Нейронные сети

Детекция объектов на изображениях

Спасёнов Алексей

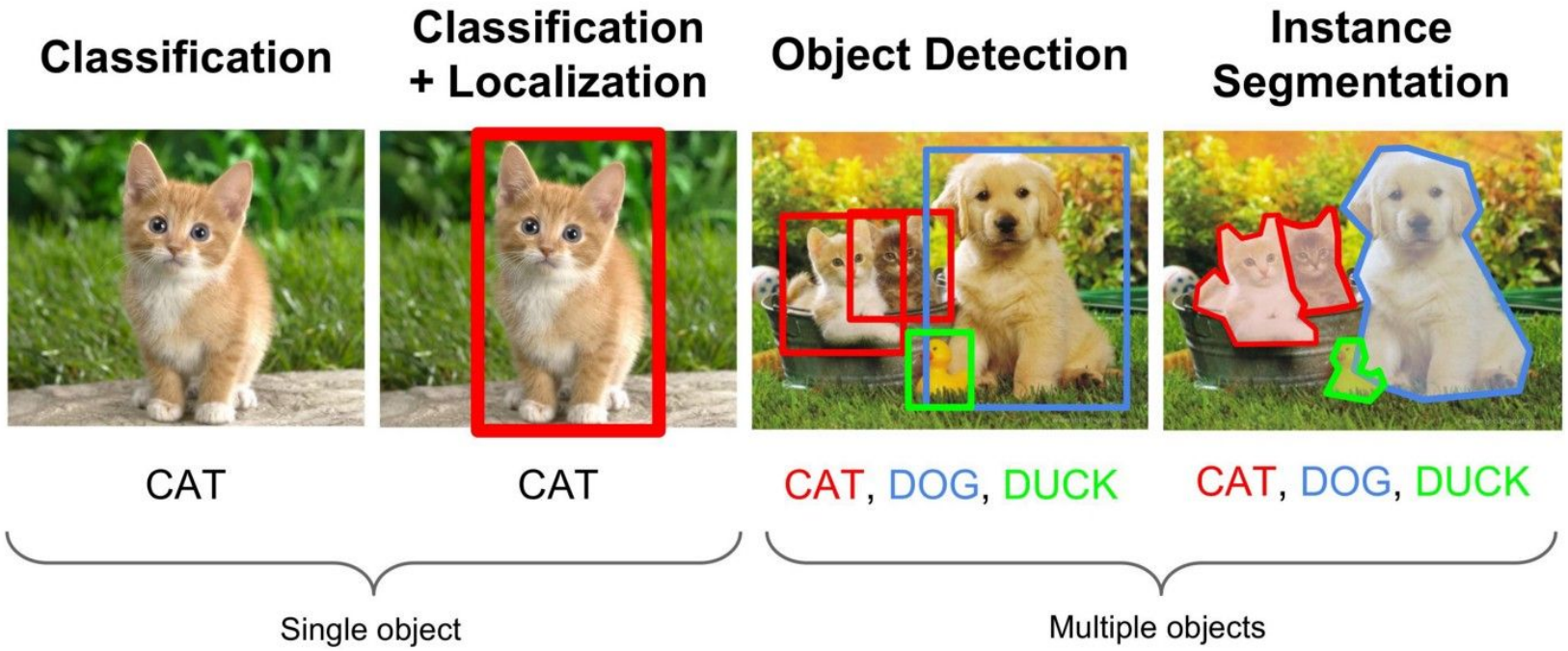
Не забывайте отмечаться и оставлять отзыв

Содержание лекции

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

Задача детекции объектов на изображении

Задача детекции объектов на изображении

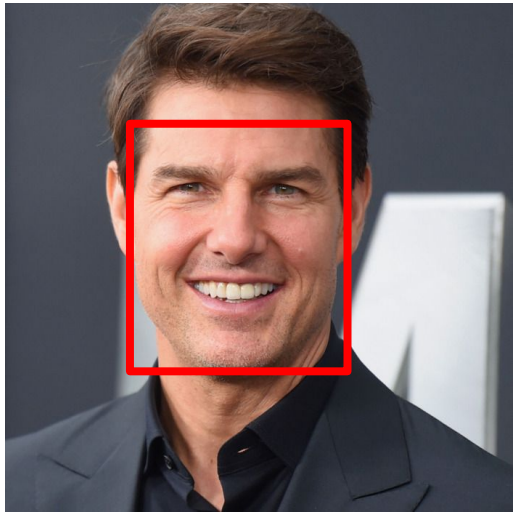


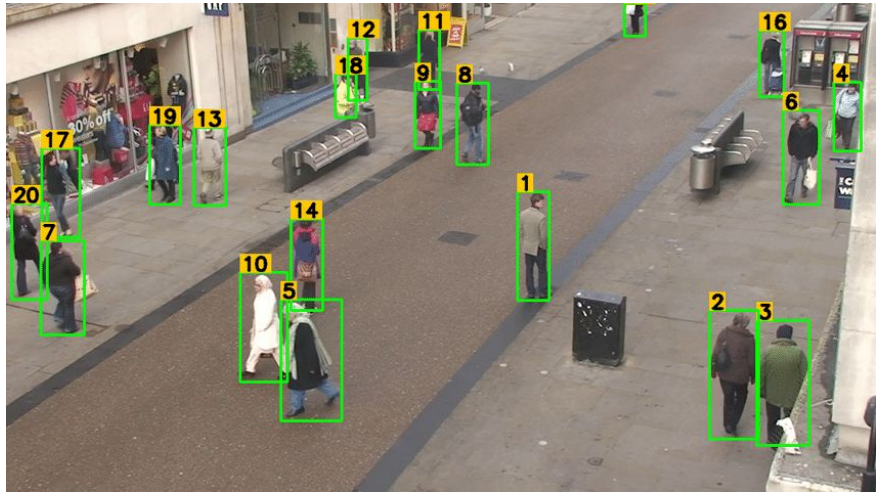
Отличия от задачи классификация:

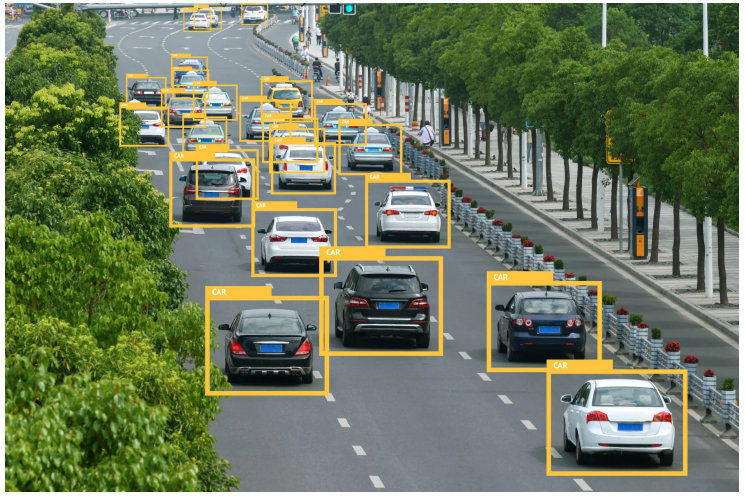
1. Изображение может содержать несколько объектов разных классов класса, в том числе могут отсутствовать вообще;
2. На изображении может находиться несколько объектов одного и того же класса;
3. Требуется локализовать объект с помощью прямоугольной зоны (bounding box).

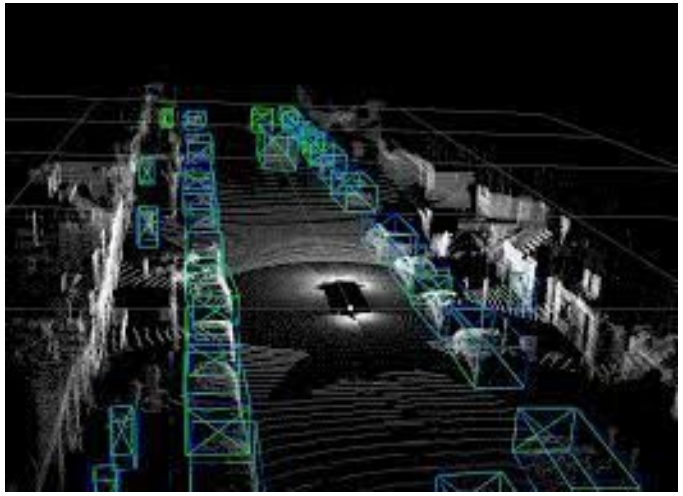
Примеры детекции объектов на изображении

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

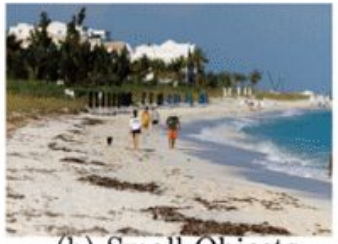
(c) Scale, Viewpoint

(*d*) Pose, Occlusion

(e) Clutter, Occlusion

(f) Blur

(g) Motion



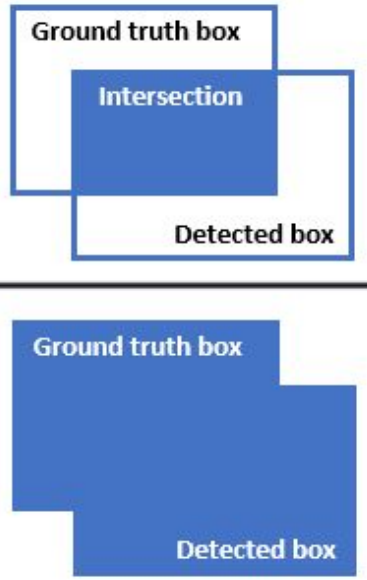
(h) Small Objects, Low Resolution

(i) Different instances of the "chair" category

(j) Small Interclass Variations: four different categories

Метрика детекции объектов на изображении

$$IoU = \frac{\text{Area of Overlap}}{\text{Area of Union}} = \frac{\text{Intersection}}{\text{Ground truth box} + \text{Detected box} - \text{Intersection}}$$



Intersection over Union (IoU).

If IoU > threshold (usually 0.5): **True Positive**

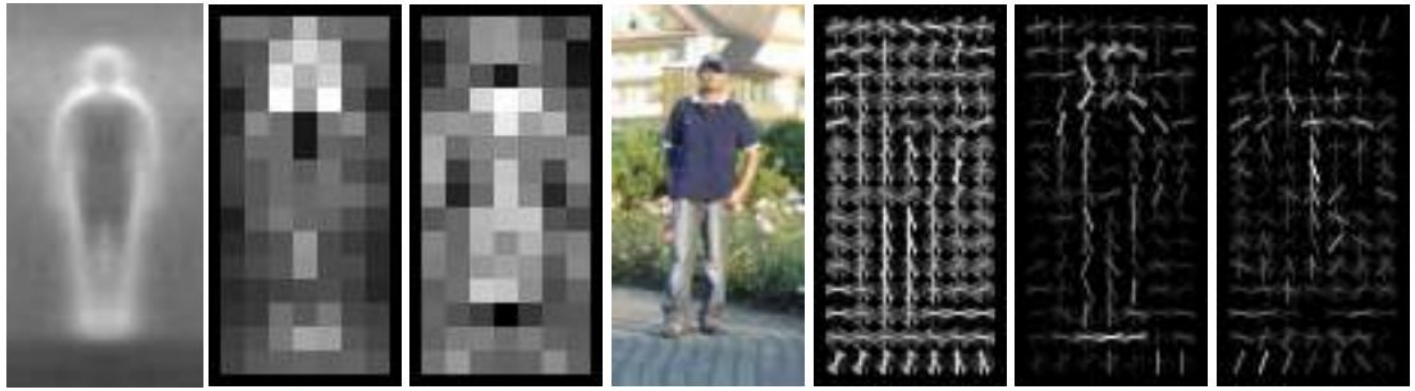
Else: **False Positive**.

If no prediction for some ground-truth object: **False Negative**.

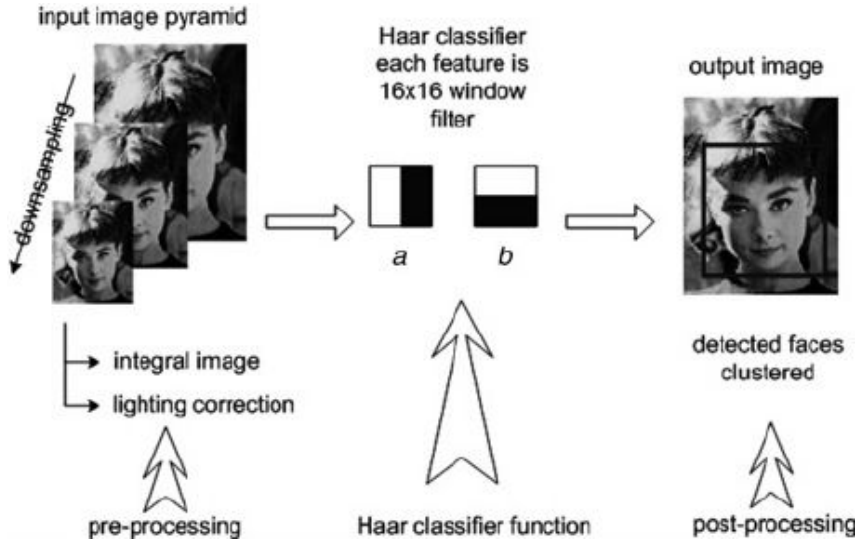
```python
def IOU(box1, box2):
    x1, y1, x2, y2 = box1
    x3, y3, x4, y4 = box2
    x_inter1 = max(x1, x3)
    y_inter1 = max(y1, y3)
    x_inter2 = min(x2, x4)
    y_inter2 = min(y2, y4)
    width_inter = abs(x_inter2 - x_inter1)
    height_inter = abs(y_inter2 - y_inter1)
    area_inter = width_inter * 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
```

