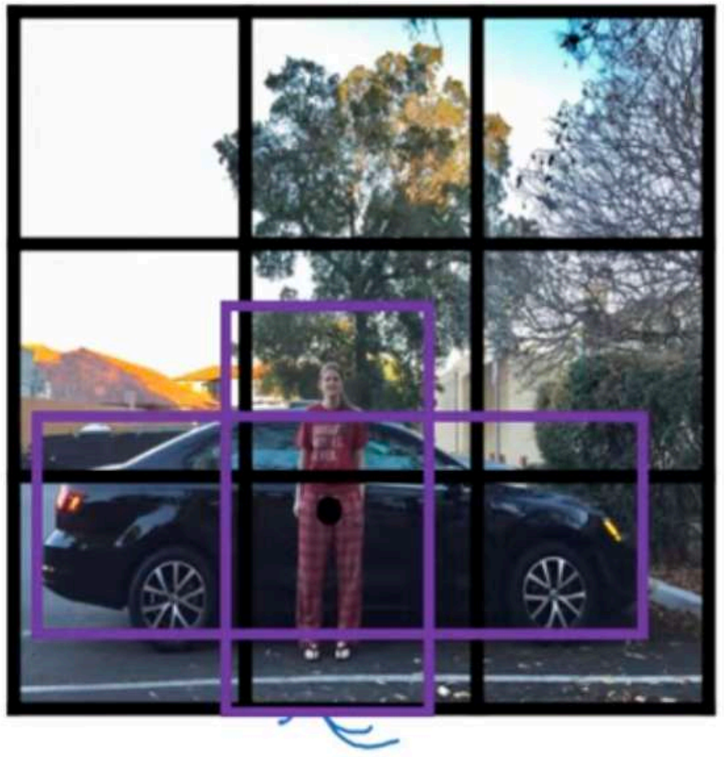


Anchor box 2:

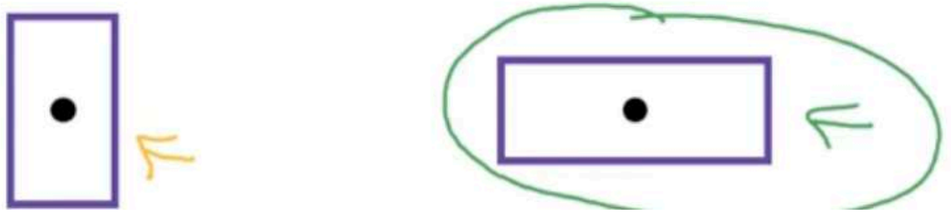


$$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \\ p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

Anchor box example

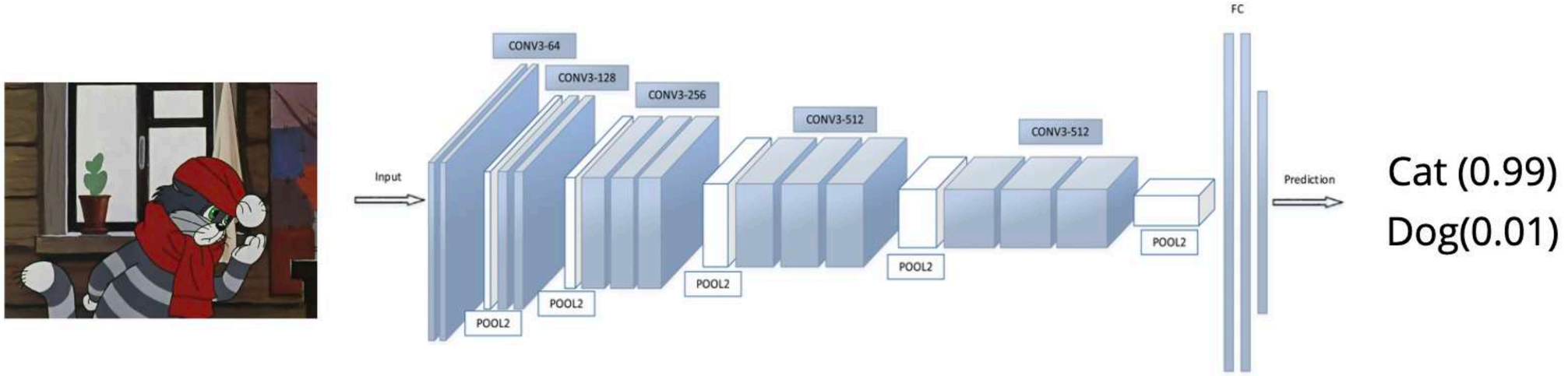


Anchor box 1: Anchor box 2:



