

Нейронные сети

Детекция объектов
на изображениях и
сегментация изображений

Спасёнов Алексей



Не забывайте
отмечаться и
оставлять
отзыв



Содержание лекции

1. Задача детекции
2. Задача сегментации



Часть 1. Задача детекции

Часть 1. Задача детекции

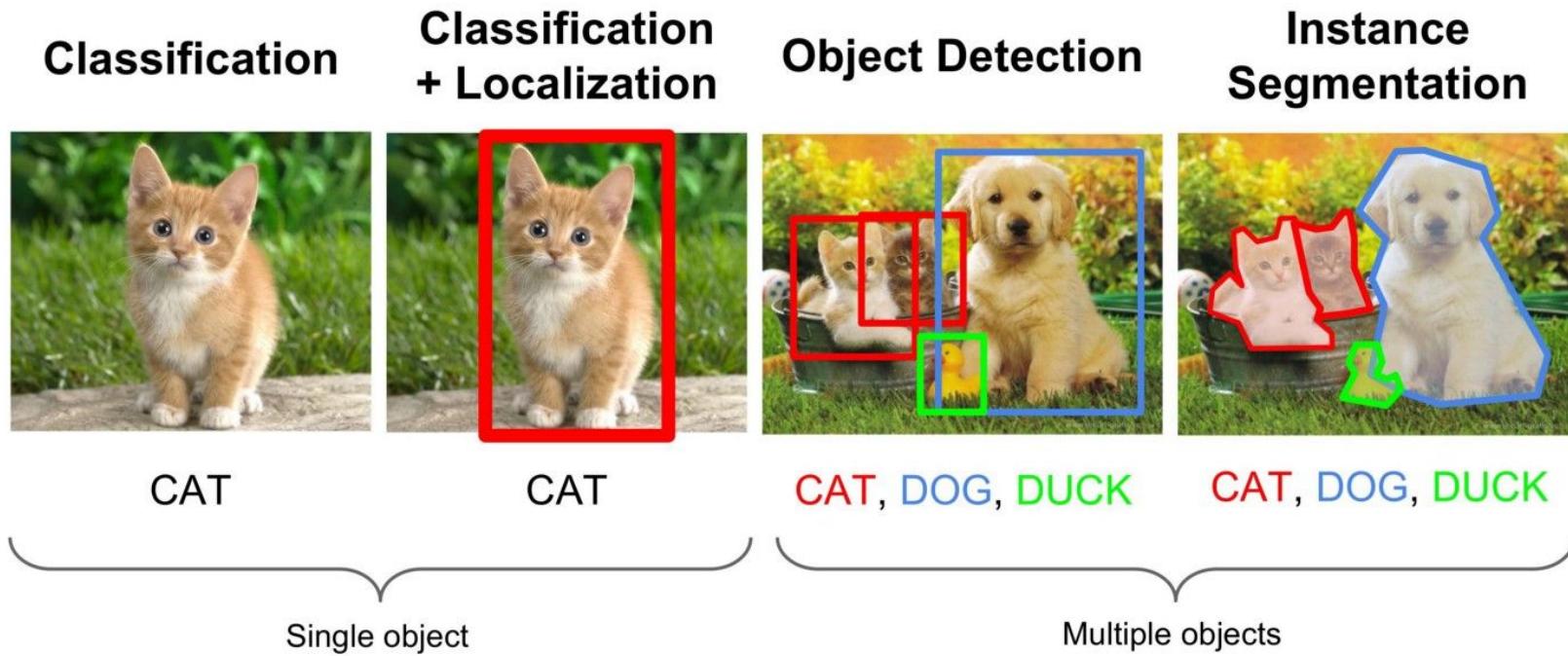
1. Задача детекции объектов на изображении
 - a. Проблемы
 - b. Метрики
2. Методы классического CV
3. Region-based детекция
 - a. RCNN
 - b. Fast RCNN
 - c. Faster RCNN
4. Single-shot детекция
 - a. SSD
 - b. YOLO
5. Формирования разметки



Задача детекции объектов на изображении

• • • •

Задача детекции объектов на изображении

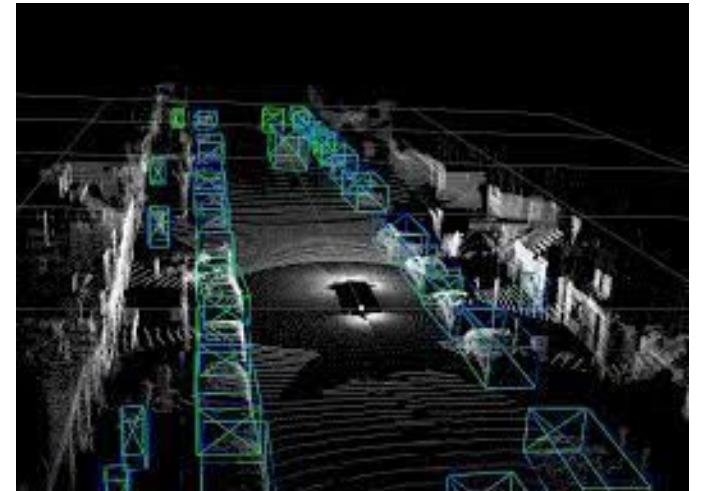
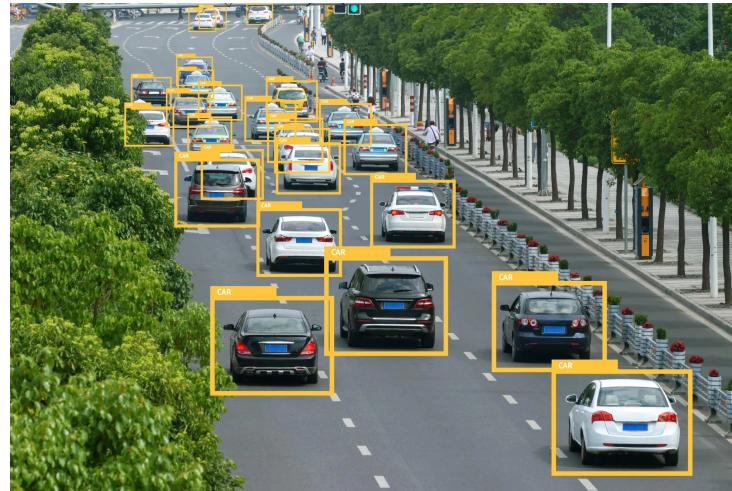
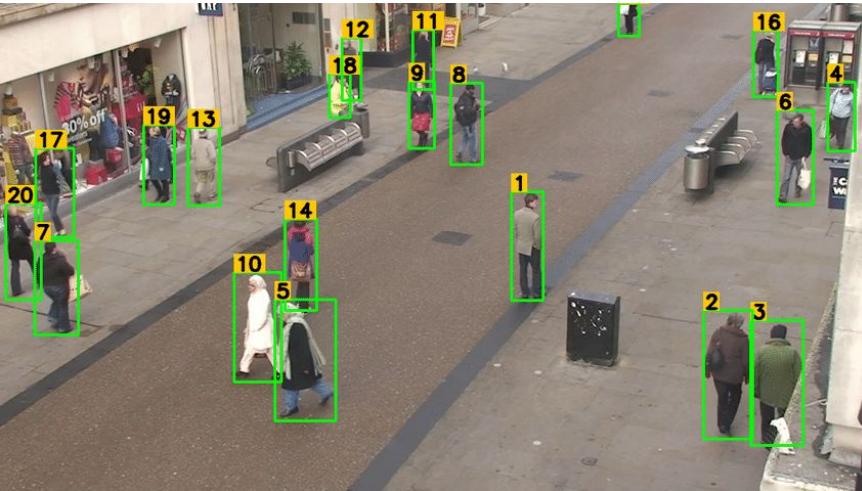
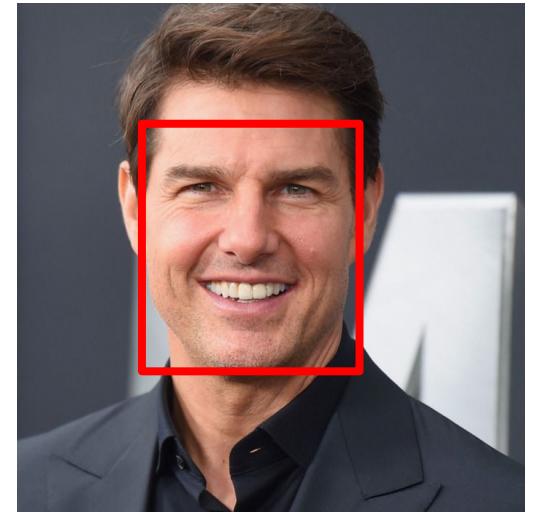


Отличия от задачи классификация:

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



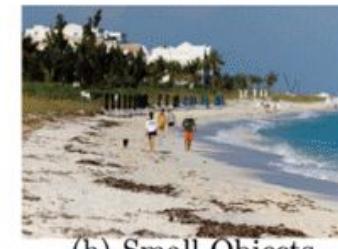
(e) Clutter, Occlusion



(f) Blur



(g) Motion



(h) Small Objects, Low Resolution



(i) Different instances of the “chair” category



(j) Small Interclass Variations: four different categories

Методы классического компьютерного зрения

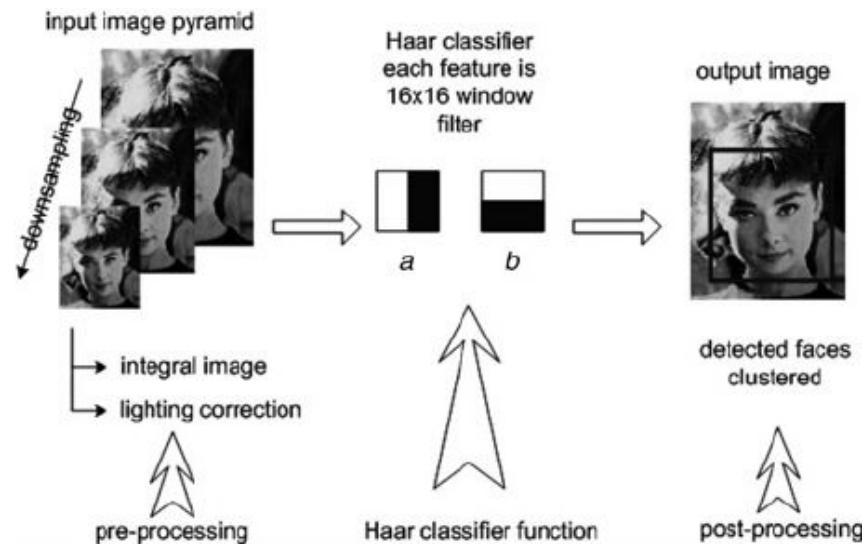
• • • •

Методы классического CV

1. HOG



2. Haar Cascades



Idea: Sliding window + image pyramid

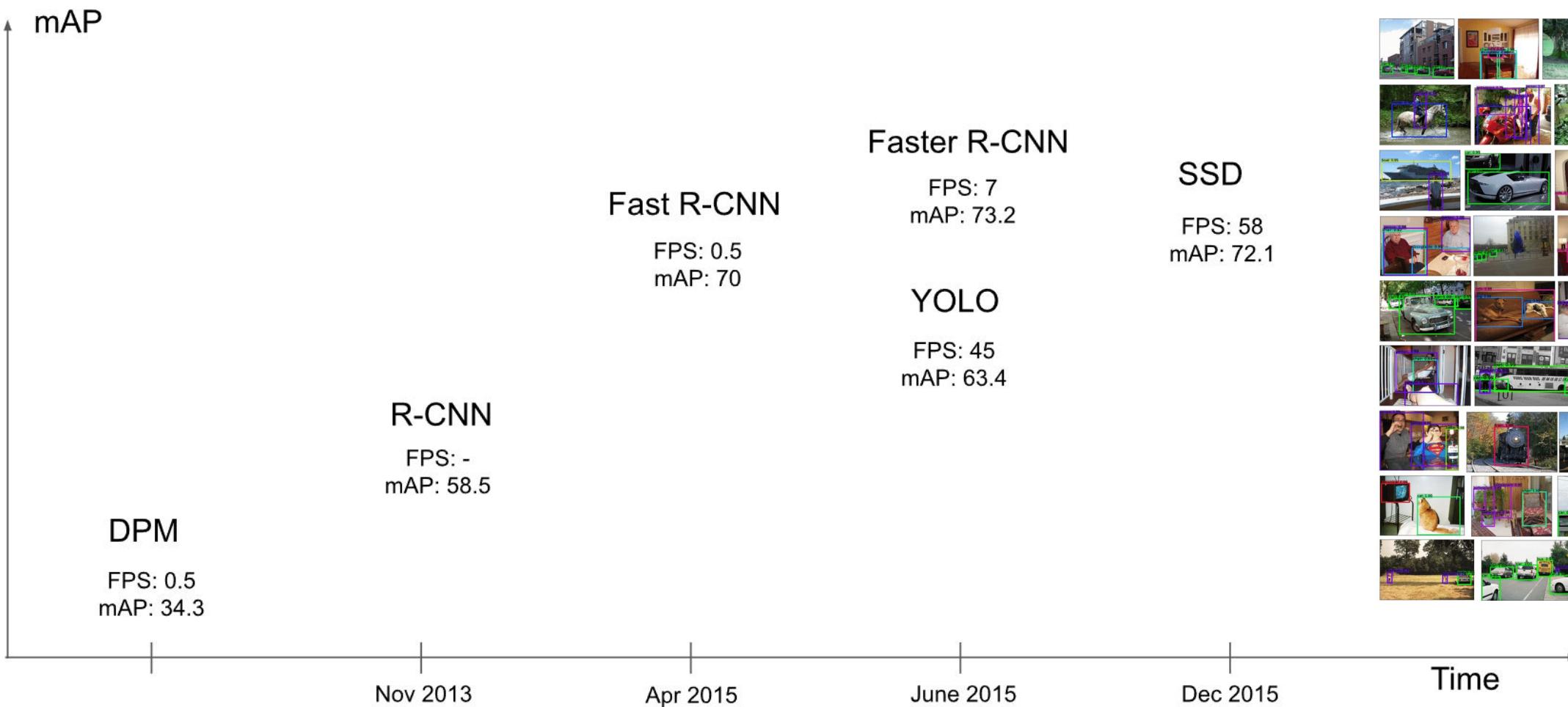
3. SIFT



Region-based детекция

• • • •

Развитие подходов



<https://paperswithcode.com/sota/object-detection-on-coco>
<https://cocodataset.org/#home>

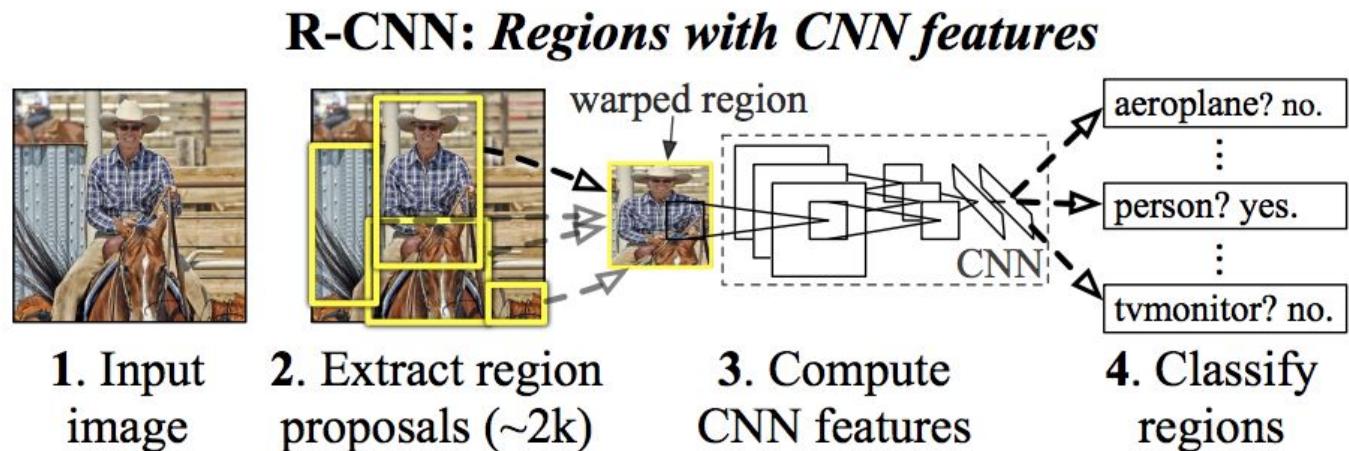
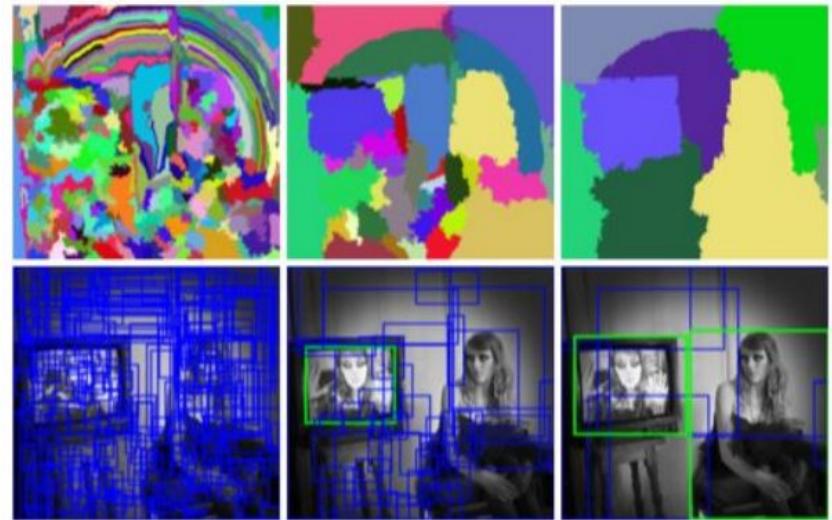
<http://host.robots.ox.ac.uk/pascal/VOC/voc2007/>

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. ...

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)



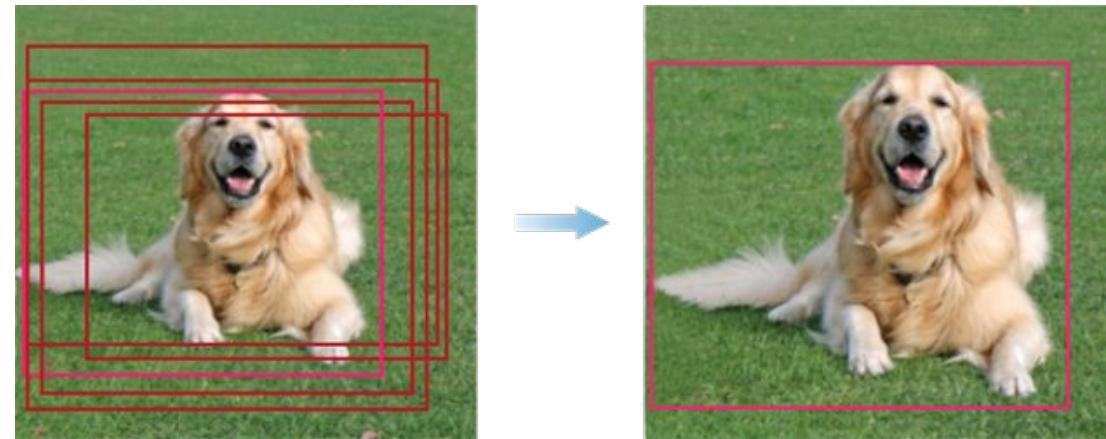
<https://arxiv.org/pdf/1311.2524v5.pdf>

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:

