

Нейронные сети

Детекция объектов
на изображениях

Елизавета Лазарева



Не забывайте
отмечаться и
оставлять
отзыв



Содержание лекции

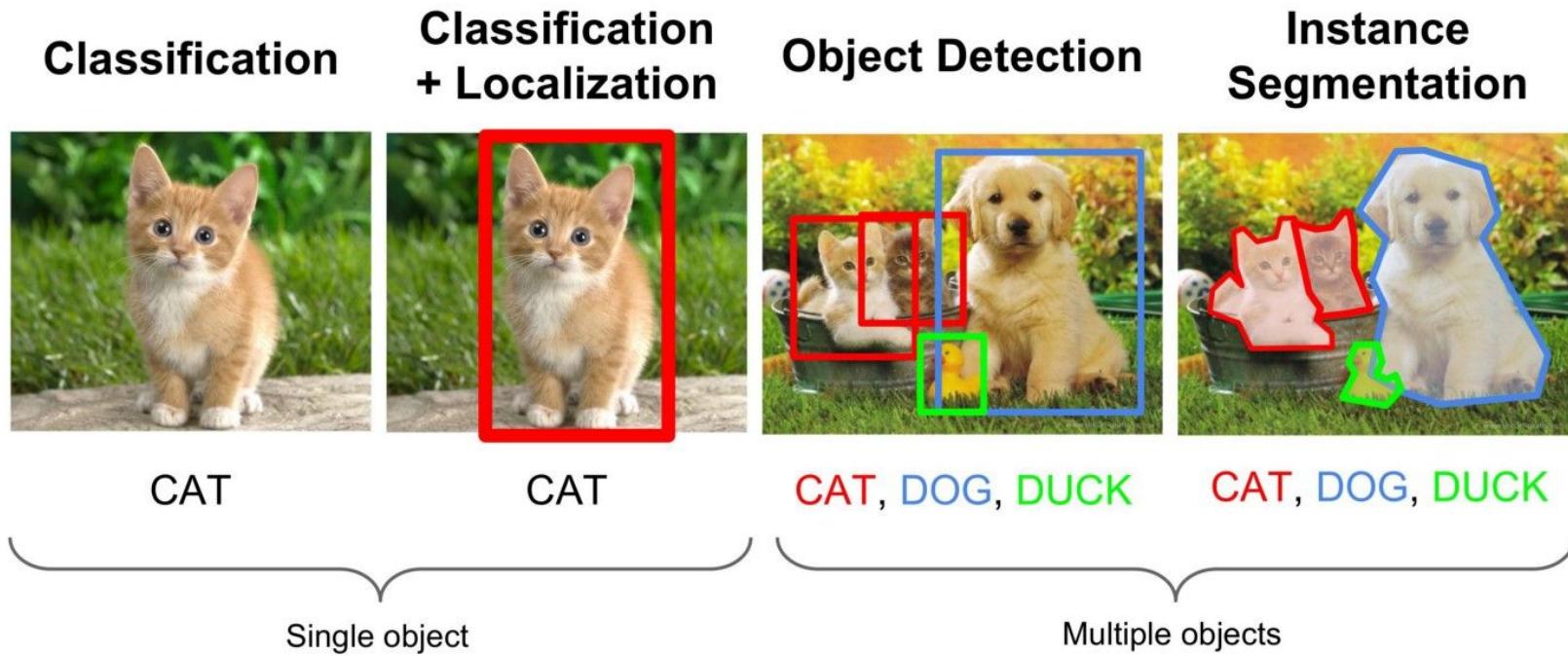
1. Задача детекции объектов на изображении
 - a. Проблемы
 - b. Метрики
2. Методы классического CV
3. Region-based детекция
 - a. RCNN
 - b. Fast RCNN
 - c. Faster RCNN
4. Single-shot детекция
 - a. SSD
 - b. YOLO



Задача детекции объектов на изображении

• • •

Задача детекции объектов на изображении

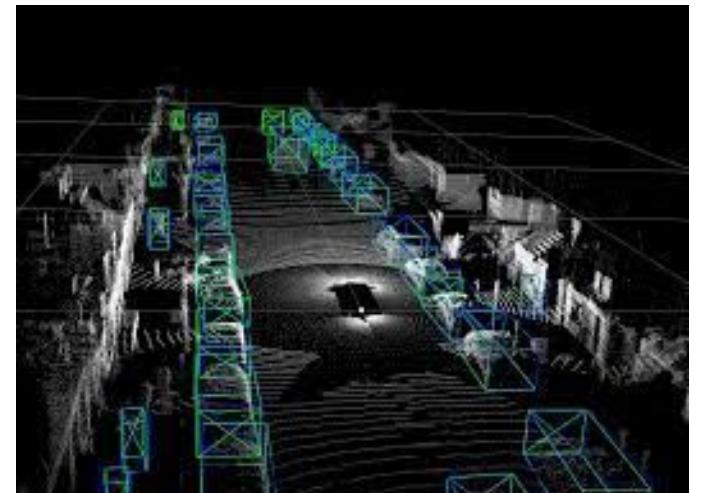
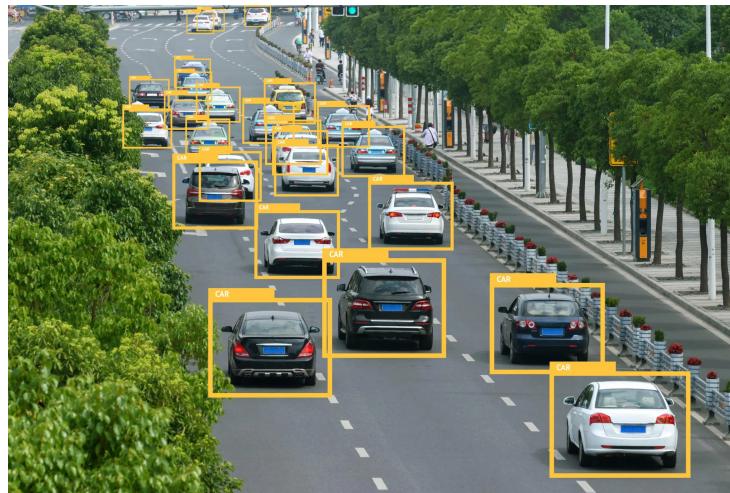
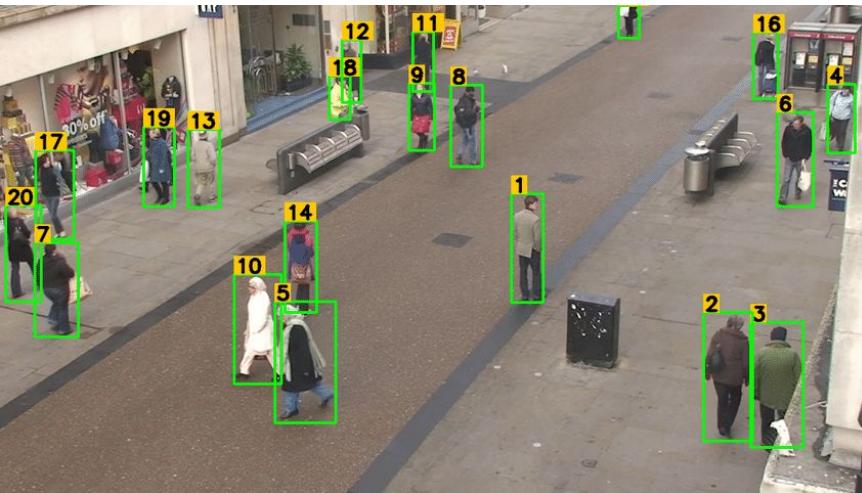
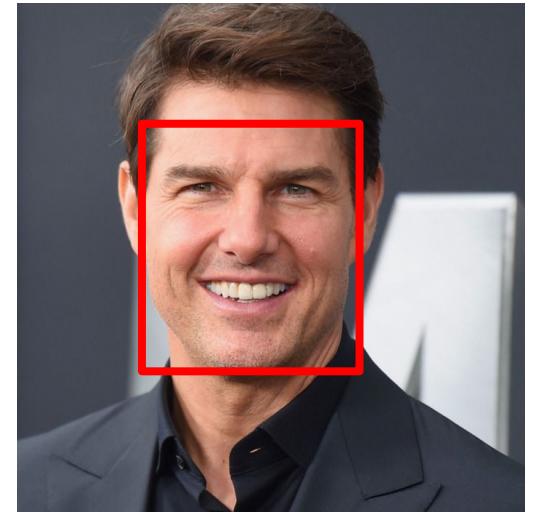


Отличия от задачи классификация:

1. Изображение может содержать несколько объектов разных классов класса, в том числе могут отсутствовать вообще;
2. На изображении может находиться несколько объектов одного и того же класса;
3. Требуется локализовать объект с помощью прямоугольной зоны (bounding box).

Примеры детекции объектов на изображении

1. Детекция лиц:
 - a. Автофокус камеры;
 - b. Распознавание лиц.
2. Беспилотные автомобили/роботы/дроны.
3. Обнаружение автотранспорта
4. Обнаружение пешеходов



Проблемы детекции объектов на изображении



(a) Illumination



(b) Deformation



(c) Scale, Viewpoint



(d) Pose, Occlusion



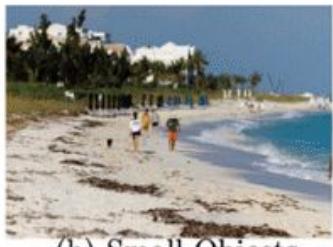
(e) Clutter, Occlusion



(f) Blur



(g) Motion



(h) Small Objects, Low Resolution



(i) Different instances of the “chair” category

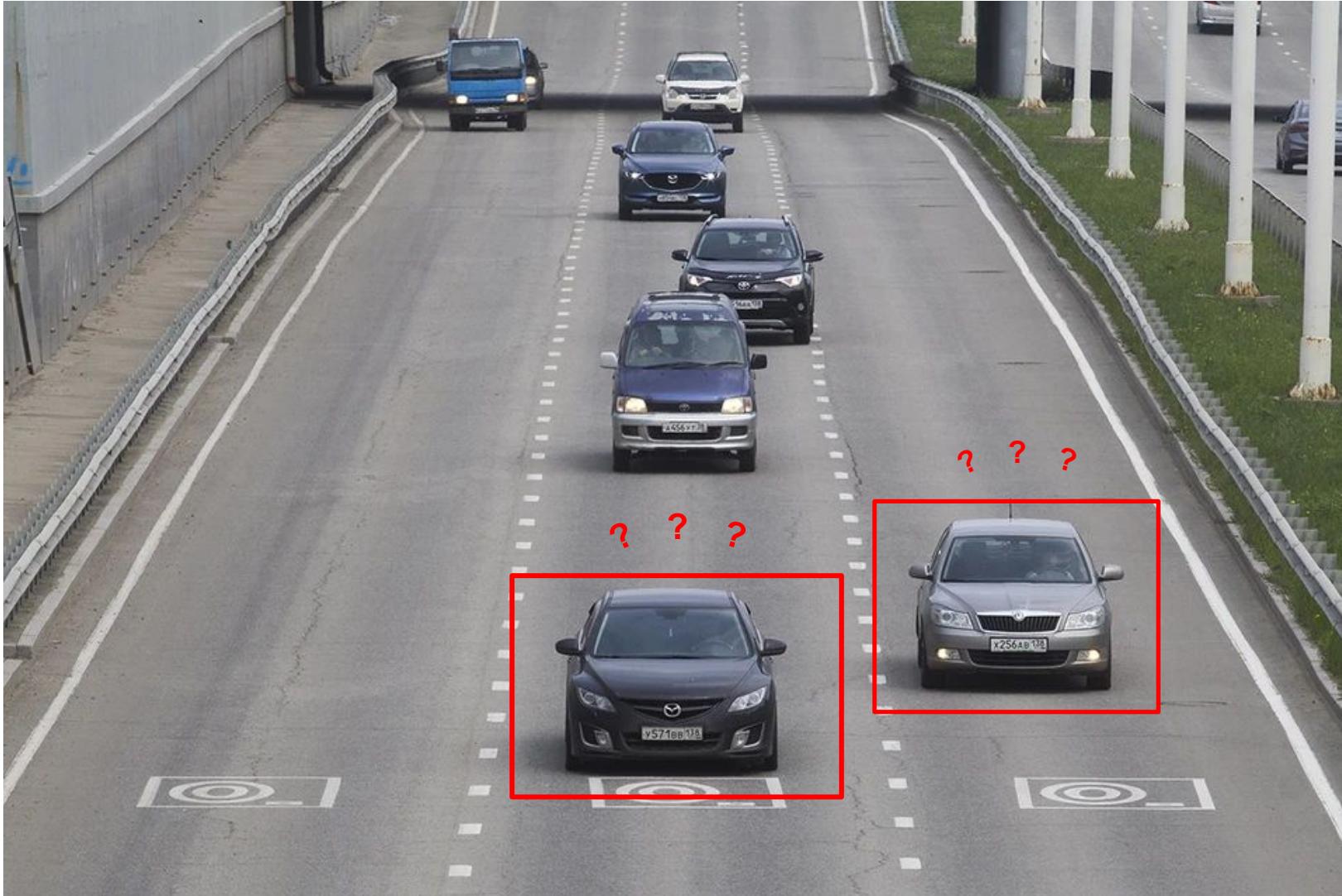


(j) Small Interclass Variations: four different categories

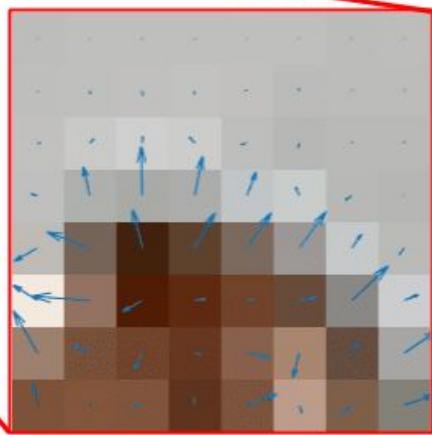
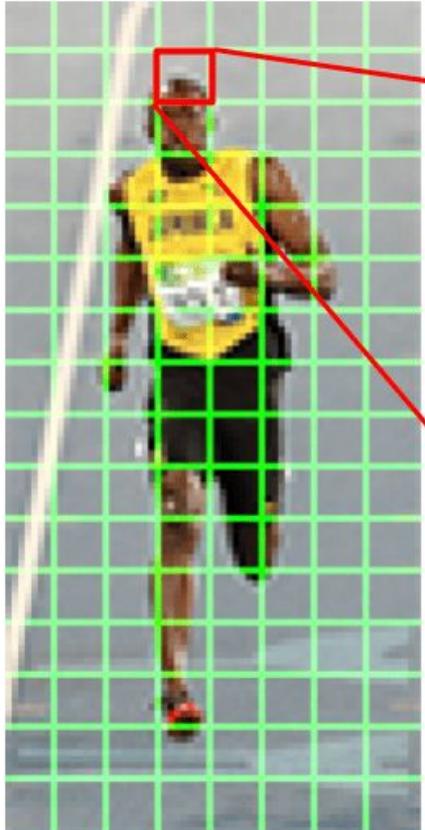
Методы классического компьютерного зрения

• • •

Методы классического CV - что можно придумать?