Методы классического CV

1. HOG



2. Haar Cascades

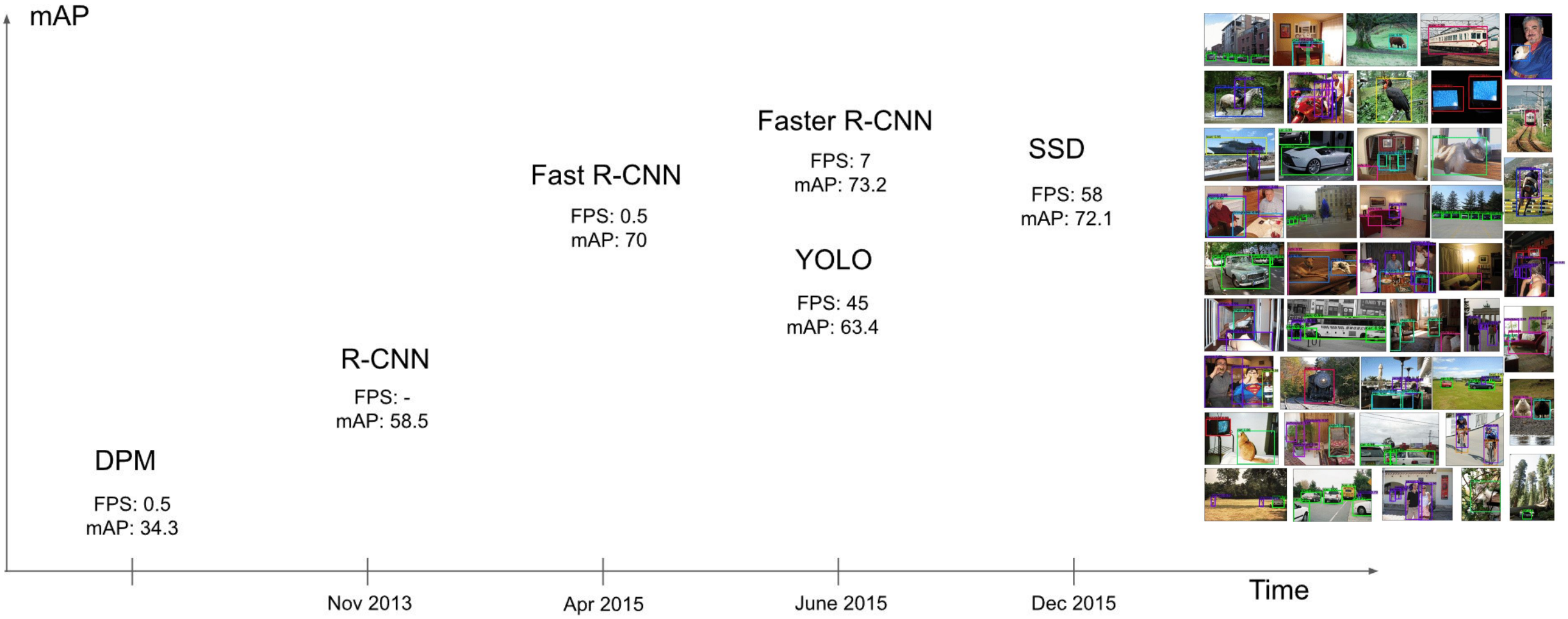


3. SIFT

Idea: Sliding window + image pyramid

Подходы на основе нейронных сетей

Развитие подходов



PASCAL VOC 2007

http://vision.stanford.edu/teaching/cs231b_spring1213/slides/dpm-slides-ross-girshick.pdf

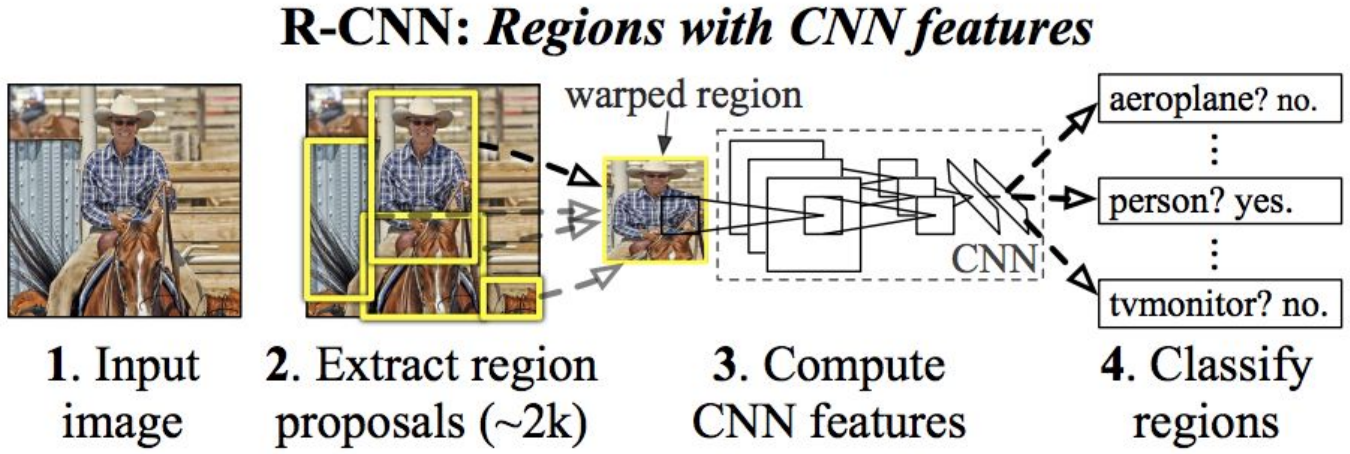
http://host.robots.ox.ac.uk/pascal/VOC/voc2007/

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. ...

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)



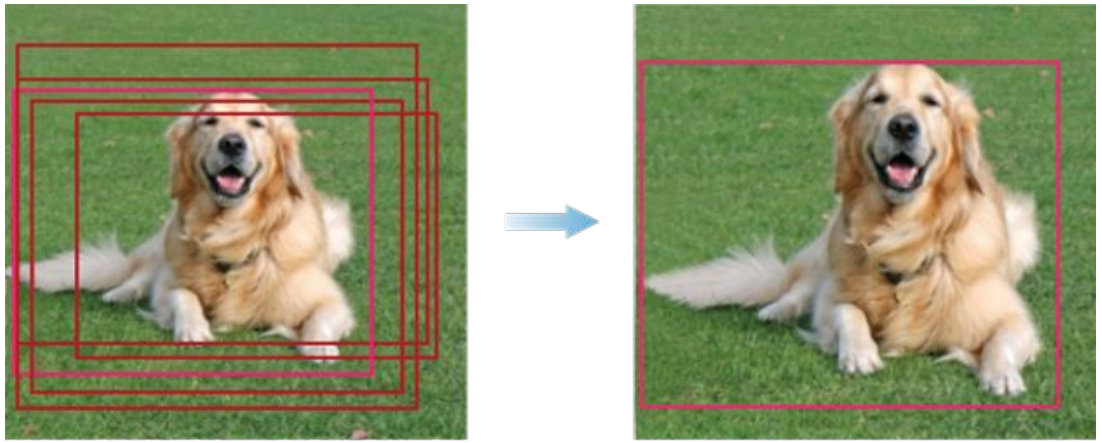
https://arxiv.org/pdf/1311.2524v5.pdf

Non Maximum Suppression (NMS), подавления немаксимумов

Problem: many highly overlapping regions for a single object. NMS discards bounding boxes with high overlap with bounding boxes with higher detection score.

NMS algorithm:

1. Sort boxes by detection scores in descending order. Store them in list L.
2. Iterate over list L:
   a. Pick first box in L.
   b. Compute its IoU with all other boxes in L.
   c. Remove boxes that have IoU with B greater than threshold.



Predictions before NMS

After applying non-maximum suppression

https://paperswithcode.com/method/non-maximum-suppression

RCNN Details

1. Classification CNN training:
   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.
2. Class-specific SVM:
   a. Позволяет проставлять несколько классов одному региону;
   b. Обучается на признаках из CNN с предпоследнего полносвязного слоя (перед классификационным).

Advantages and disadvantages of RCNN

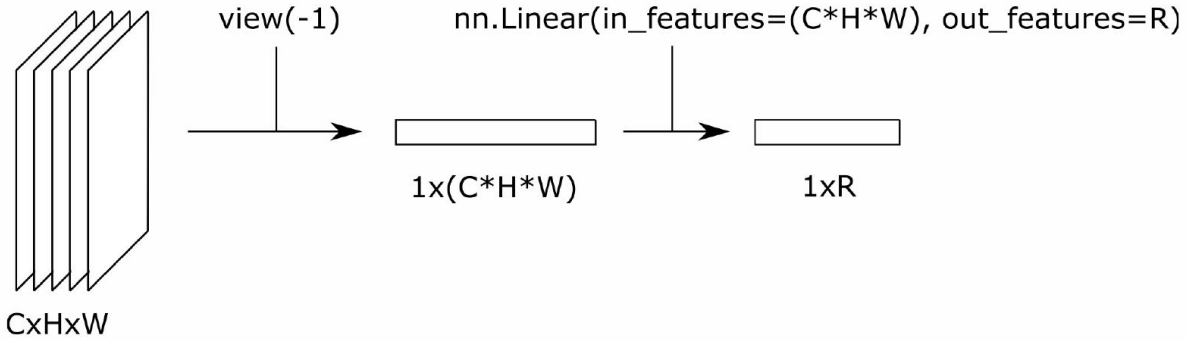
Pros:

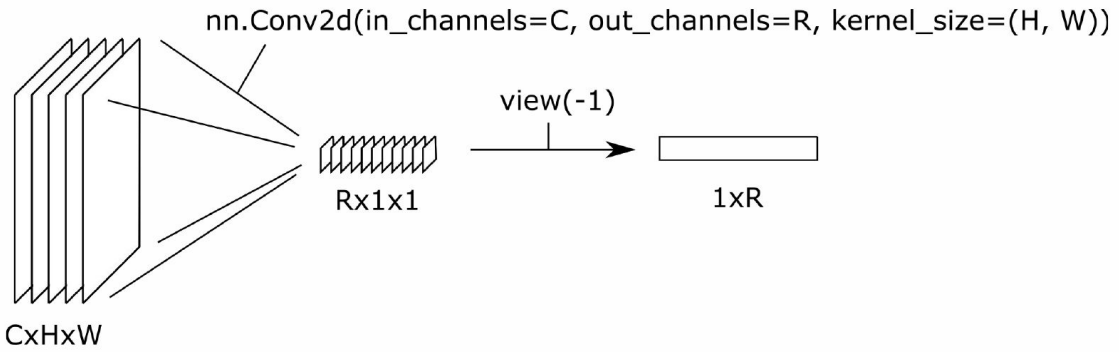
1. First “good” object detector (53.7% mAP on PASCAL VOC 2010).

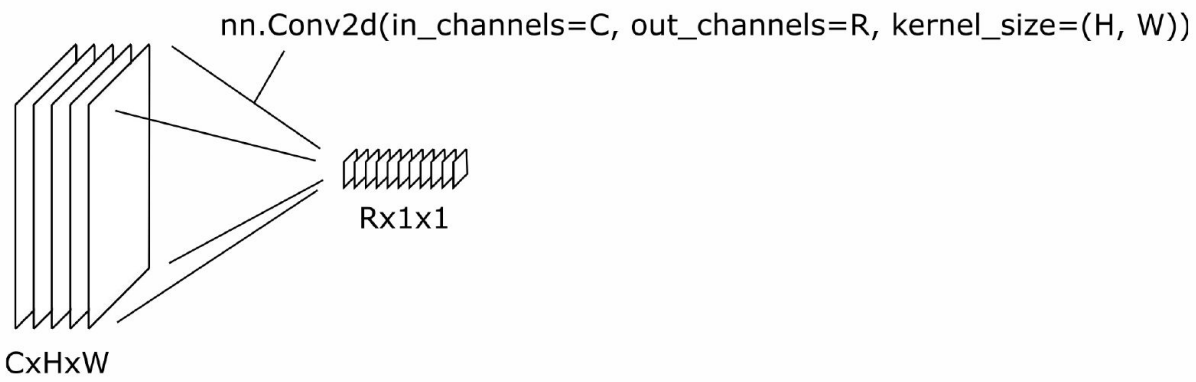
Cons:

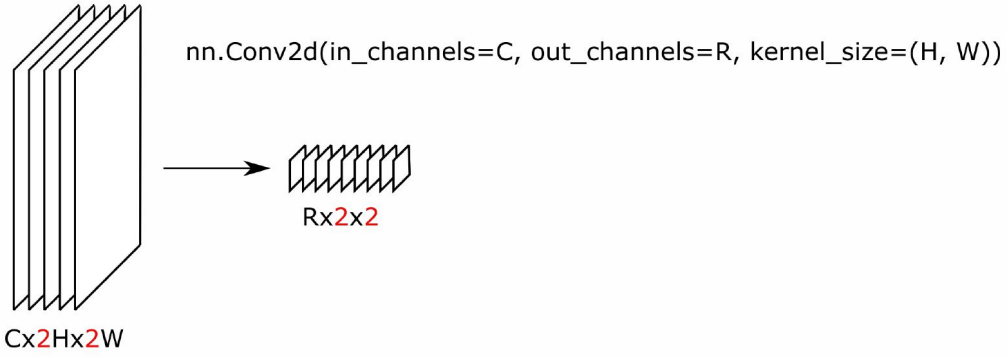
1. Too slow: 13s/image on a GPU or 53s/image on a CPU. ~2000 region proposals. Duplication of calculations in case of overlapping region proposals (no computation sharing).
2. Warps image to some standard size, does not care about object scale / aspect ratio.
3. Training is both computationally and memory expensive.
4. Multiple detection stages which is not very convenient.

Имитация линейного слоя с помощью Conv2d?