- 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

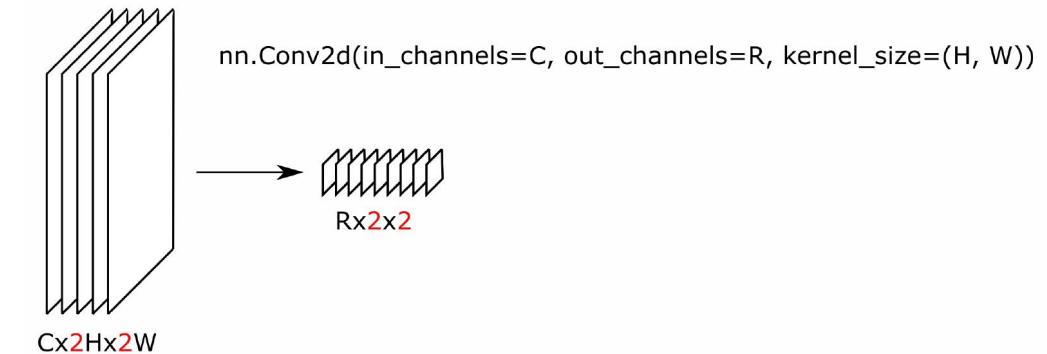
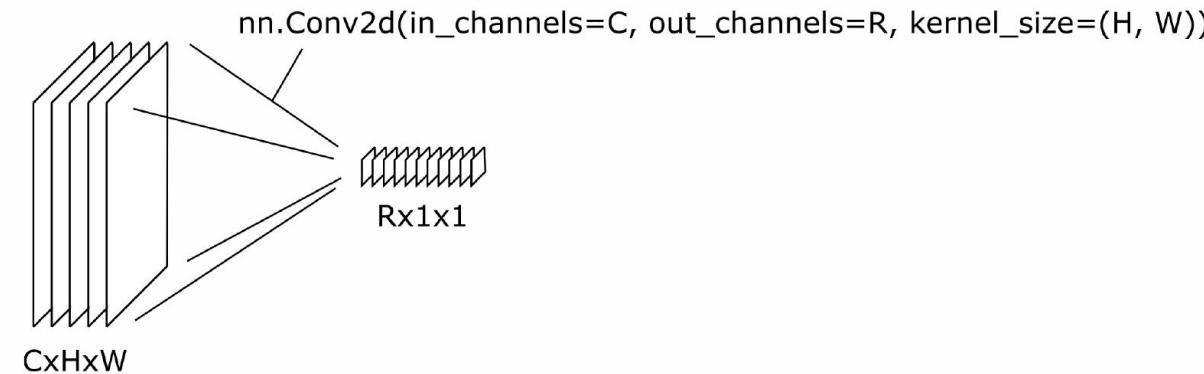
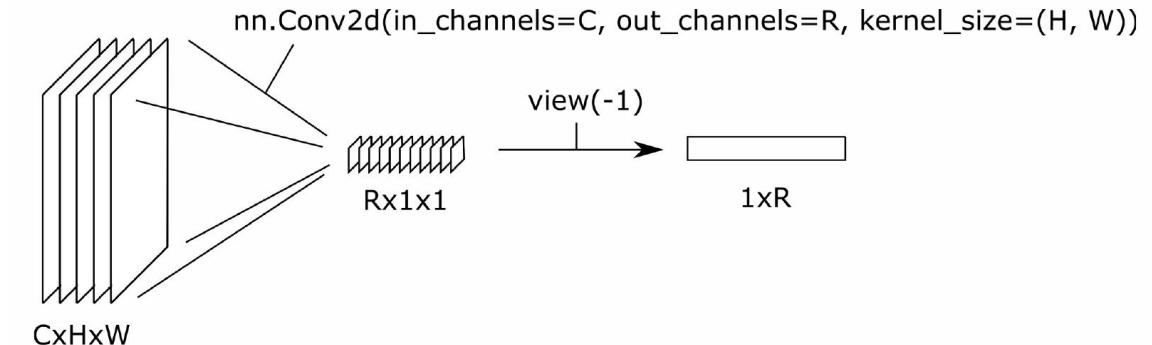
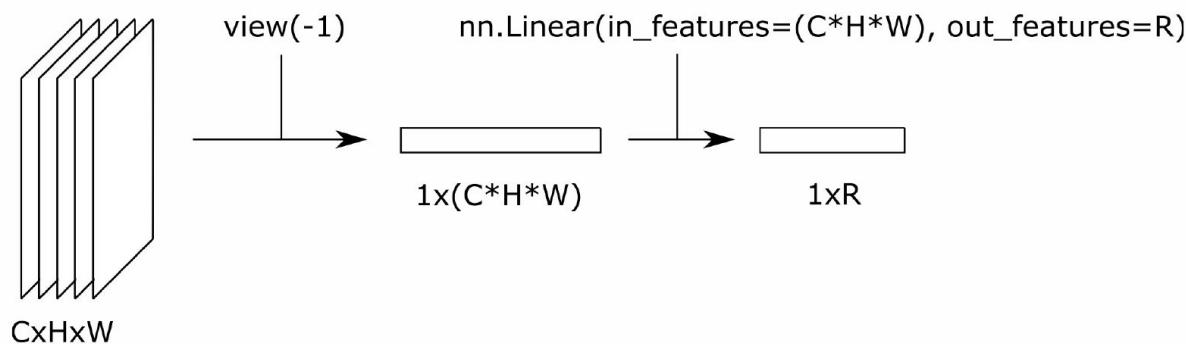
Плюсы:

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

Минусы:

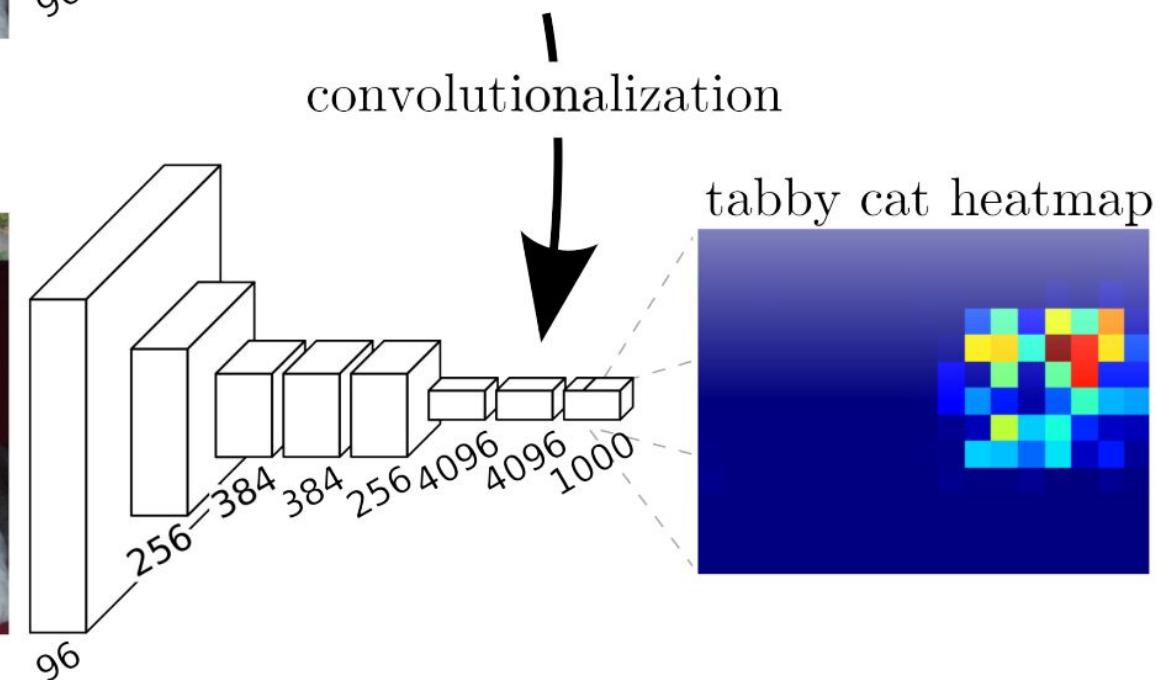
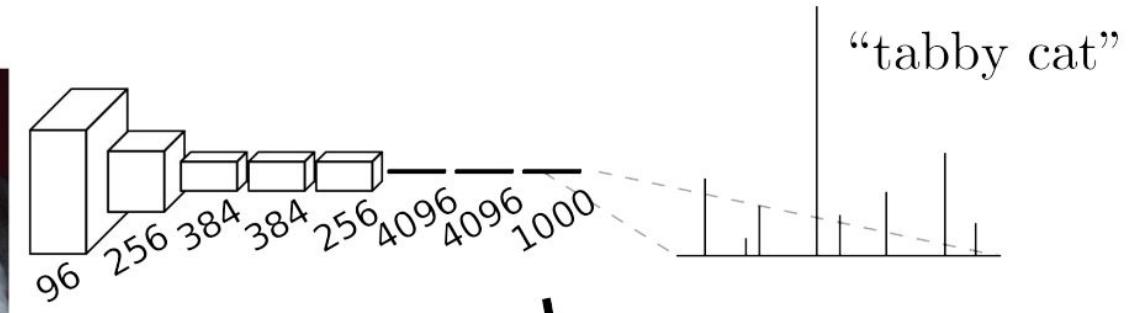
1. Крайне медленный: 13s/изображение на GPU or 53s/изображение на CPU.
~2000 region proposals. Дублирование расчетов в случае совпадающих region proposals.
2. Модель масштабирует изображение до определенного стандартного размера, может теряться информация о размере объекта / соотношении сторон.
3. Сложный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

Имитация линейного слоя с помощью Conv2d?

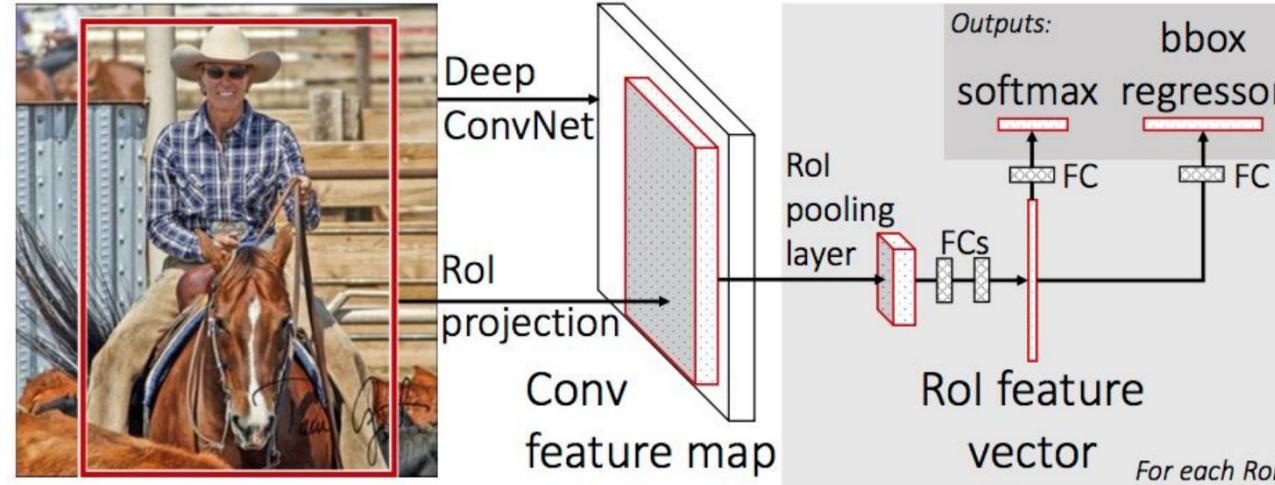


Fully Convolutional Network (FCN)

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

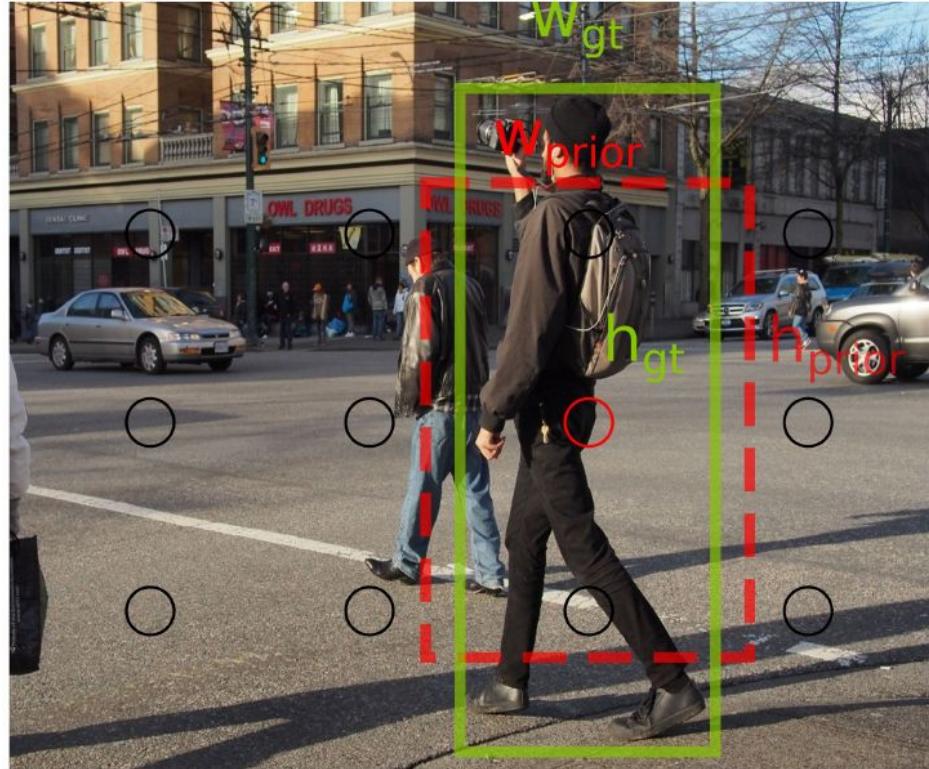
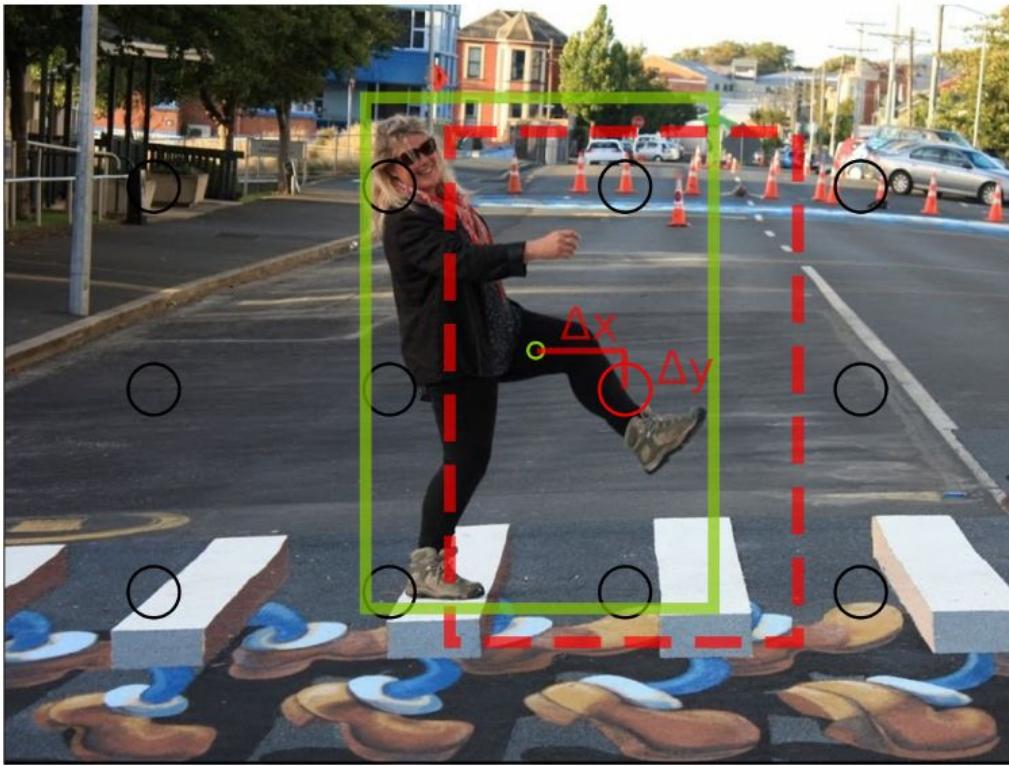


input	0.88	0.44	0.14	0.16	0.37	0.77	0.96	0.27
	0.19	0.45	0.57	0.16	0.63	0.29	0.71	0.70
	0.66	0.26	0.82	0.64	0.54	0.73	0.59	0.26
	0.85	0.34	0.76	0.84	0.29	0.75	0.62	0.25
	0.32	0.74	0.21	0.39	0.34	0.03	0.33	0.48
	0.20	0.14	0.16	0.13	0.73	0.65	0.96	0.32
	0.19	0.69	0.09	0.86	0.88	0.07	0.01	0.48
	0.83	0.24	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 CxWxH.
3. Для каждого region proposal:
 - a. Вырезаем его из карты признаков, используя ROI Pooling. Результатом операции является часть карты признаков, где $(W_{\text{image}}/w_{\text{proposal}})$ и $(H_{\text{image}}/h_{\text{proposal}})$ ее линейные размеры.
 - b. Полученный тензор передается в fully connected слои, в них предсказывается класс объекта и поправки координат bounding box-а (Δx , Δy , 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: $(\Delta x, \Delta y, k_h, k_w)$.

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

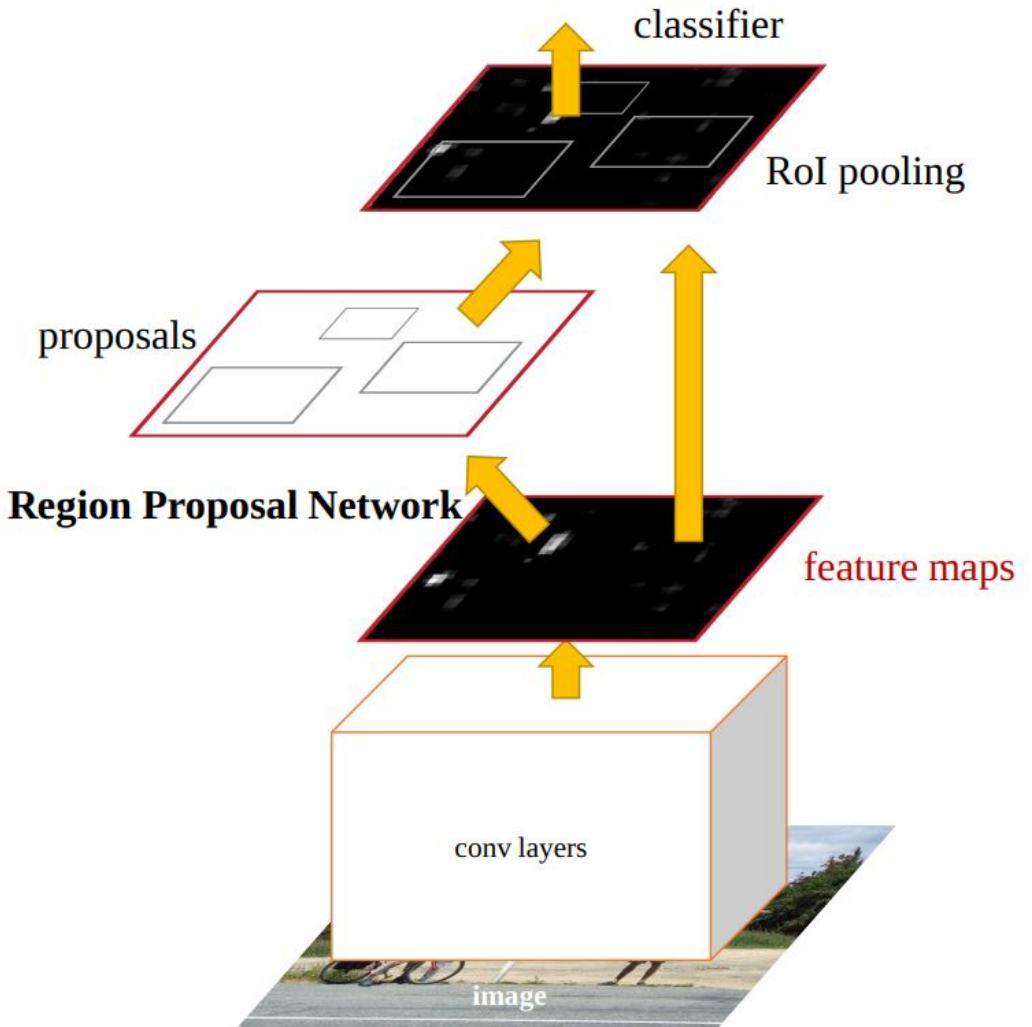
Fast RCNN детали реализации

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.

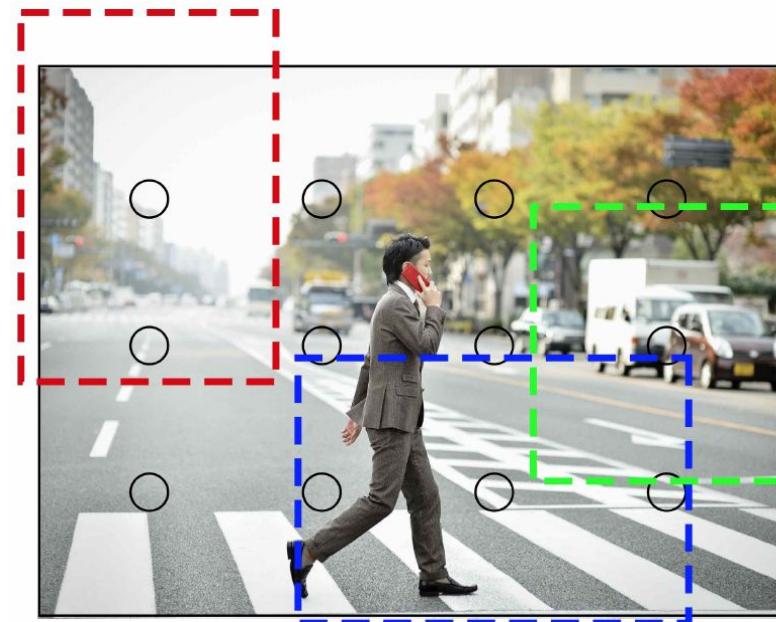
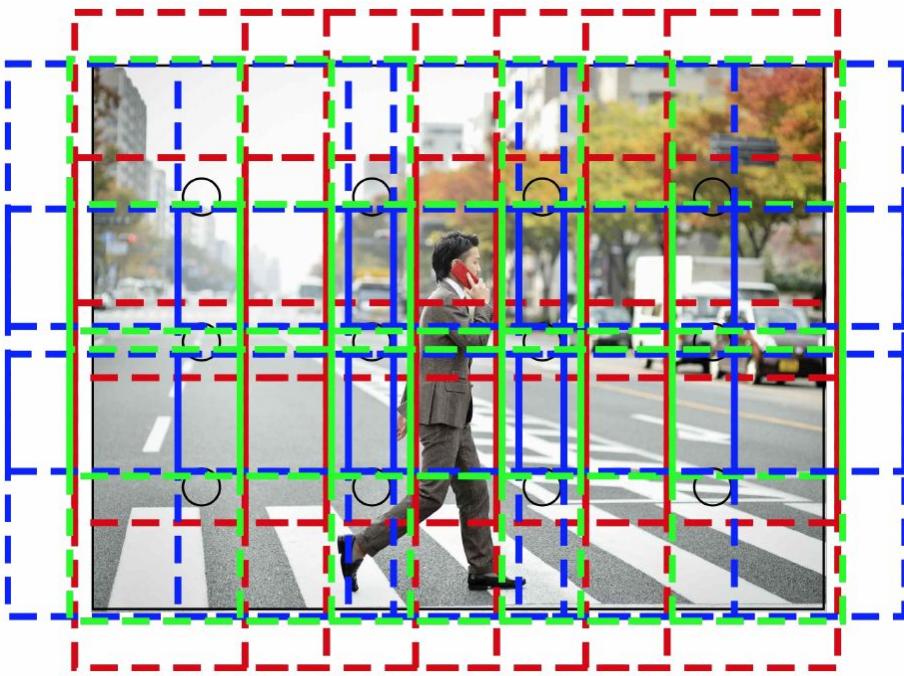
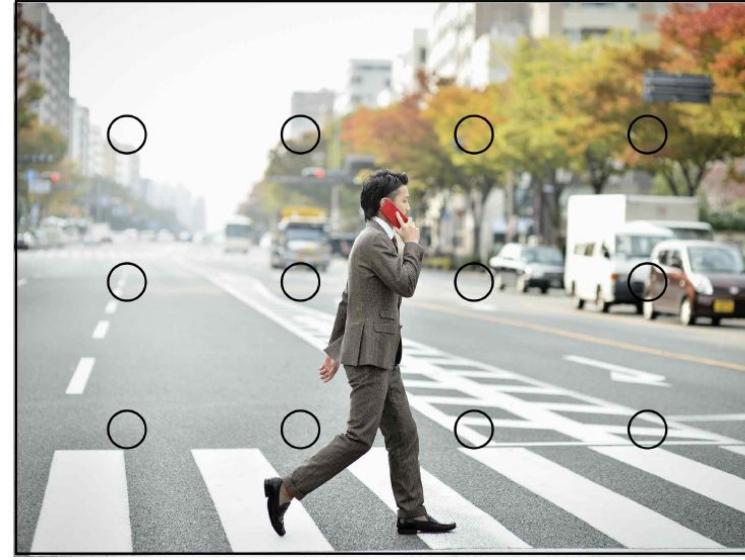
Faster RCNN

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. Результат передается в RoI pooling слой, а после в регрессор и классификатор Fast RCNN.