Histogram Oriented Gradients

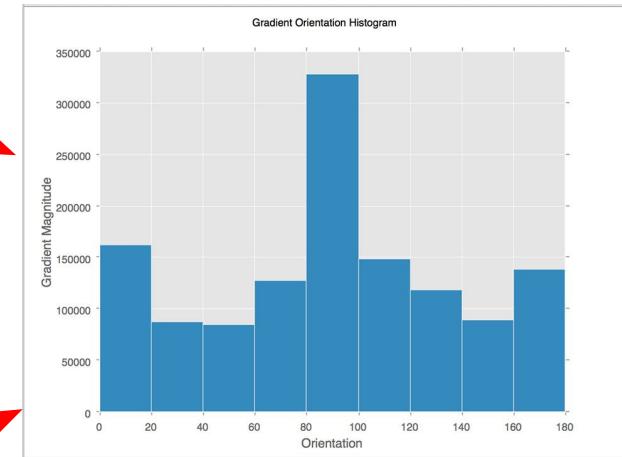


2	3	4	4	3	4	2	2
5	11	17	13	7	9	3	4
11	21	23	27	22	17	4	6
23	99	165	135	85	32	26	2
91	155	133	136	144	152	57	28
98	196	76	38	26	60	170	51
165	60	60	27	77	85	43	136
71	13	34	23	108	27	48	110

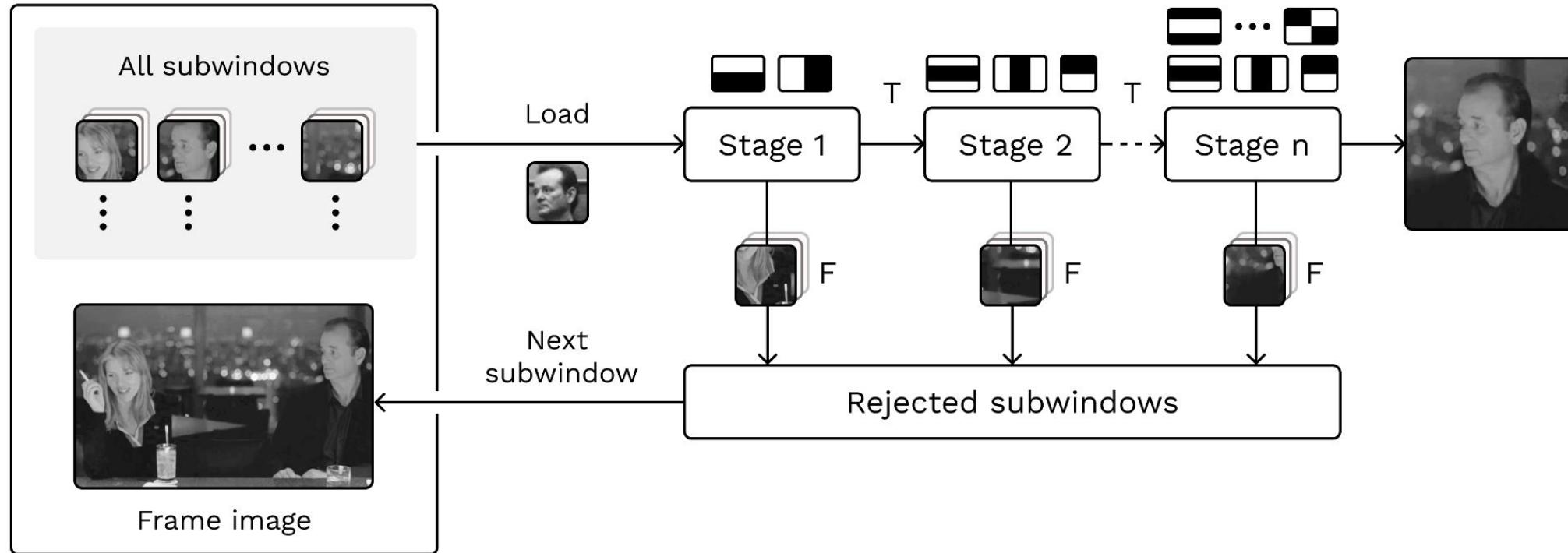
Gradient Magnitude

80	36	5	10	0	64	90	73
37	9	9	179	78	27	169	166
87	136	173	39	102	163	152	176
76	13	1	168	159	22	125	143
120	70	14	150	145	144	145	143
58	86	119	98	100	101	133	113
30	65	157	75	78	165	145	124
11	170	91	4	110	17	133	110

Gradient Direction

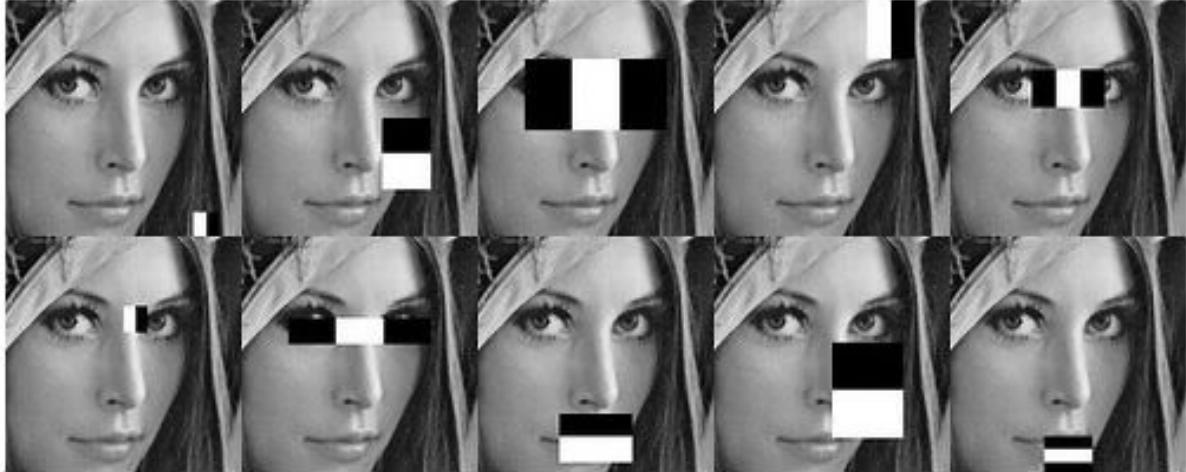


Haar Cascades



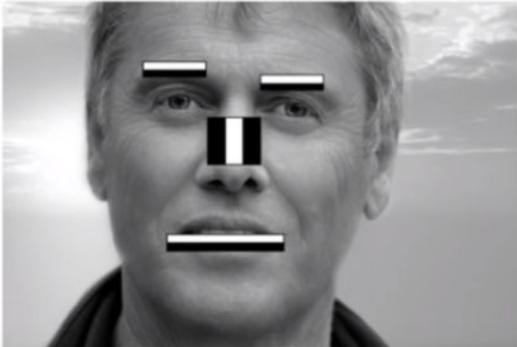
1. Изображение разбиваем на окна и считаем признаки Хаара.
2. Окно проходит через каскад классификаторов.
3. Только окна, прошедшие все стадии, считаются лицами.

Haar Cascades



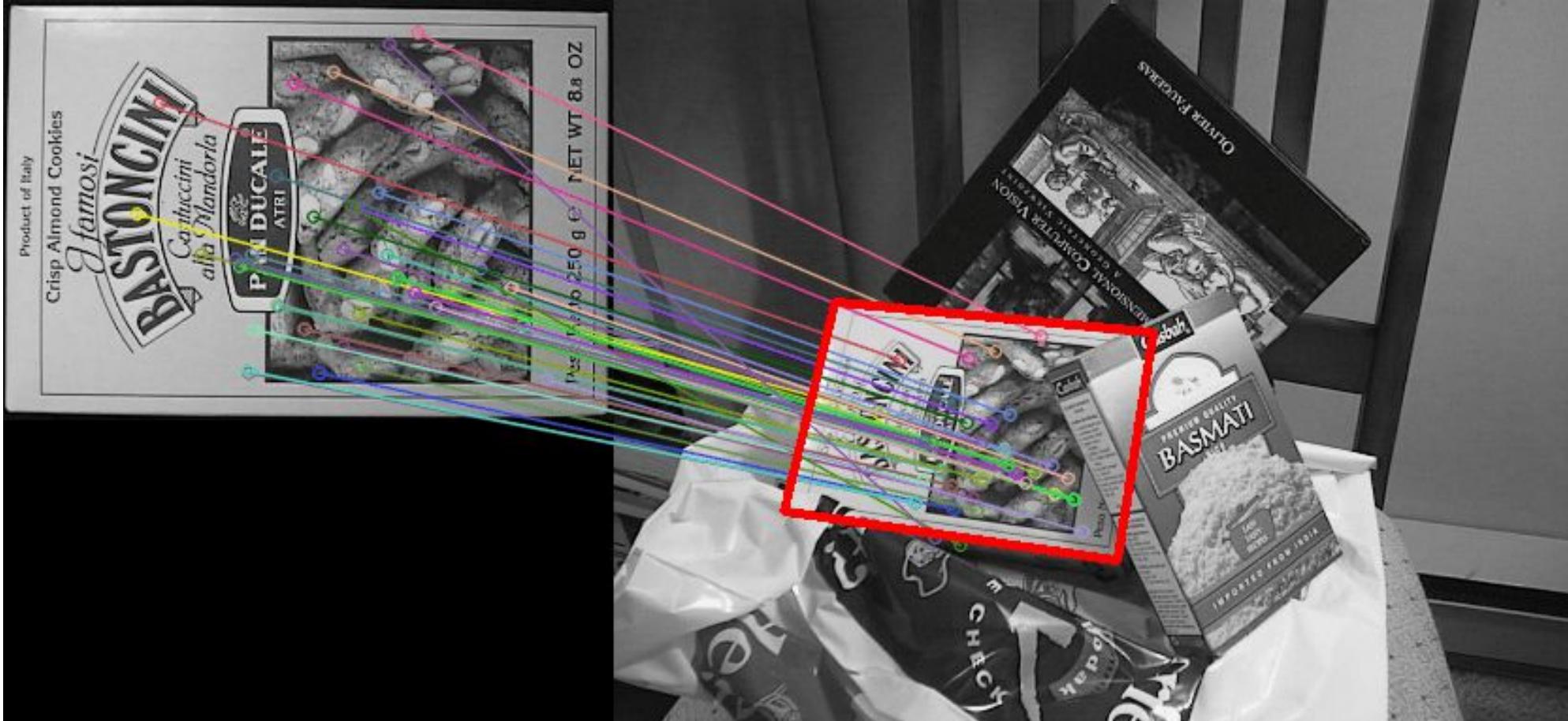
Признак Хаара – разность яркости между чёрной и белой областями фильтра.

Вычисляем значения признаков на каждом участке изображения.



WE CAN REPRESENT THE MOST RELEVANT FEATURES WITH HAAR-FEATURES !!!

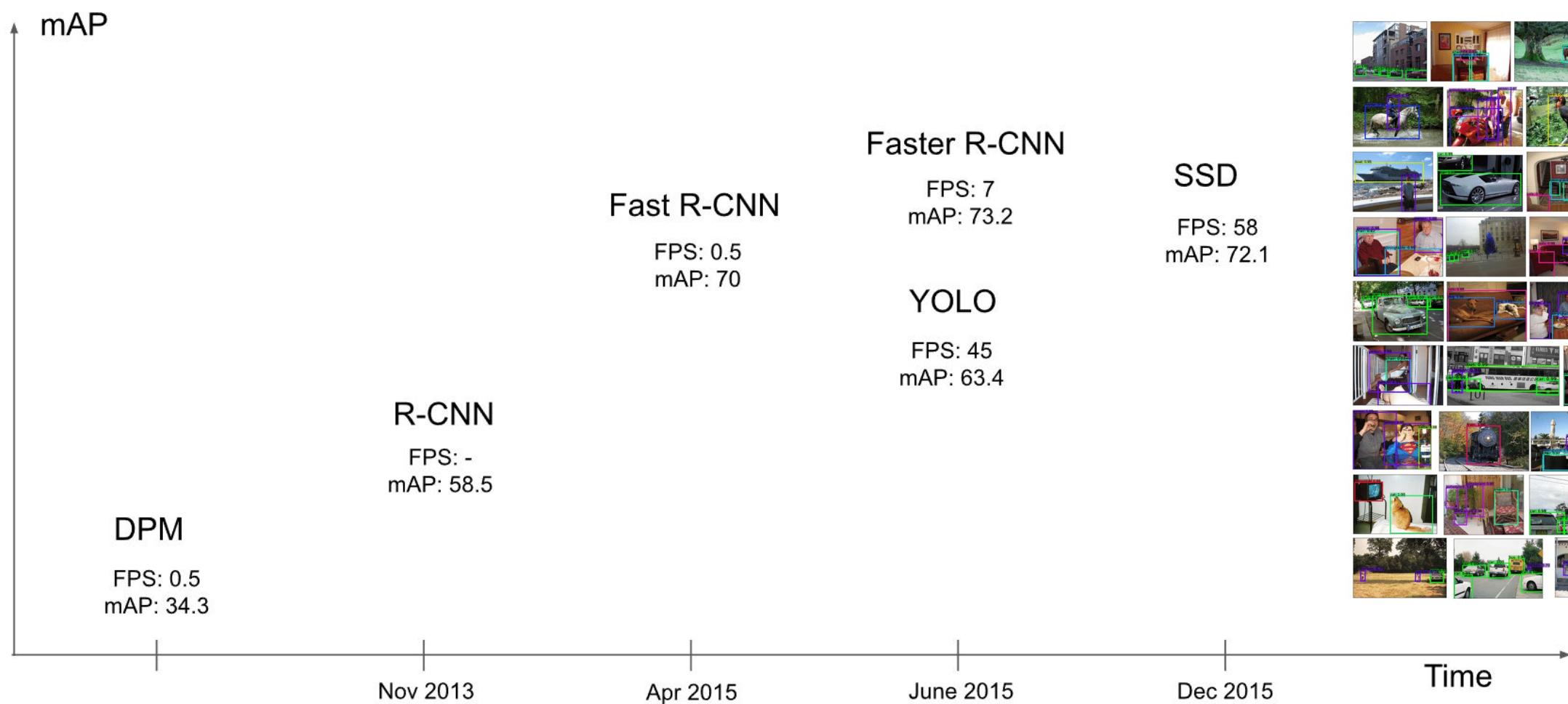
Scale-Invariant Feature Transform (SIFT)