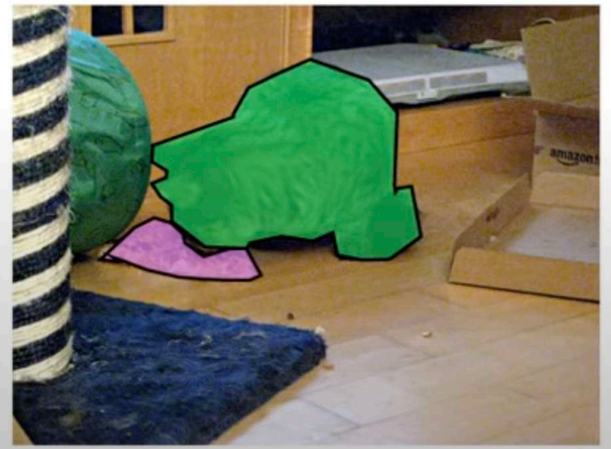
$$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \\ p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

Classification

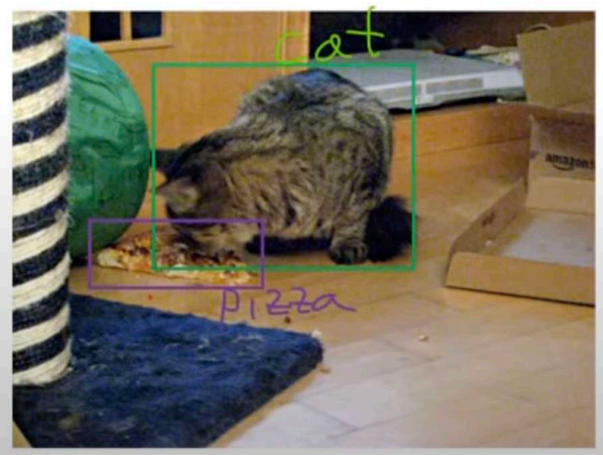


Другие задачи компьютерного зрения

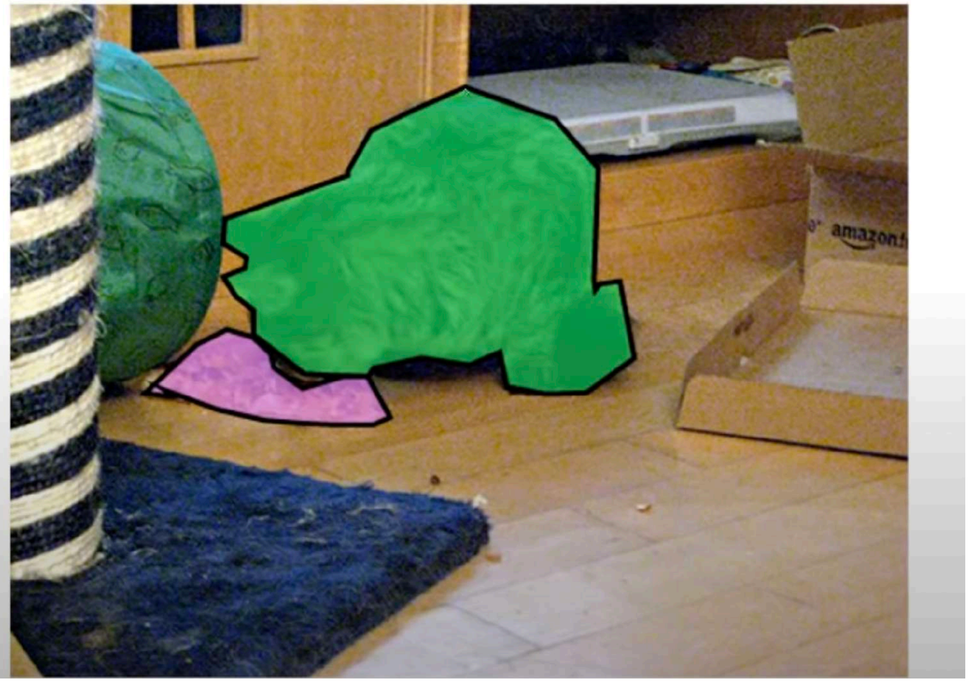
Семантическая сегментация
Semantic Segmentation