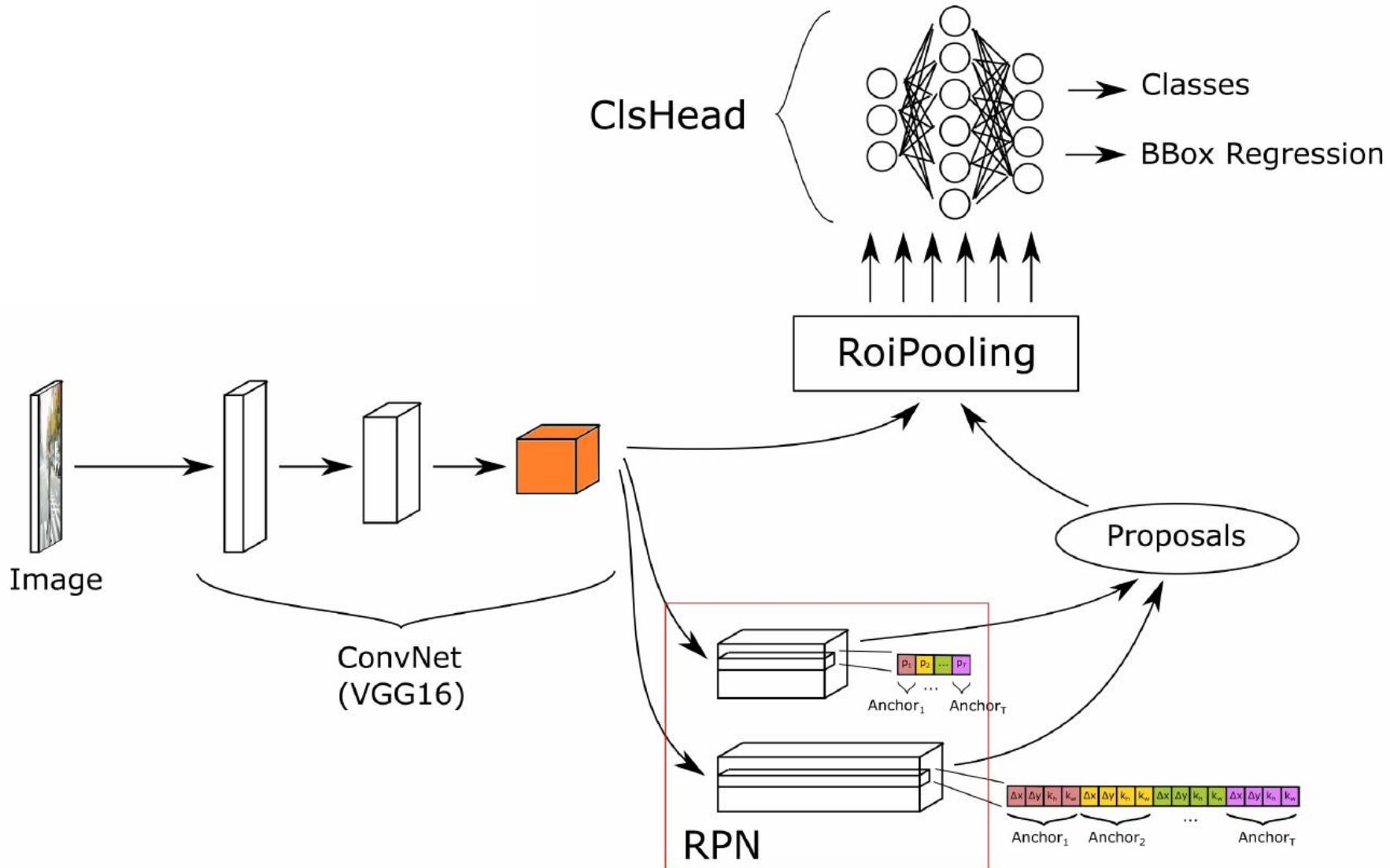


<https://arxiv.org/pdf/1506.01497.pdf>

Anchor Boxes



Faster RCNN



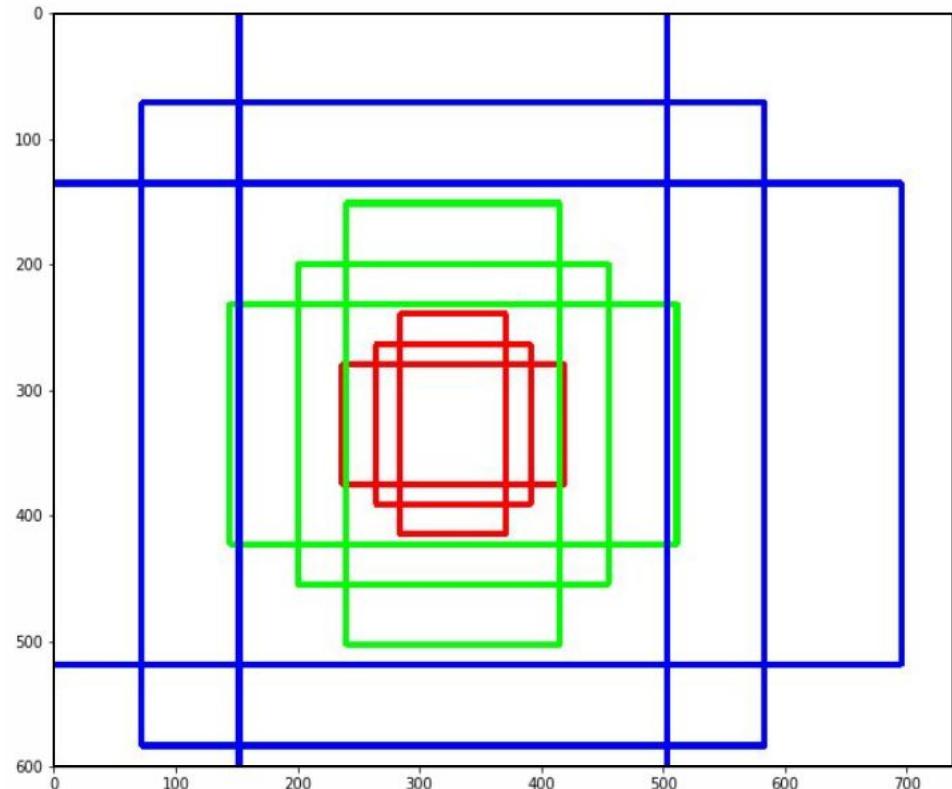
Scale problem

Проблема:

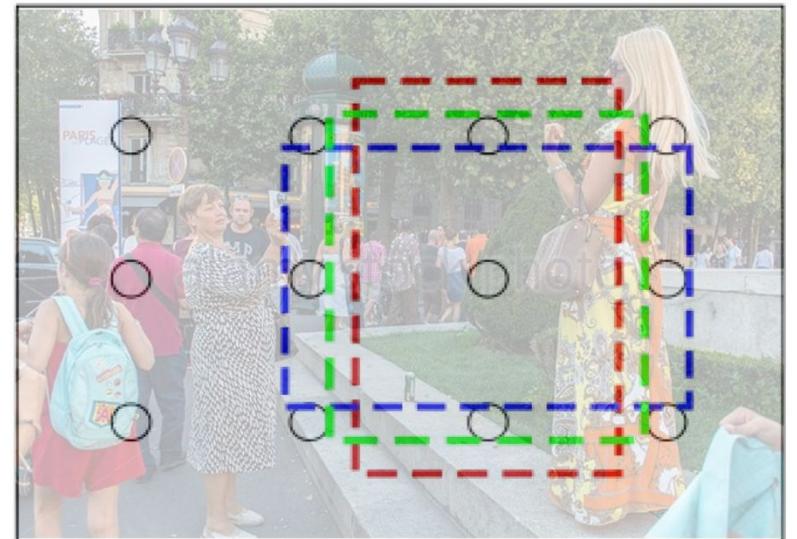
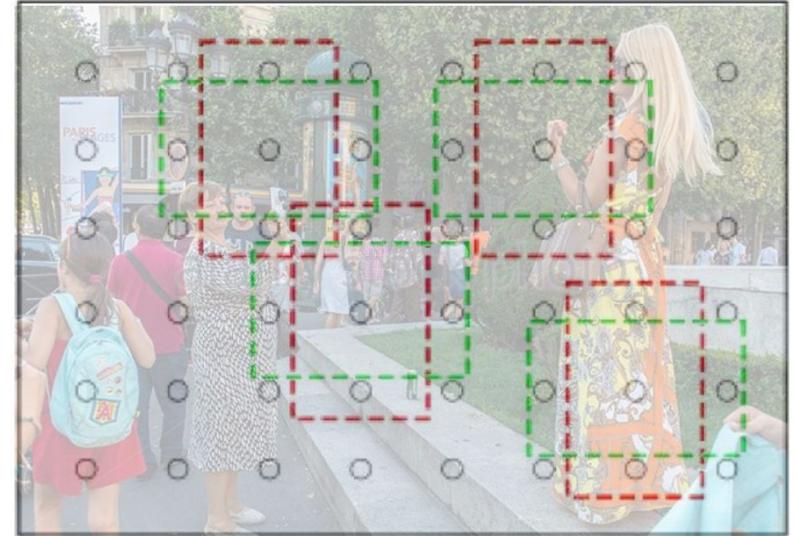
Для каждой точки мы генерируем одинаковое количество больших и маленьких box-ов. При таком подходе box-ы небольшого размера будут “разрежены”. Хотя мы имеем возможность расположить их в большем количестве.

Решение:

Генерируем region proposal-ы разных размеров!

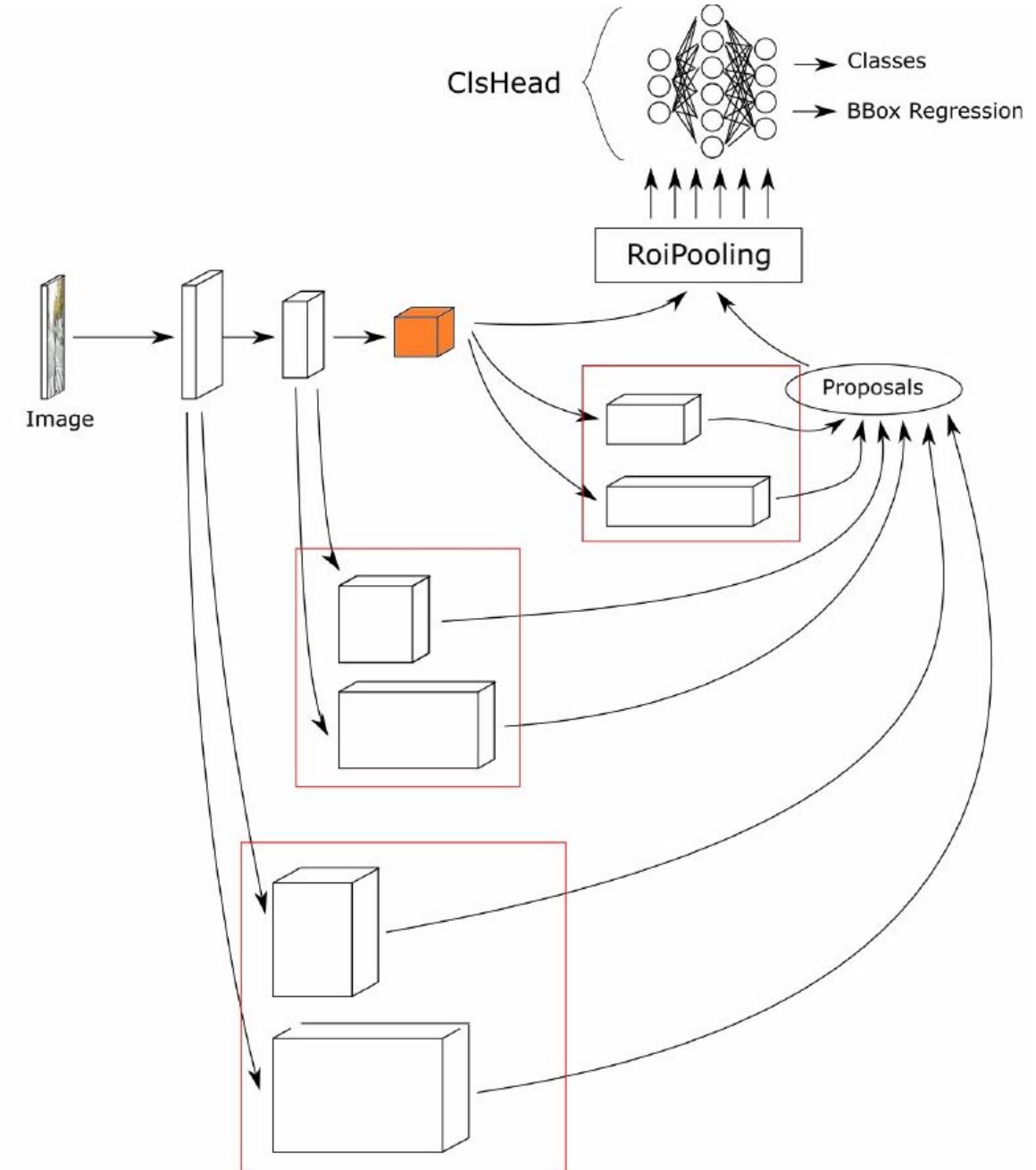
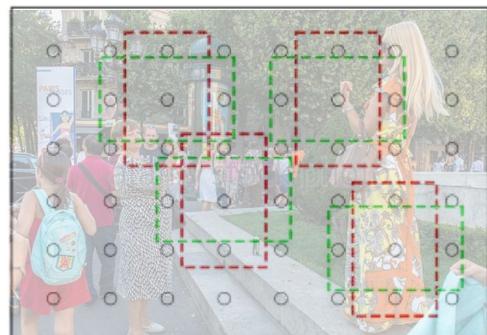
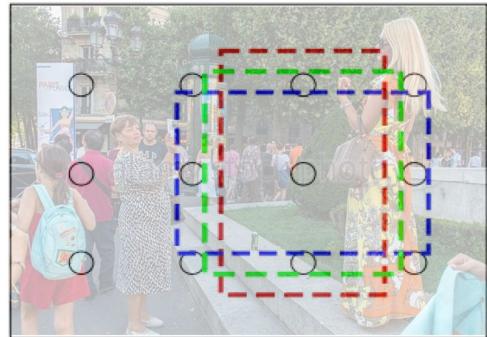


Different Scales



Multiple Scales

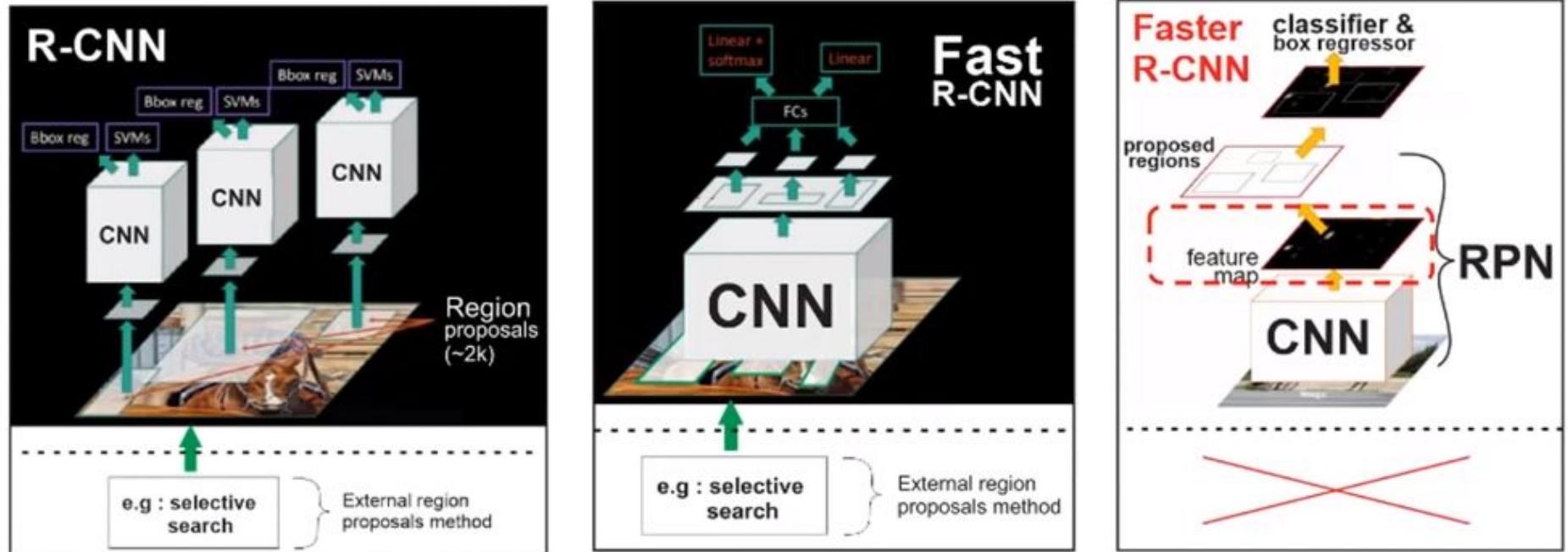
1. Предыдущие CNN тензоры используются для генерации proposal.
2. Proposal-ы с разных слоев сохраняются в один массив.



Faster RCNN резюме

1. RPN можно рассматривать как механизм “внимания” сети.
2. Классификация и регрессия в RPN являются superpixel-wise подходом.
3. Подход до сих пор актуален. Mask RCNN базируется на Faster RCNN с resnet в качестве бэкбона.
4. R-FCN дальнейшее развитие Faster RCNN.