Region-based детекция



Развитие подходов



<https://paperswithcode.com/sota/object-detection-on-coco>

<https://cocodataset.org/#home>

<http://host.robots.ox.ac.uk/pascal/VOC/voc2007/>

Подходы на основе нейронных сетей

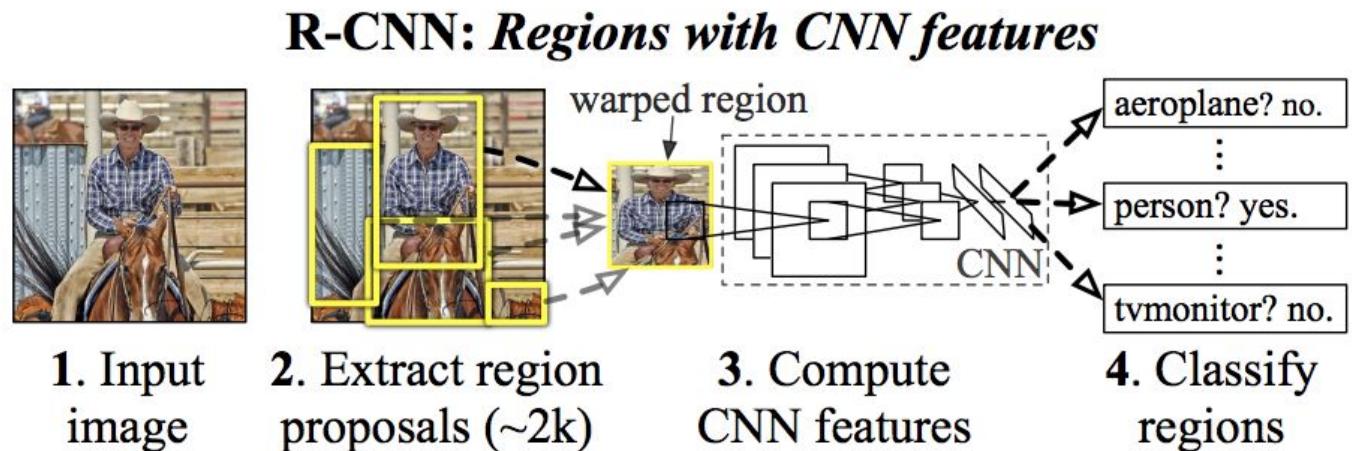
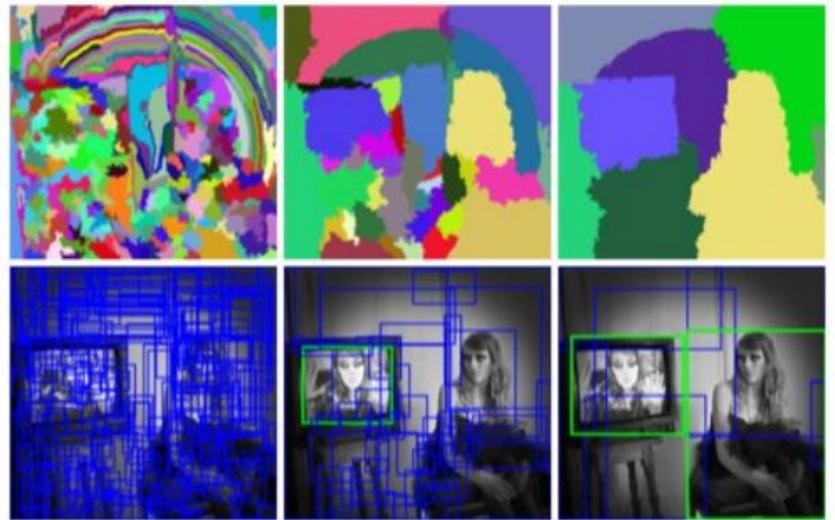
1. Region-based
 - a. RCNN
 - b. Fast RCNN
 - c. Faster RCNN
2. One-shot:
 - a. SSD
 - b. YOLO
3. Cascaded Detectors:
 - a. MTCNN
 - b. ...
4. Transformer-based
 - a. DETR
 - b. ...

Умеем решать задачу классификации с помощью CNN. Как решить задачу детекции?



RCNN (Regions with CNN features)

1. Генерируем region proposals (Selective Search)
2. Выбираем каждый регион, пропускаем через CNN, обученную для классификации изображений.
3. Выбираем регионы, прошедшие по порогу, и применяем Non Maximum Suppression (NMS)



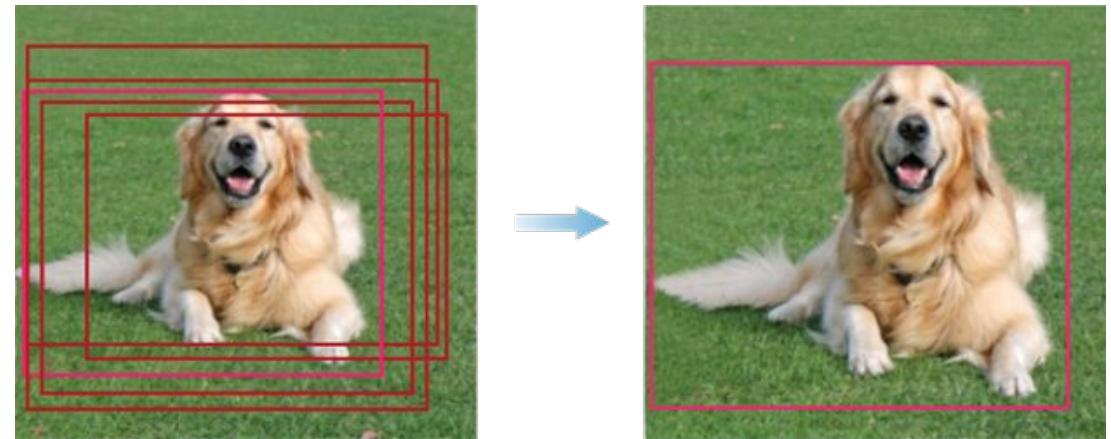
<https://arxiv.org/pdf/1311.2524v5.pdf>

Non Maximum Suppression (NMS), подавления немаксимумов

Проблема: selective search создает много пересекающихся регионов для одного объекта. NMS отбрасывает bounding box-ы с низким значением detection score при их пересечении с другими регионами, имеющими более высокий score.

NMS алгоритм:

1. Отсортируем box-ы по detection score в порядке убывания.
Сохраним их в массив L.
2. Проходим по L:
 - a. Берем регион.
 - b. Считаем его IoU со всеми остальными box-ами из L.
 - c. Удаляем регионы с IoU превосходящим порог.



RCNN детали реализации

Обучение CNN для классификации:

- a. Используем предобученную классификационную CNN сеть (авторы оригинальной статьи брали AlexNet, обученную на ImageNet);
- b. Заменяем последний классификационный слой и добавляем дополнительный класс “background”;
- c. Создаём датасет с кропами объектов;
- d. Переобучаем сеть.

Достоинства и недостатки RCNN

Плюсы:

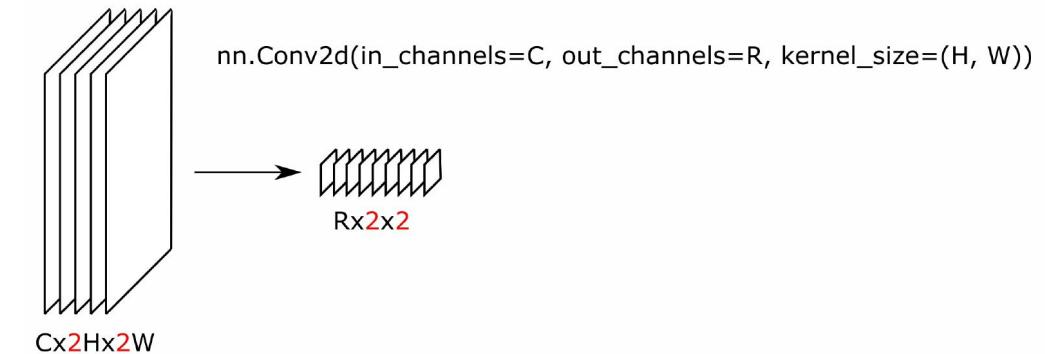
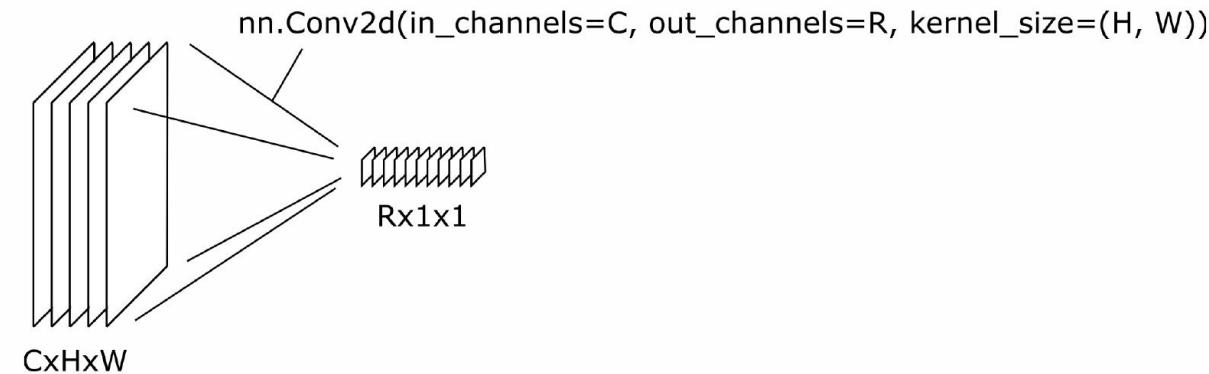
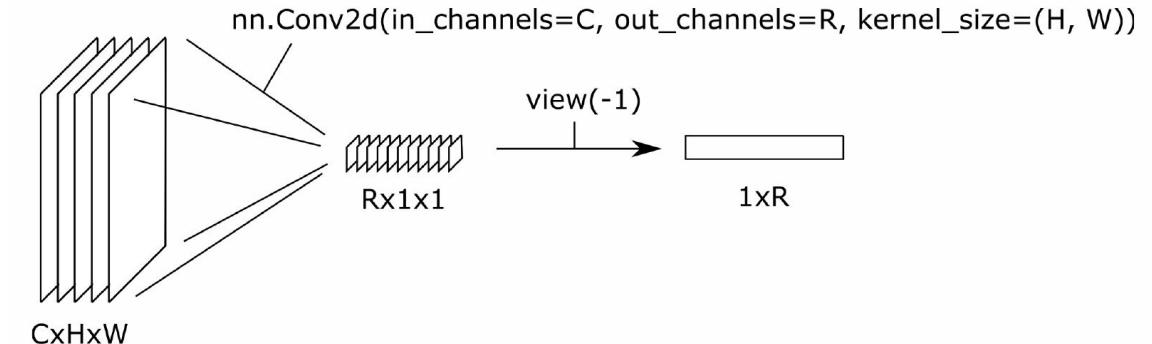
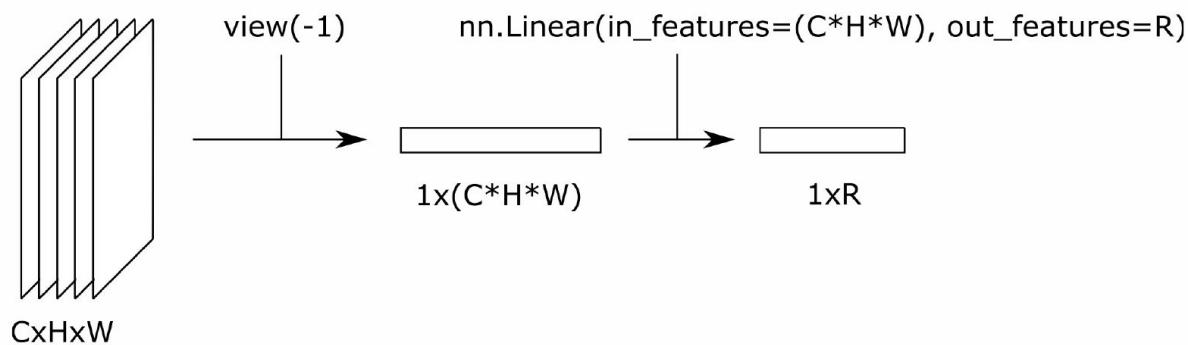
1. Первый “хороший” детектор (53.7% mAP on PASCAL VOC 2010).

Минусы:

1. Крайне медленный: 13s/изображение на GPU or 53s/изображение на CPU.
~2000 region proposals. Дублирование расчетов в случае совпадающих region proposals.
2. Модель масштабирует изображение до определенного стандартного размера, может теряться информация о размере объекта / соотношении сторон.
3. Сложный процесс обучения.
4. Несколько стадий детекции усложняют работу с моделью.

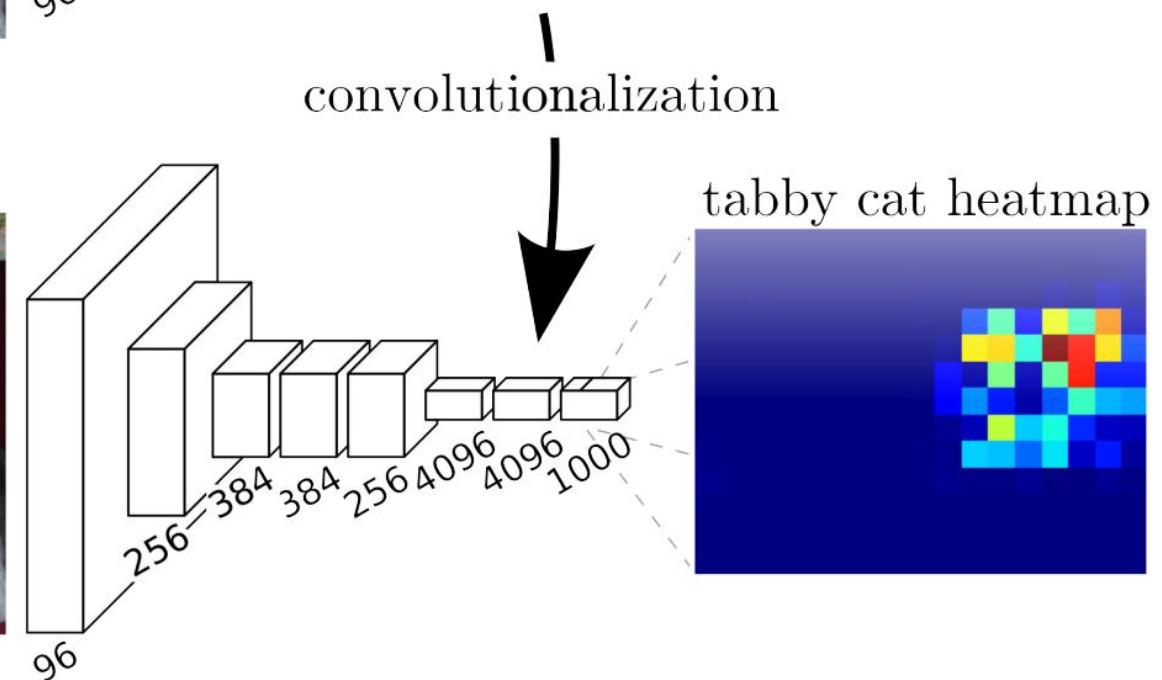
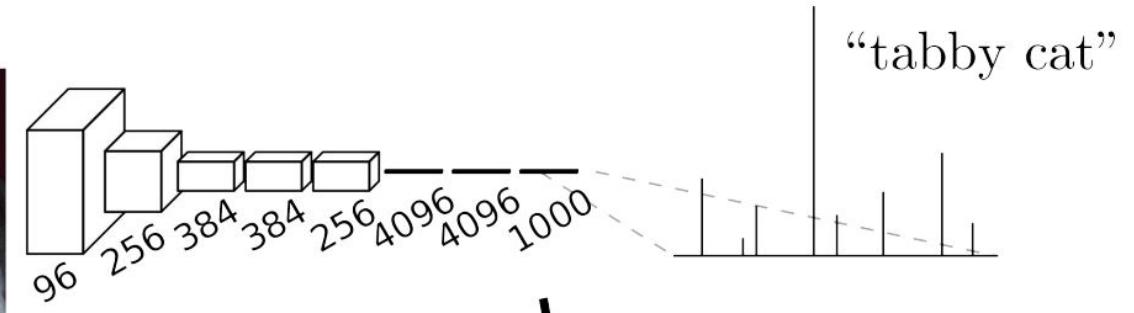
Как можно оптимизировать?

Имитация линейного слоя с помощью Conv2d?