Обнаружение объектов
Object Detection

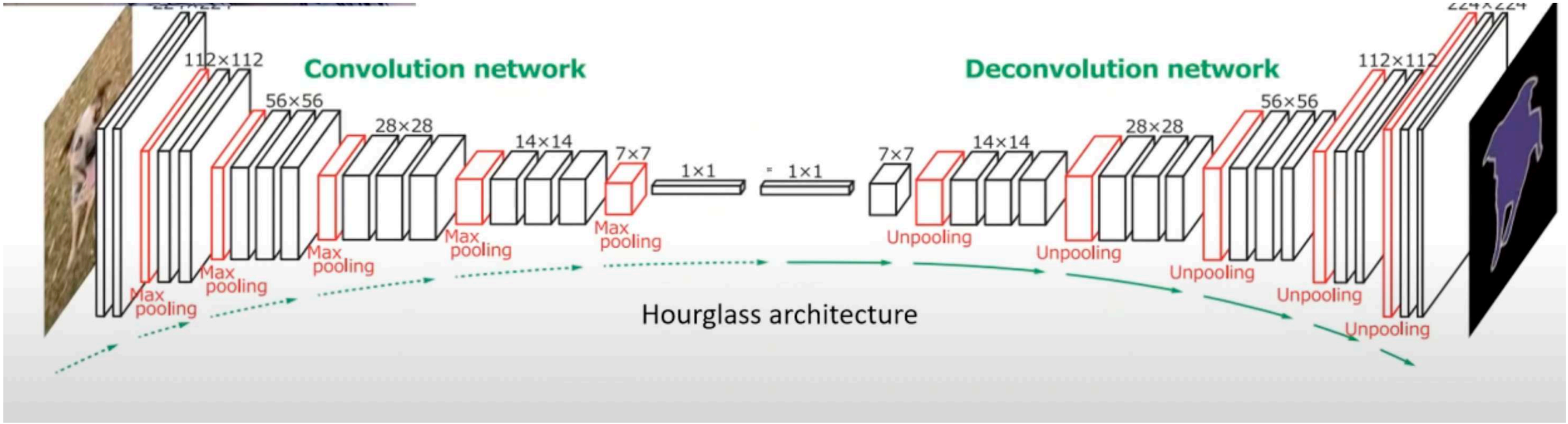


Семантическая сегментация
Semantic Segmentation

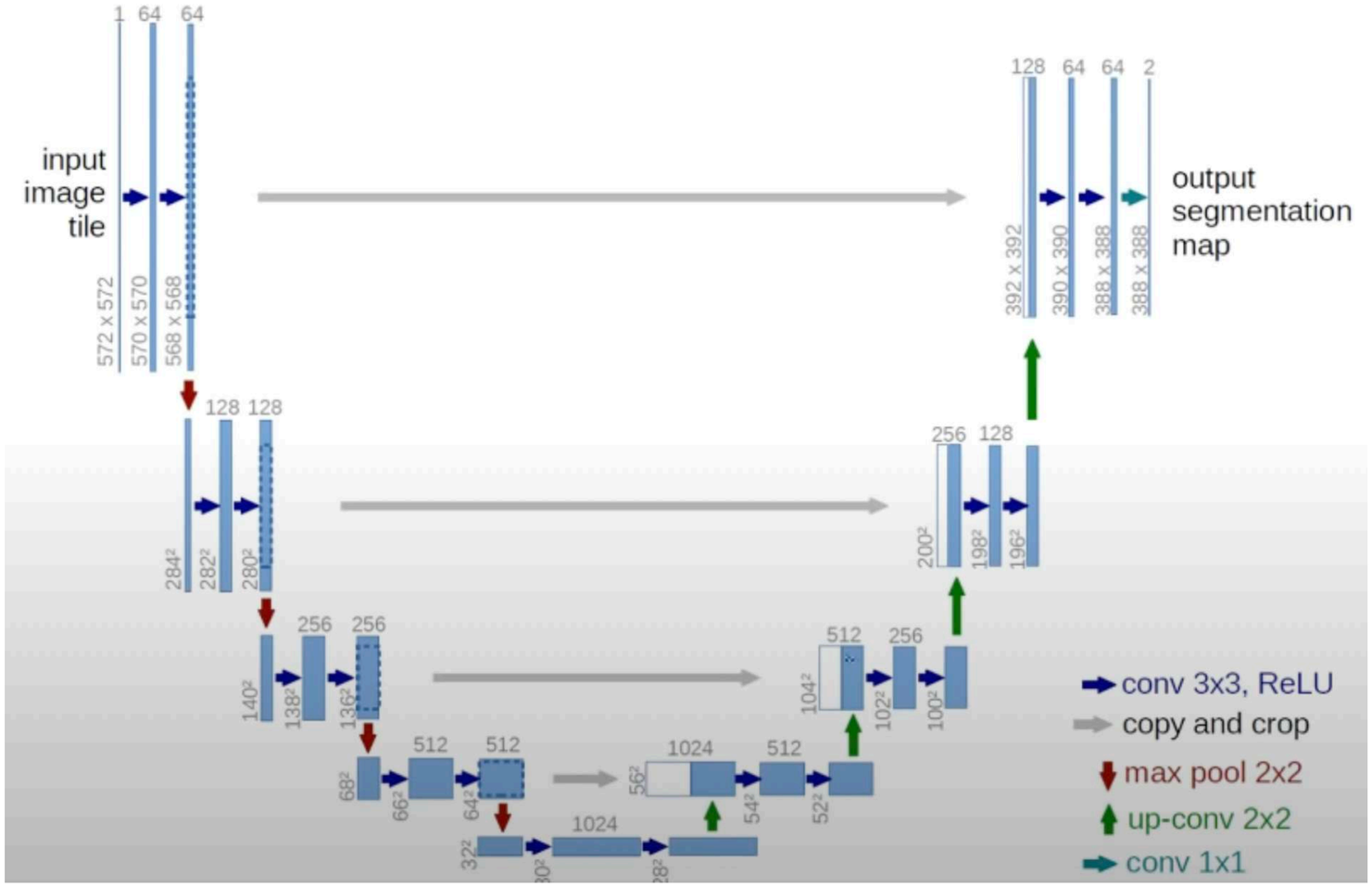


1. Сегментация = попиксельная классификация
2. Не требует большого объема тренировочных данных.
3. Все сегментационные сети - это архитектуры вида FCN.