RCNN / Fast RCNN / Faster RCNN

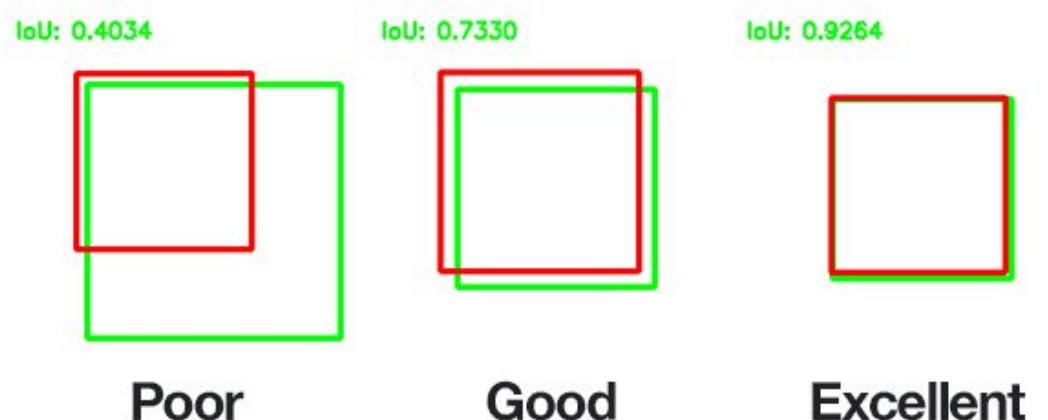
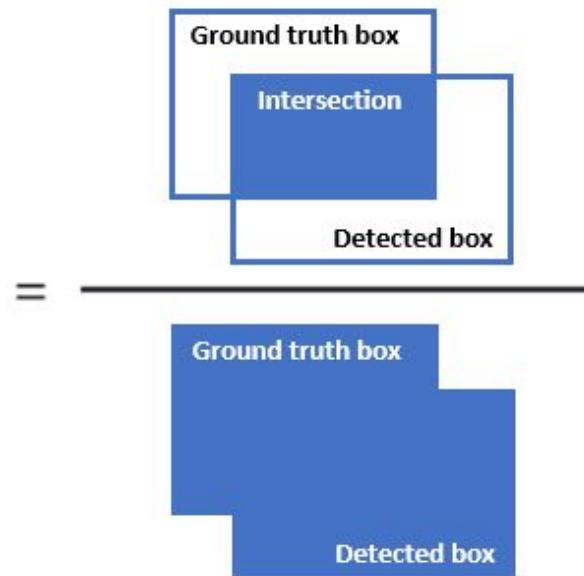


	R-CNN	Fast R-CNN	Faster R-CNN
Test time per image	50 seconds	2 seconds	0.2 seconds
Speed-up	1x	25x	250x
mAP (VOC 2007)	66.0%	66.9%	66.9%

* 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**

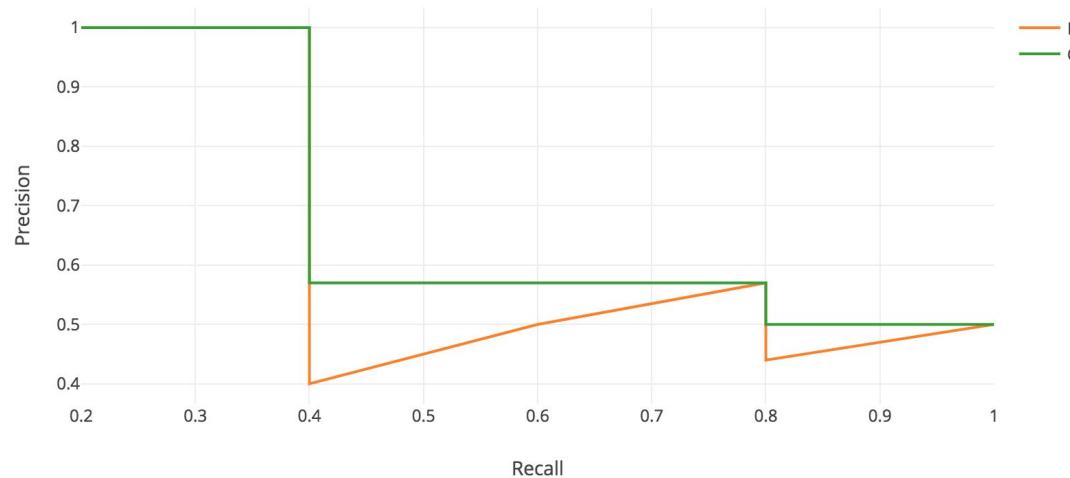
Иначе: **False Positive**.