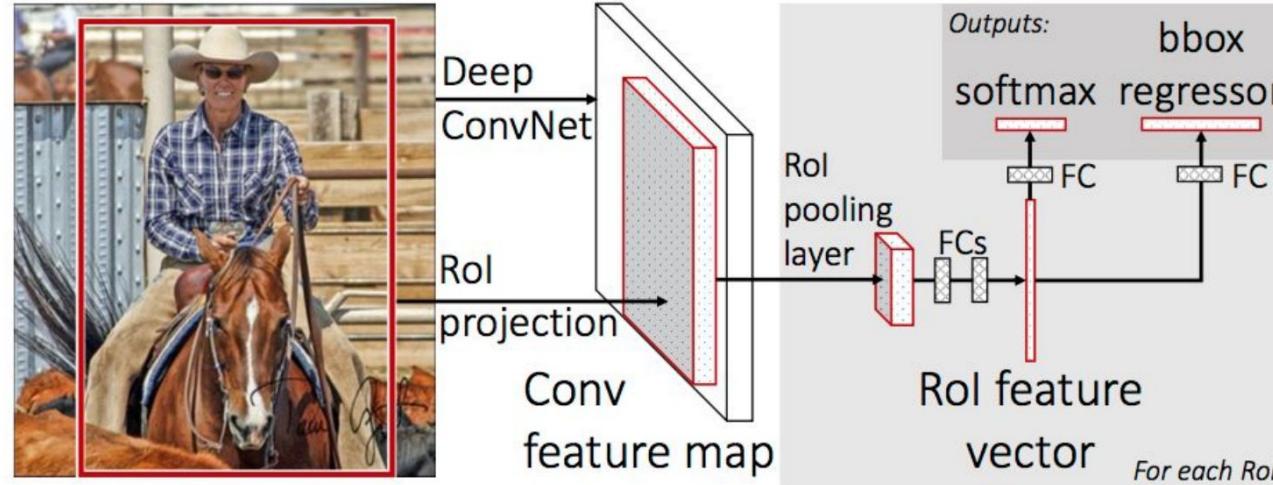


Fully Convolutional Network (FCN)

1. Свертка с фиксированным фильтром может быть применена к тензору произвольной высоты и ширины.
2. Только fully connected слои не позволяют применять CNN к изображениям разного размера.



Fast RCNN

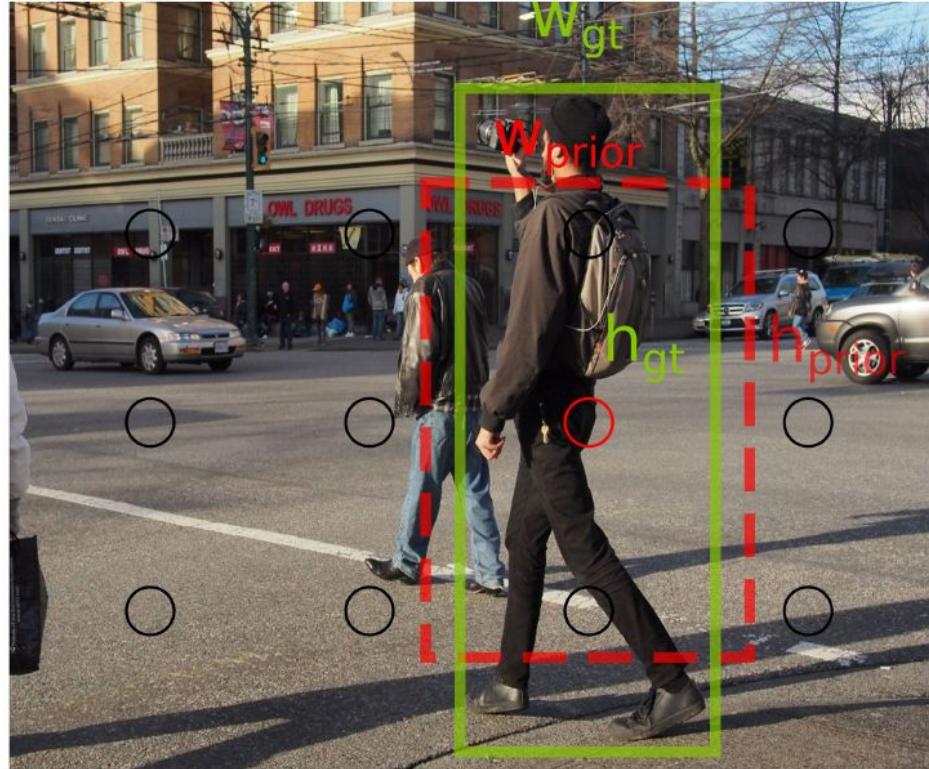
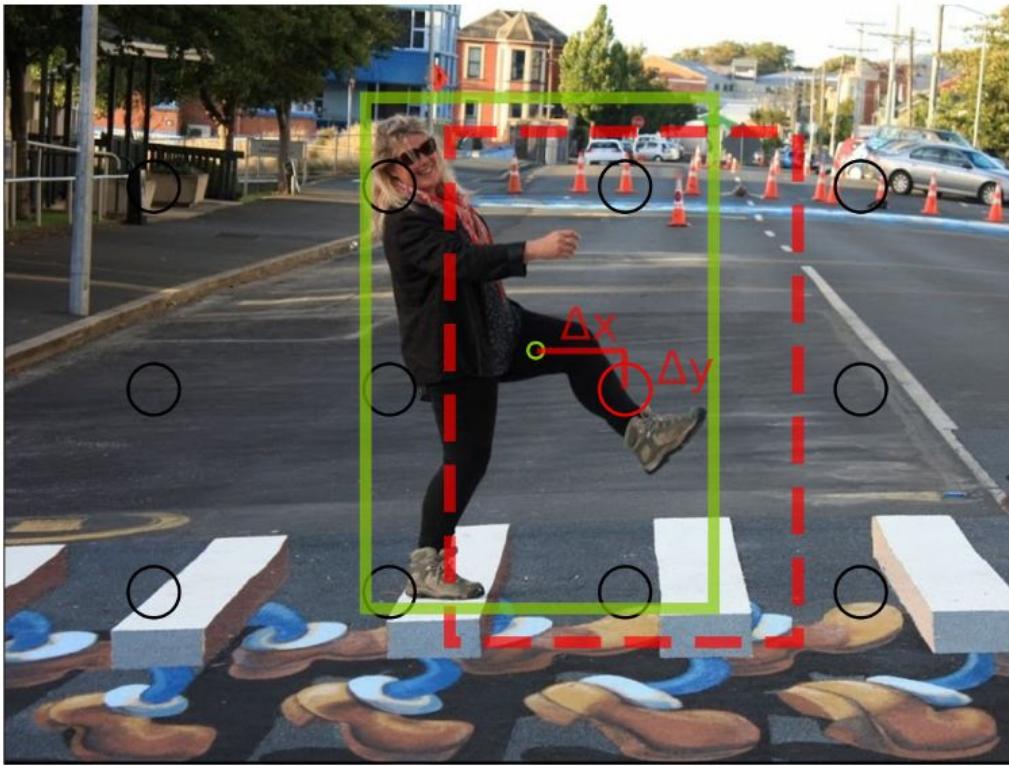


input	0.88	0.44	0.14	0.16	0.37	0.77	0.96	0.27
	0.19	0.45	0.57	0.16	0.63	0.29	0.71	0.70
	0.66	0.26	0.82	0.64	0.54	0.73	0.59	0.26
	0.85	0.34	0.76	0.84	0.29	0.75	0.62	0.25
	0.32	0.74	0.21	0.39	0.34	0.03	0.33	0.48
	0.20	0.14	0.16	0.13	0.73	0.65	0.96	0.32
	0.19	0.69	0.09	0.86	0.88	0.07	0.01	0.48
	0.83	0.24	0.97	0.04	0.24	0.35	0.50	0.91

1. Генерируем region proposal-ы с помощью selective search.
2. Передаем изображение на вход **fully convolutional network**, получаем карты признаков с размерами $C \times W \times H$.
3. Для каждого region proposal:
 - a. Вырезаем его из карты признаков и применяем Region of Interest Pooling. Эта операция превращает участок произвольного размера (h, w, c) в область фиксированных размеров (например, $7 \times 7 \times c$).
 - b. Полученный тензор передается в fully connected слои, в них **предсказывается класс объекта и поправки координат bounding box-а** ($\Delta x, \Delta y, k_h, k_w$).
 - c. Поправки координат применяются к исходному region proposal.
4. После используется алгоритм Non Maximum Suppression.

<https://arxiv.org/pdf/1504.08083.pdf>

BBox Regression



BBox regression vector: $(\Delta x, \Delta y, k_h, k_w)$.

$$k_h = h_{gt} / h_{prior}$$
$$k_w = w_{gt} / w_{prior}$$

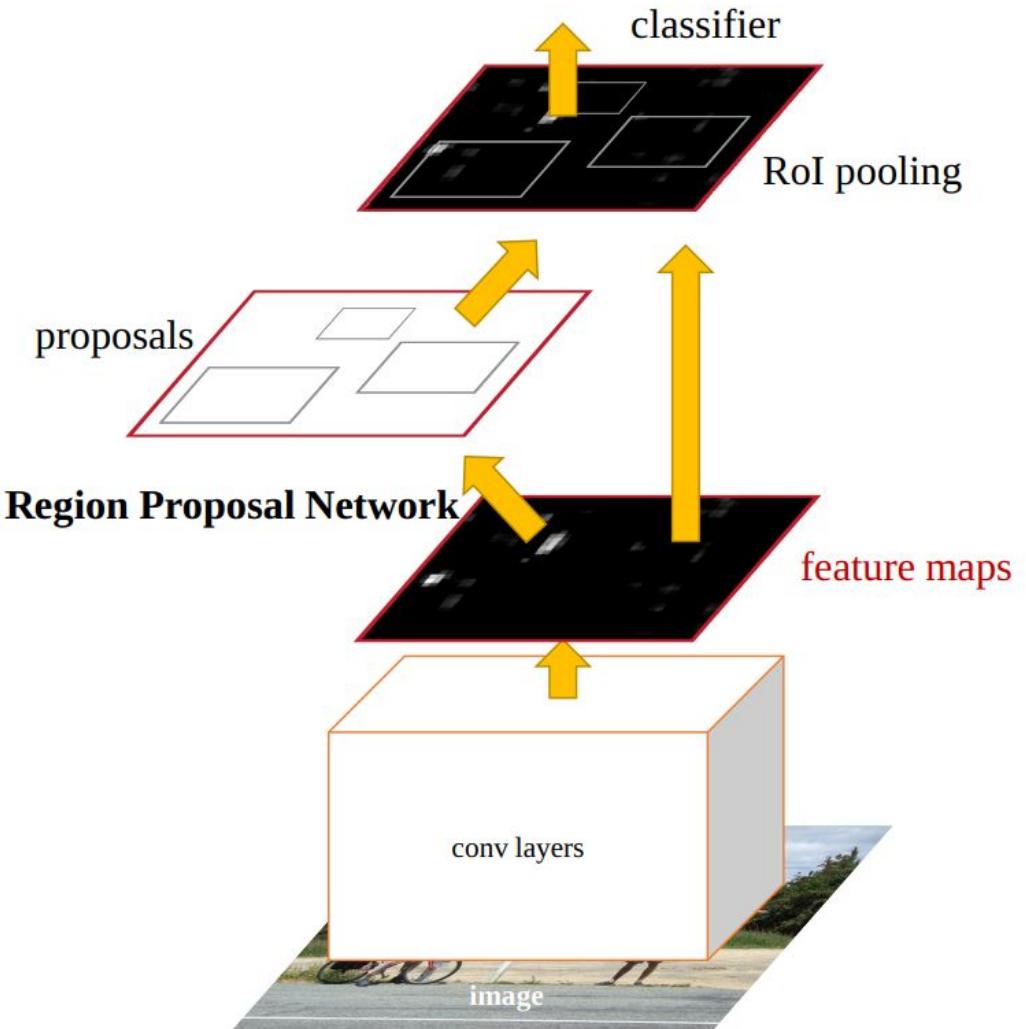
Fast RCNN детали реализации

1. RoI Pooling слой дифференцируем, поэтому Fast RCNN может быть обучена end-to-end.
2. Обе функции потерь для классификации и регрессии bounding box-ов оптимизируются совместно.
3. Сверточная бэкбон сеть применяется к изображению один раз, но fully connected слои применяются к каждому pooled region proposal => авторы применили SVD разложение на матрицах весов fully connected слоев => 30% speedup.
4. VGG16 вместо AlexNet.

Главный недостаток Fast RCNN: техника генерации region proposal.

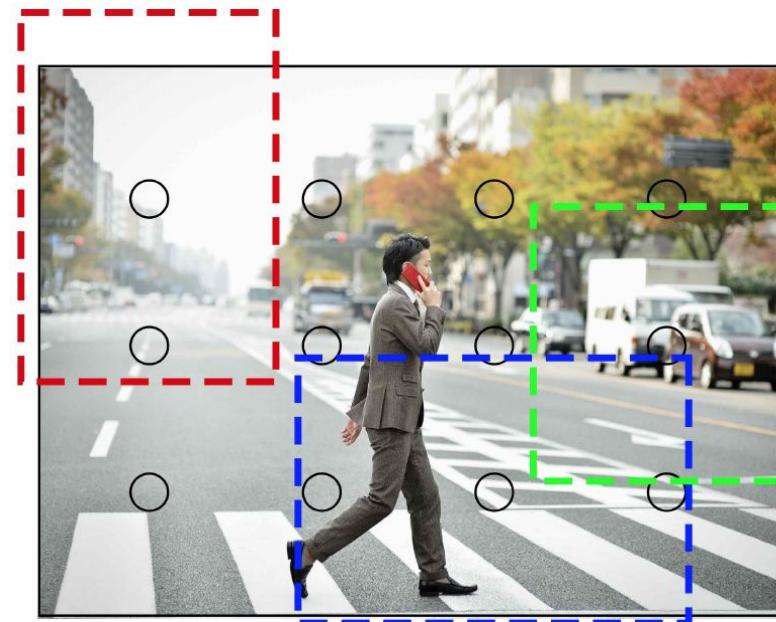
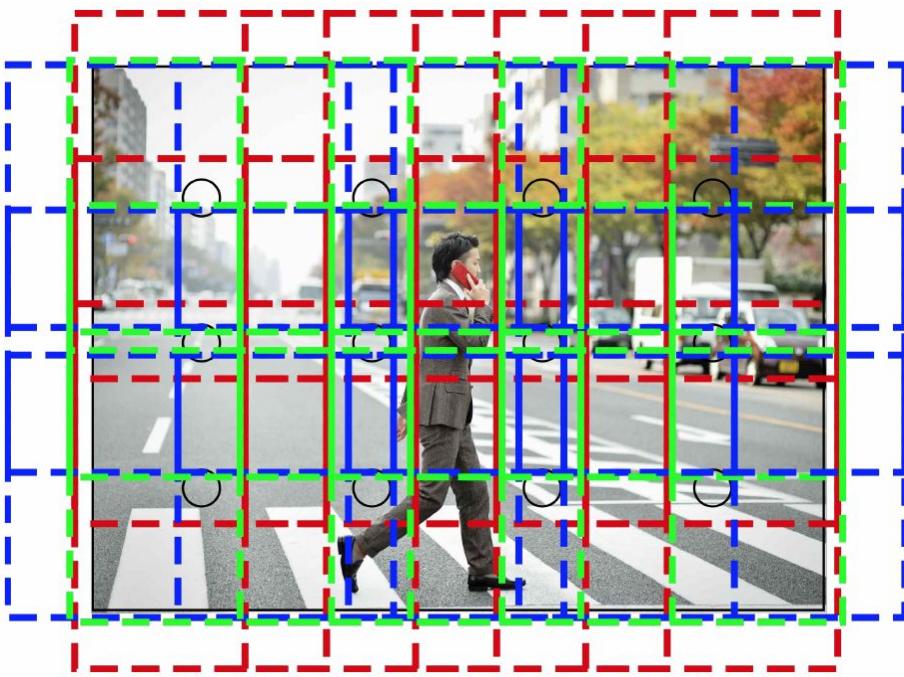
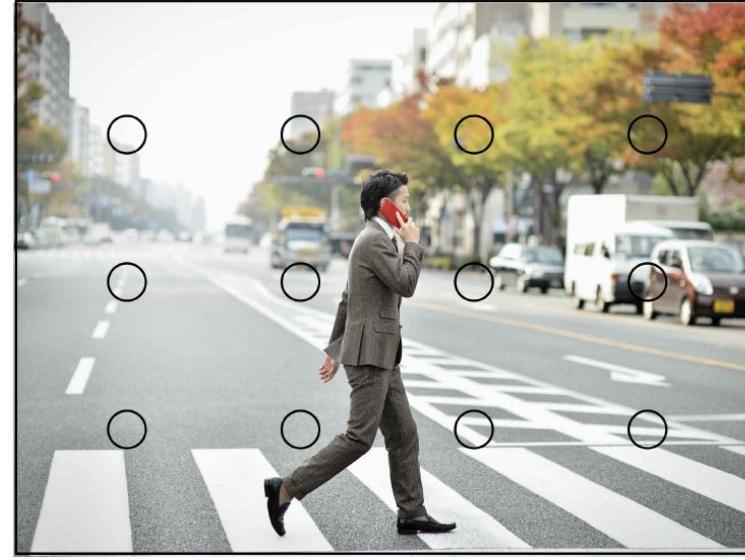
Faster RCNN

1. Дополнительная нейронная сеть для генерации region proposal-ов располагается после fully convolutional части сети и называется - Region Proposal Network (RPN).
2. Используются anchor box-ы с различными размерами и соотношением сторон.
3. RPN предсказывает “объектность”.
4. NMS применяется к сгенерированным регионам.
5. Результат передается в RoI pooling слой, а после в регрессор и классификатор Fast RCNN.