Full convolution to the rescue!



U-net'15


input image tile
output segmentation map
conv 3x3, ReLU
copy and crop
max pool 2x2
up-conv 2x2
conv 1x1

Detection

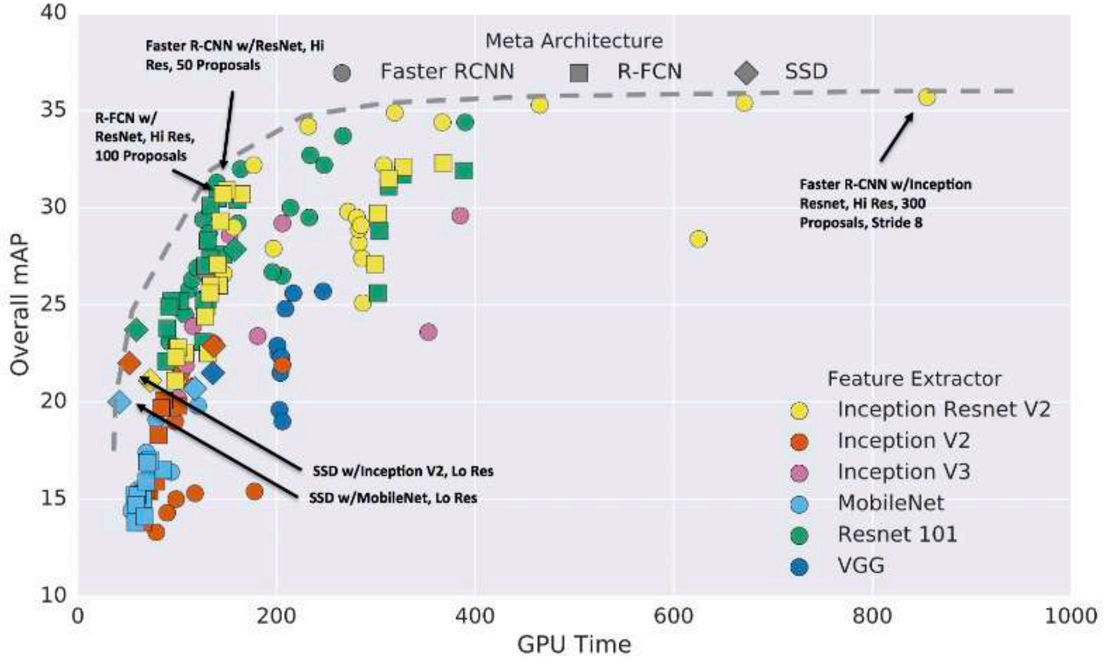
One-shot (быстрые)