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
```

mean Average Precision (mAP)

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



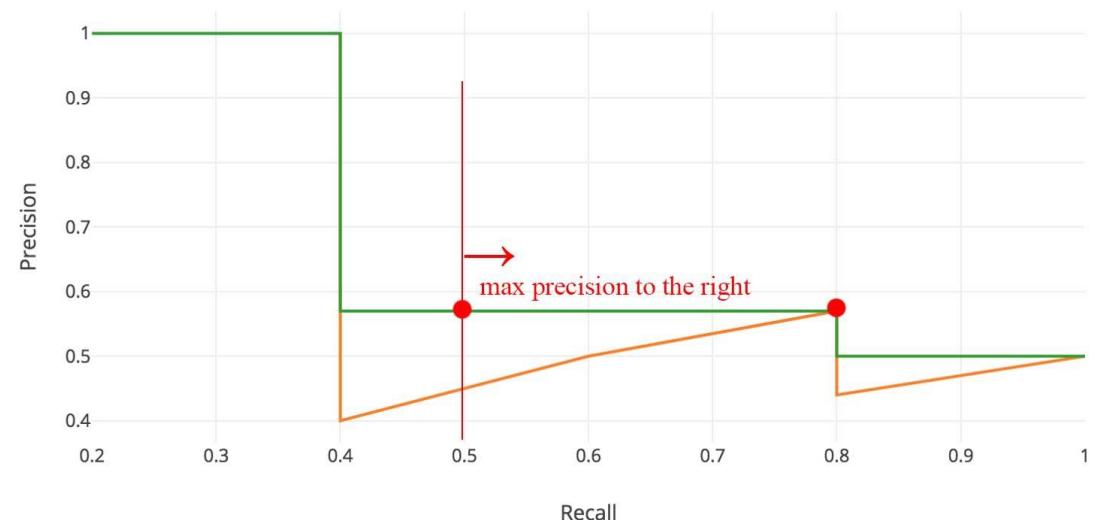
$$AP = \frac{1}{11} \times (AP_r(0) + AP_r(0.1) + \dots + AP_r(1.0))$$

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.

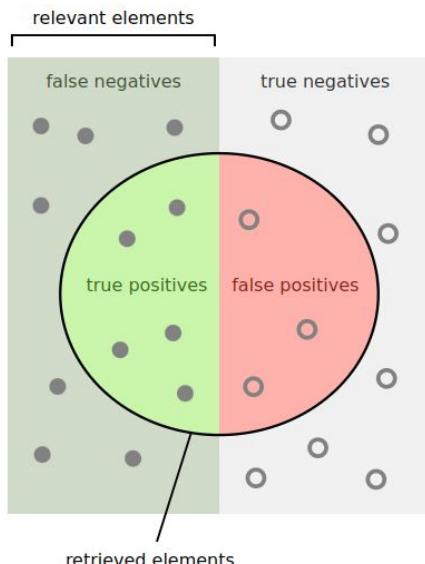
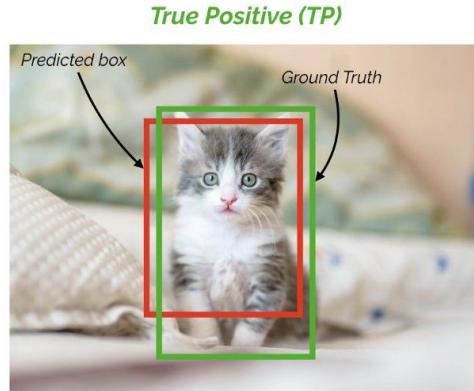
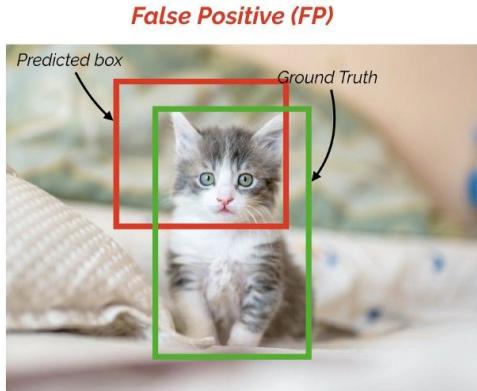
Vary threshold => obtain different Precision and Recall.

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



mean Average Precision (mAP)

If IoU threshold = 0.5

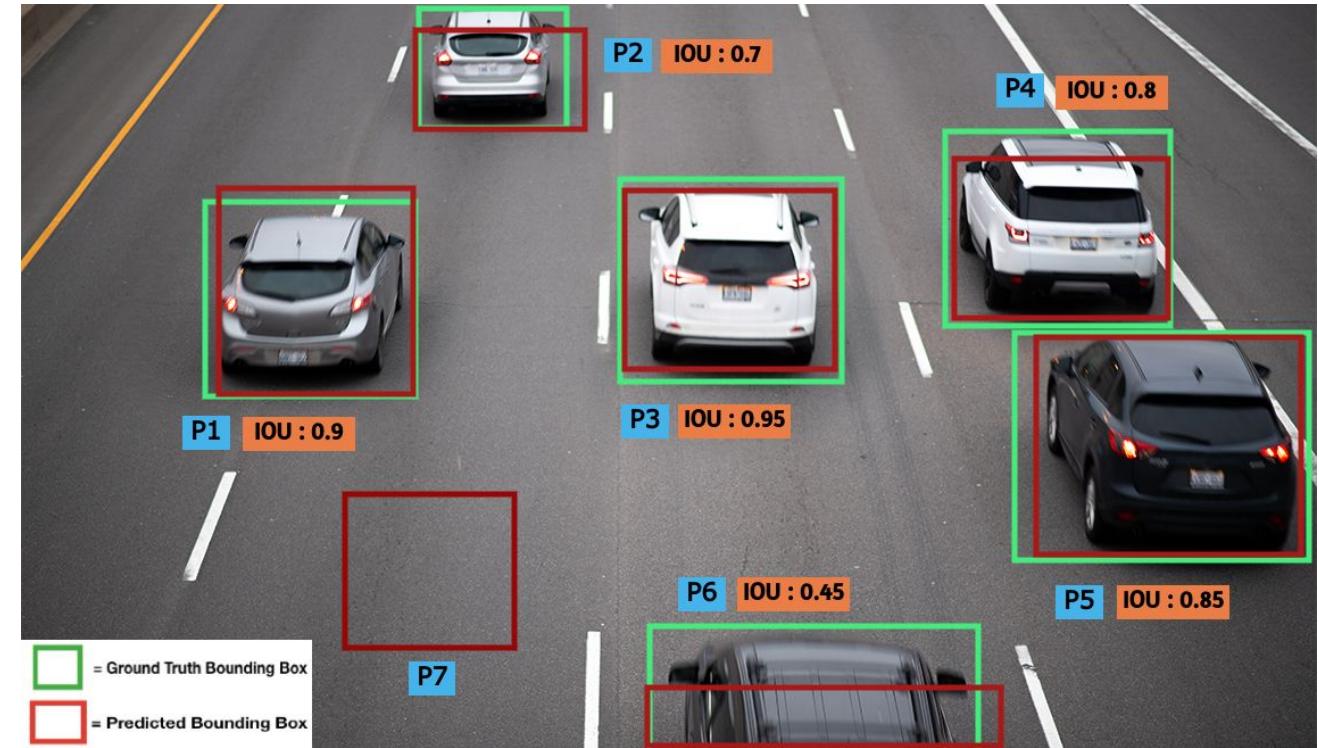


How many retrieved items are relevant?

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 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)

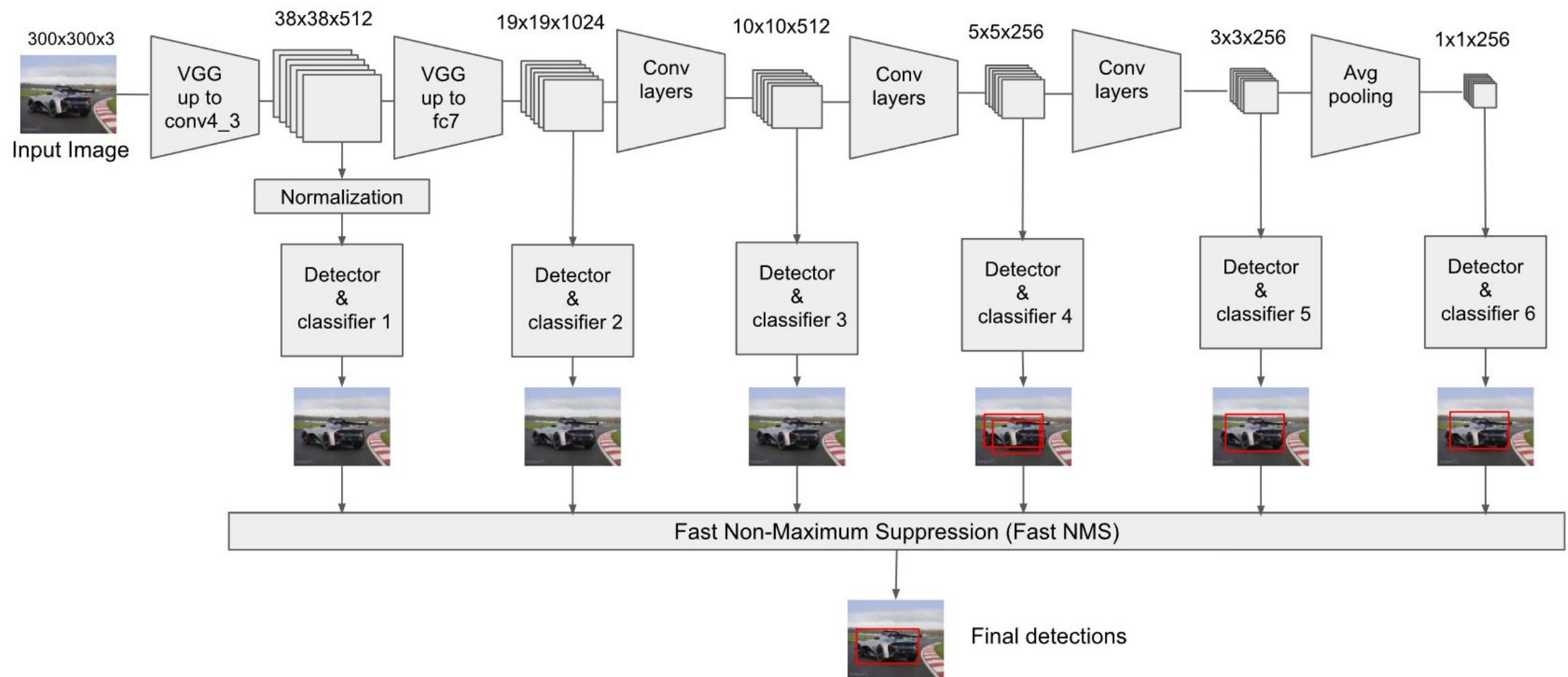
Single-shot детекция

• • • •

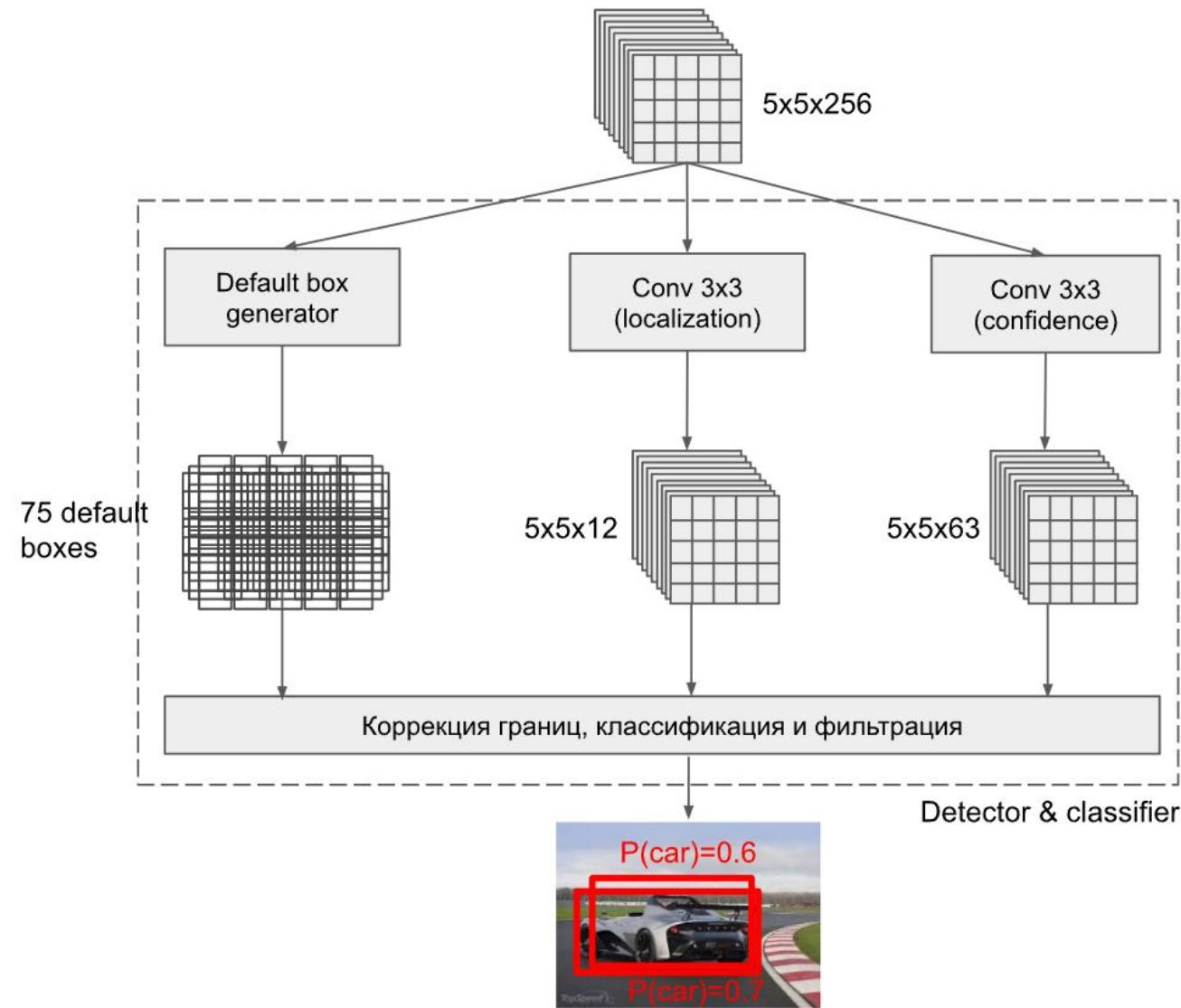
Single shot approaches

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.

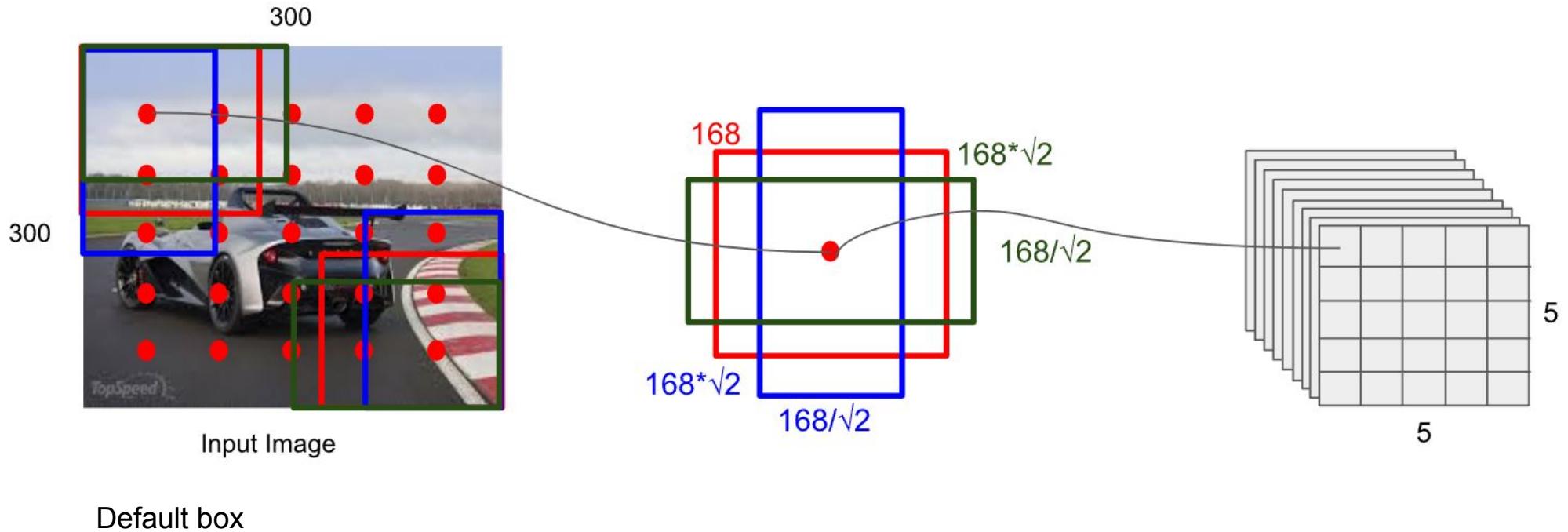
Single Shot Detector (1)



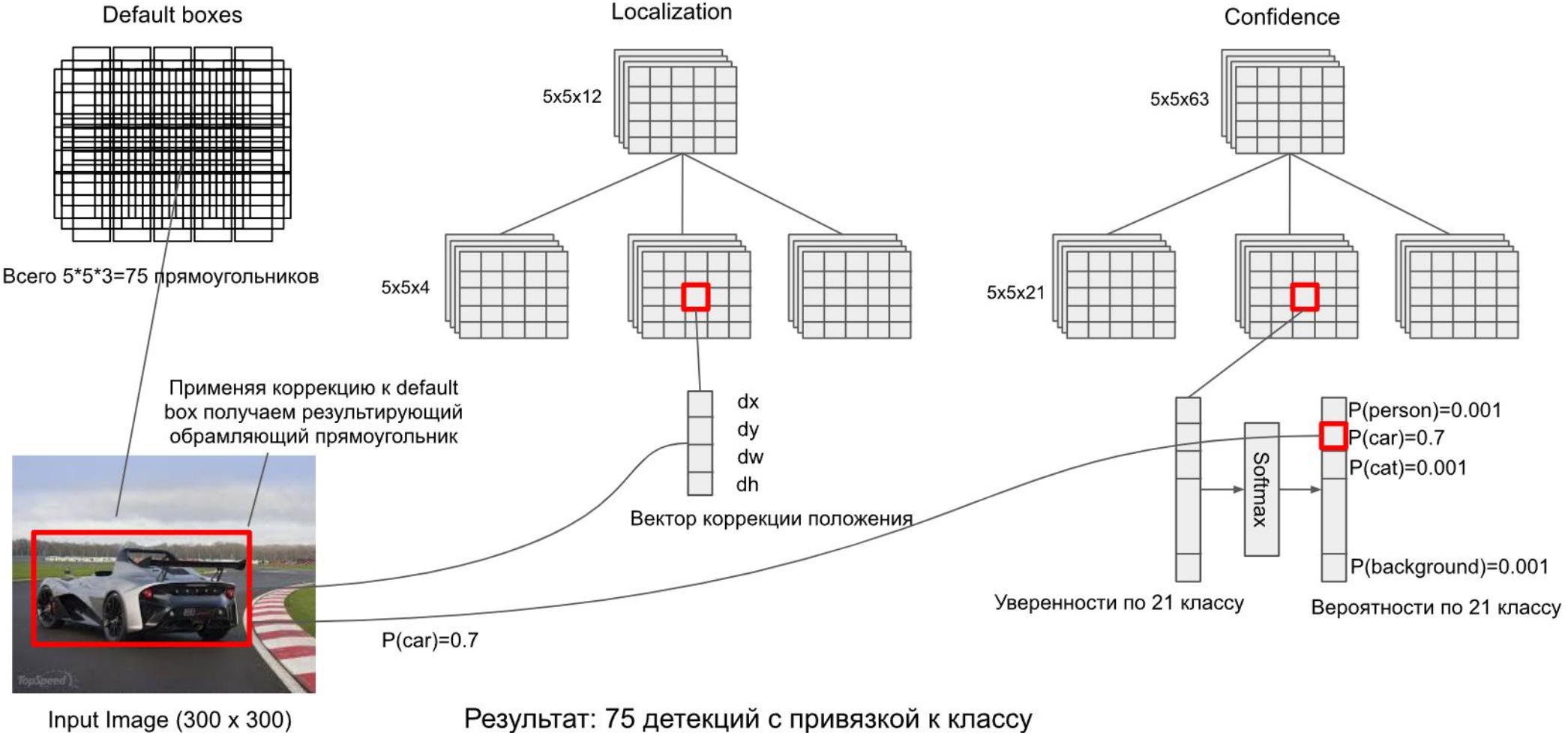
Single Shot Detector (2)



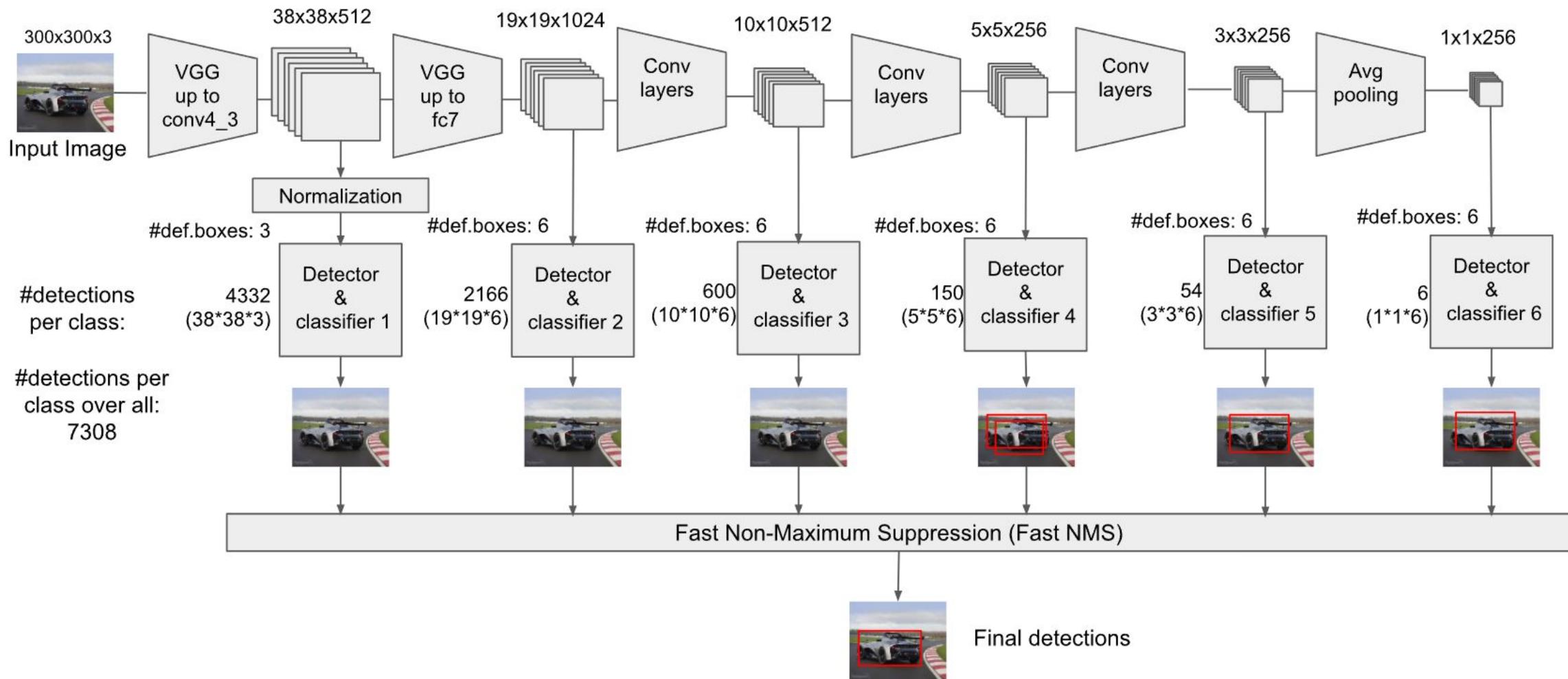
Single Shot Detector (3)



Single Shot Detector (4)



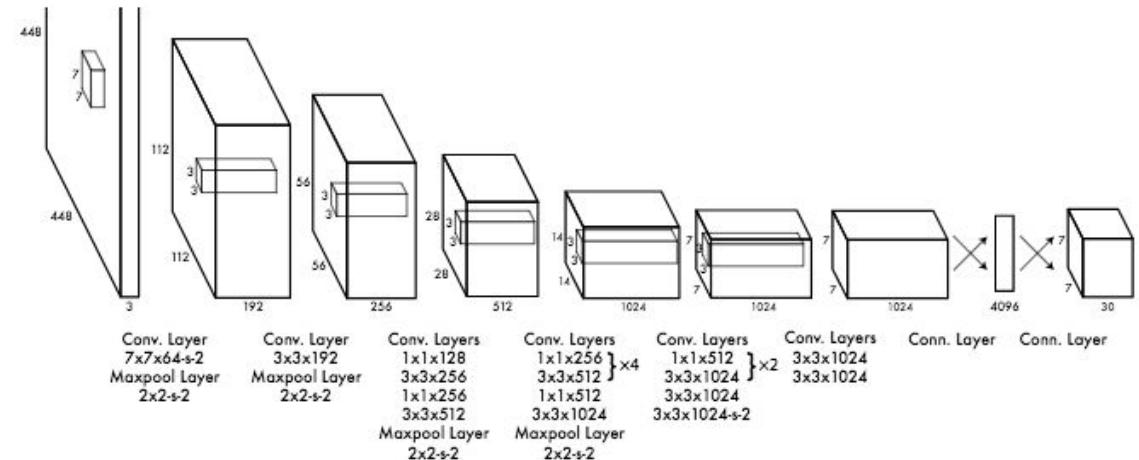
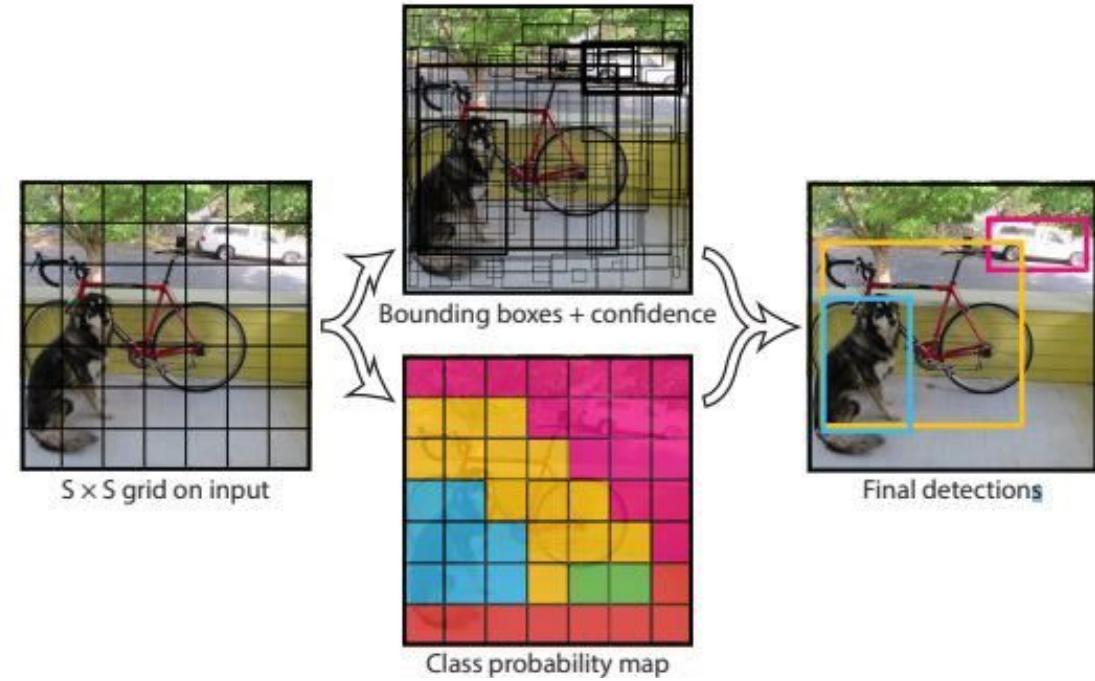
Single Shot Detector (5)



YOLO (You Only Look Once)

1. Принимаем изображение на вход и разбиваем на сетку
2. Каждая ячейка отвечает за предсказание объекта, относящегося к ней
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:

[GitHub - ultralytics/ultralytics: NEW - YOLOv8 🚀 in PyTorch > ONNX > OpenVINO > CoreML > TFLite](#)

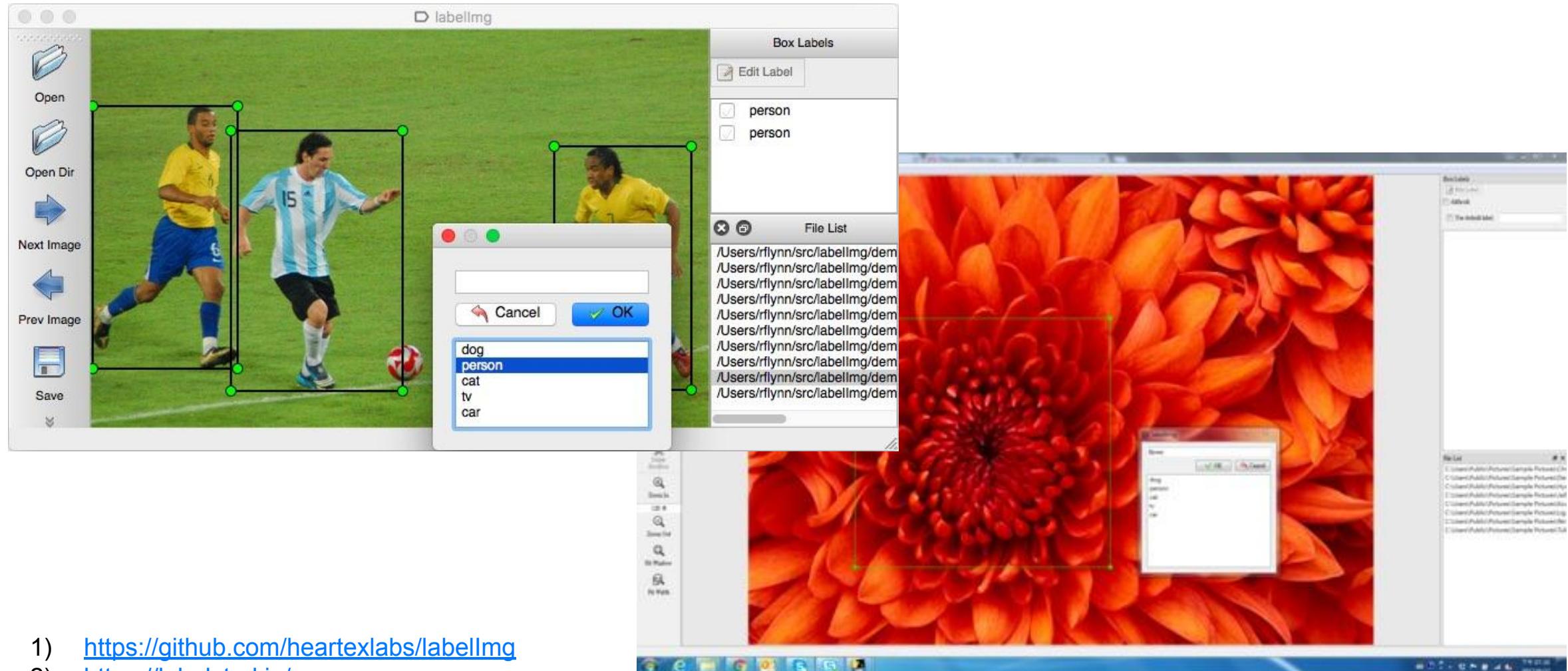


<https://arxiv.org/pdf/1506.02640.pdf>
<https://pjreddie.com/publications/yolo/>

Формирования разметки



Инструменты создания разметки изображений



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

Вопросы?

Часть 2. Задача сегментации

Часть 2. Задача сегментации

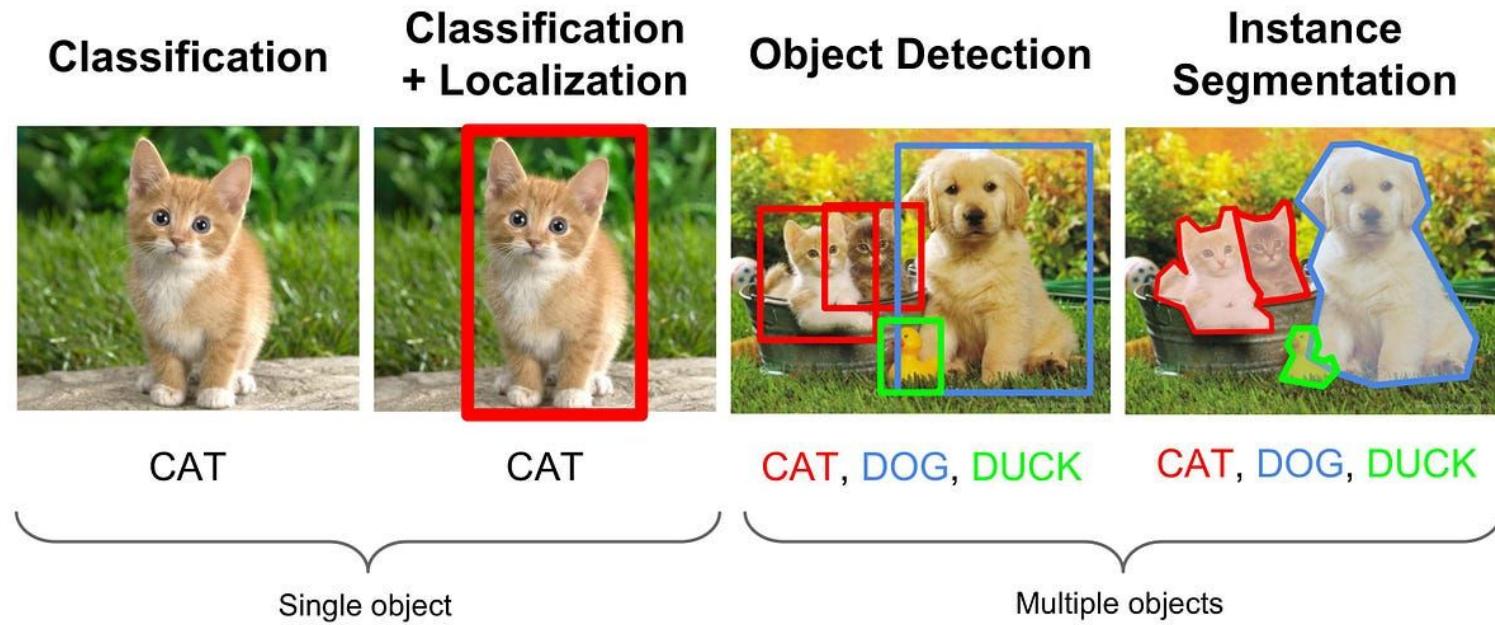
1. Задача сегментации и ее виды
2. Семантическая сегментация
3. Объектная сегментация



Сегментация

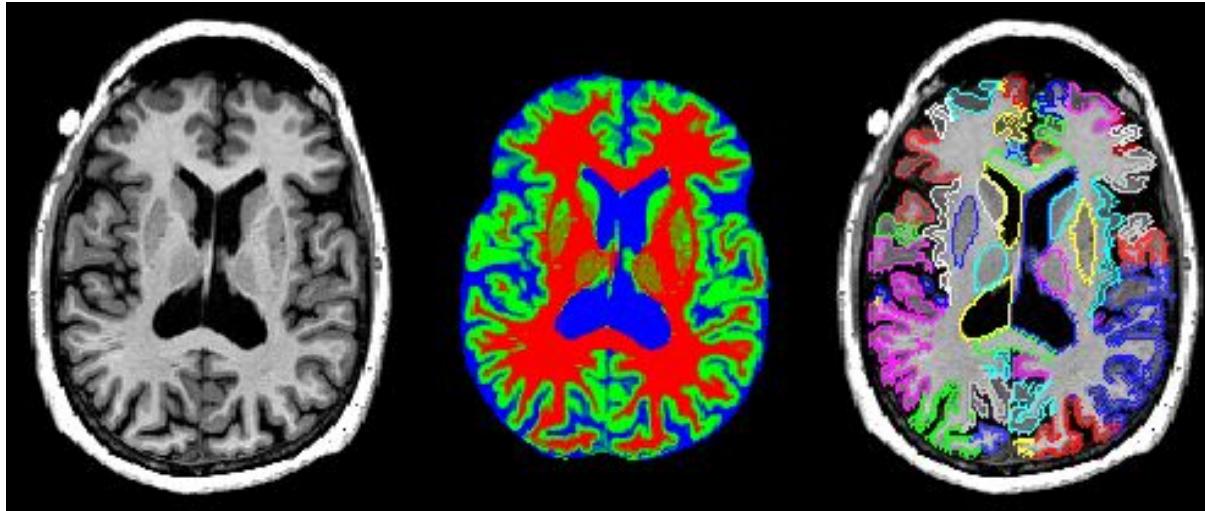
• • •

Локализация объектов



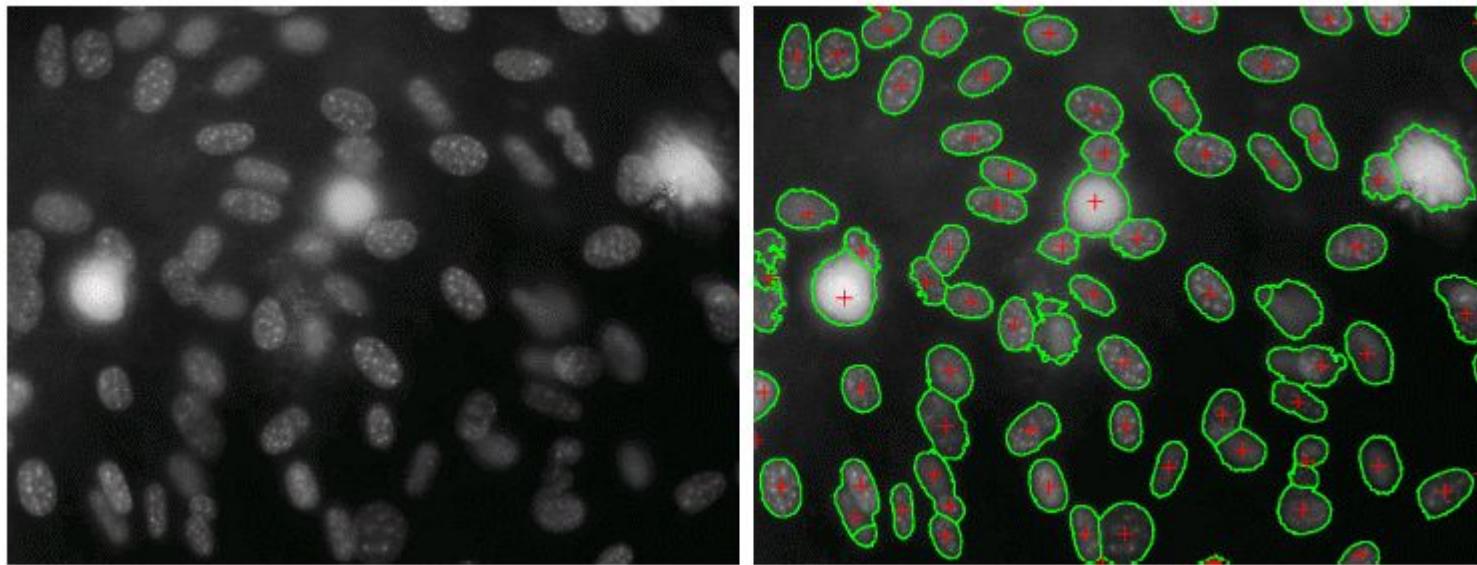
- Другой частный случай локализации – **сегментация**
- В сегментации положение объекта задается **попиксельной маской**

Сегментация - примеры



Томографические снимки

Сегментация - примеры



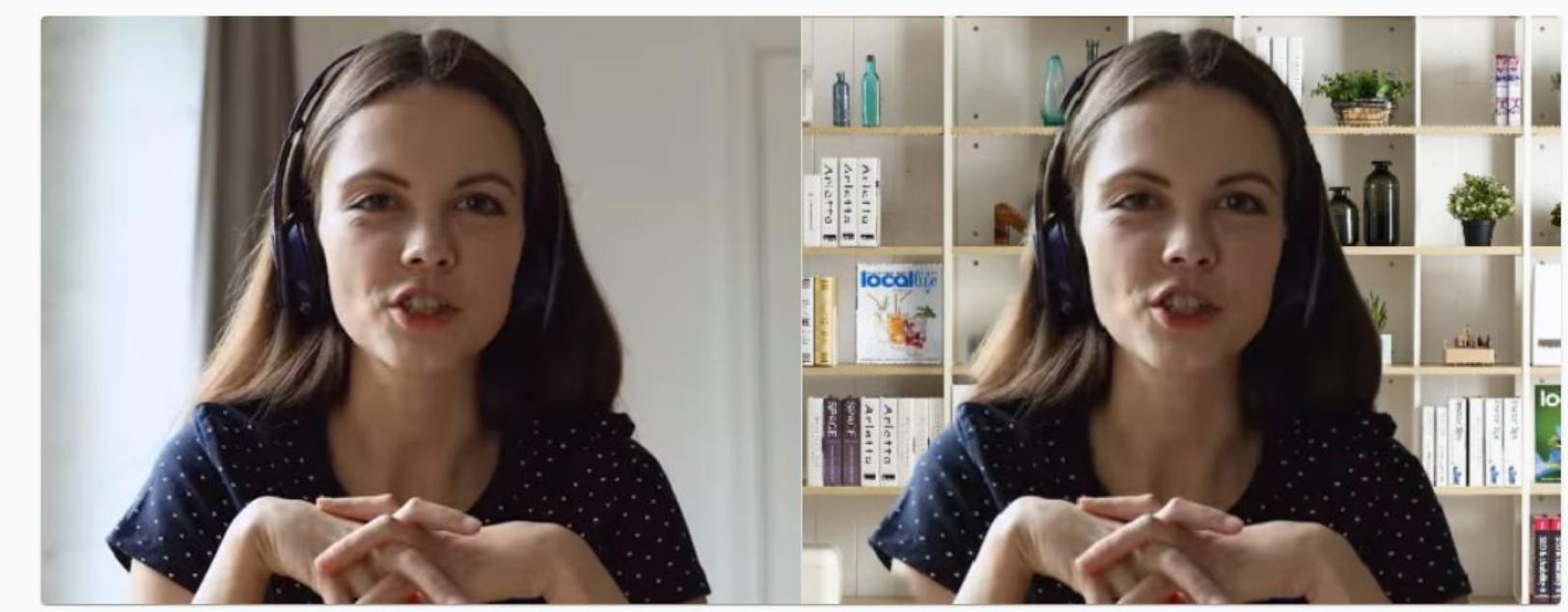
Снимки с микроскопа

Сегментация - примеры