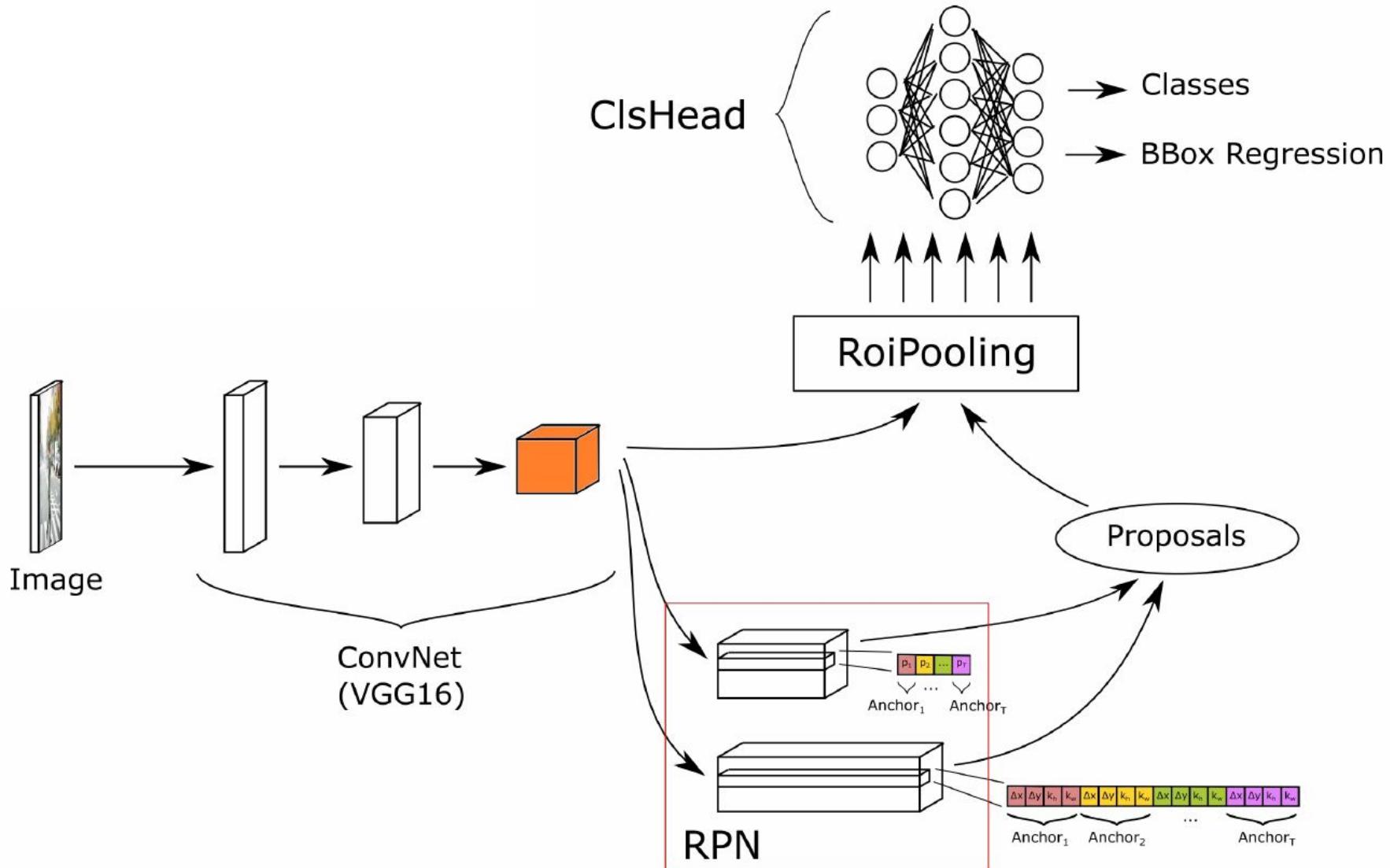


<https://arxiv.org/pdf/1506.01497.pdf>

Anchor Boxes



Faster RCNN



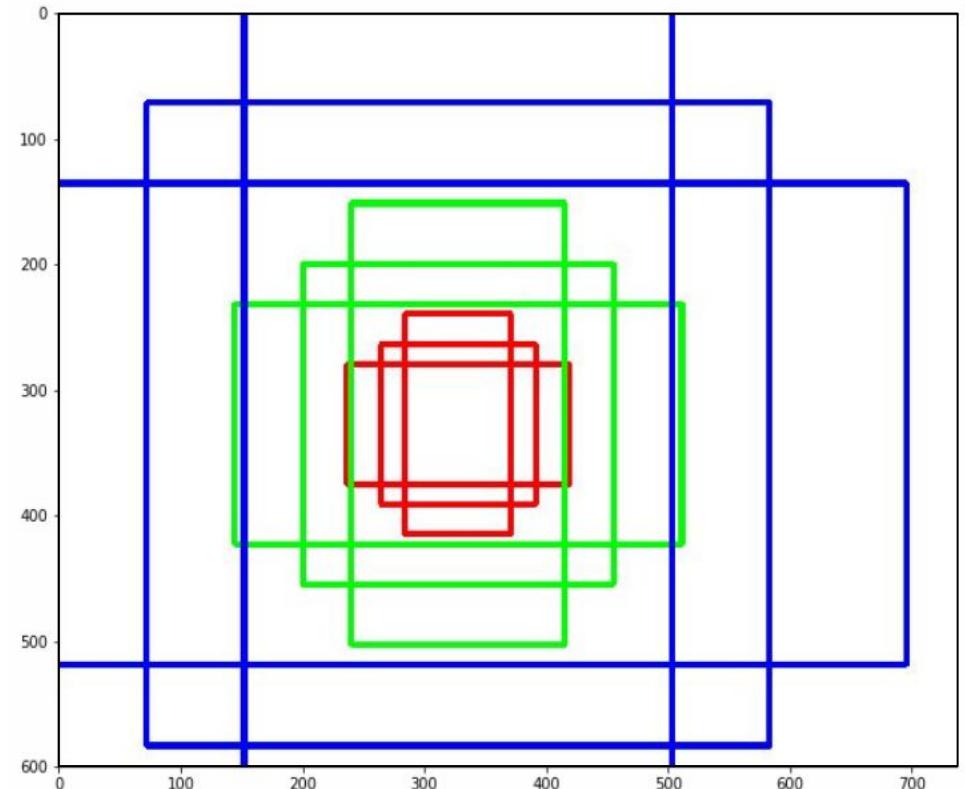
Scale problem

Проблема:

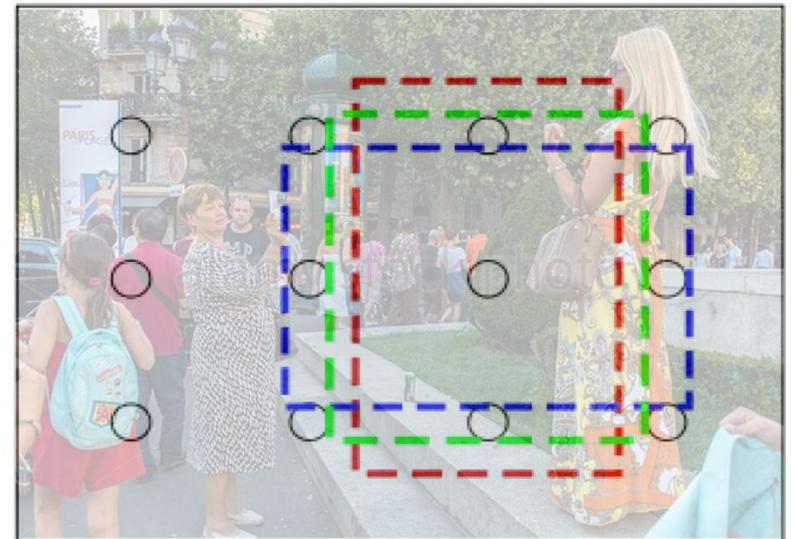
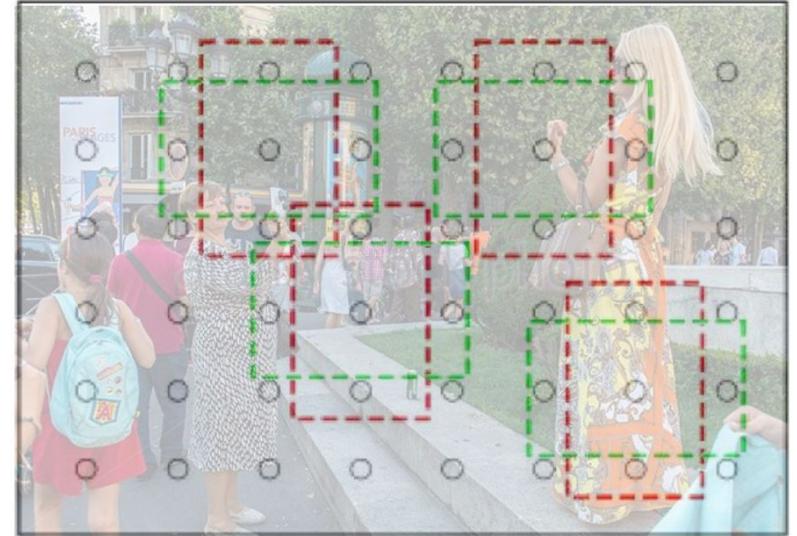
Для каждой точки мы генерируем одинаковое количество больших и маленьких box-ов. При таком подходе box-ы небольшого размера будут “разрежены”. Хотя мы имеем возможность расположить их в большем количестве.

Решение:

Генерируем region proposal-ы разных размеров с разной плотностью покрытия!

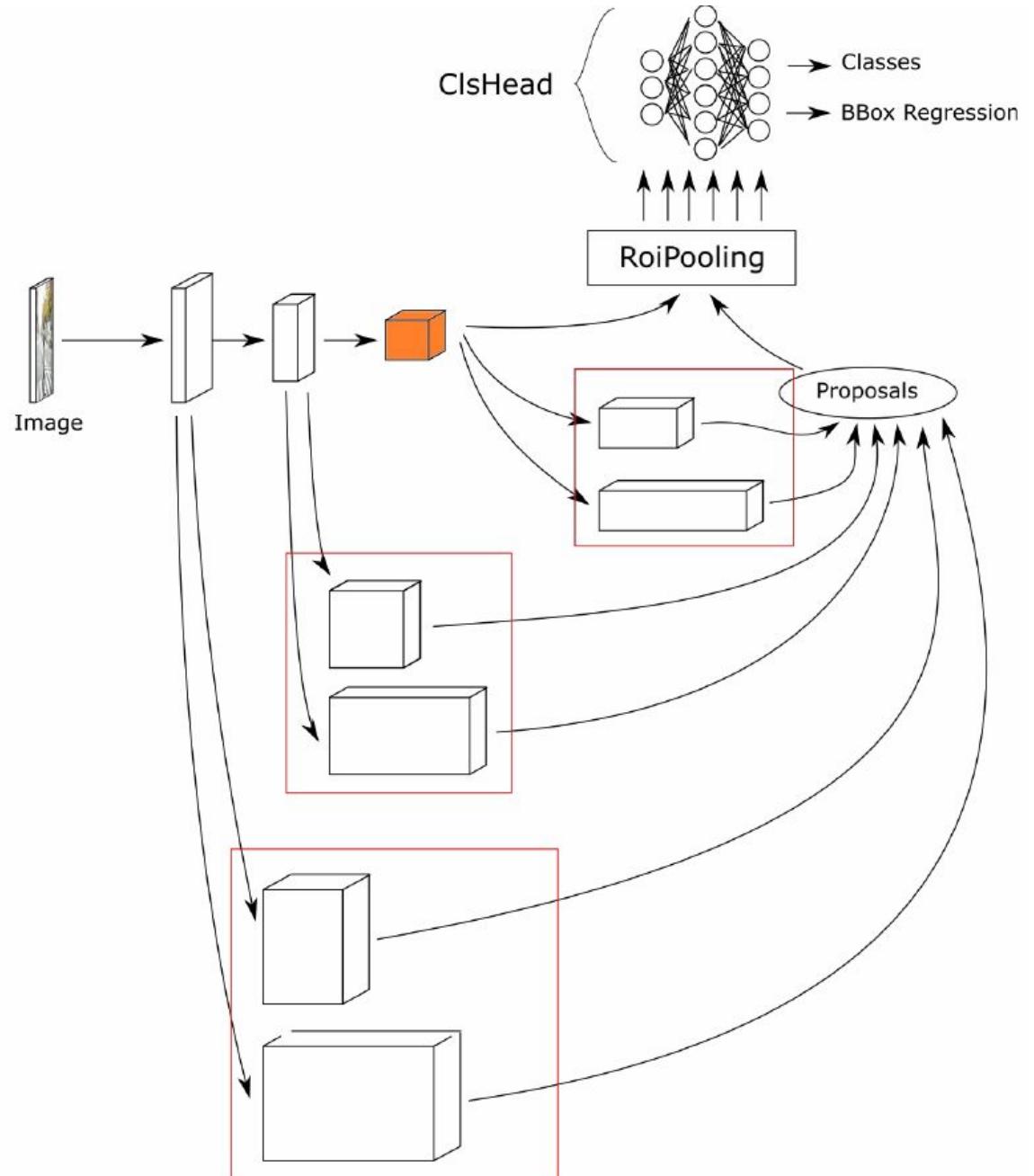
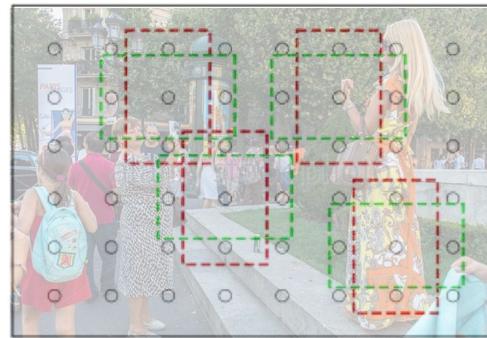
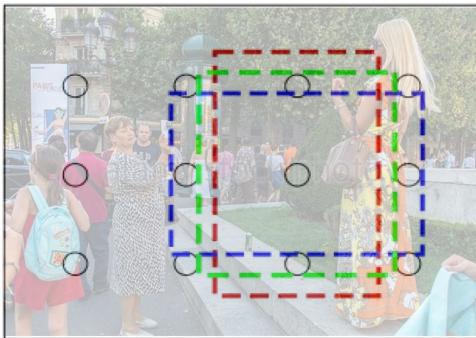