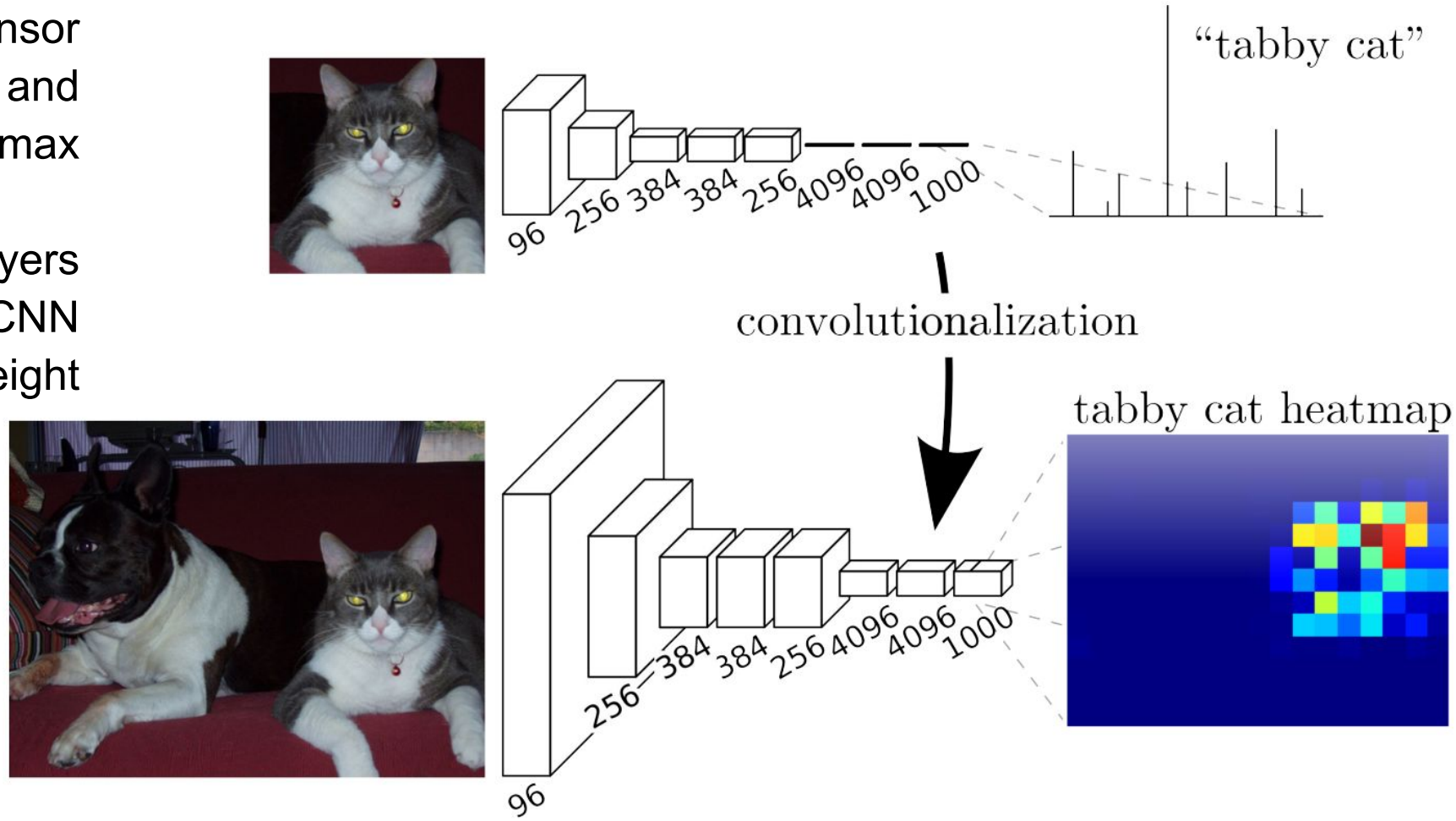






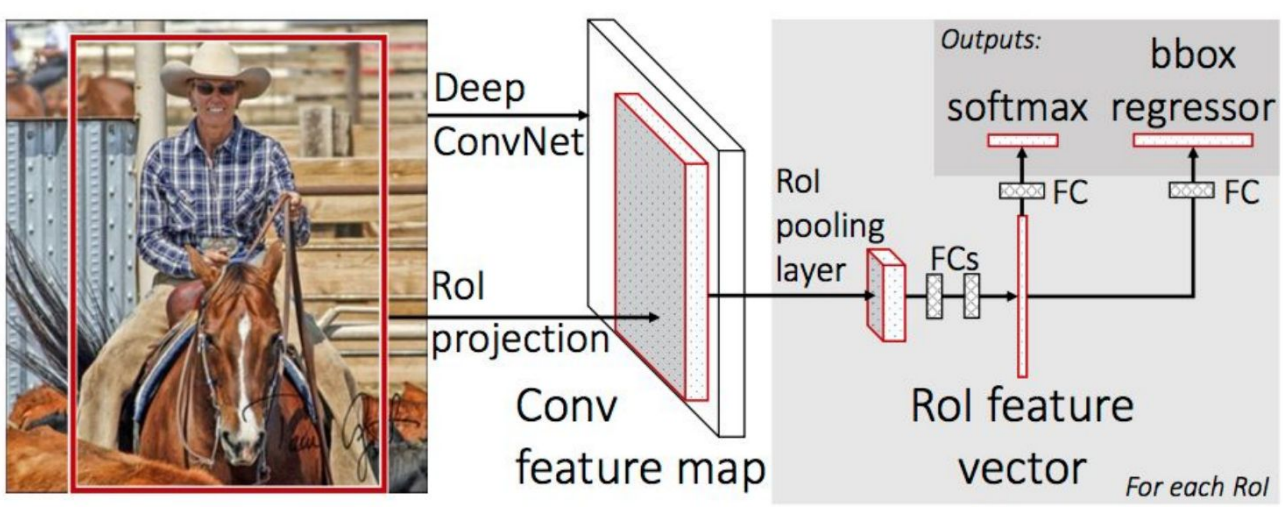
Fully Convolutional Network (FCN)

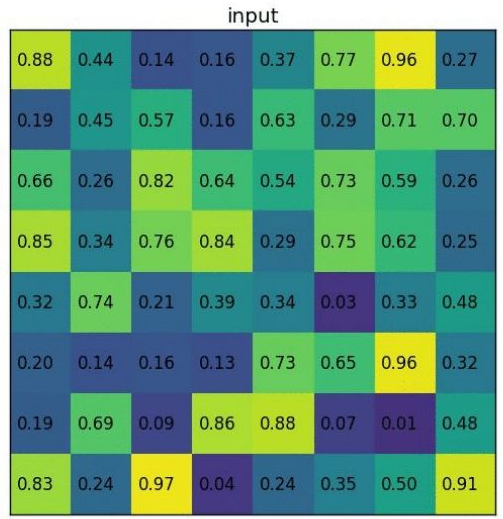
1. Convolution with fixed filter can be applied to a tensor of arbitrary height and width. Same for max pooling.
2. Only fully connected layers do not allow to apply CNN to image of arbitrary height and width.



https://arxiv.org/pdf/1411.4038.pdf

Fast RCNN

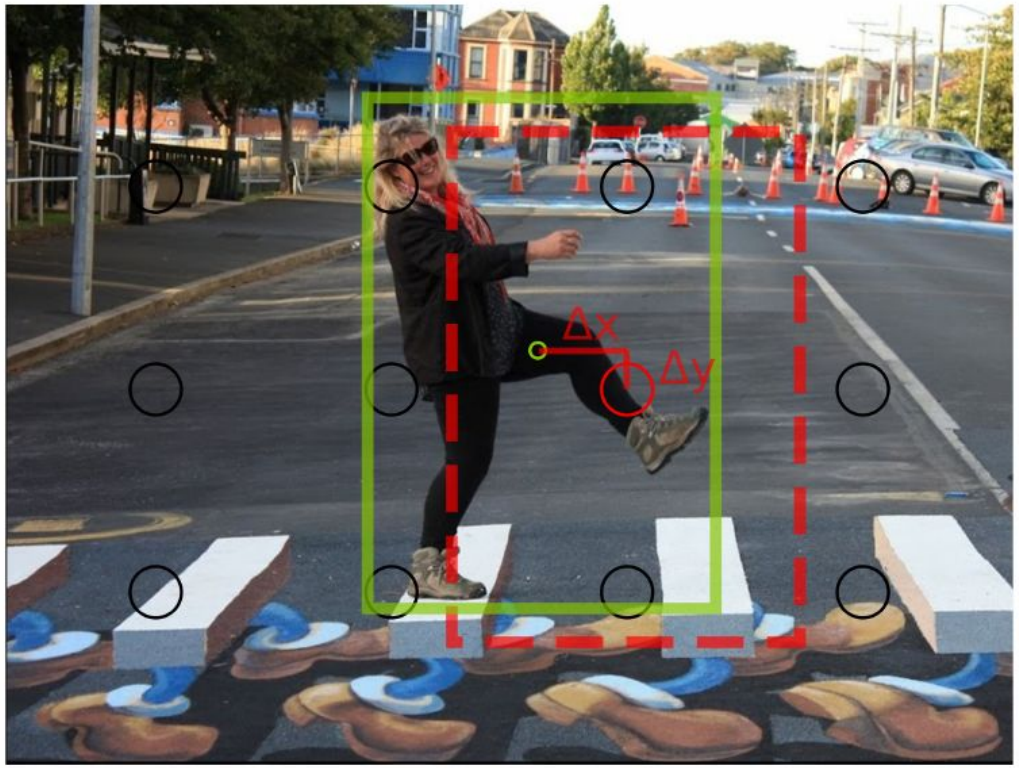


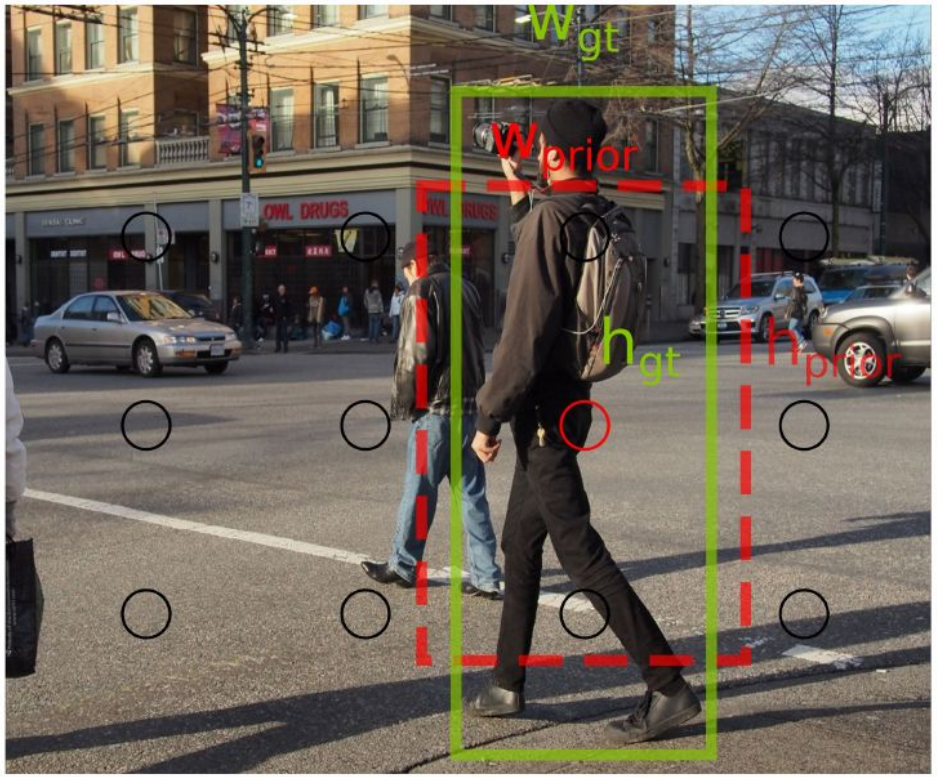


1. Same region proposal technique (selective search).
2. Feed full image to fully convolutional network, obtain feature map with size CxWxH.
3. For each proposal:
   a. Crop regions from feature map that corresponding to proposal using RoI Pooling. The result is fixed-size feature map of size approximately ($W_{image}/w_{proposal}$) and ($H_{image}/h_{proposal}$).
   b. Feed pooled result to fully connected layers, predict object class and bounding box regression (Δx, Δy, k_h, k_w).
   c. Adjust bbox with bounding box regression vector.
4. Apply Non Maximum Suppression.

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

BBox Regression





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

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

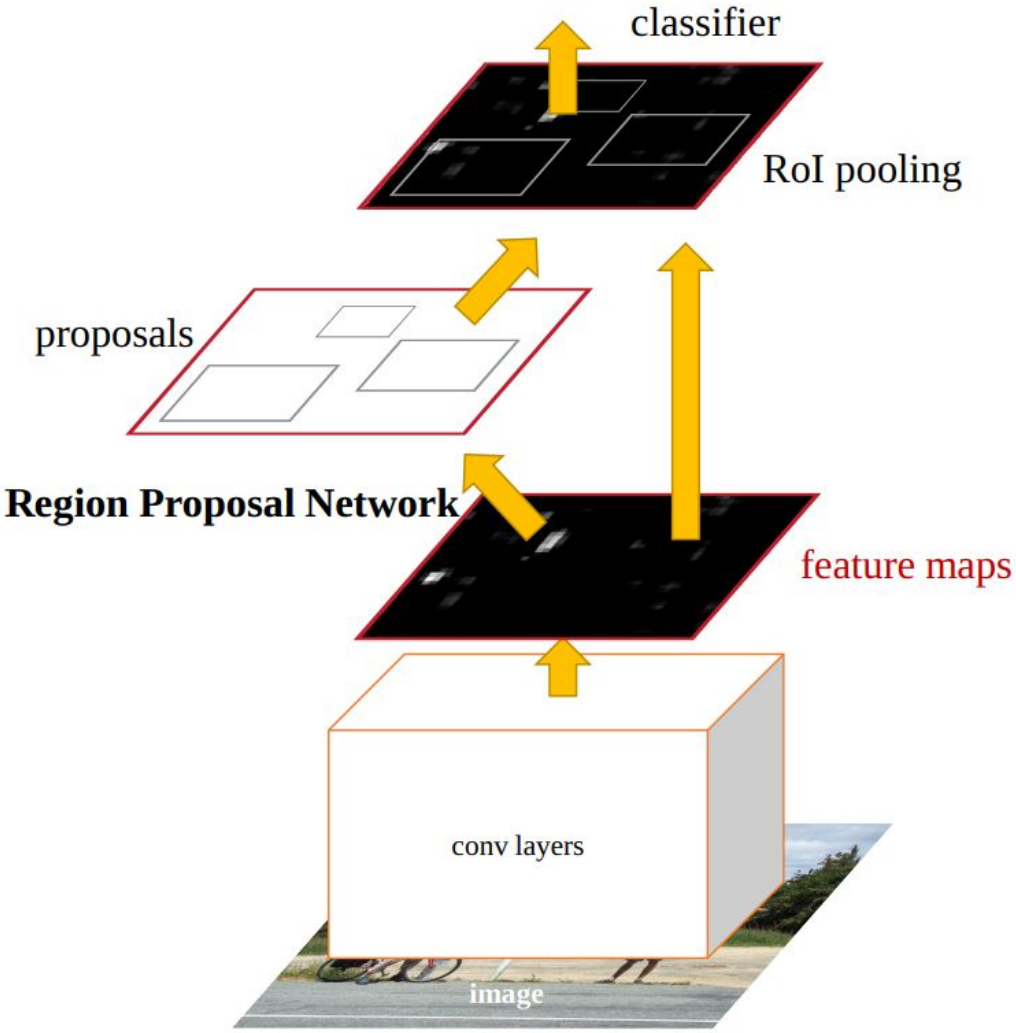
Fast RCNN Details

1. RoI Pooling layer is differentiable, so Fast RCNN can be trained end-to-end.
2. Both loss functions for classification and bounding box regression are optimized jointly.
3. Convolutions are applied to input image only once, but fully connected layers are applied to every pooled region proposal => authors use truncated SVD on fully connected layers weight matrices => 30% speedup.
4. VGG16 instead of AlexNet.