Спутниковые снимки / аэрофото

Сегментация - примеры



Сегментация "фона"

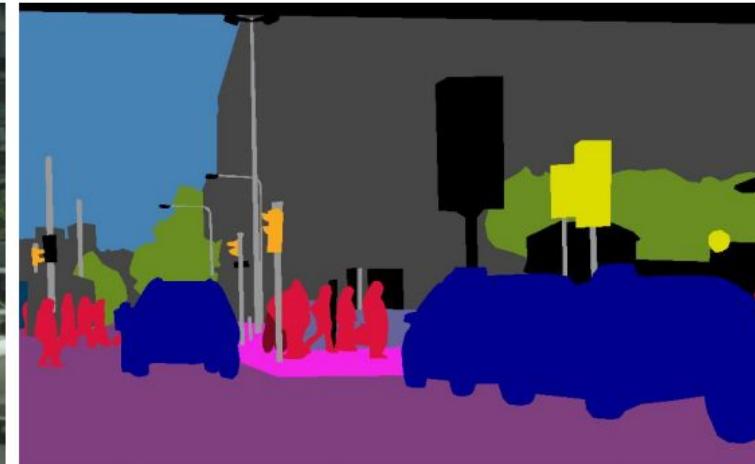
Сегментация - виды

- Сегментация позволяет более точно **определять форму интересующих объектов**
- Есть несколько видов сегментации с **разной полнотой извлекаемой информации:**
 - Semantic Segmentation
 - Instance Segmentation
 - Panoptic Segmentation

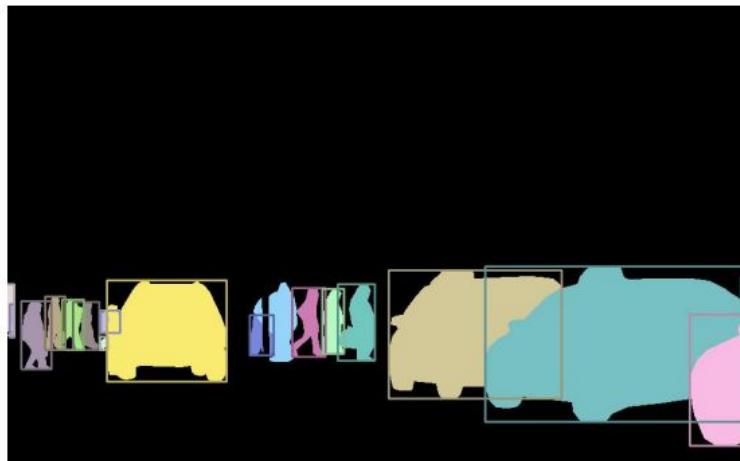
Сегментация - виды



(a) image



(b) semantic segmentation



(c) instance segmentation



(d) panoptic segmentation

Сегментация - виды

- Семантическая (semantic) - каждому пикселю (кроме фона) ставится в соответствие номер класса
- Объектная (instance) - разделяются маски разных экземпляров одного класса
- Panoptic
 - для классов типа “небо”, “дорожное полотно” и т.д. (stuff) только класс,
 - для остальных (things) - отдельная маска

Семантическая сегментация

...

Семантическая сегментация

- Семантическая сегментация \sim = пиксельная классификация
 - На входе в модель - изображение (RGB, RGBD, ...), облако точек, ...
 - На выходе - набор масок разных классов
- Рассмотрим бинарный случай - классы “фон” и “объект”

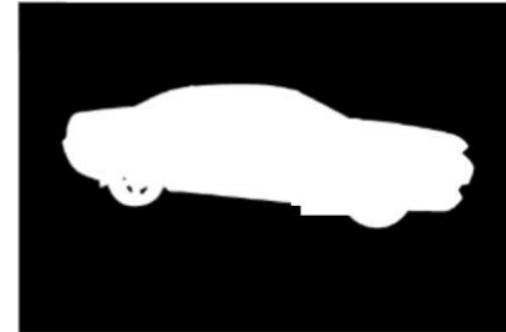
Семантическая сегментация

RGB



Семантическая сегментация

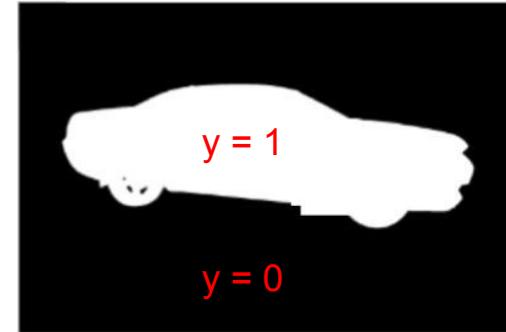
RGB



Ground-truth
маска

Семантическая сегментация

RGB



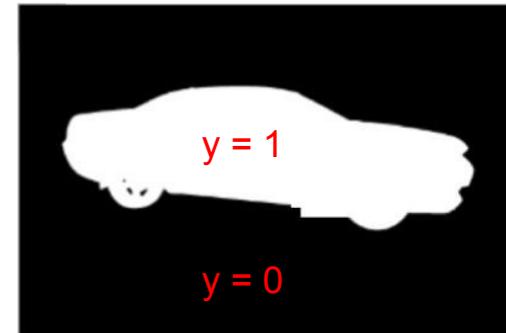
Ground-truth
маска

Семантическая сегментация

RGB



Модель



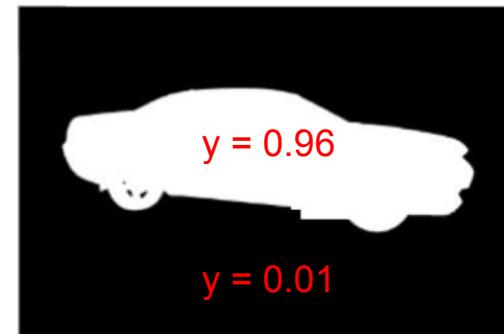
Ground-truth
маска

Семантическая сегментация

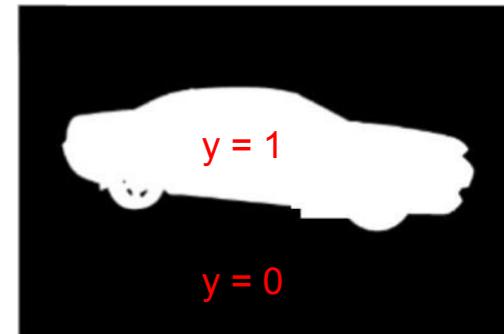
RGB



Модель



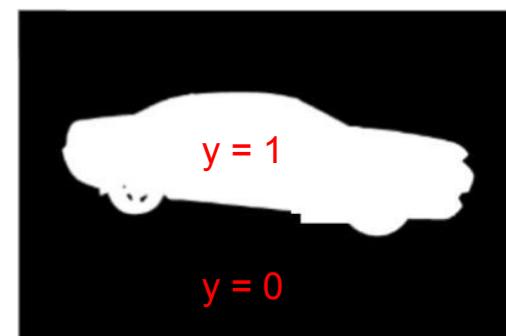
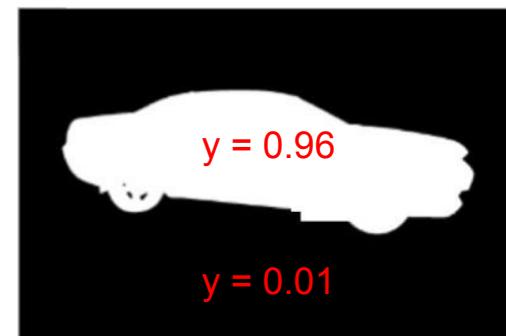
Ground-truth
маска



Семантическая сегментация



Модель



Ground-truth
маска



Функция
потерь

Семантическая сегментация - функция потерь

- Что использовать в качестве лосса?
 - Попиксельная классификация -> Cross-Entropy?

$$BCE(y, \bar{y}) = -y \log (\bar{y}) - (1 - y) \log (1 - \bar{y})$$

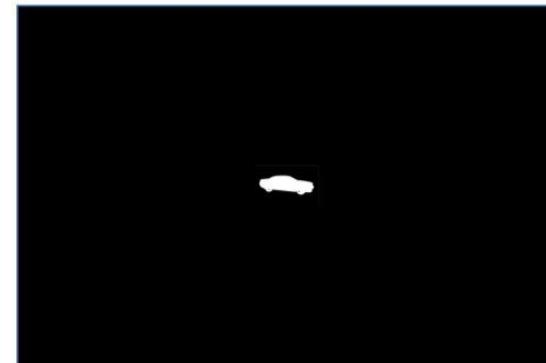
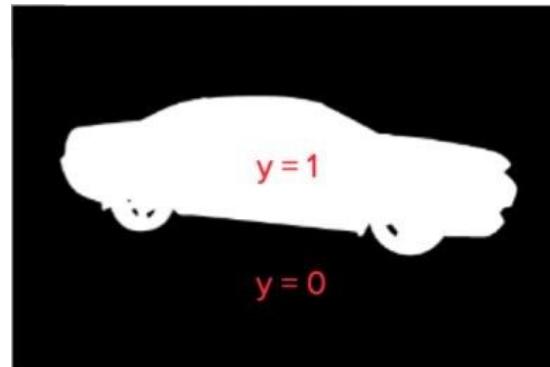
- Считаем в каждом пикселе, усредняем по площади

Семантическая сегментация - функция потерь

- Что использовать в качестве лосса?
 - Попиксельная классификация -> Cross-Entropy?

$$BCE(y, \bar{y}) = -y \log (\bar{y}) - (1 - y) \log (1 - \bar{y})$$

- Считаем в каждом пикселе, усредняем по площади

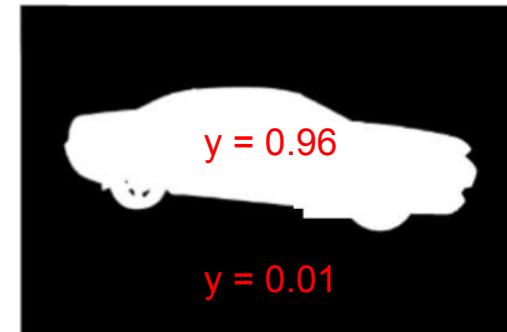


Семантическая сегментация - дисбаланс классов

- Аналогично задаче детектирования, может быть сильный дисбаланс классов