Different Scales



Multiple Scales

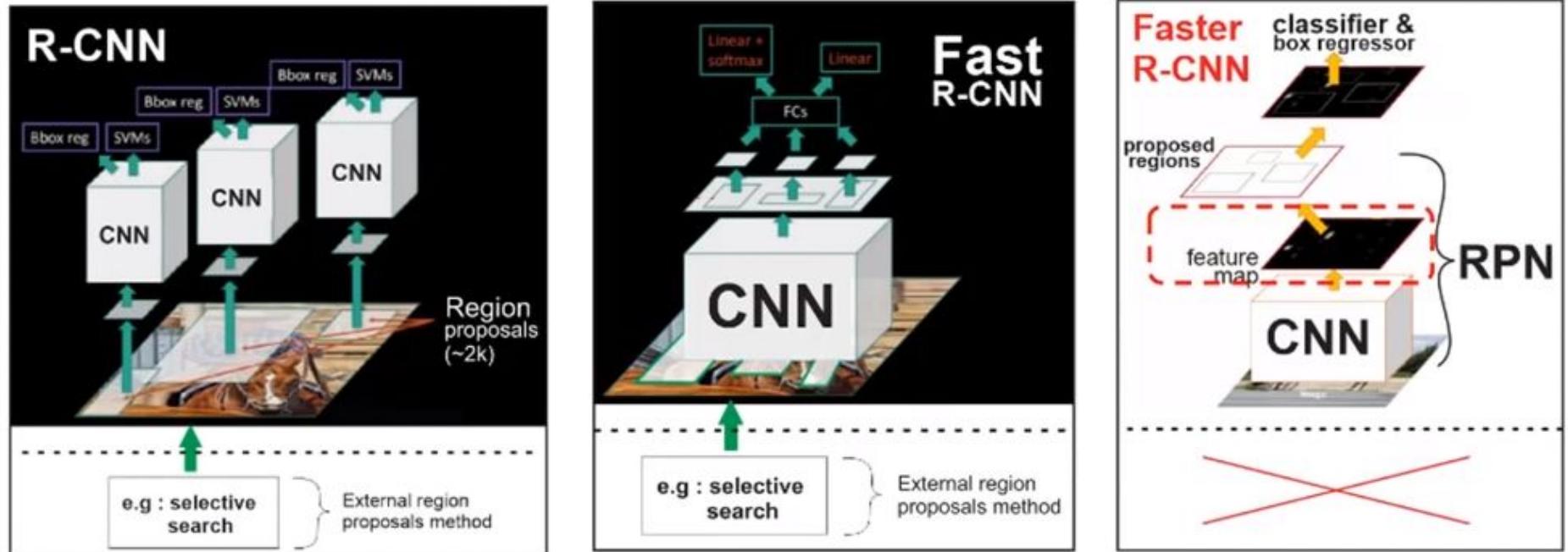
1. Предыдущие CNN тензоры используются для генерации proposal.
2. Proposal-ы с разных слоев сохраняются в один массив.



Faster RCNN резюме

1. Создается сетка anchors разного размера и формы.
2. RPN отбирает наилучшие anchors и работает anchor-wise. Для каждой точки feature map-ы предсказывается:
 - 2K вероятностей наличия объекта,
 - 4K смещений,где K – это кол-во anchors.
3. Два уточнения боксов.
Anchor → (RPN) proposal → (Fast R-CNN, class-specific) final box.
4. Подход до сих пор актуален. Mask RCNN базируется на Faster RCNN с resnet в качестве бэкбона.
5. R-FCN дальнейшее развитие Faster RCNN.

RCNN / Fast RCNN / Faster RCNN

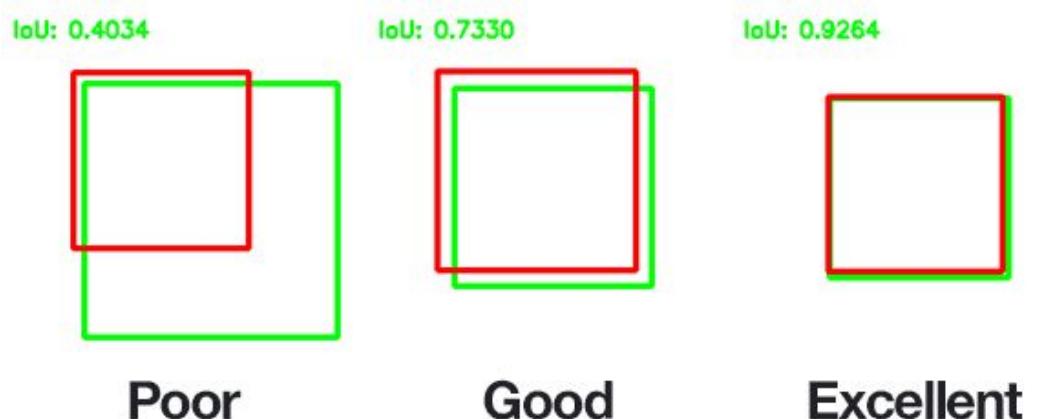
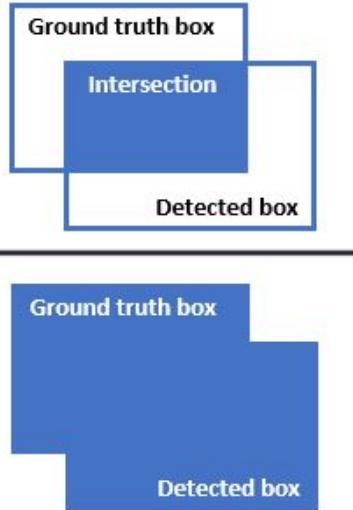


	R-CNN	Fast R-CNN	Faster R-CNN
Test time per image	50 seconds	2 seconds	0.2 seconds
Speed-up	1x	25x	250x
mAP (VOC 2007)	66.0%	66.9%	66.9%

* Standford lecture notes on CNN by Fei Fei Li and Andrej Karpathy

Метрика детекции объектов на изображении

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



Intersection over Union (IoU).

Если $\text{IoU} >$ порог (обычно 0.5): **True Positive**

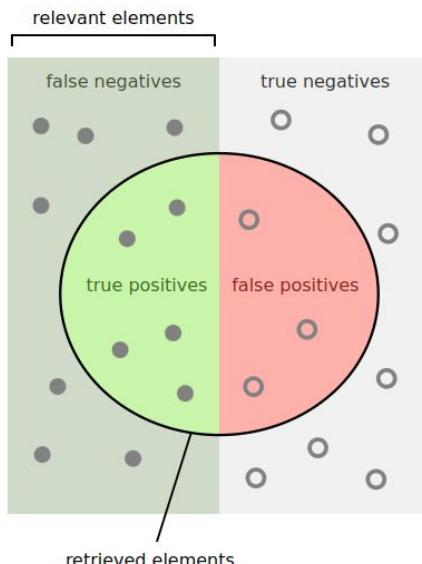
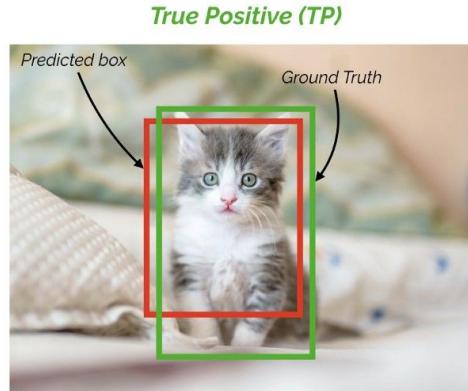
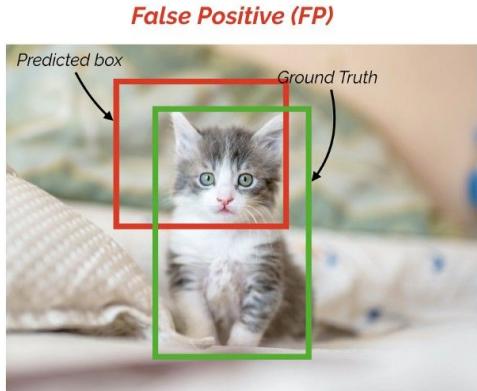
Иначе: **False Positive**.

Если нет предсказания для ground-truth объекта: **False Negative**.

```
def IOU(box1, box2):
    x1, y1, x2, y2 = box1
    x3, y3, x4, y4 = box2
    x_inter1 = max(x1, x3)
    y_inter1 = max(y1, y3)
    x_inter2 = min(x2, x4)
    y_inter2 = min(y2, y4)
    width_inter = abs(x_inter2 - x_inter1)
    height_inter = abs(y_inter2 - y_inter1)
    area_inter = width_inter * height_inter
    width_box1 = abs(x2 - x1)
    height_box1 = abs(y2 - y1)
    width_box2 = abs(x4 - x3)
    height_box2 = abs(y4 - y3)
    area_box1 = width_box1 * height_box1
    area_box2 = width_box2 * height_box2
    area_union = area_box1 + area_box2 - area_inter
    iou = area_inter / area_union
    return iou
```

Precision & Recall

If IoU threshold = 0.5

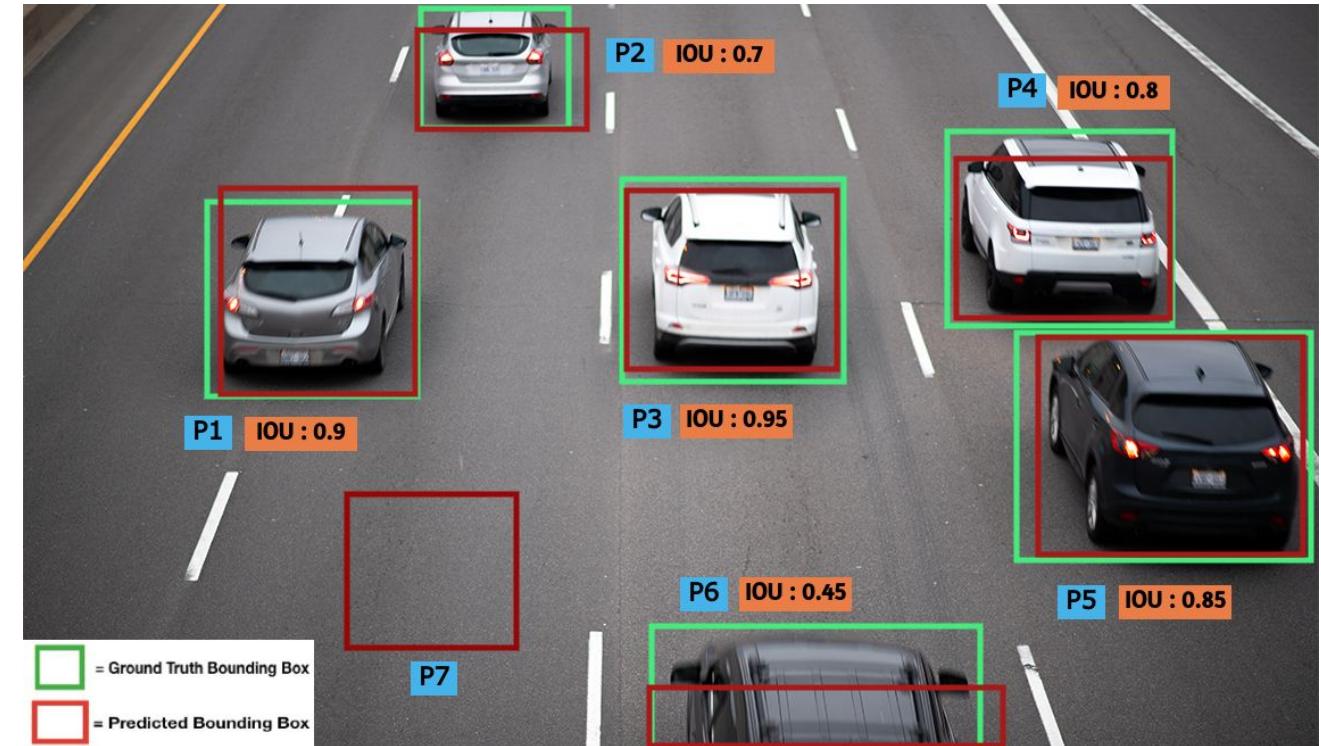


How many retrieved items are relevant?

How many relevant items are retrieved?

$$\text{Precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}$$

$$\text{Recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$$



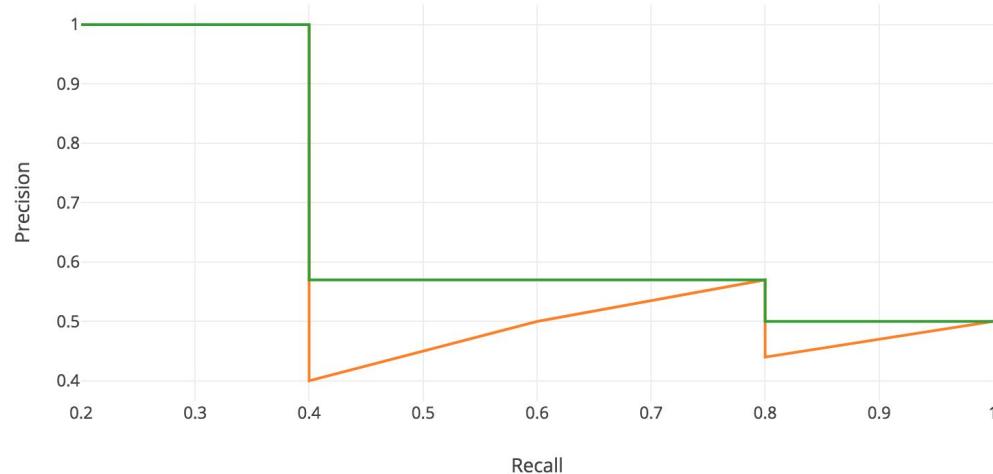
If IoU threshold = 0.8 then precision = 4/7 and recall = 4/6.

If IoU threshold = 0.5 then precision = 5/7 and recall = 5/6.

If IoU threshold = 0.2 then precision = 6/7 and recall = 6/6.

mean Average Precision (mAP)

Average Precision (AP) = площадь под кривой precision-recall.