Main disadvantage of Fast RCNN: region proposal technique.

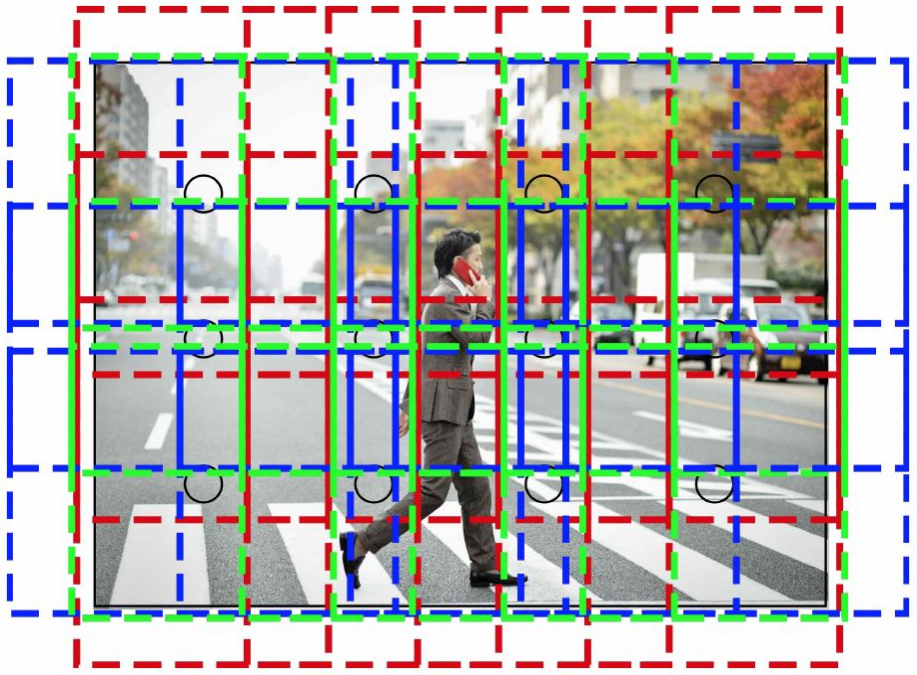
Faster RCNN

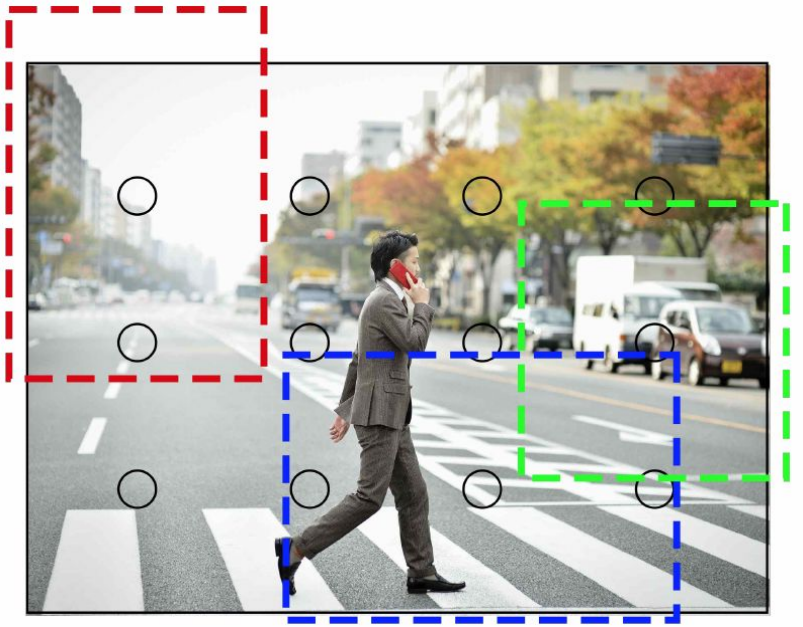
1. Additional network for object proposal generation on top of base fully convolutional part - Region Proposal Network (RPN).
2. Concept of Anchor Boxes with different size and aspect ratios.
3. RPN predicts “objectness”.
4. NMS is applied to generated proposals. Corresponding regions of conv feature map are passed to RoI pooling layer and the result is passed to Fast RCNN head.



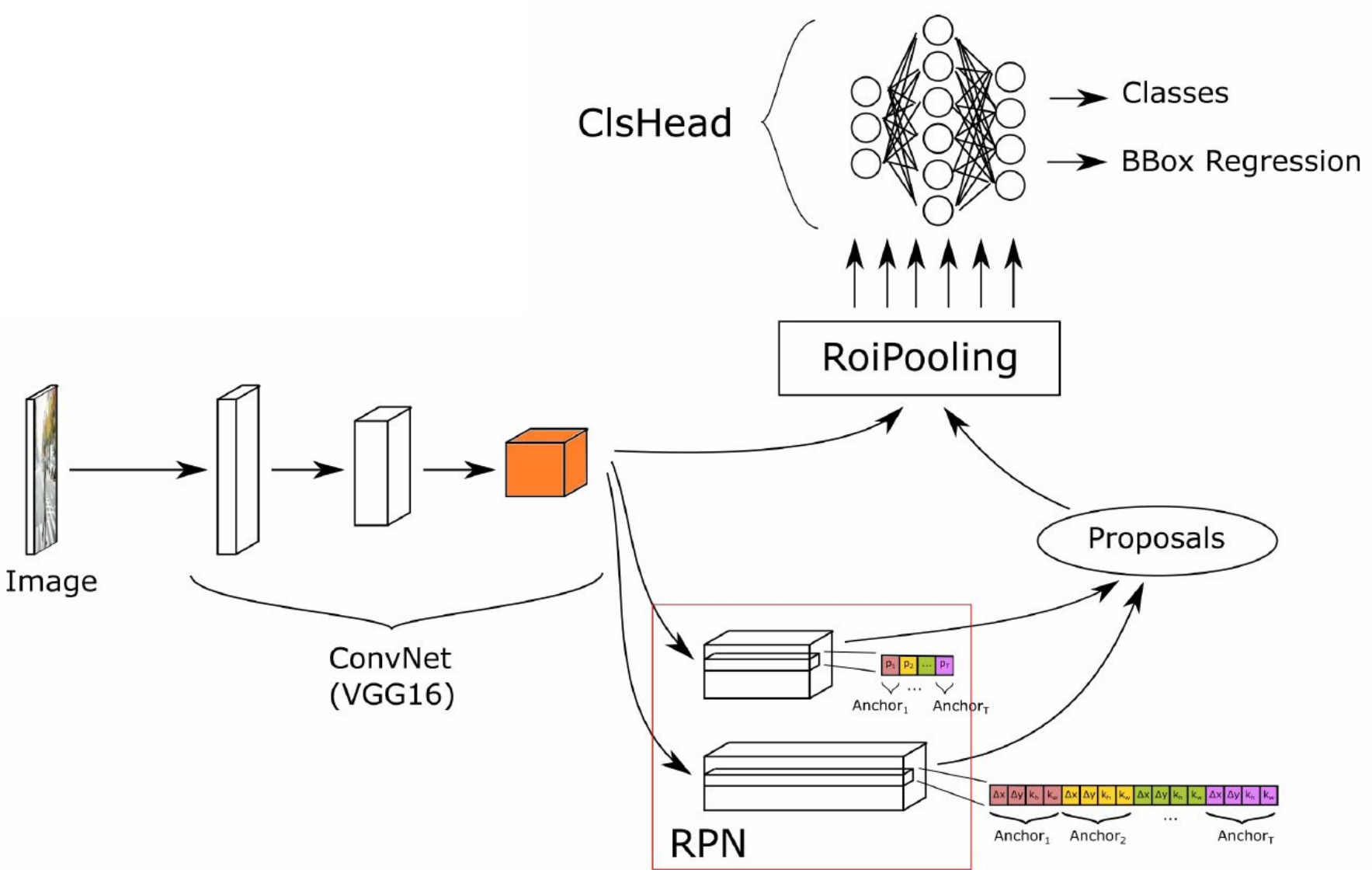
https://arxiv.org/pdf/1506.01497.pdf

Anchor Boxes





Faster RCNN


ClsHead
Classes
BBox Regression
RoiPooling
Image
ConvNet
(VGG16)
Proposals
RPN
Anchor1
AnchorT
Anchor1
Anchor2
AnchorT

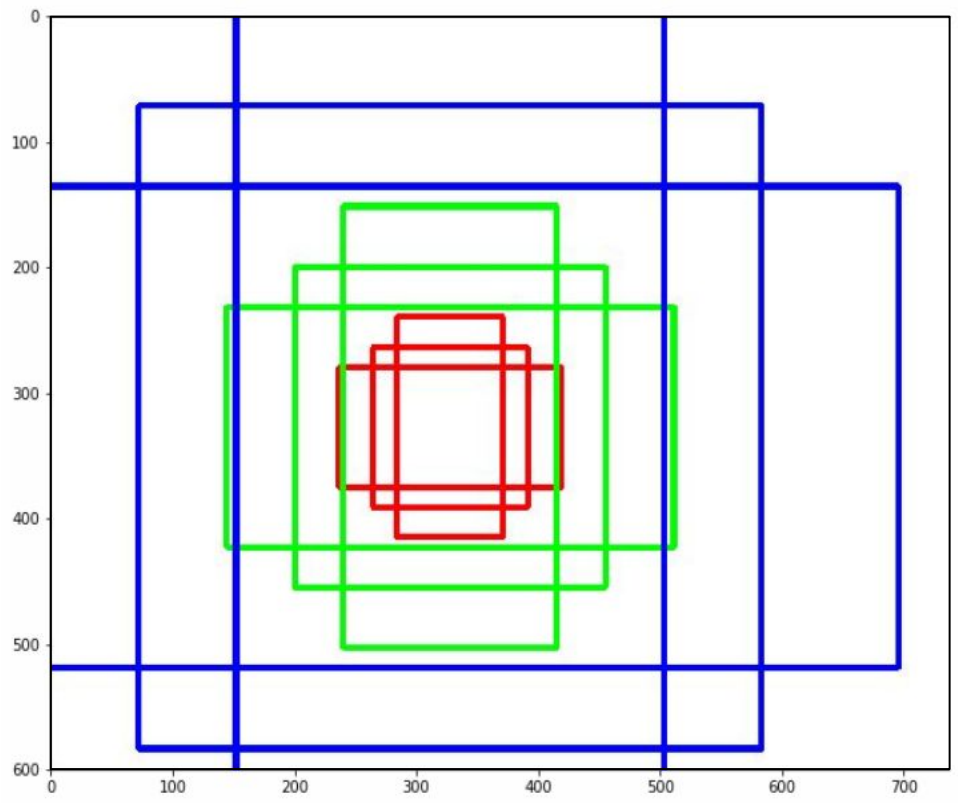
Scale problem

Problem:

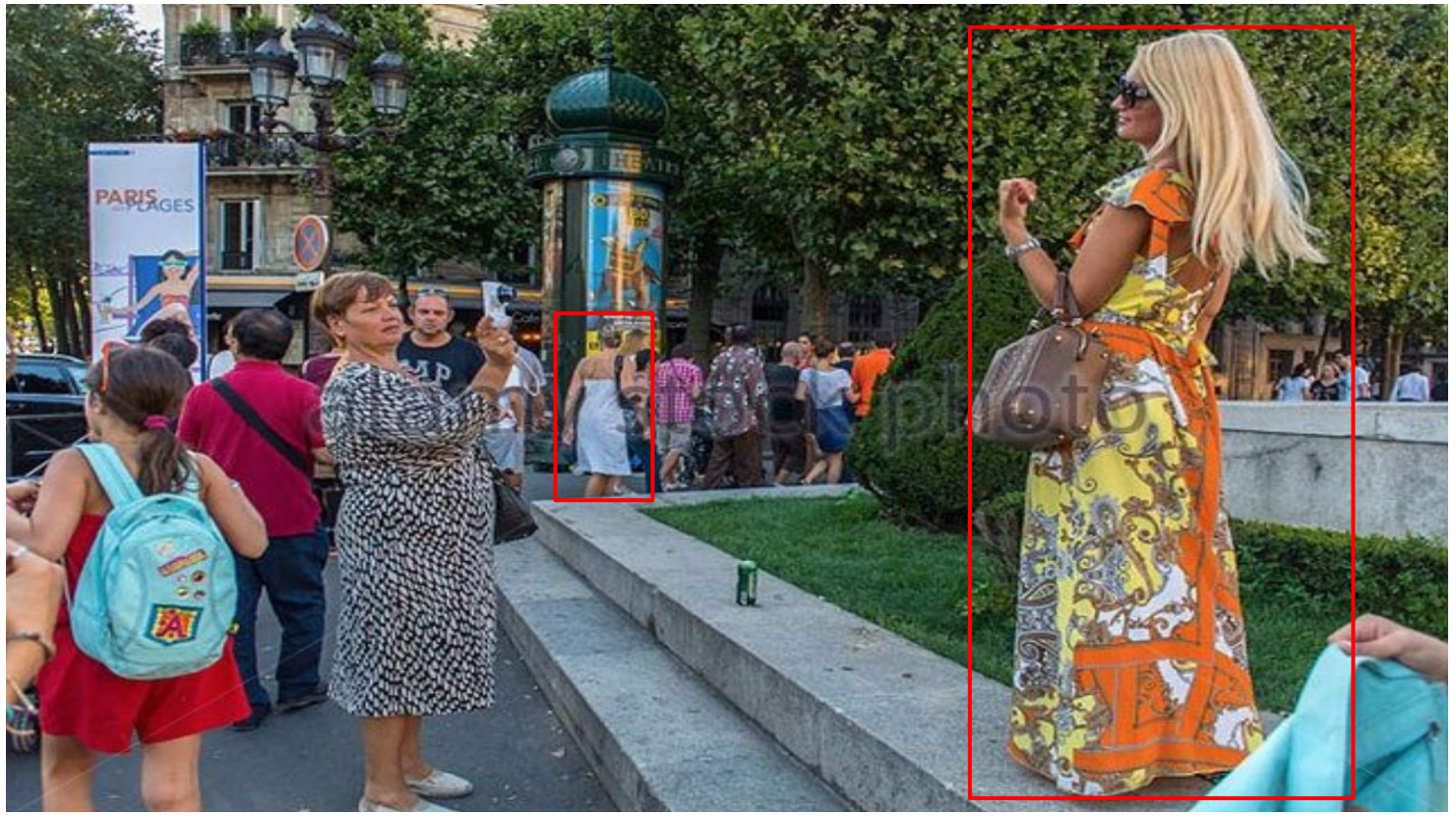
For each “location” we have same number of big and small boxes. Smaller boxes are more “sparse”. But the image can fit more small objects than big ones.