YOLO, SSD, RetinaNet,

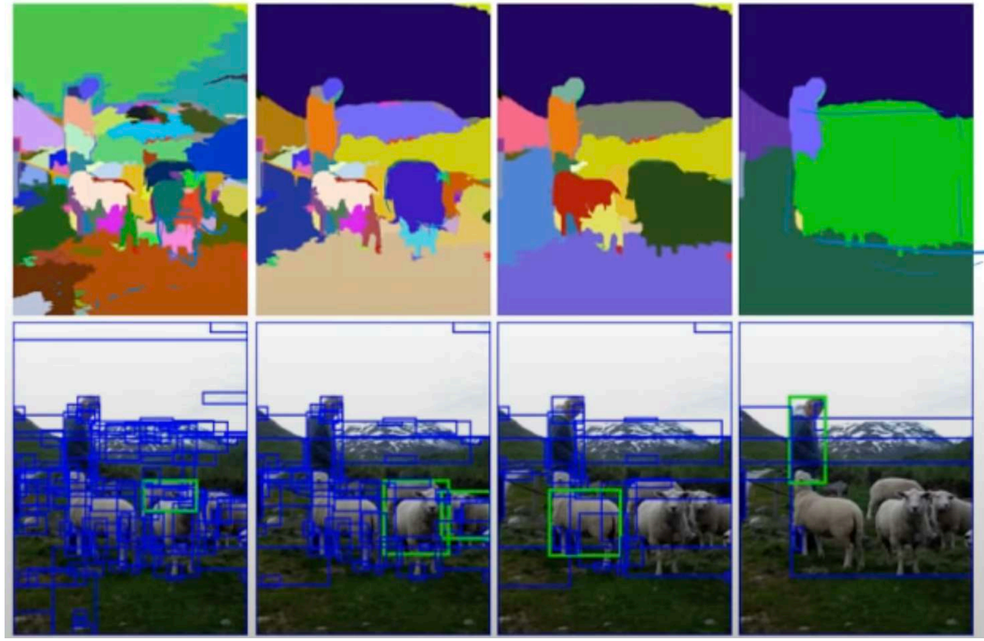
SqueezeNet, DetectNet

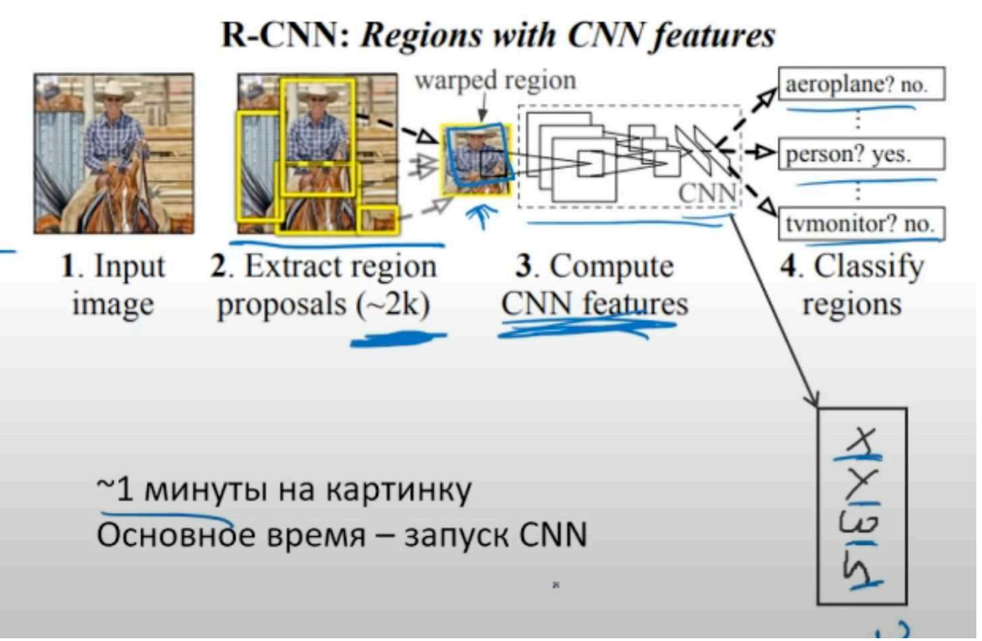
Two-shot (точные)