- Возможные решения:
 - Веса для балансировки вклада разных пикселей
 - Focal Loss
 - Другие функции (см. дальше)

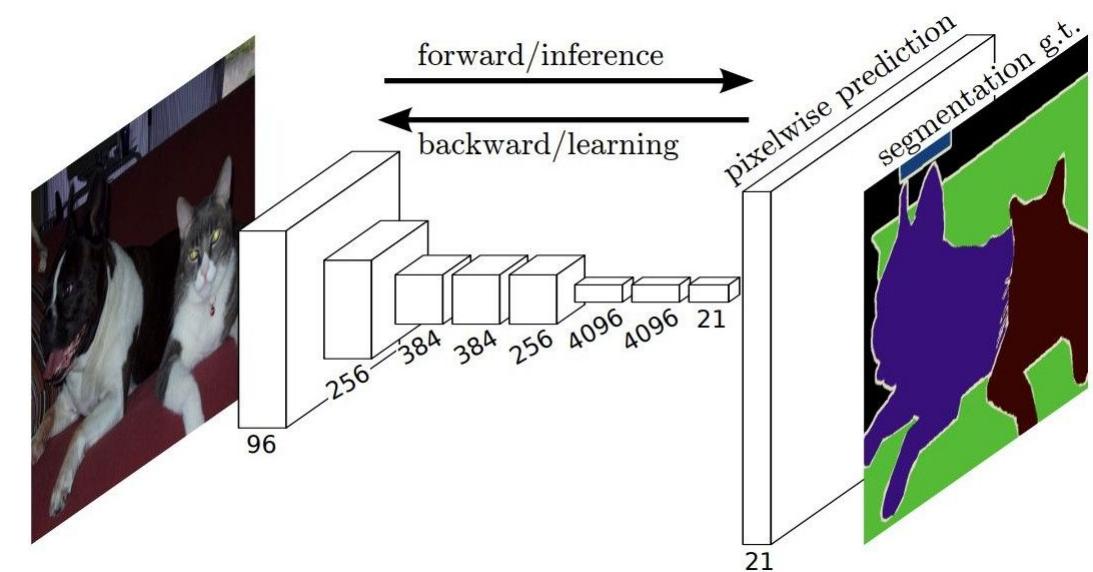
Семантическая сегментация - архитектура

- Какой может быть архитектура модели?
- "Интерфейс" модели:
 - На входе - изображение, $H \times W \times C$
 - На выходе - К масок, $H \times W \times K$



Fully-Convolutional Network (FCN) (2014)

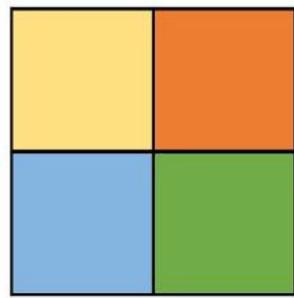
- Fully convolutional networks for Semantic Segmentation (2014)
 - Backbone (VGG)
 - Upsampling для приведения к исходному размеру
- Проблемы:
 - Узкое бутылочное горлышко
 - Резкое увеличение H/W



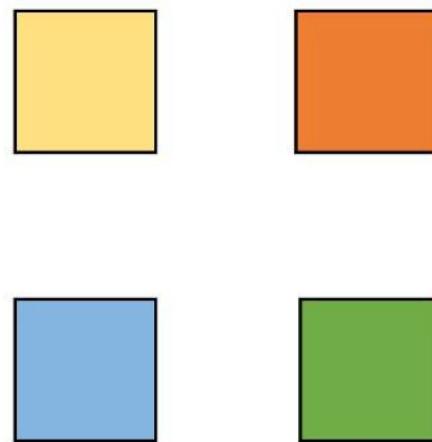
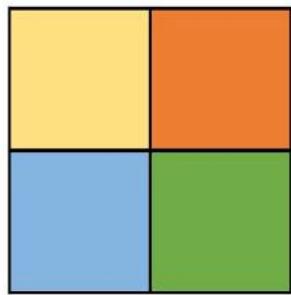
Interlude

- Как увеличить размер (H, W) карты активаций?
 - Интерполяция значений (Upsampling)
 - Транспонированная свертка (Transposed Conv)

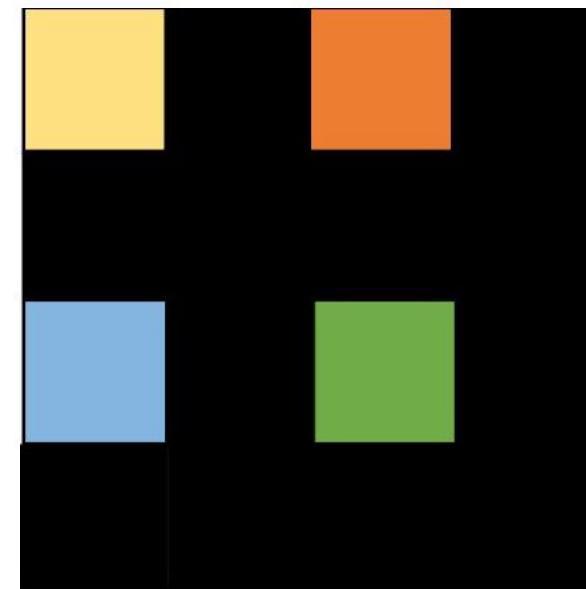
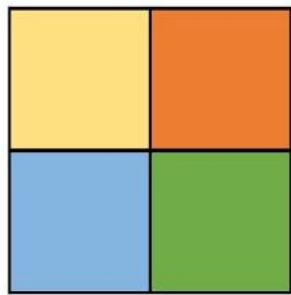
Upsampling



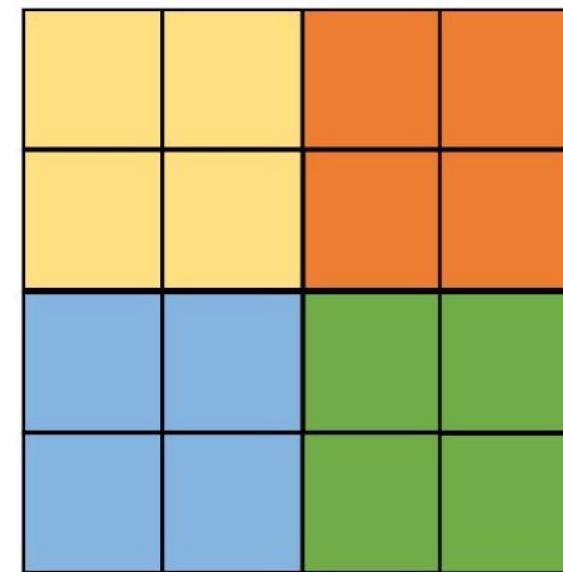
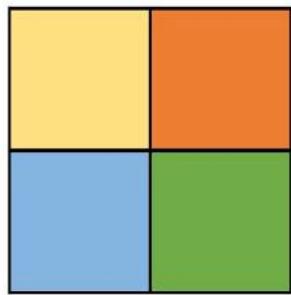
Upsampling



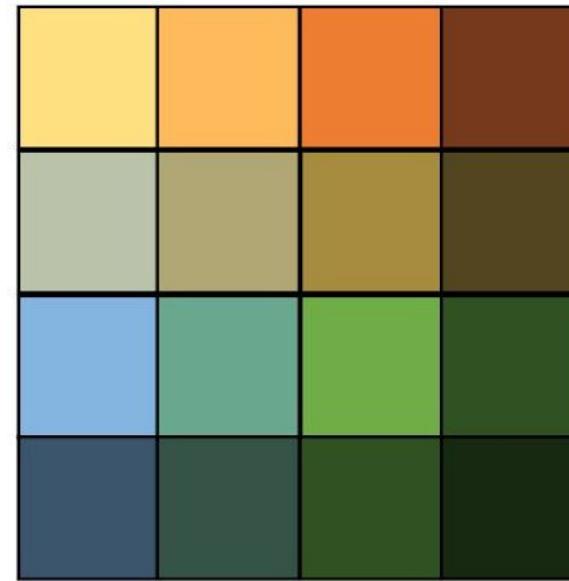
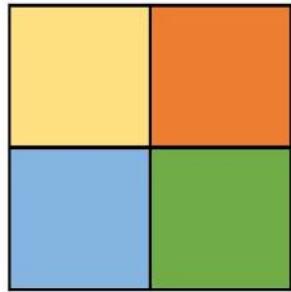
Upsampling



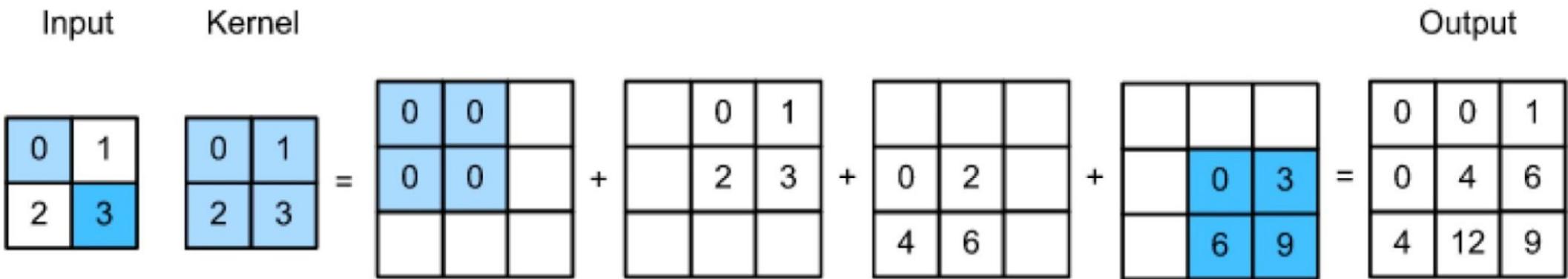
Upsampling



Upsampling



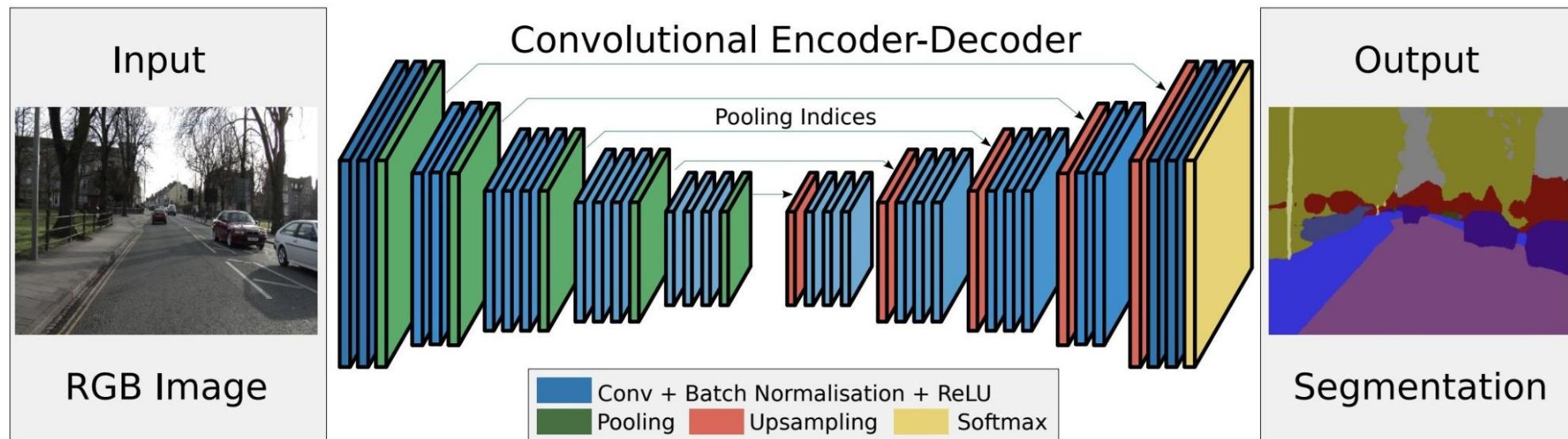
Transposed Conv



- Вместо "сворачивания" с ядром (как в обычном ConvLayer) происходит "разворачивание" входного сигнала
- Значения входного сигнала выступают весами перед ядром транспонированной свертки

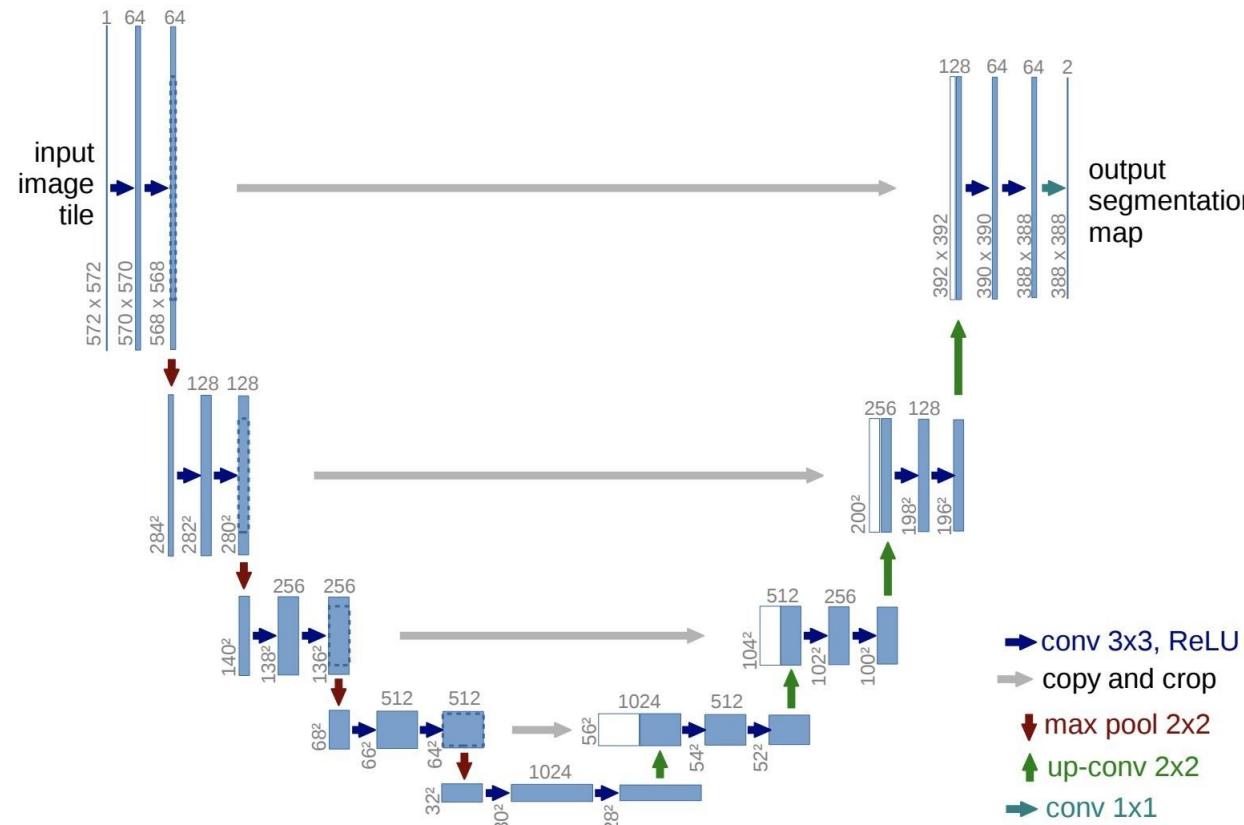
SegNet (2015)

- SegNet: A Deep Convolutional Encoder-Decoder Architecture
 - Симметричная архитектура вида Encoder-Decoder
 - Постепенный Upsampling



UNet (2015)

- U-Net: CNNs for Biomedical Image Segmentation
 - Добавили горизонтальные связи к Encoder-Decoder



UNet (2015)

- Сильное улучшение сегментации на границах объектов
- Всего лишь 30 изображений 512x512 для обучения!

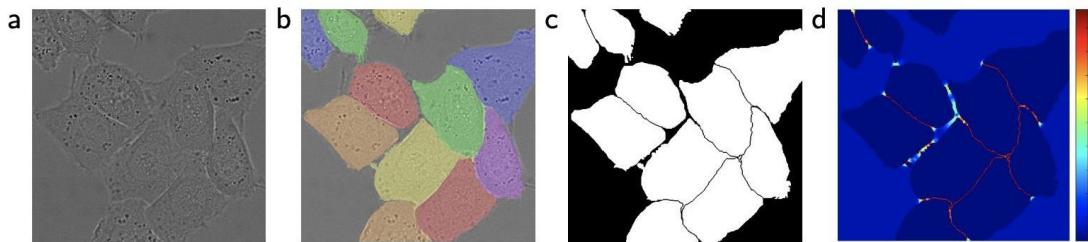
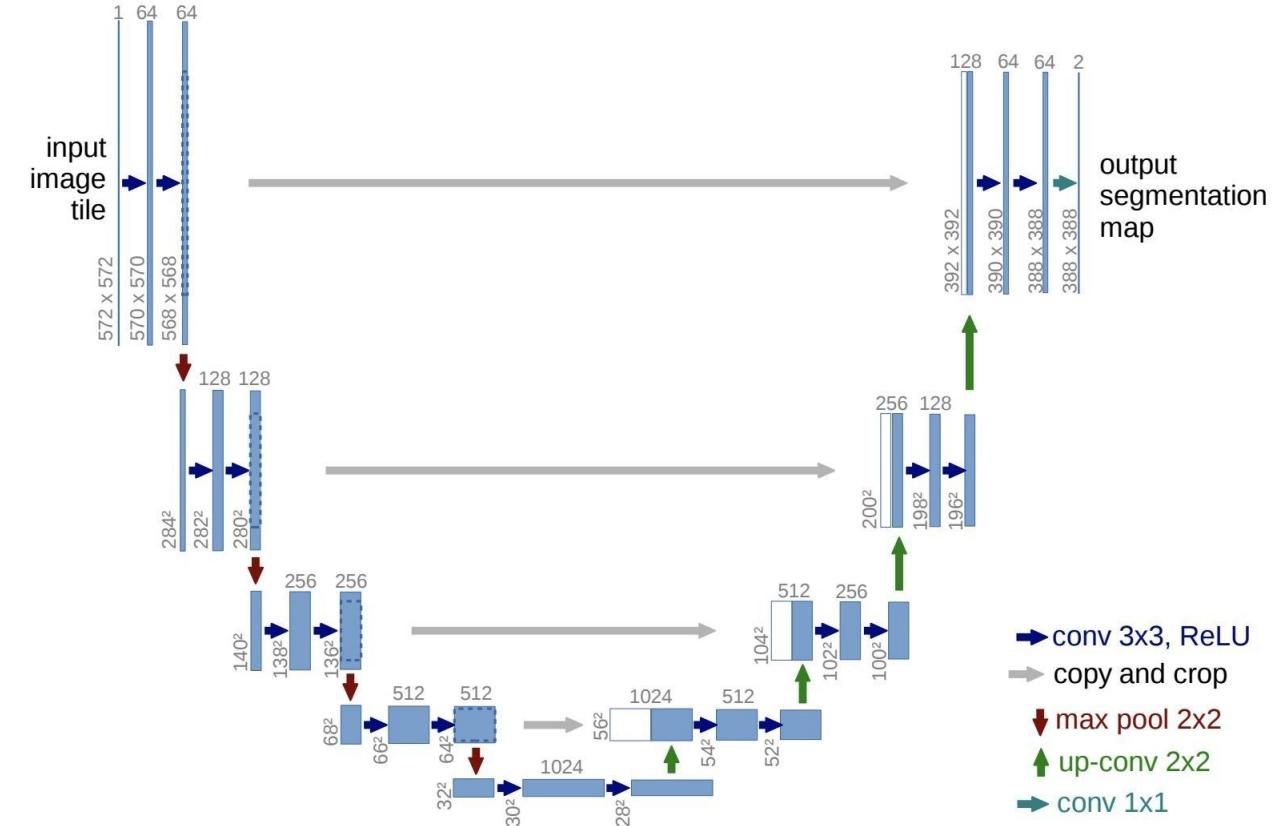


Fig. 3. HeLa cells on glass recorded with DIC (differential interference contrast) microscopy. (a) raw image. (b) overlay with ground truth segmentation. Different colors indicate different instances of the HeLa cells. (c) generated segmentation mask (white: foreground, black: background). (d) map with a pixel-wise loss weight to force the network to learn the border pixels.

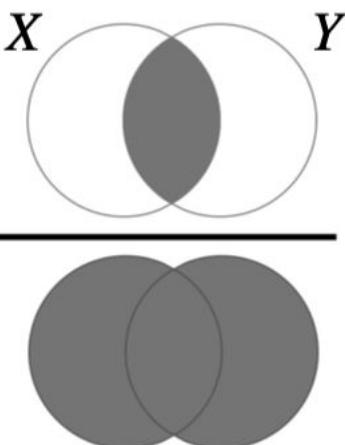


UNet-like