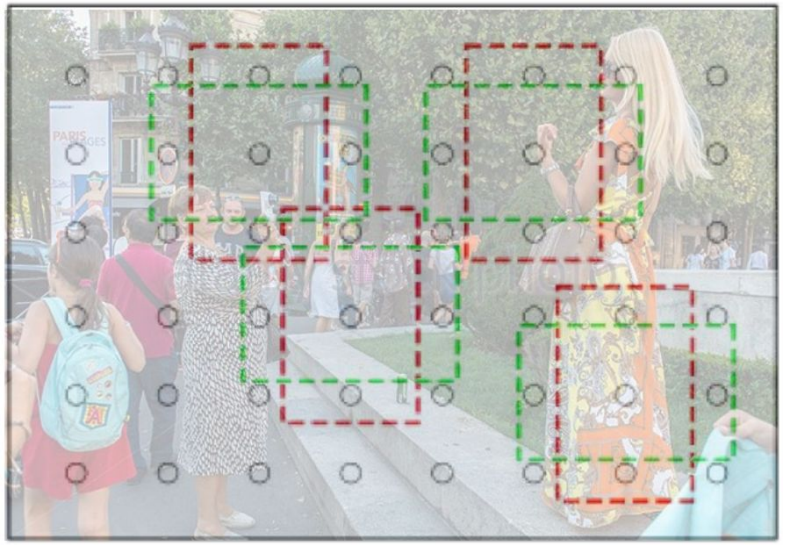
Solution:

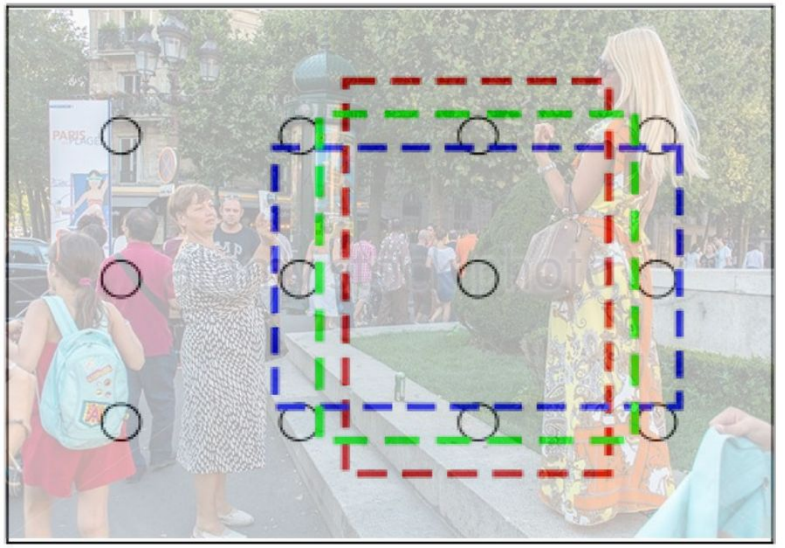
Let’s detect at different scales!



Different Scales

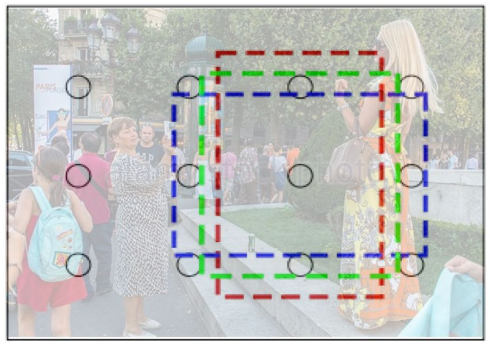


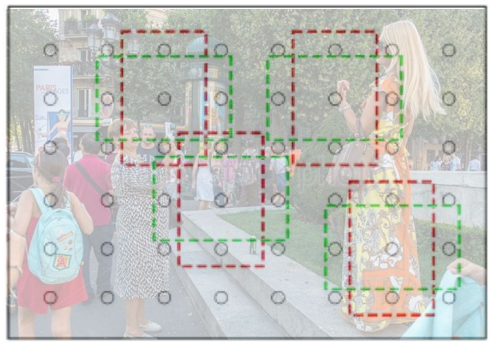


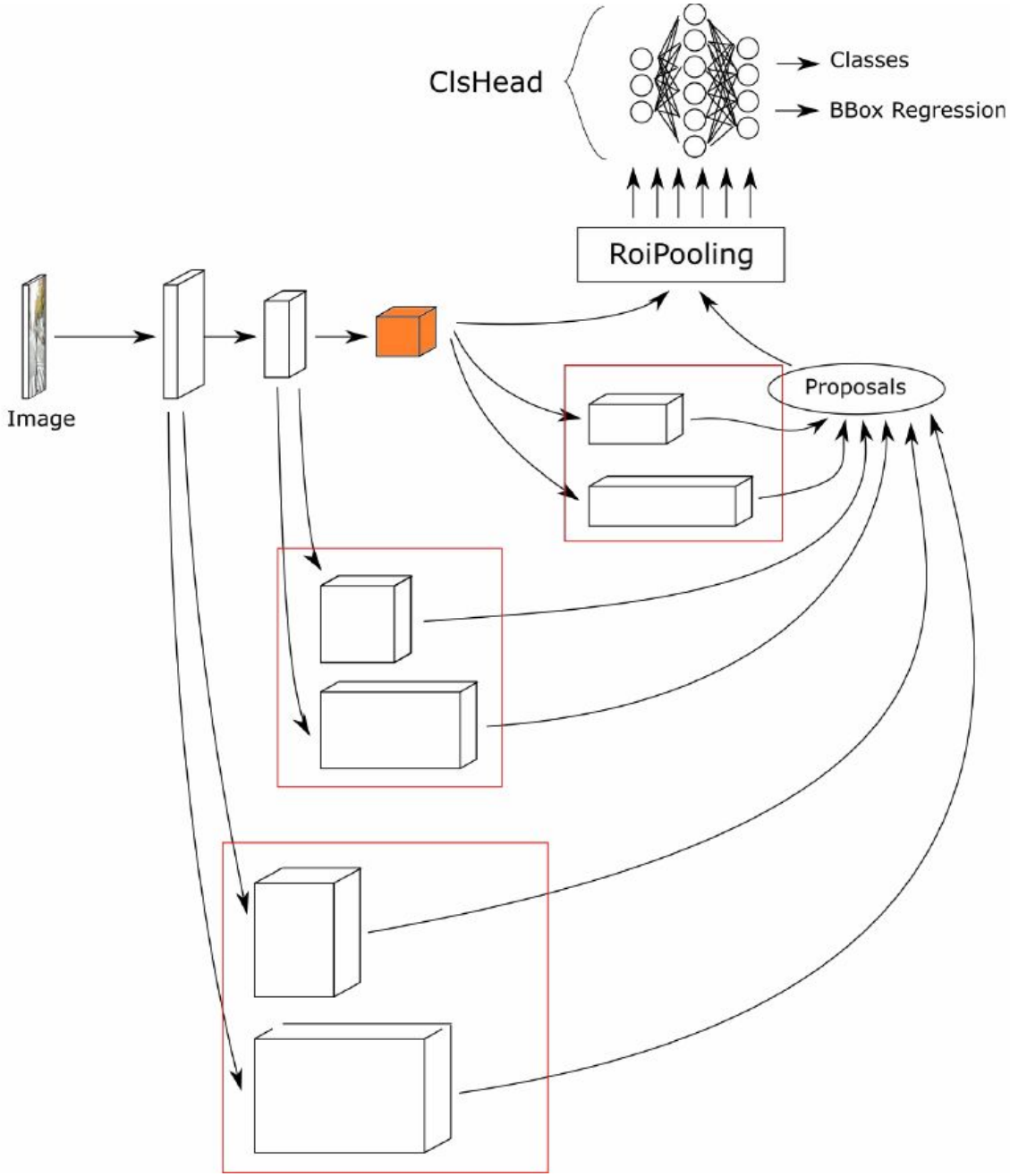


Multiple Scales

1. Previous CNN tensors are used for proposal generation
2. Proposals from different scales are stored in one array.



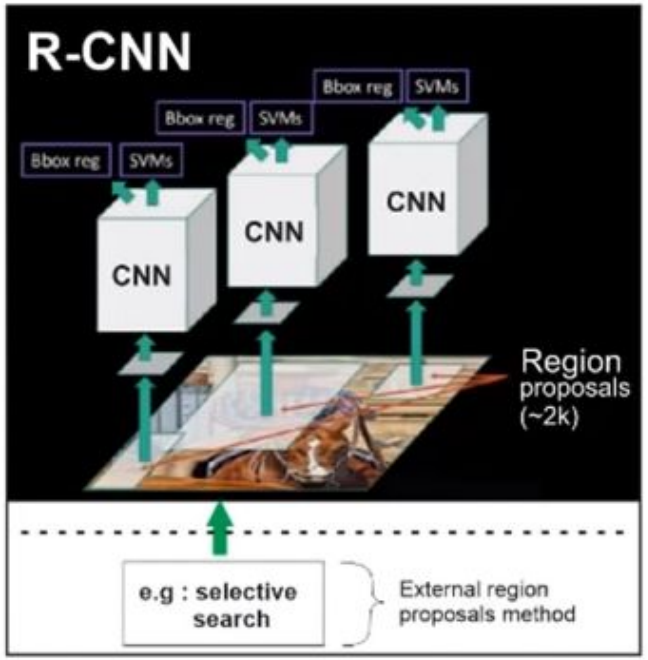


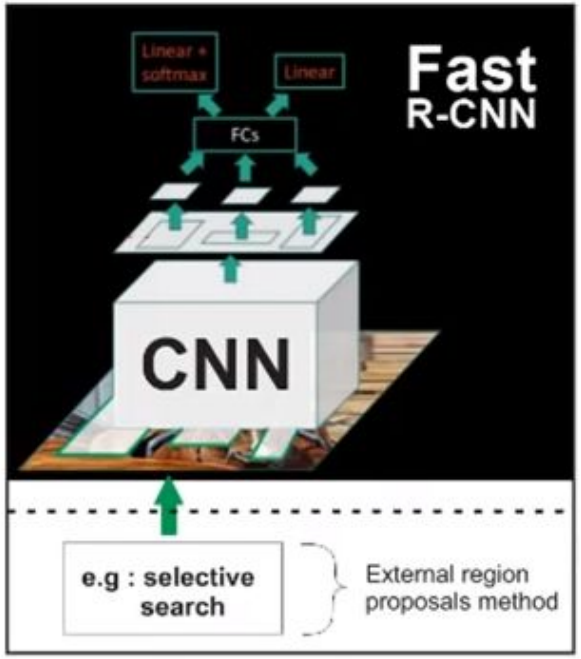


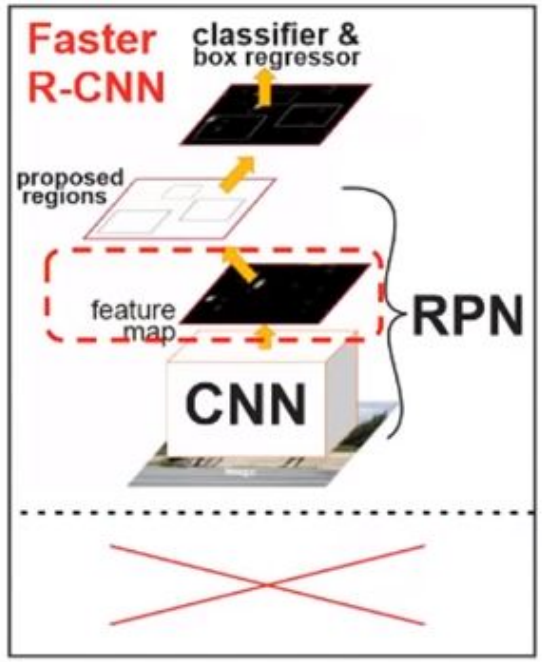
Faster RCNN Details

1. RPN can be viewed as “attention” mechanism of the network.
2. Classification and Regression are predicted in RPN in superpixel-wise manner.
3. Still used. Mask RCNN is based on Faster RCNN with resnet as a backbone.
4. R-FCN is a further development of Faster RCNN.