R-FCN, Fast RCNN, Faster-RCNN



R-CNN’13



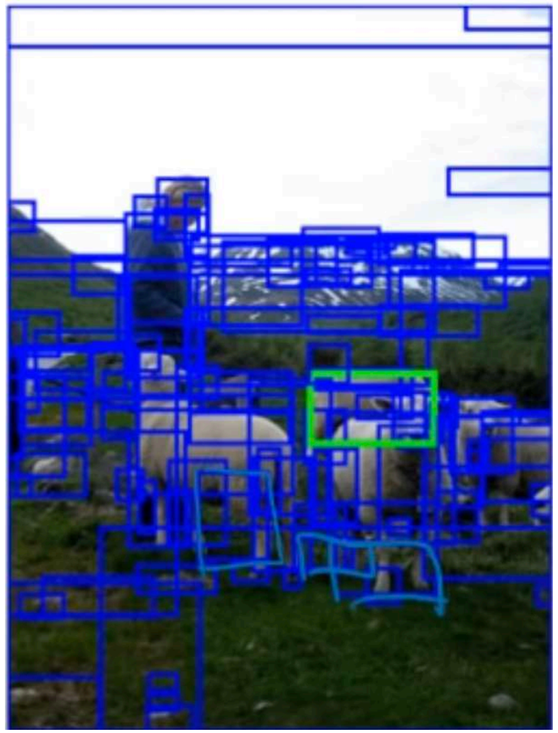


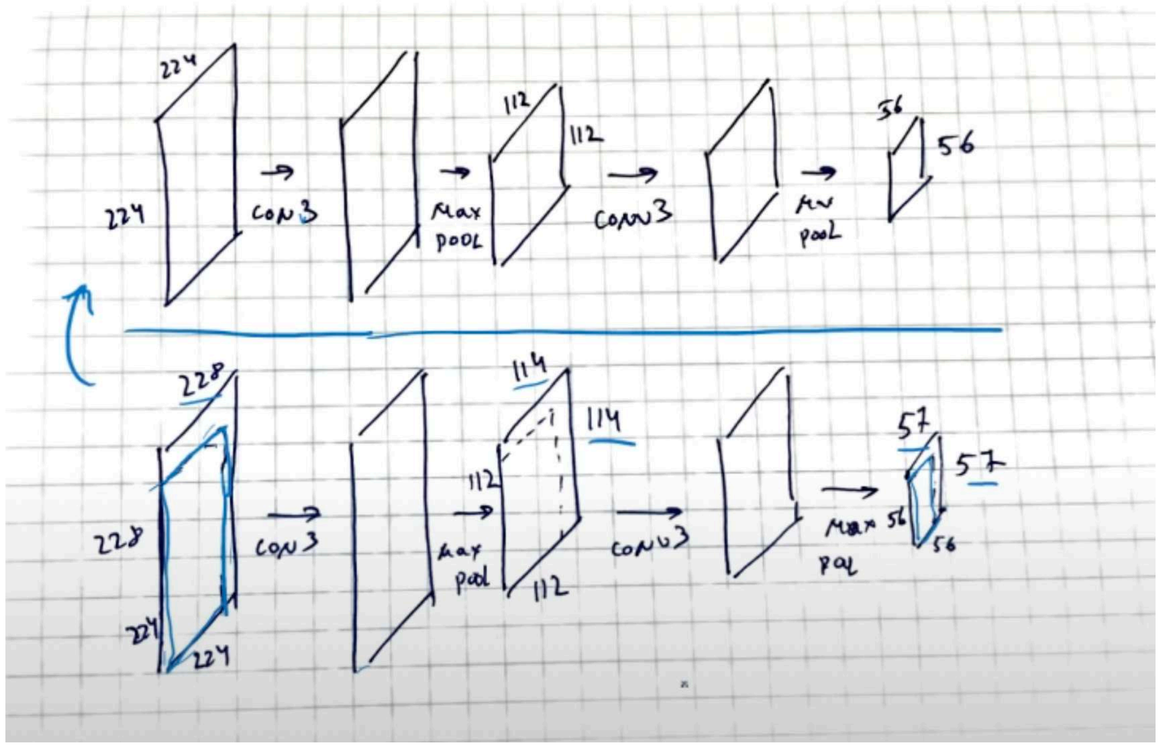
- Мы используем какой-то внешний алгоритм как генератор гипотезы объектов, с помощью него генерируем заданное количество гипотез, чтобы обеспечить достаточно высокую полноту. Затем каждую из этих гипотез мы вырезаем из изображения и подаем на вход классификатору. Метод генерации гипотез, который использовался в R-CNN (регионально сверточных сетях), - один из методов иерархической сегментации, то есть этот метод последовательно разбивает изображение на однородные сегменты.

Недостатки R-CNN :

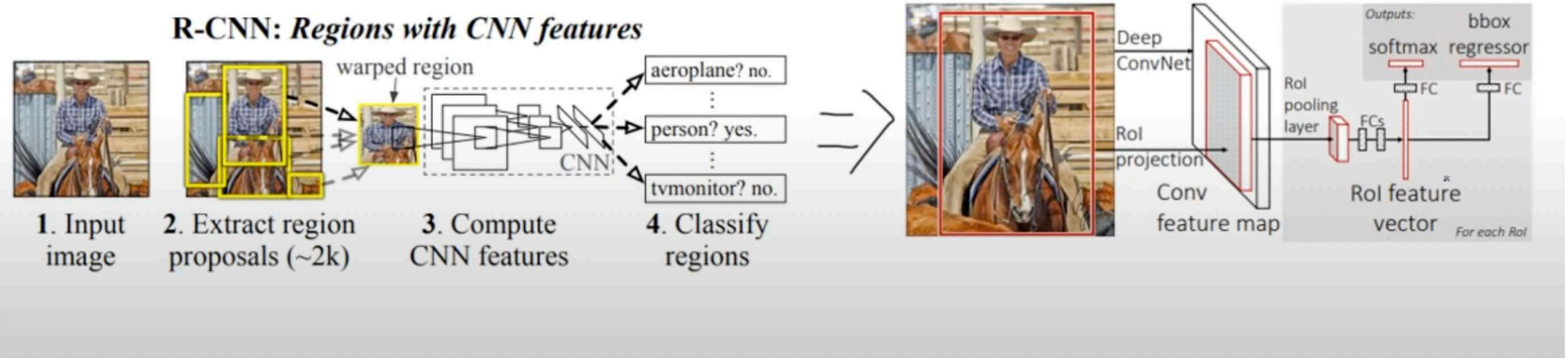
1. Вычисляем нейросетевые признаки независимо для каждого окна-гипотезы. Поскольку они пересекаются, это ведет к избыточным вычислениям
2. Нужно масштабировать фрагменты-гипотез до нужного разрешения
3. Сложная процедура обучения
4. Зависимость от внешнего алгоритма генерации гипотез