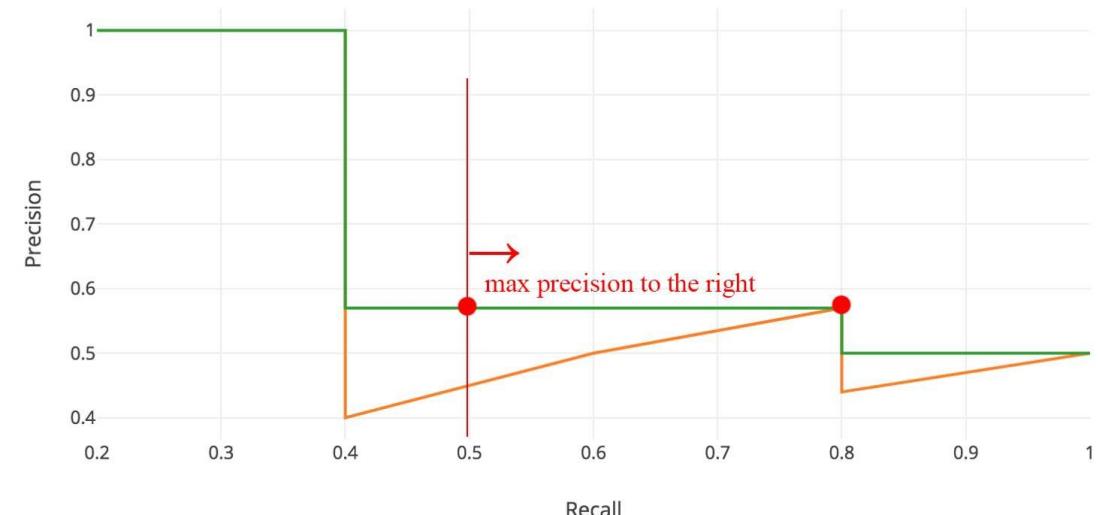
$$AP = \frac{1}{11} \times (AP_r(0) + AP_r(0.1) + \dots + AP_r(1.0))$$

Average Precision для класса = average maximum Precision, который можно получить для значения Recall в [0.0, 0.1, ..., 0.9, 1.0]

mAP = усредненный AP по всем классам в датасете.

Меняем порог => Получаем разные precision и recall.

- оранжевая линия - PR кривая
- зеленая линия - максимальный Precision, который можно получить для данного уровня Recall.



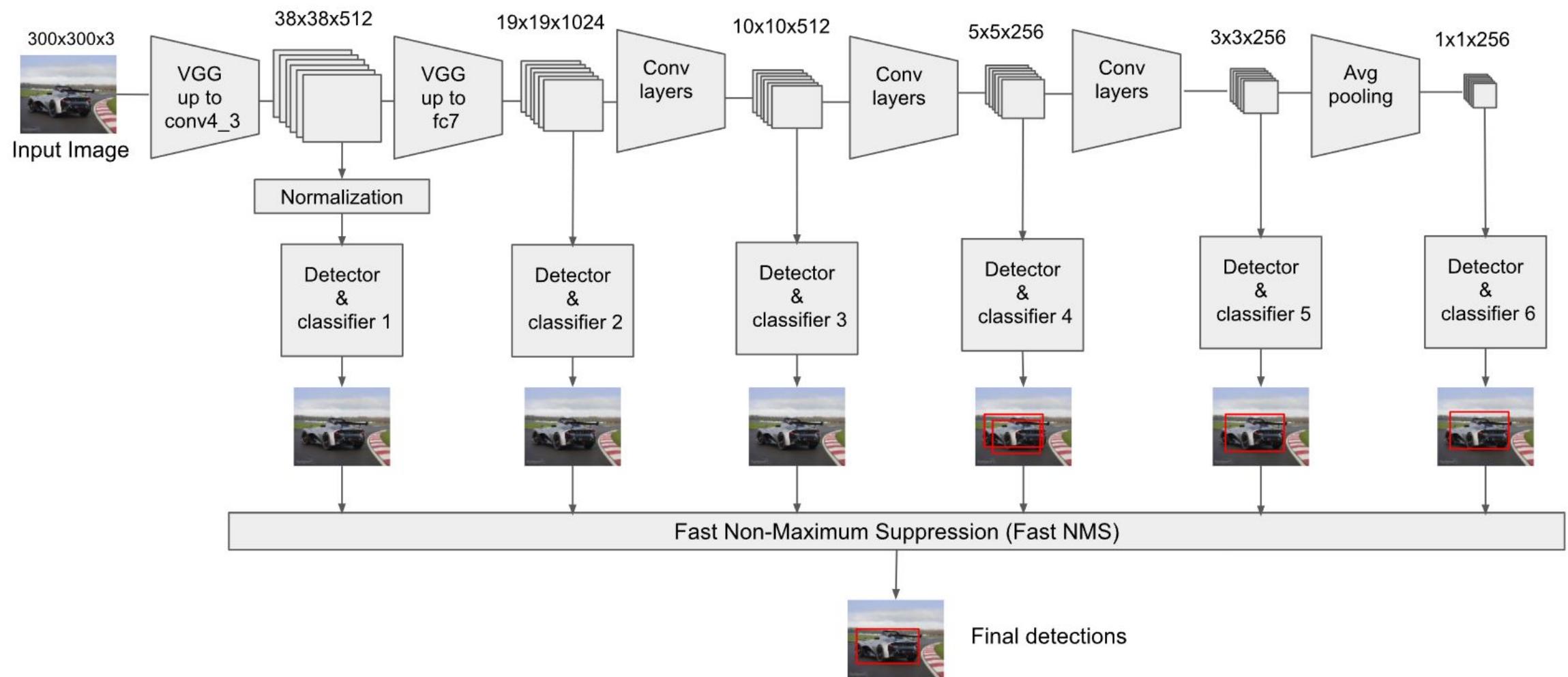
Single-shot детекция



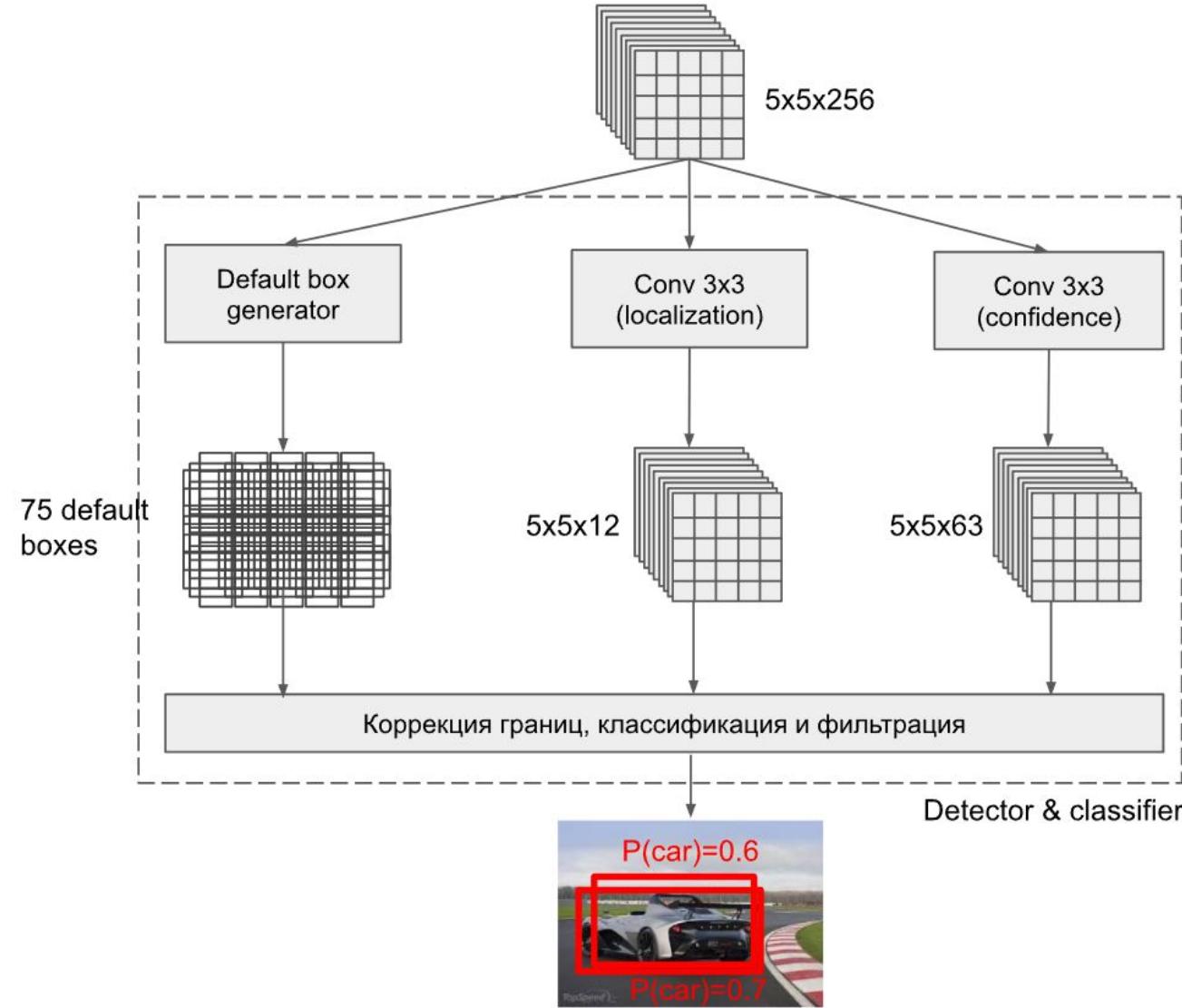
Single shot approaches

Основная идея: не используем region proposals и ROI pooling.
Работаем с одной нейронной сетью, которая предсказывает классы
и регрессионные метки для bounding-box-ов для каждого возможного
положения.

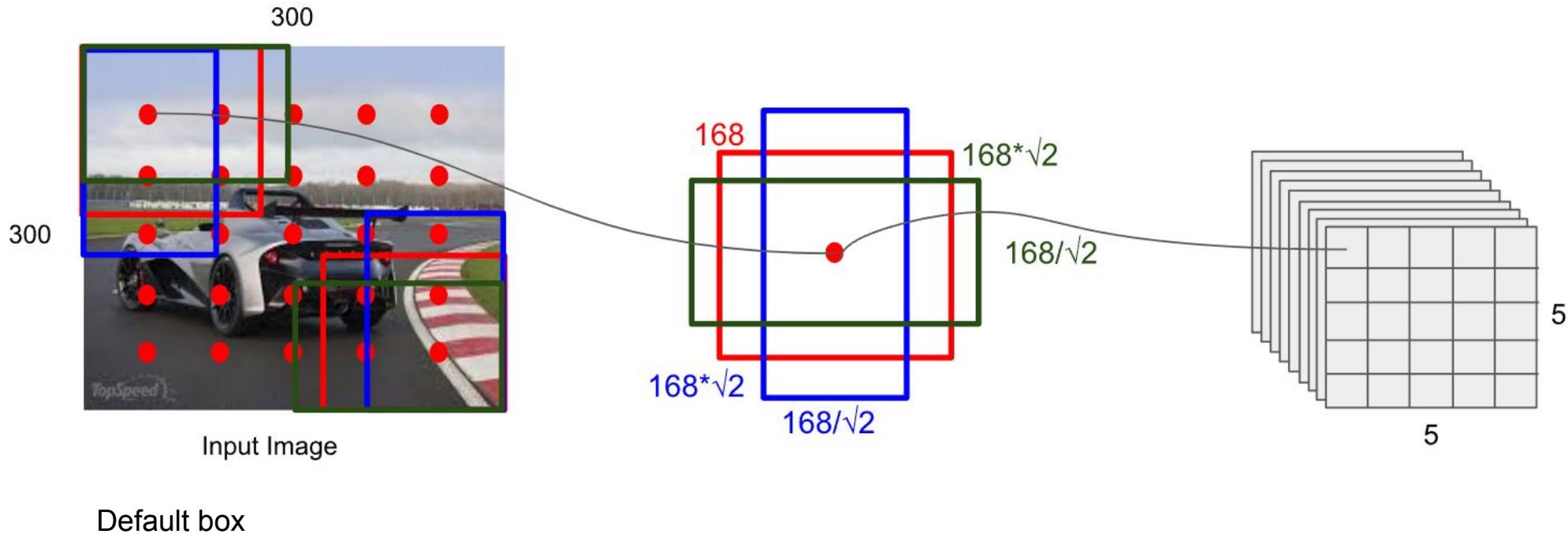
Single Shot Detector (1)



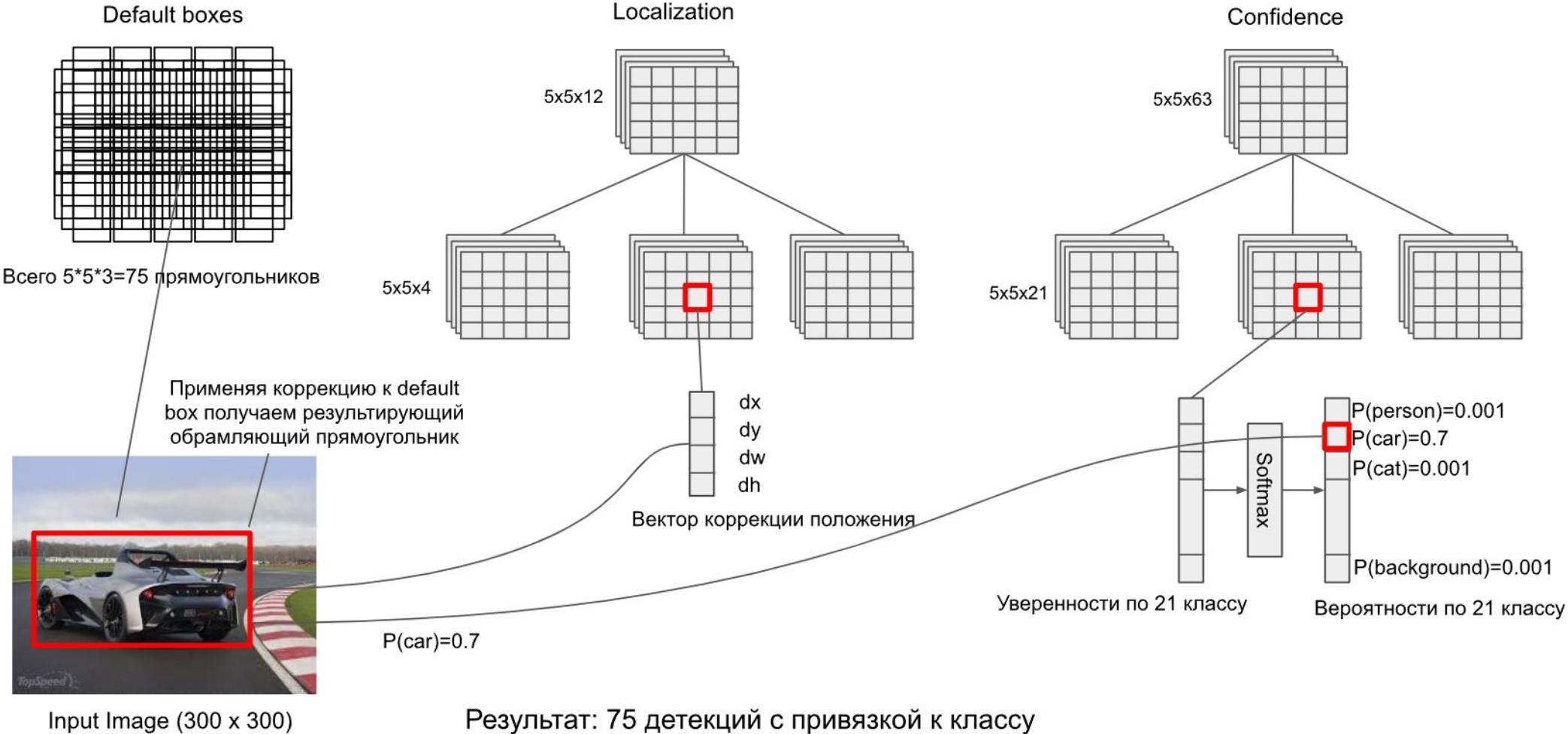
Single Shot Detector (2)