RCNN / Fast RCNN / Faster RCNN





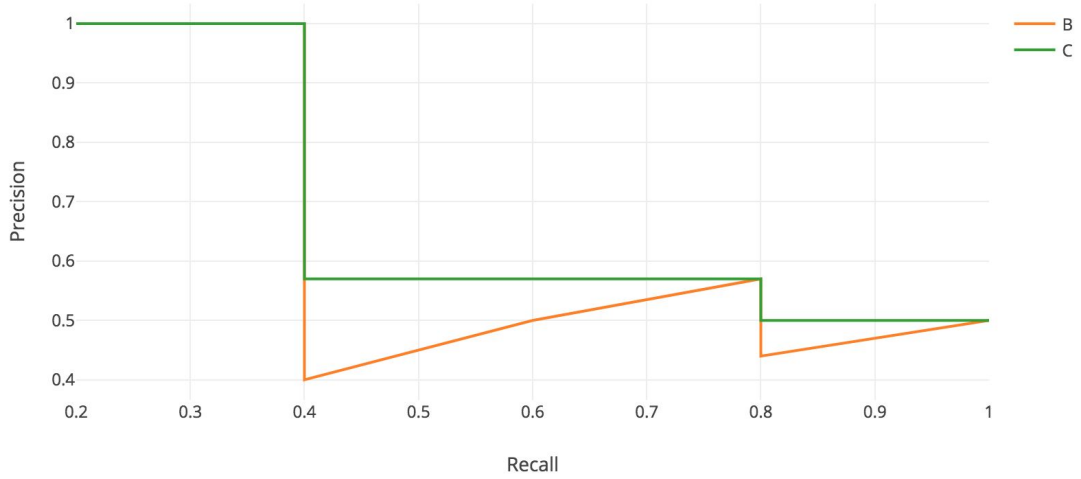


	R-CNN	Fast R-CNN	Faster R-CNN
Test time per image	50 seconds	2 seconds	0.2 seconds
Speed-up	1x	25x	250x
mAP (VOC 2007)	66.0%	66.9%	66.9%

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

mean Average Precision (mAP)

The general definition for the Average Precision (AP) is finding the area under the precision-recall curve above.



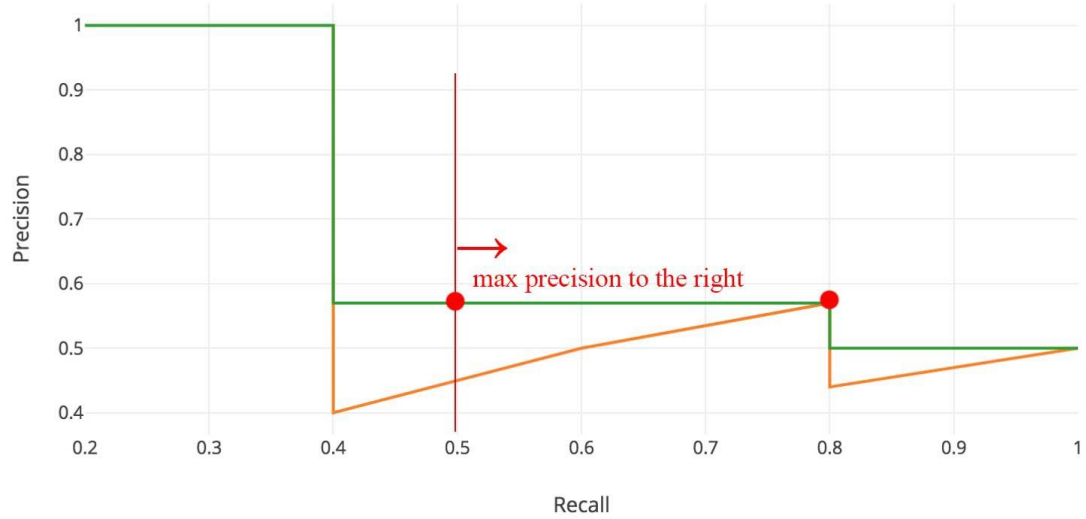
Vary threshold => obtain different Precision and Recall.

- Orange line - PR curve
- Green line - maximum Precision we can get for specific Recall level.

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

Average Precision for specific class: average maximum Precision we can get for Recall values in [0.0, 0.1, ..., 0.9, 1.0]

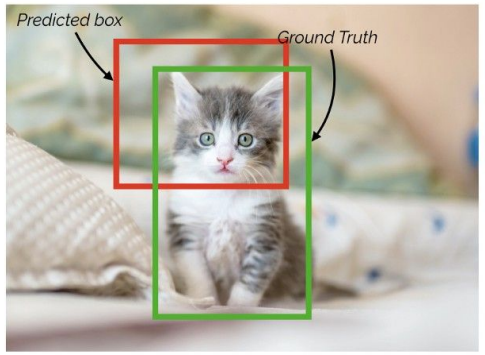
mAP metric is mean of APs for all classes in dataset.



mean Average Precision (mAP)

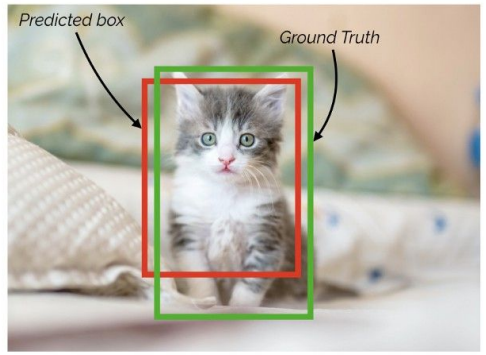
If IoU threshold = 0.5

False Positive (FP)

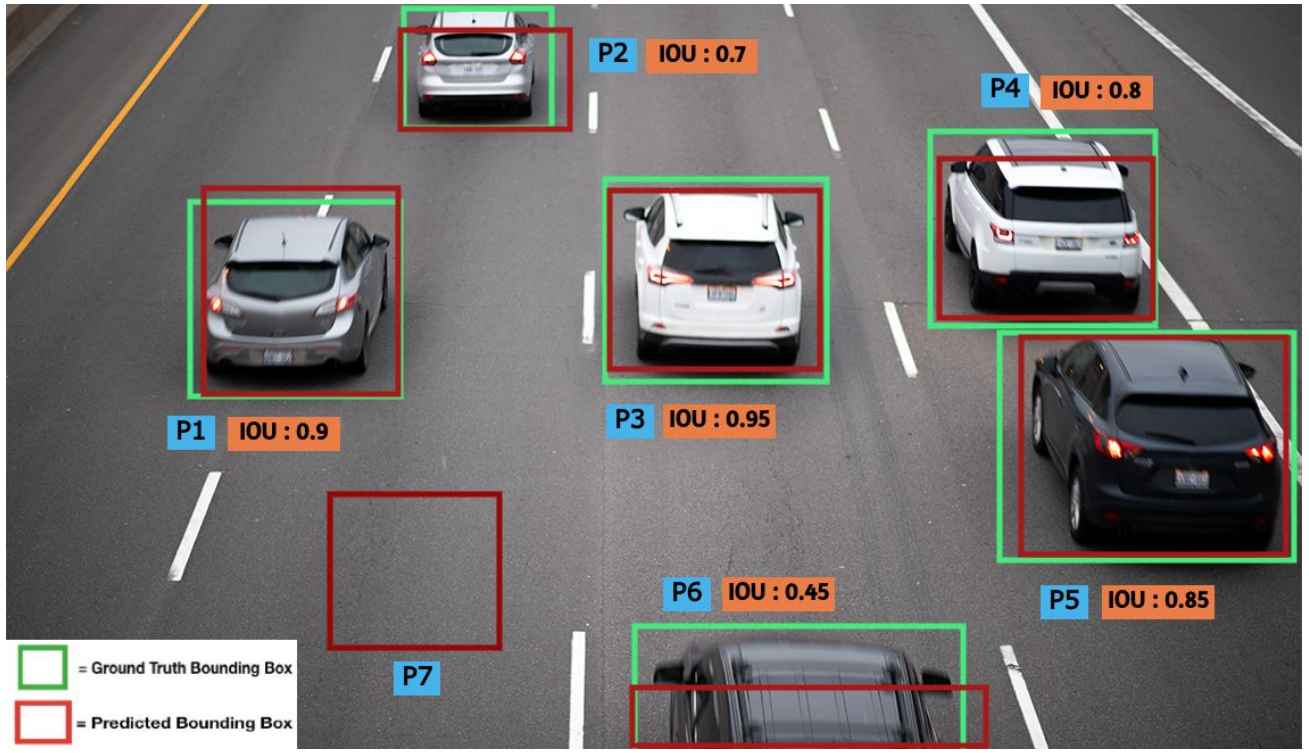


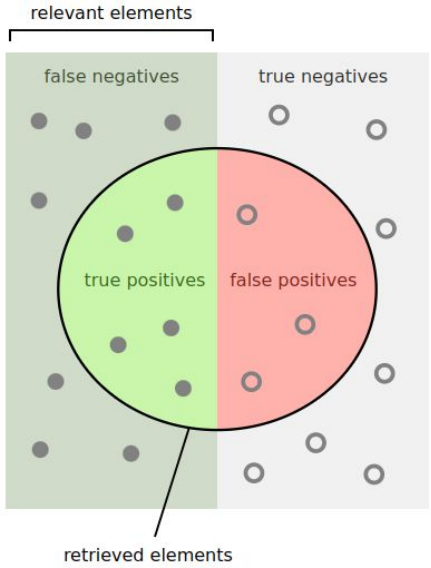
IoU = ~0.3

True Positive (TP)



IoU = ~0.7





How many retrieved items are relevant?

$$\text{Precision} = \frac{\text{true positives}}{\text{retrieved elements}}$$

How many relevant items are retrieved?

$$\text{Recall} = \frac{\text{true positives}}{\text{relevant elements}}$$

If IoU threshold = 0.8 then precision is 66.67%. (4 out of 6 are considered correct)

If IoU threshold = 0.5 then precision is 83.33%. (5 out of 6 are considered correct)

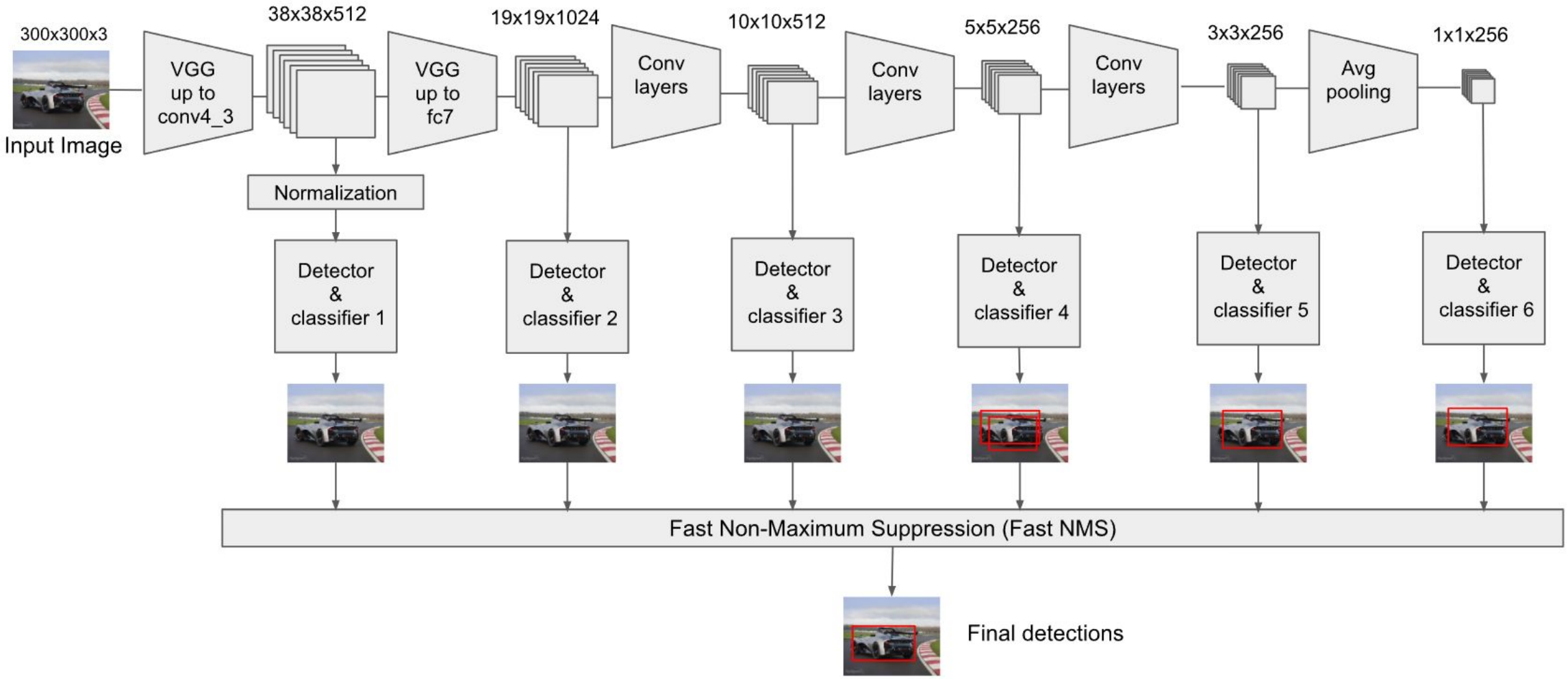
If IoU threshold = 0.2 then precision is 100%. (6 out of 6 are considered correct)