Full convolution



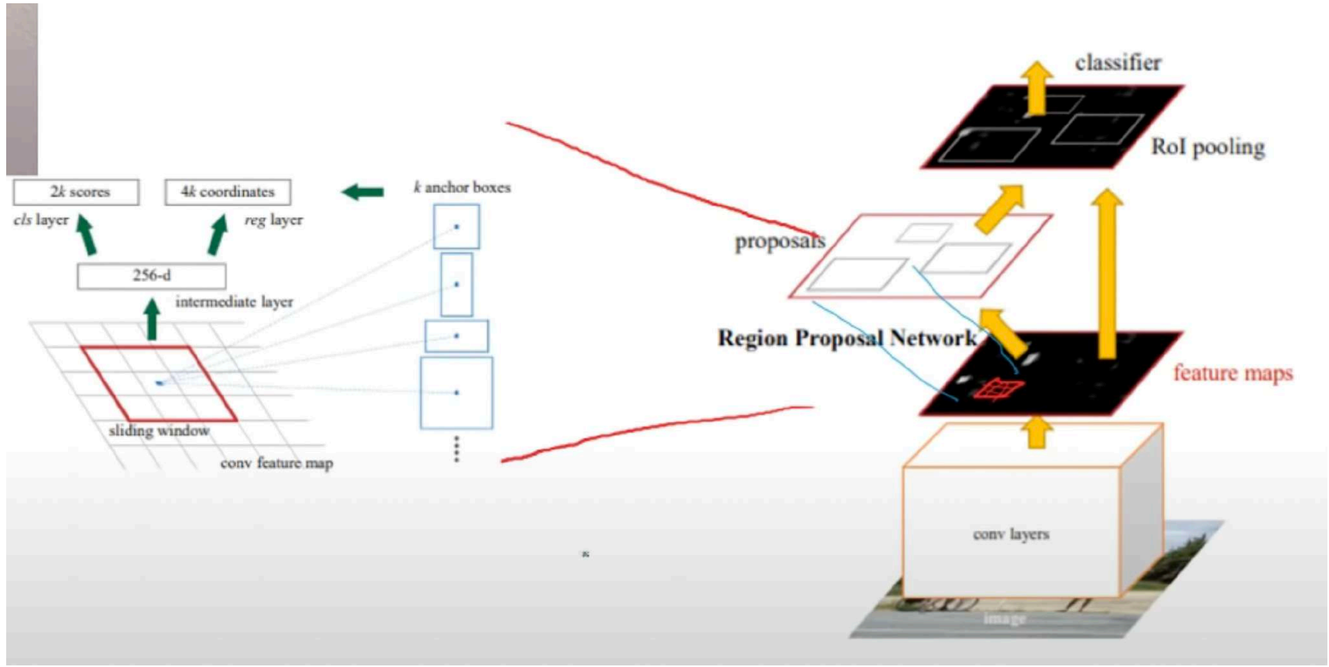


R-CNN vs Fast R-CNN



- ~3 секунды на картинку
- Основное время - поиск регионов

Faster R-CNN’15

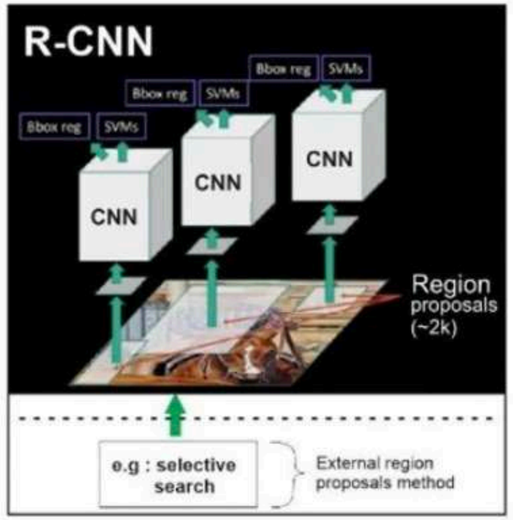


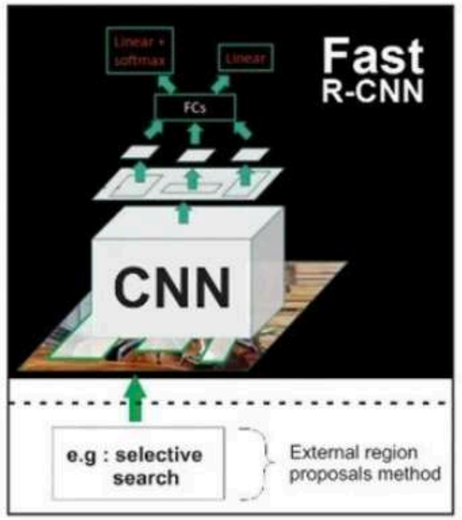
- Вычисляем proposals самой сетью

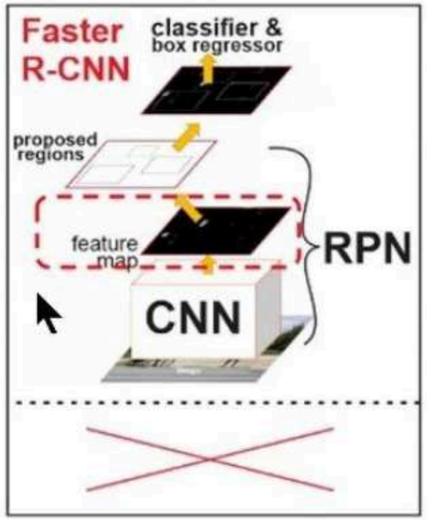
Faster R-CNN – пример нейросетевой архитектуры, которая используется до сих пор. По сути Faster R-CNN = Fast R-CNN + RPN (нейросетевого генератора гипотез). Нейросетевой генератор гипотез – маленькая нейросеть, которая по тем же самым сверточным признакам генерирует гипотезы. Наиболее вероятные гипотезы подаются на вход ROI-pooling слою и затем классифицируются более мощным классификатором.