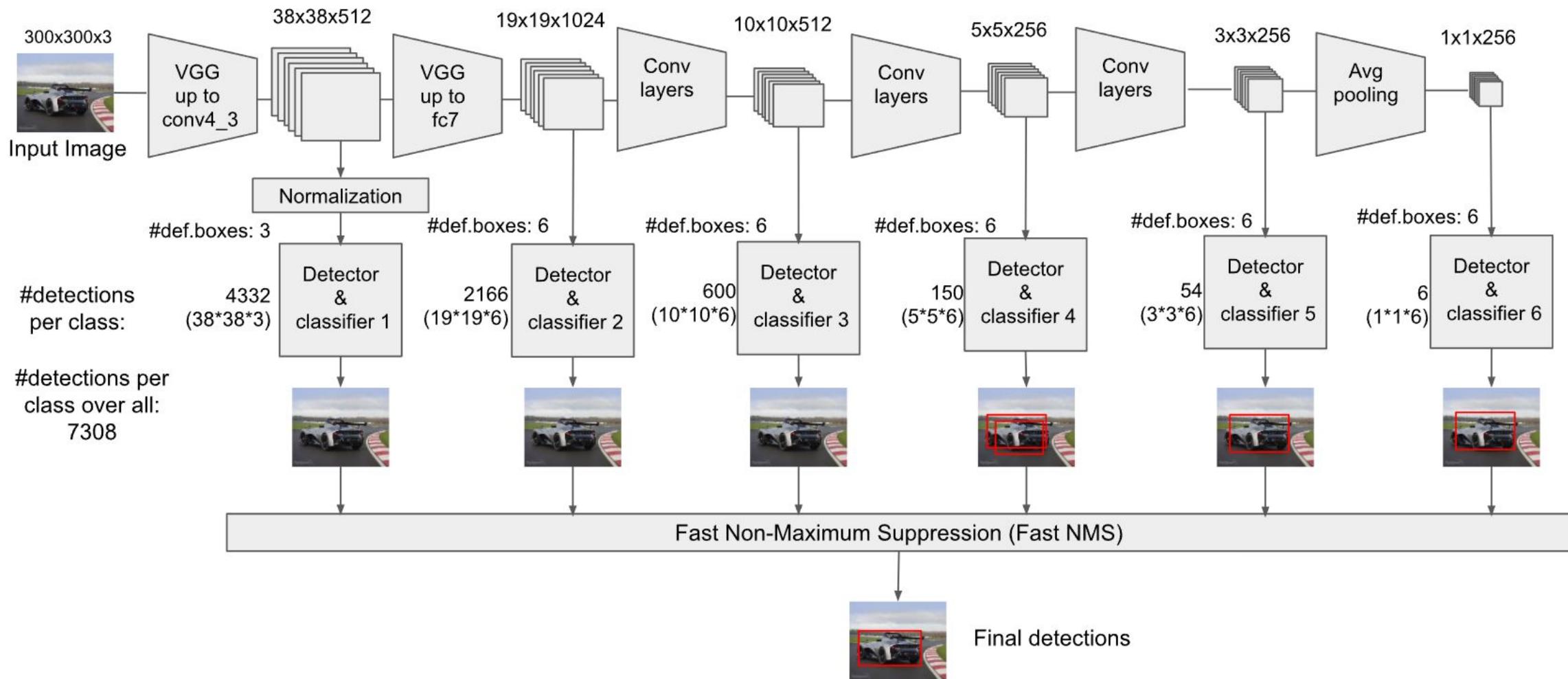
Single Shot Detector (3)



Single Shot Detector (4)



Single Shot Detector (5)

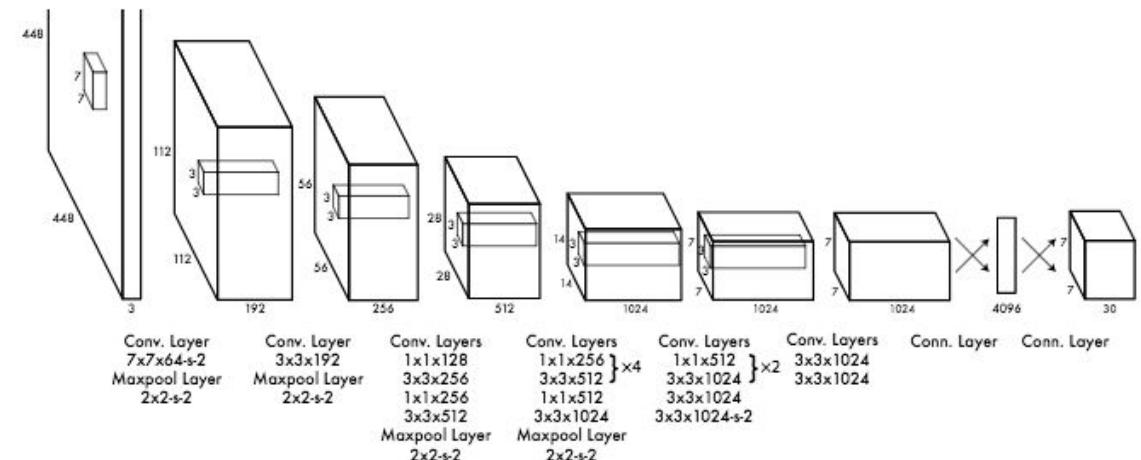
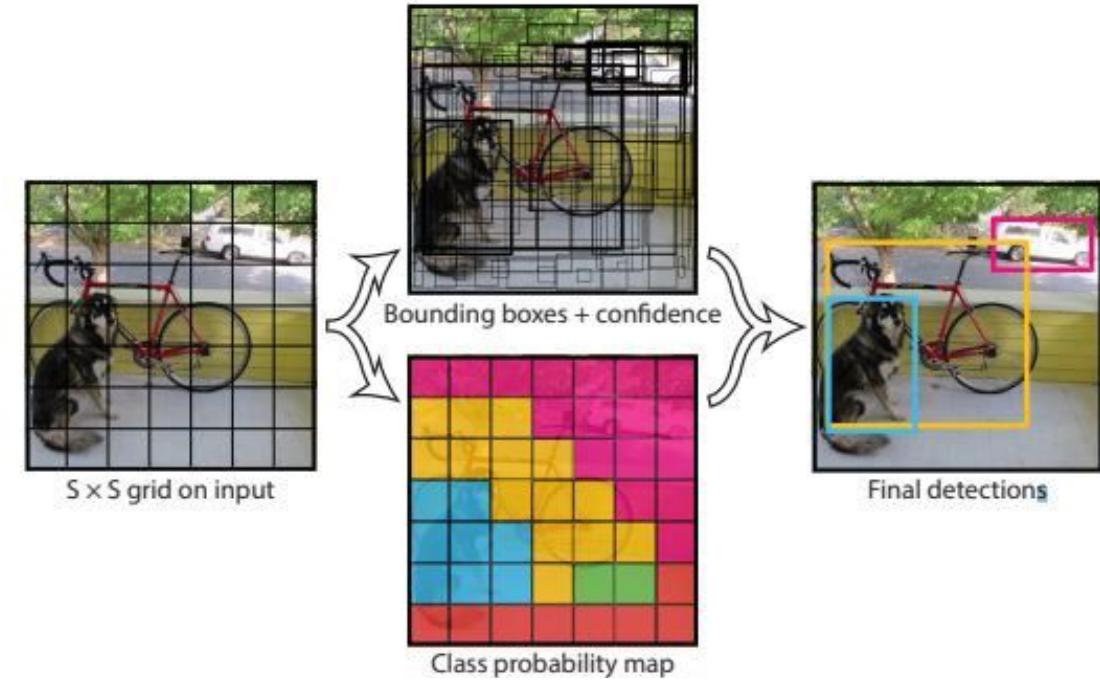


YOLO (You Only Look Once)

1. Принимаем изображение на вход и разбиваем на сетку
2. Каждая ячейка отвечает за предсказание объекта, с центром лежащим внутри этой ячейки
3. Для каждой ячейки предсказываем bounding box и вероятность объекта и класса
4. Фильтруем боксы с помощью NMS
5. Модель улучшается до сих пор различными авторами, с последней версией можно ознакомиться тут:

YOLO v10:

<https://github.com/ultralytics/ultralytics>



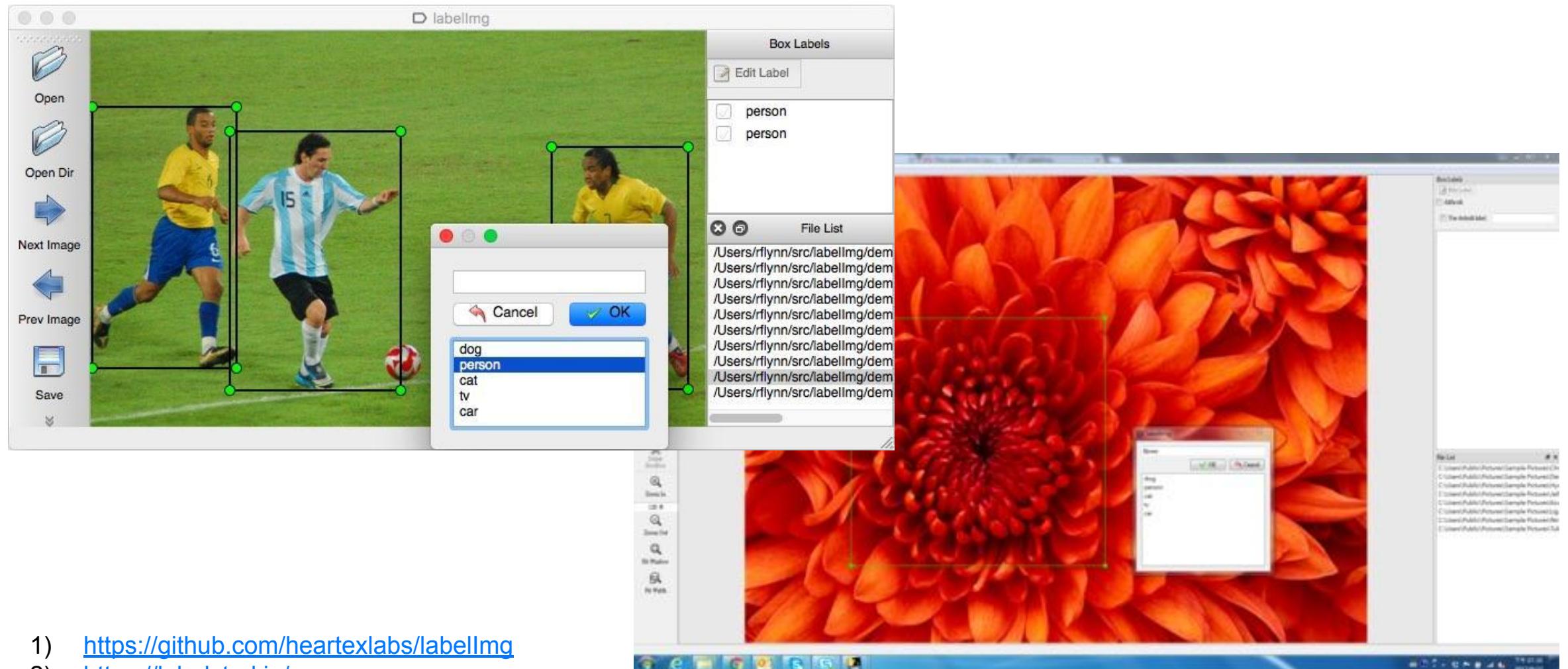
<https://arxiv.org/pdf/1506.02640.pdf>

<https://pjreddie.com/publications/yolo/>

Формирования разметки

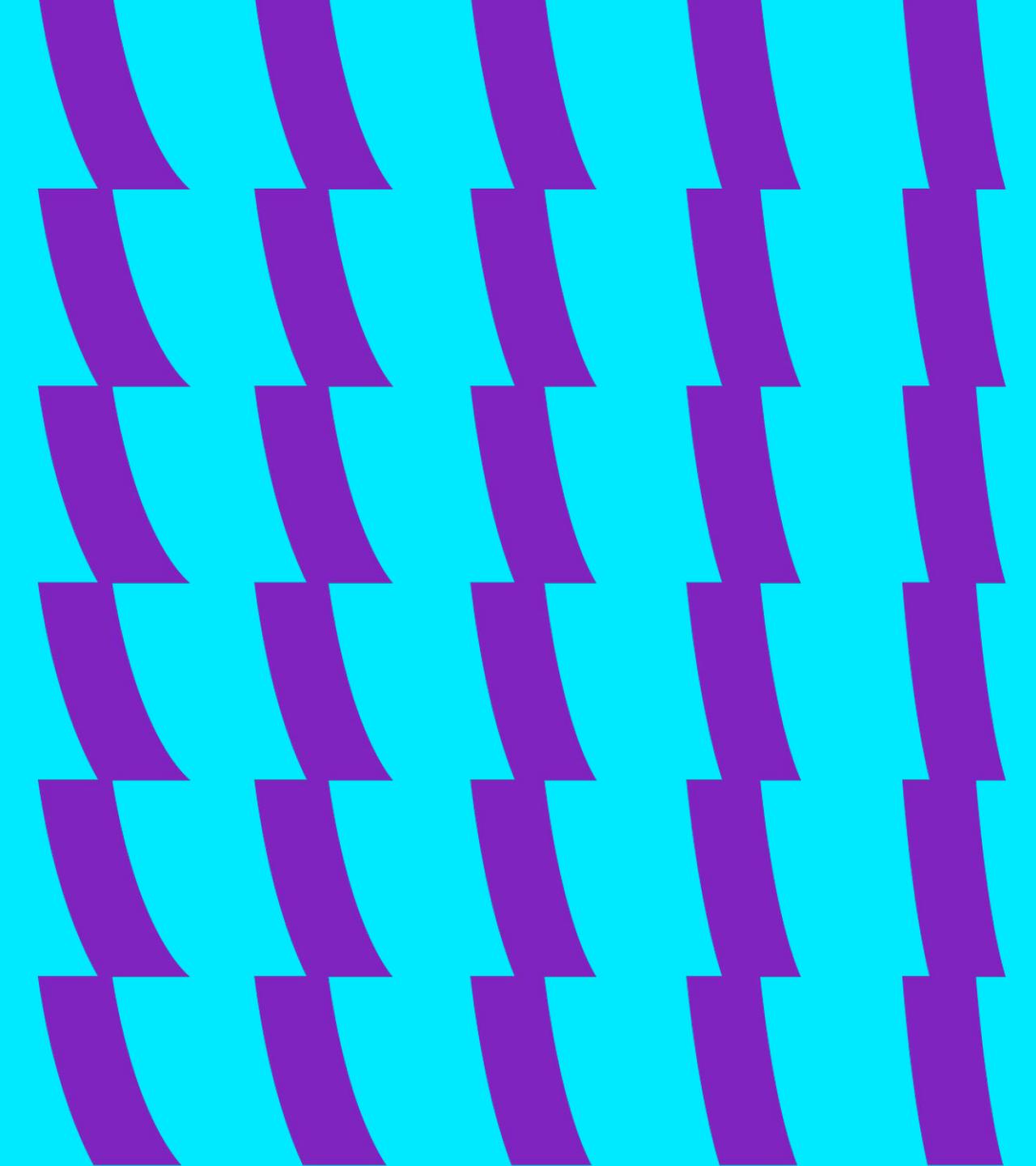


Инструменты создания разметки изображений



- 1) <https://github.com/heartexlabs/labelImg>
- 2) <https://labelstud.io/>
- 3) <https://www.cvtai.ai/>

Вопросы?





Спасибо
за внимание!