Single shot approaches

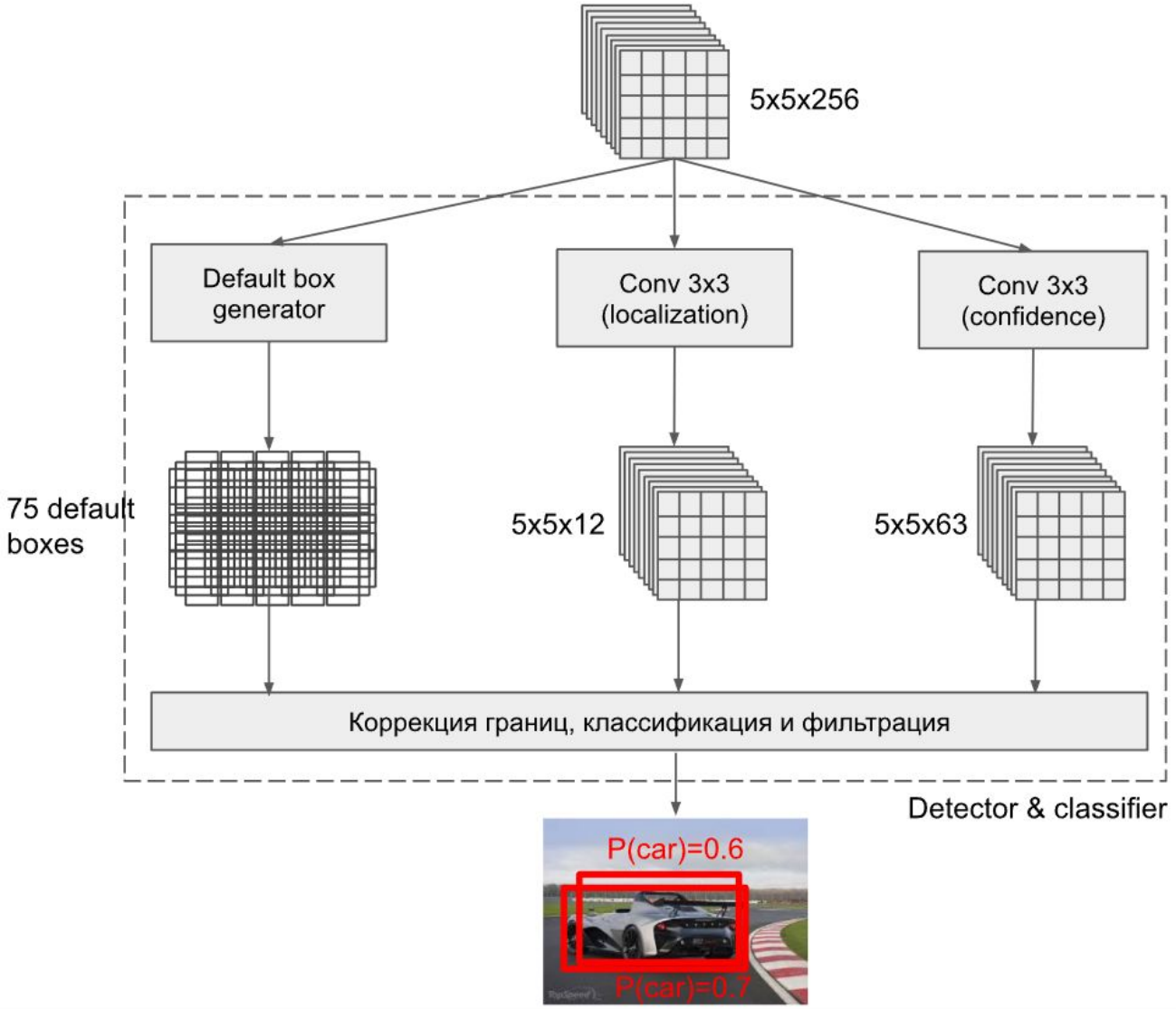
Single shot approaches

1. Main idea: No region proposals, no RoI pooling. One network to predict classification and regression scores for each possible object location (anchor-boxes).
2. BTW, should not be confused with one-shot learning.

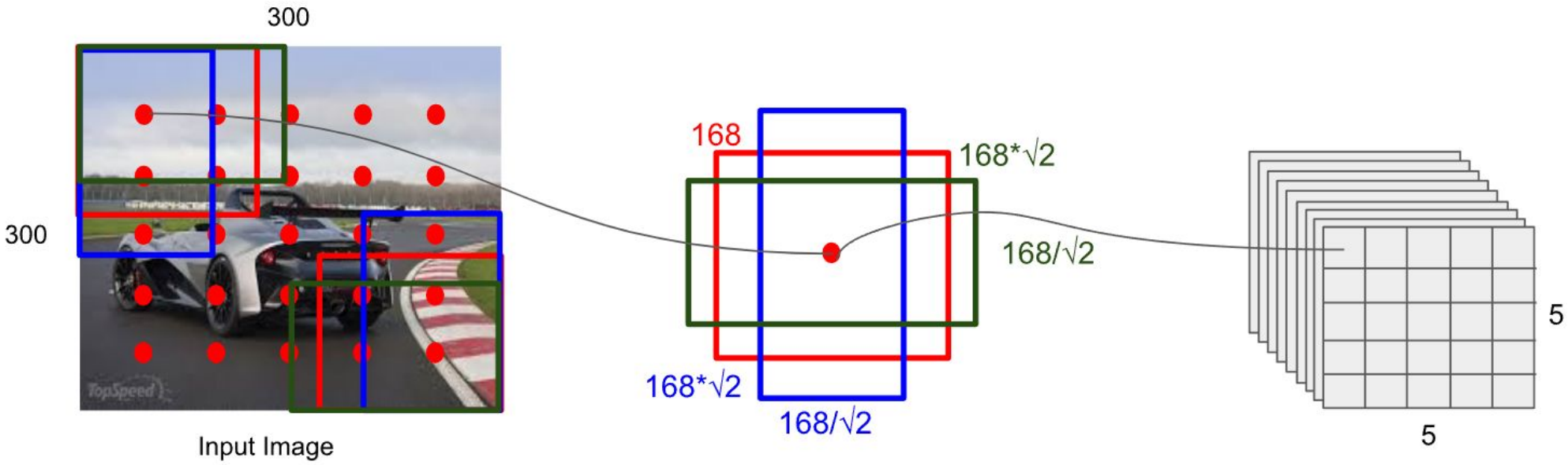
Single Shot Detector (1)



Single Shot Detector (2)

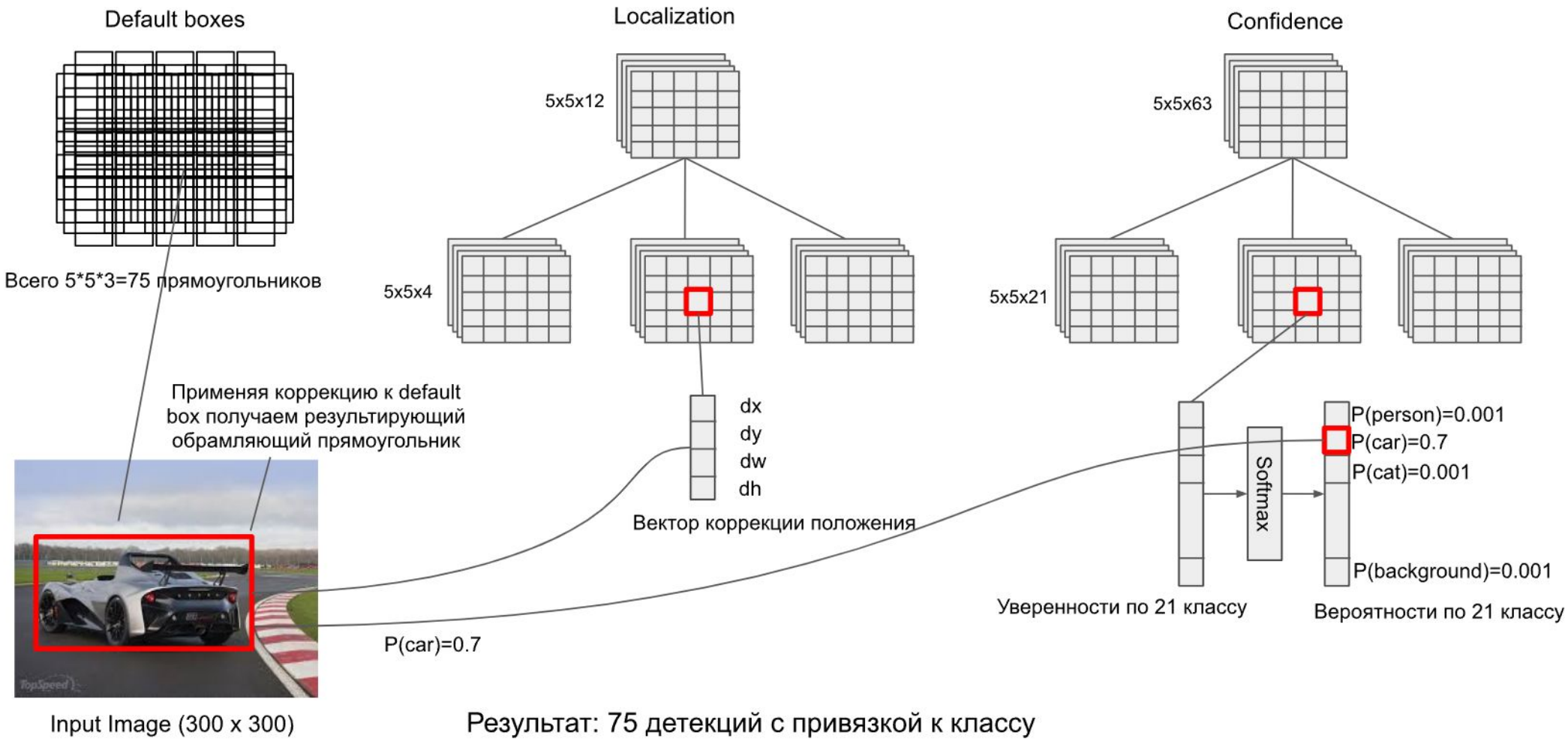


Single Shot Detector (3)

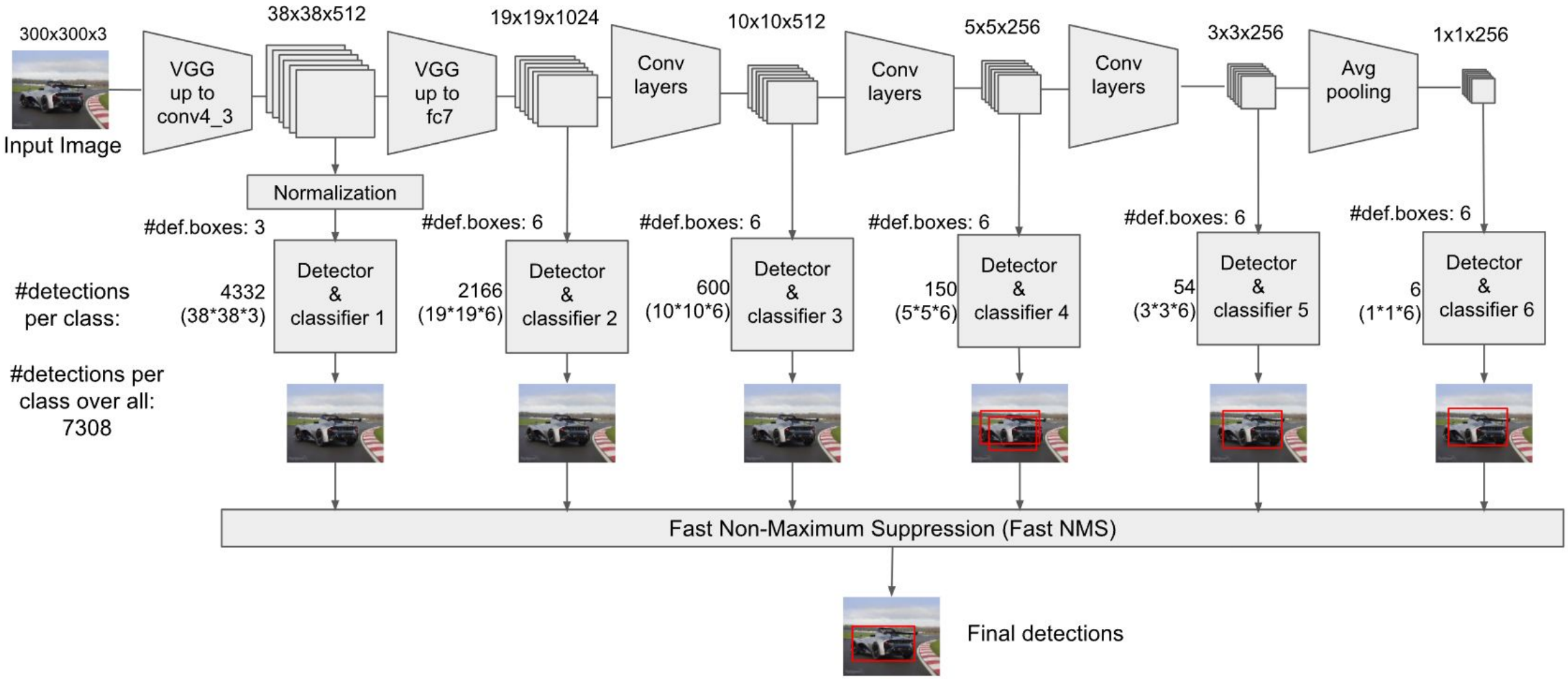


Default box

Single Shot Detector (4)