- Маленькое скользящее окно по карте признаков (feature map)
- Маленькая нейросеть для
  - Классификации объект/не объект
  - Регрессии bbox
- Позиция окна показывает локализацию объекта относительно изображения
- Регрессия bbox показывает положение bbox относительно положения скользящего окна

Two shot: performance

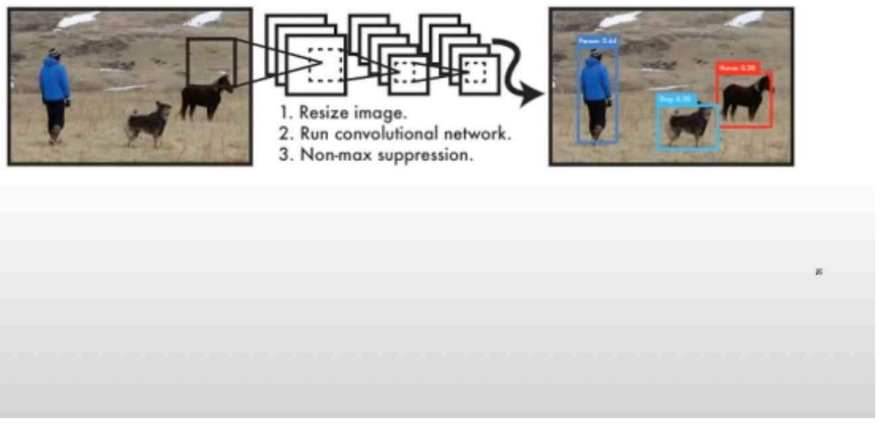


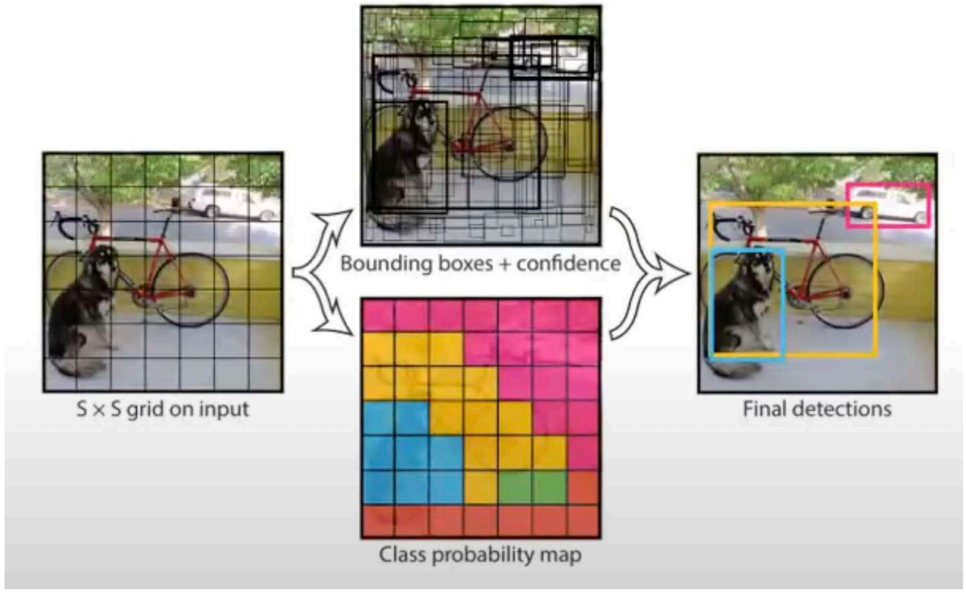




	R-CNN	Fast R-CNN	Faster R-CNN
Test time per image	50 seconds	2 seconds	0.2 seconds
Speed-up	1x	25x	250x
mAP (VOC 2007)	66.0%	66.9%	66.9%

YOLO (You Only Look Once)





YOLO

Несколько фактов о YOLO:

1. Самая быстрая архитектура - 170 рамок в секунду на изображении 256 на 256

2. Вышла в 2015 году, сейчас уже третья версия из 2018ого

3. Не самая точная, но быстрая за счет небольшой потери качества

4. От чего страдает YOLO - плохо работает с мелкими объектами. Как пример - Вам будет тяжело выделить отдельную птицу из стаи. С этим пытаются бороться, но тяжко. И на краях изображения тоже могут возникнуть проблемы

Резюме

- Скользящее окно позволяет свести задачу детекции к задаче классификации
- Для работы с разными масштабами/пропорциями рассматриваем гипотезы (окна) разных размеров и масштабов
- На один объект получаем множество откликов, поэтому приходится подавлять «слабые» (NMS)
- Считаем скользящие окна по картам признаков
- Для работы с разными размерами объектов лучше собирать признаки с разных слоев нейросети
- Ключевые методы – YOLO, Faster R-CNN