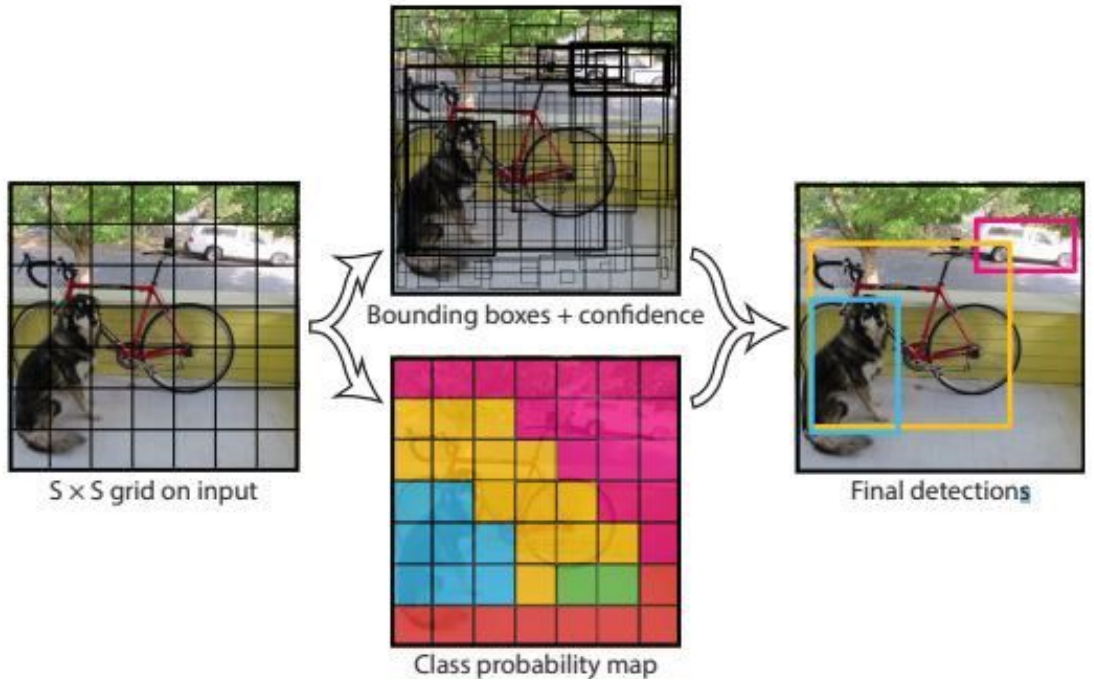


Single Shot Detector (5)



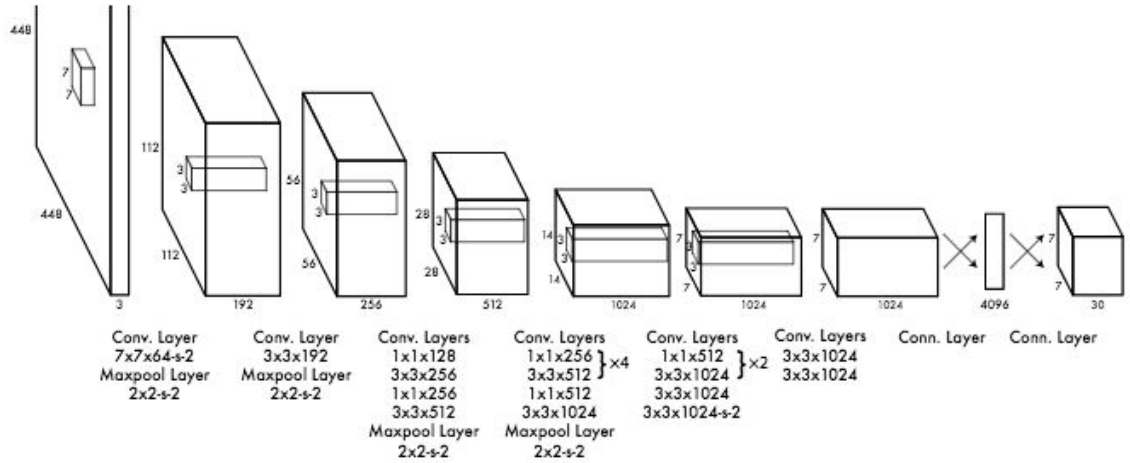
YOLO (You Only Look Once)

1. Tensor of size SxSx(B*5+C). B is amount of anchor boxes for each cell. C is amount of classes. (B=2, S=7, C=20)
2. Realtime performance
3. “Darknet” framework



Further reading on YOLO:

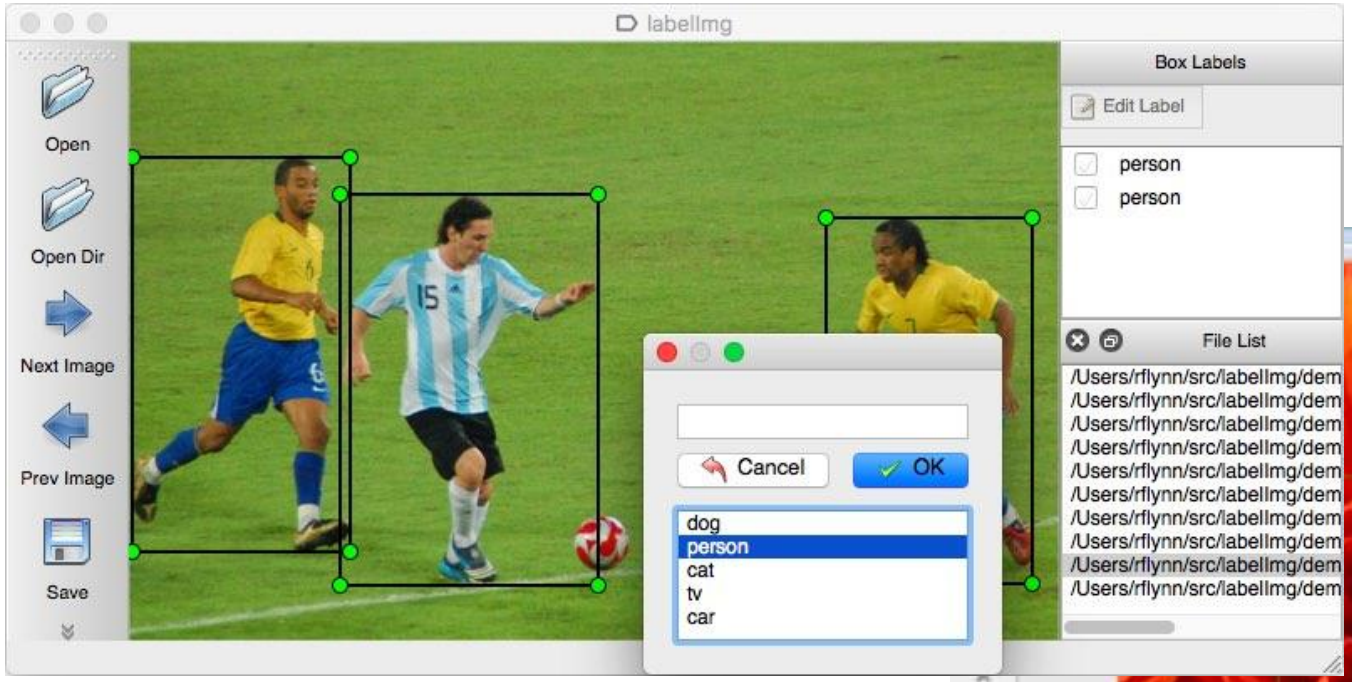
1. YOLO v2 (a.k.a. YOLO9000): more classes.
2. YOLO v3: log. reg. for each class. Helps with overlapping objects. Multiscale anchor boxes.
3. YOLO v4: lots of small improvements
4. YOLO v5: very good quality/performance balance.

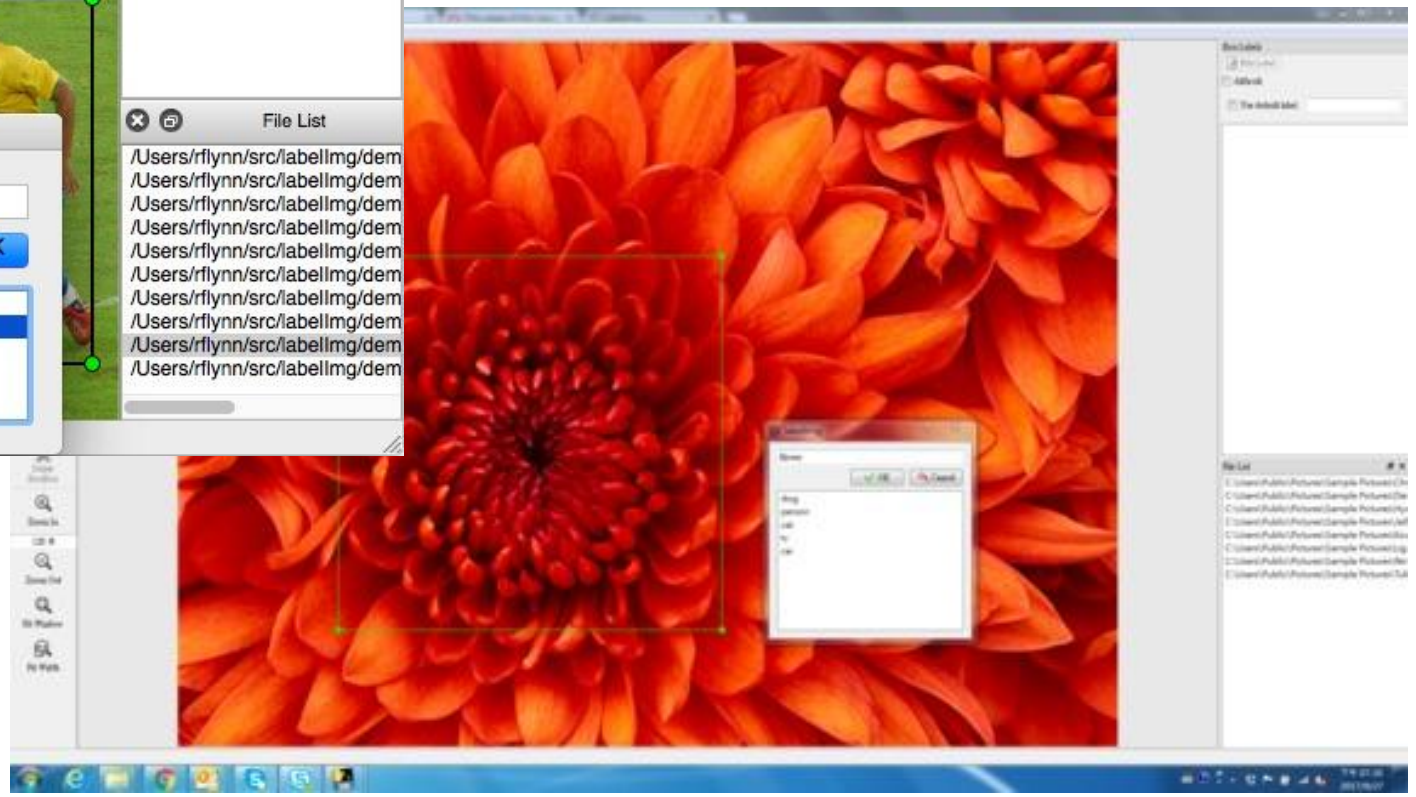


https://arxiv.org/pdf/1506.02640.pdf
https://pjreddie.com/publications/yolo/

Формирования разметки

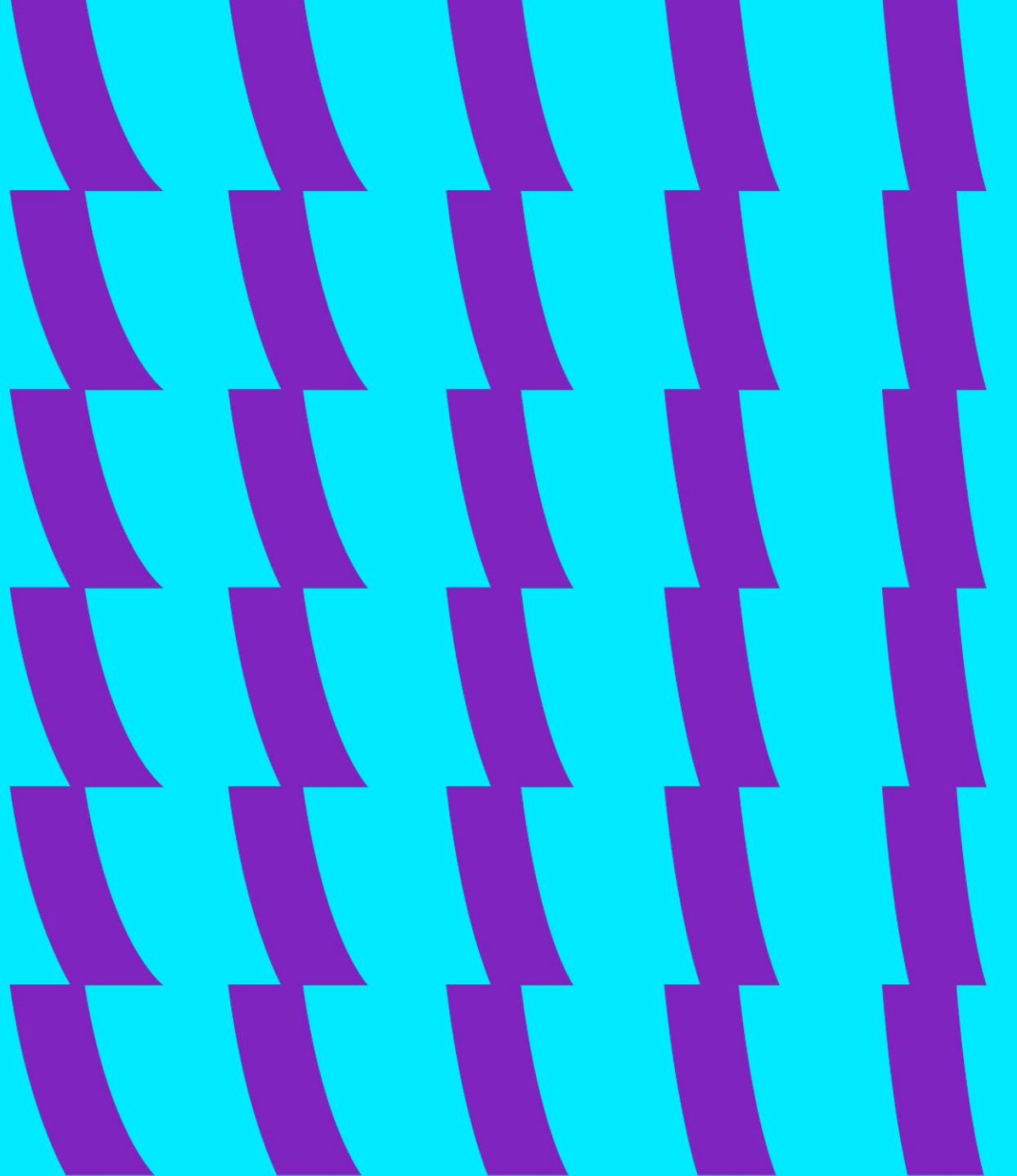
LabelImg





https://github.com/heartexlabs/labelImg

Вопросы?



Спасибо
за внимание!