- Из конкретной архитектуры для сегментации UNet давно превратился в "подход" для задач image-to-image:
 - Сегментация (subj)
 - Колоризация (предсказание цветных каналов для grayscale-входа)
 - InPainting ("закрашивание" пустот)
 - ...
- UNet-like сеть =
 - encoder (resnet, efficientnet, ...) +
 - decoder

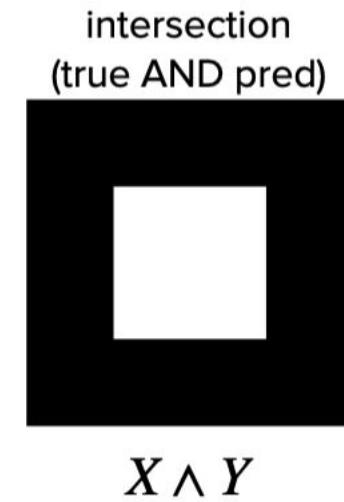
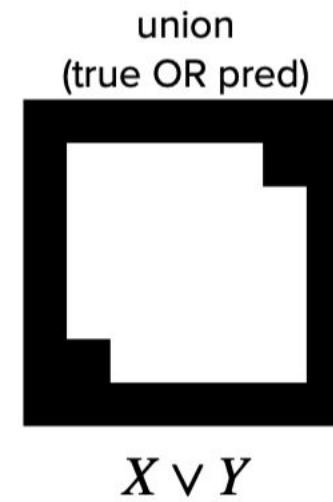
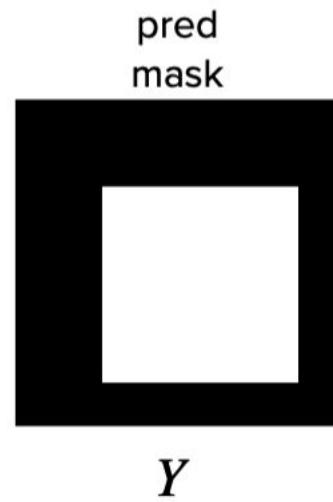
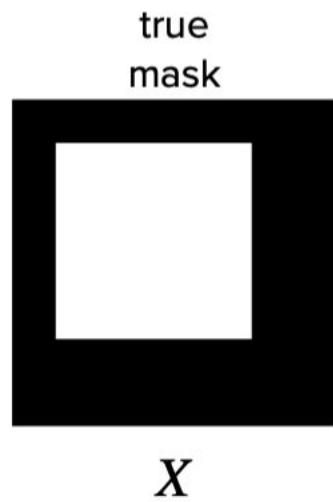
Beyond BCE

- Применение кросс-энтропии может сломаться о дисбаланс классов (и не только в сегментации)
- Вспомним, что в детекторах объектов говорили про понятие **Intersection-over-Union (IoU)** (синоним - Jaccard Index):

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

$$Jaccard(X, Y) = \frac{|X \wedge Y|}{|X \vee Y|}$$

Jaccard Index для сегментации

- По аналогии определим Jaccard Index для пары сегментационных масок:



$$Jaccard(X, Y) = \frac{|X \wedge Y|}{|X \vee Y|}$$

Jaccard Index для сегментации

- Напрямую оптимизировать Jaccard Index нельзя
- Но можно аппроксимировать его, например:

$$J_{seg} = \frac{\sum_{x,y} M_{gt}(x,y)*M_{pred}(x,y)}{\sum_{x,y} M_{gt}(x,y)+M_{pred}(x,y)-M_{gt}(x,y)*M_{pred}(x,y)}$$

- Получить из этого лосс можно, например, так:

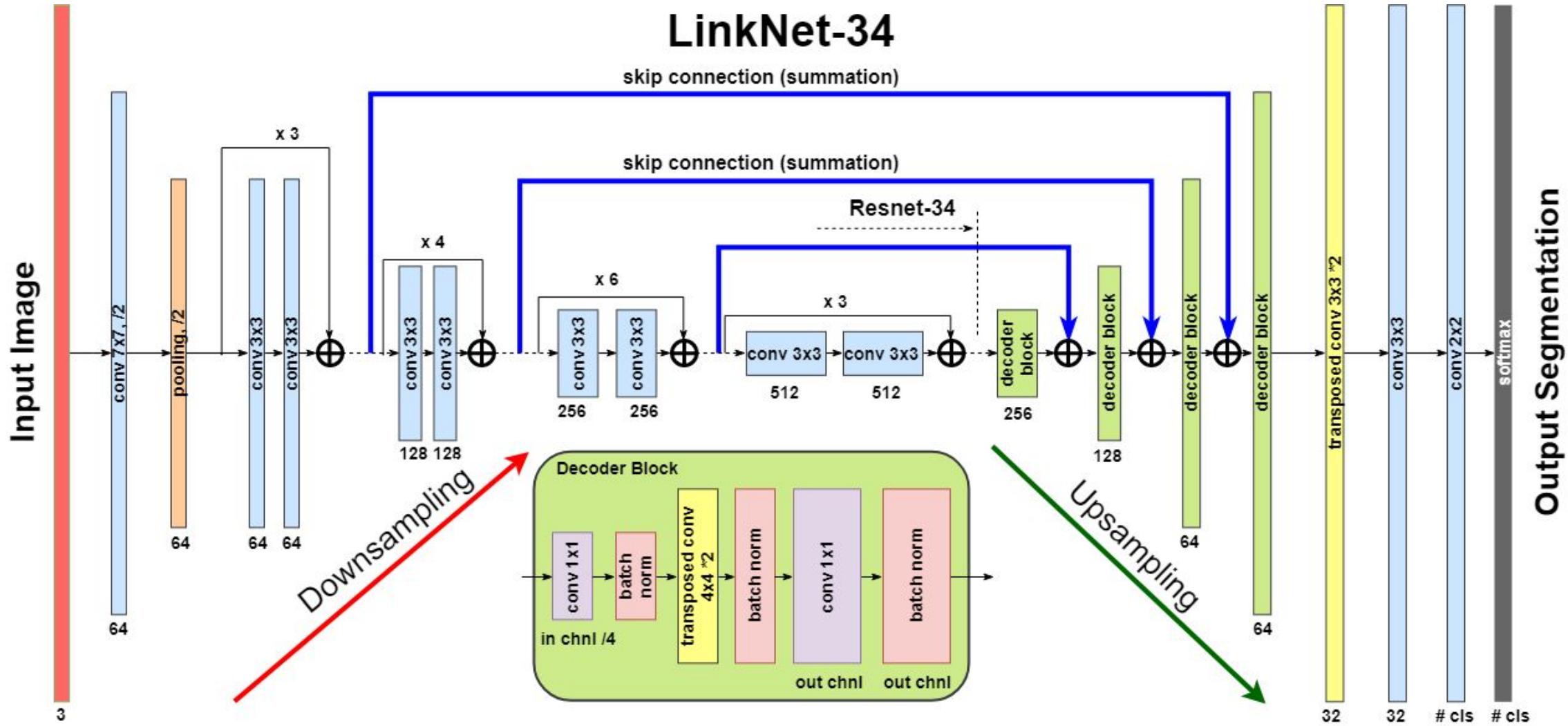
$$Loss_J = 1 - \log(J_{seg})$$

- Часто комбинируют с BCE:

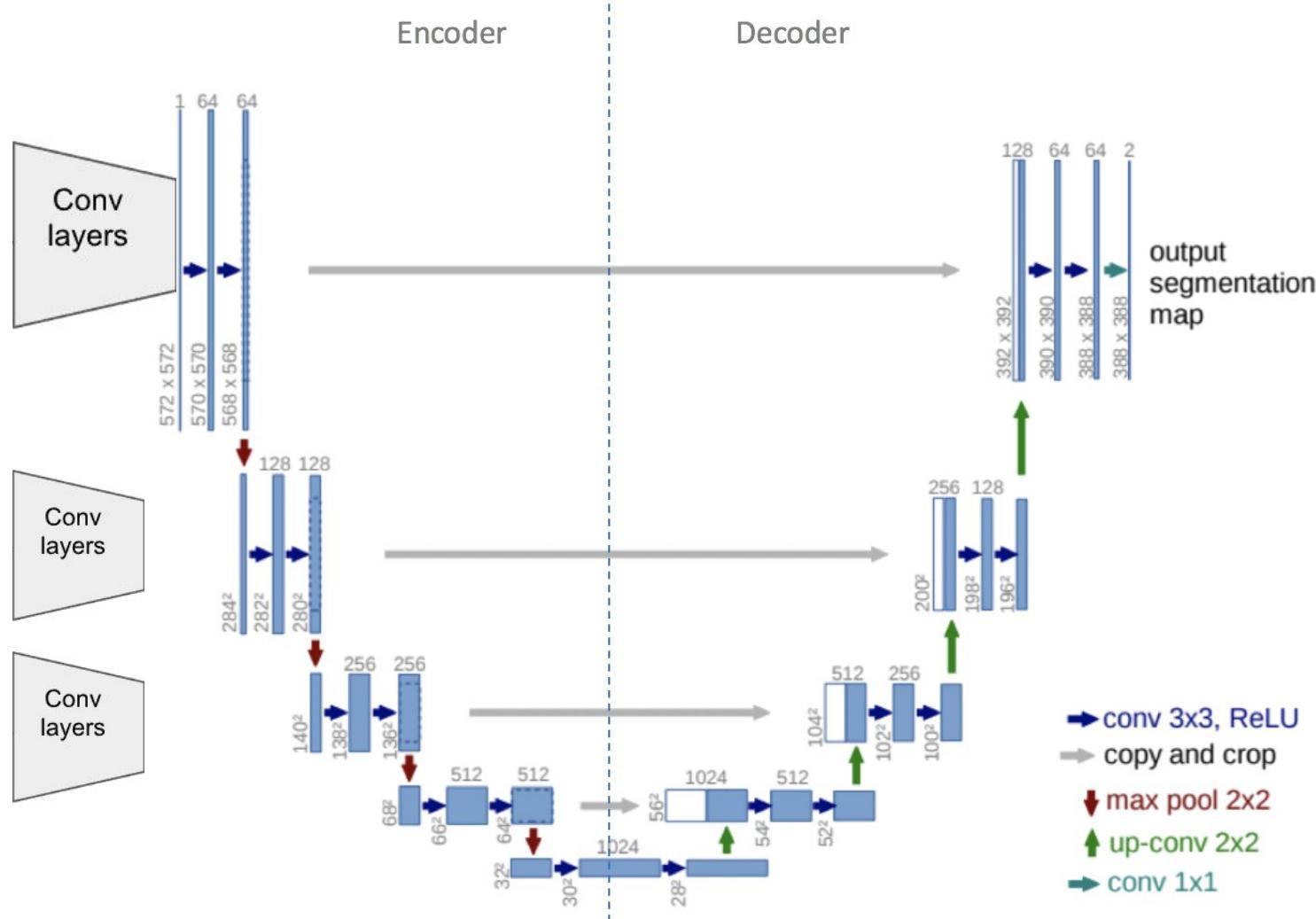
$$Loss = \alpha * Loss_{BCE} + (1 - \alpha) * Loss_J$$

Ternaus Net

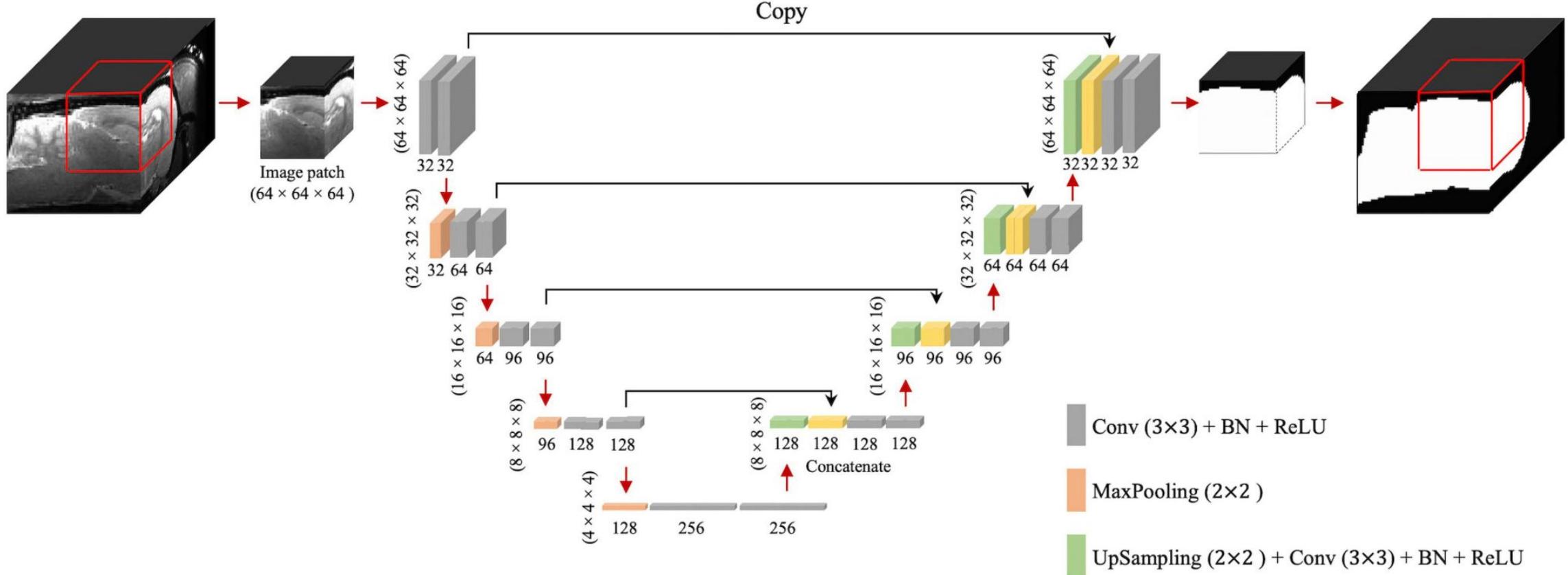
LinkNet-34



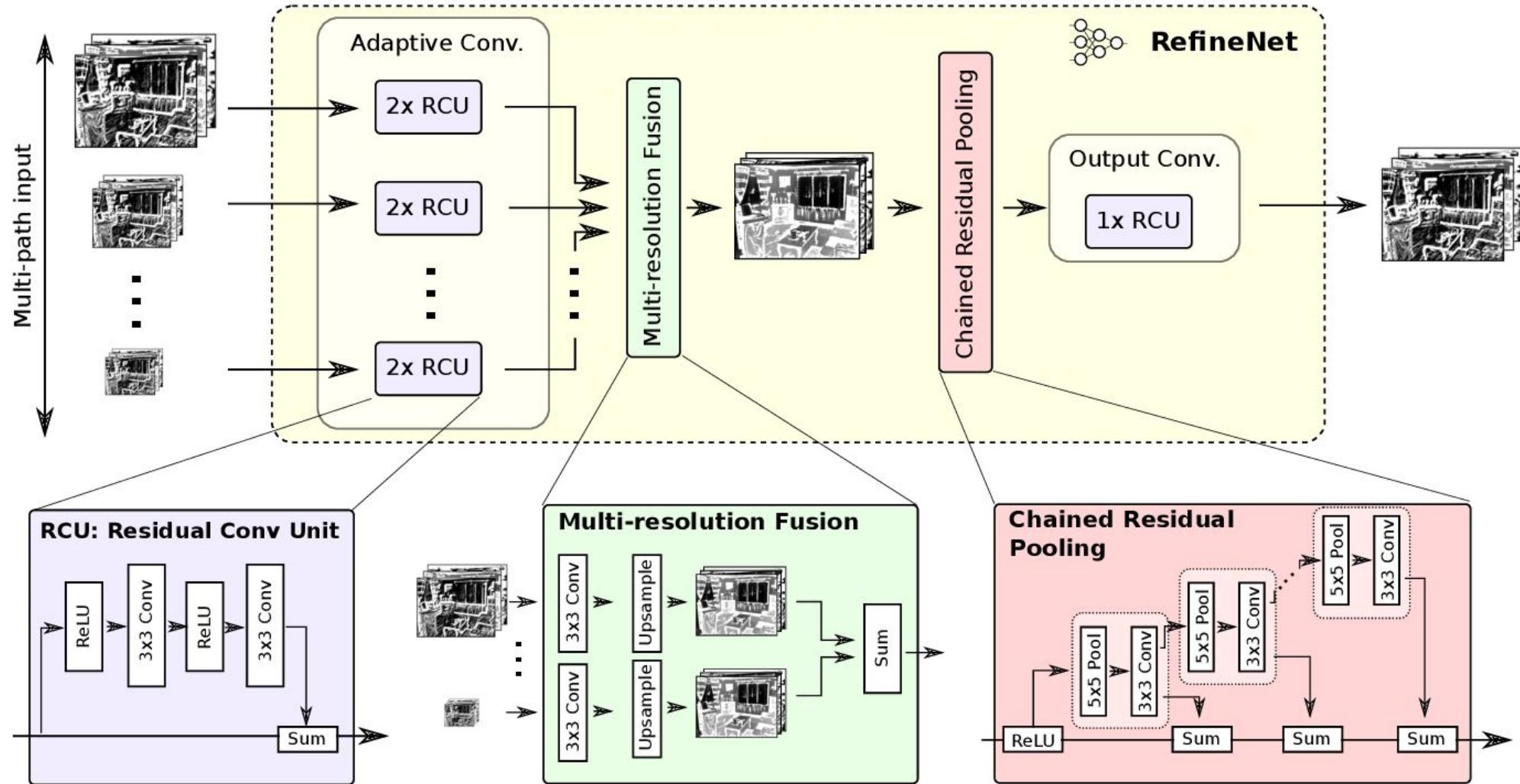
Multi-Input U-Net



3D U-Net

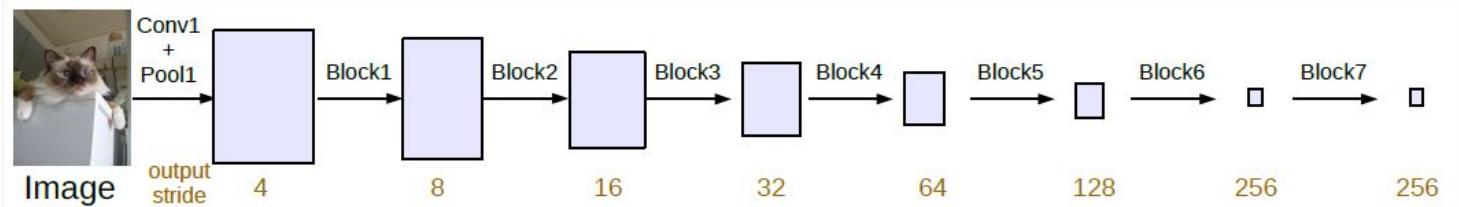


RefineNet

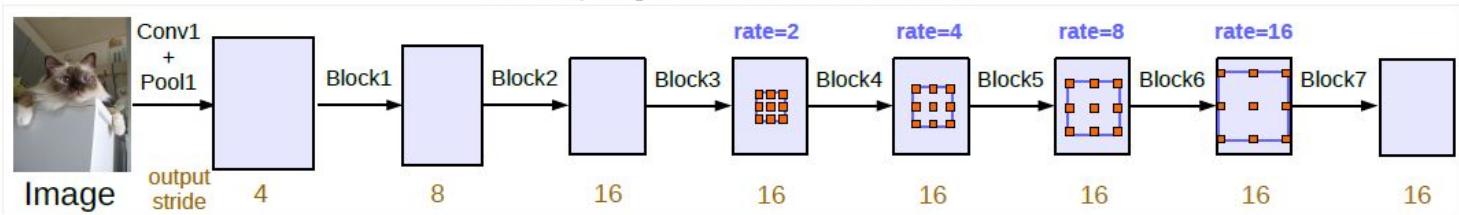


U-Net, но с очень специфической архитектурой кодировщика

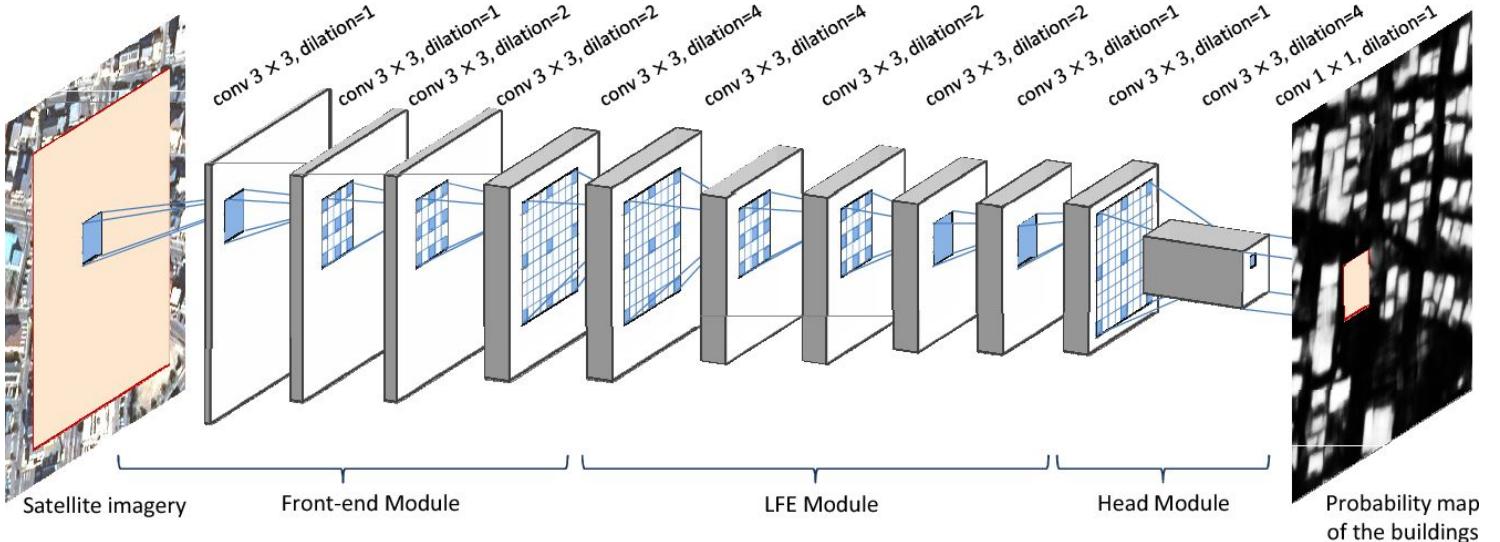
Dilated Convolutions



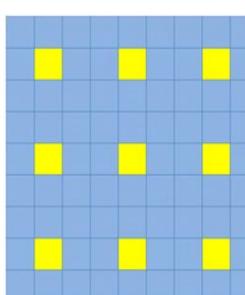
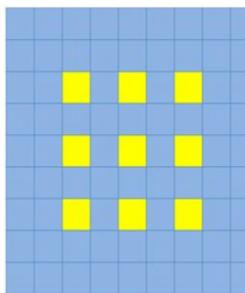
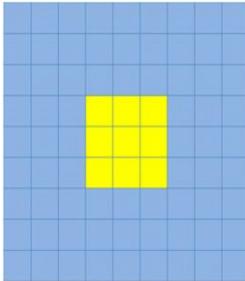
(a) Going deeper without atrous convolution.



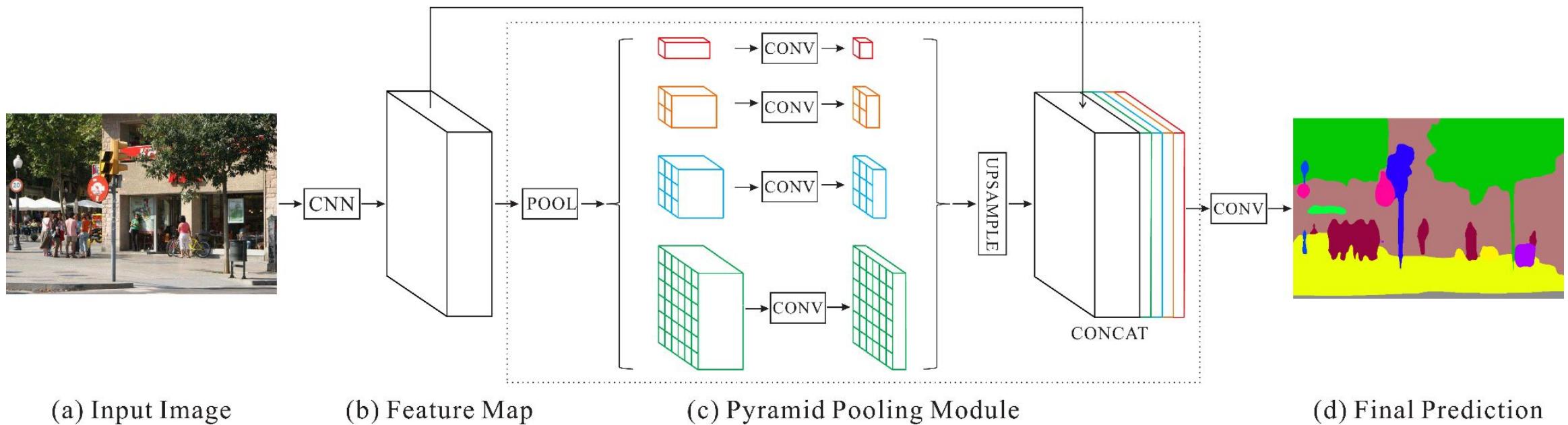
(b) Going deeper with atrous convolution. Atrous convolution with $rate > 1$ is applied after block3 when *output_stride* = 16.



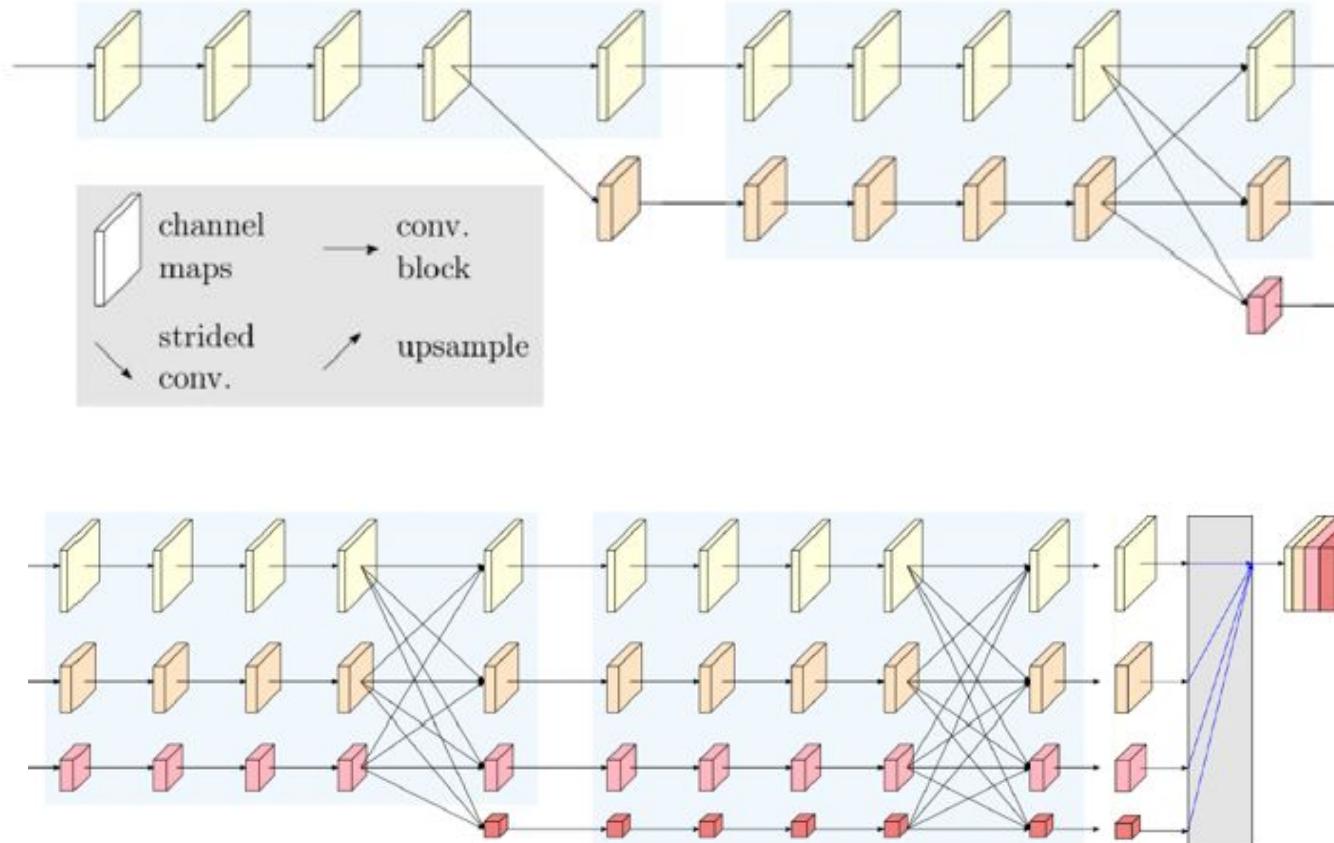
[Rethinking Atrous Convolution for Semantic Image Segmentation](#)



PSP Net (2016)



HRNet

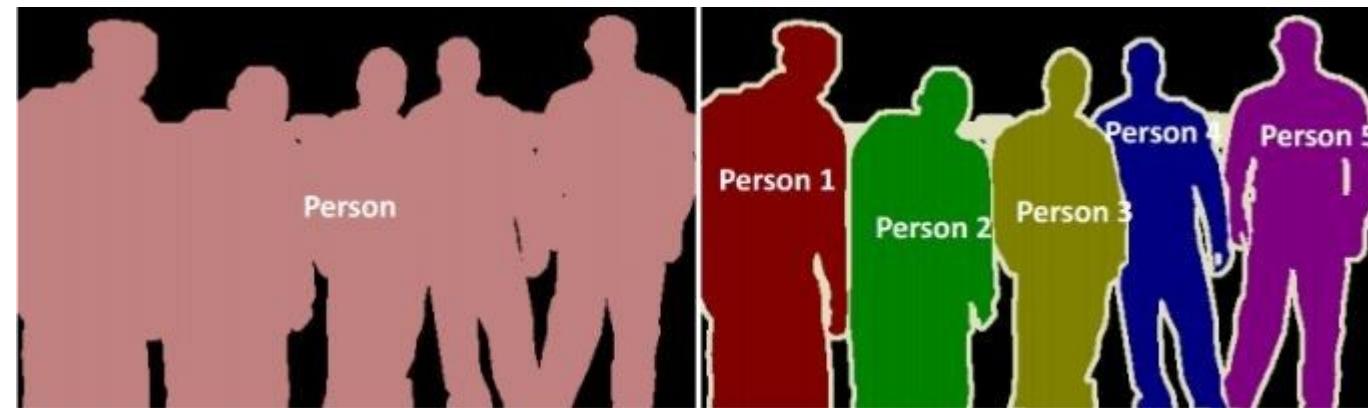


Объектная сегментация

• • •

Объектная сегментация

- Если объекты не “касаются” и не перекрывают друг друга, то разделить их маски не составит труда и после semantic segmentation
- ... Но так бывает не всегда

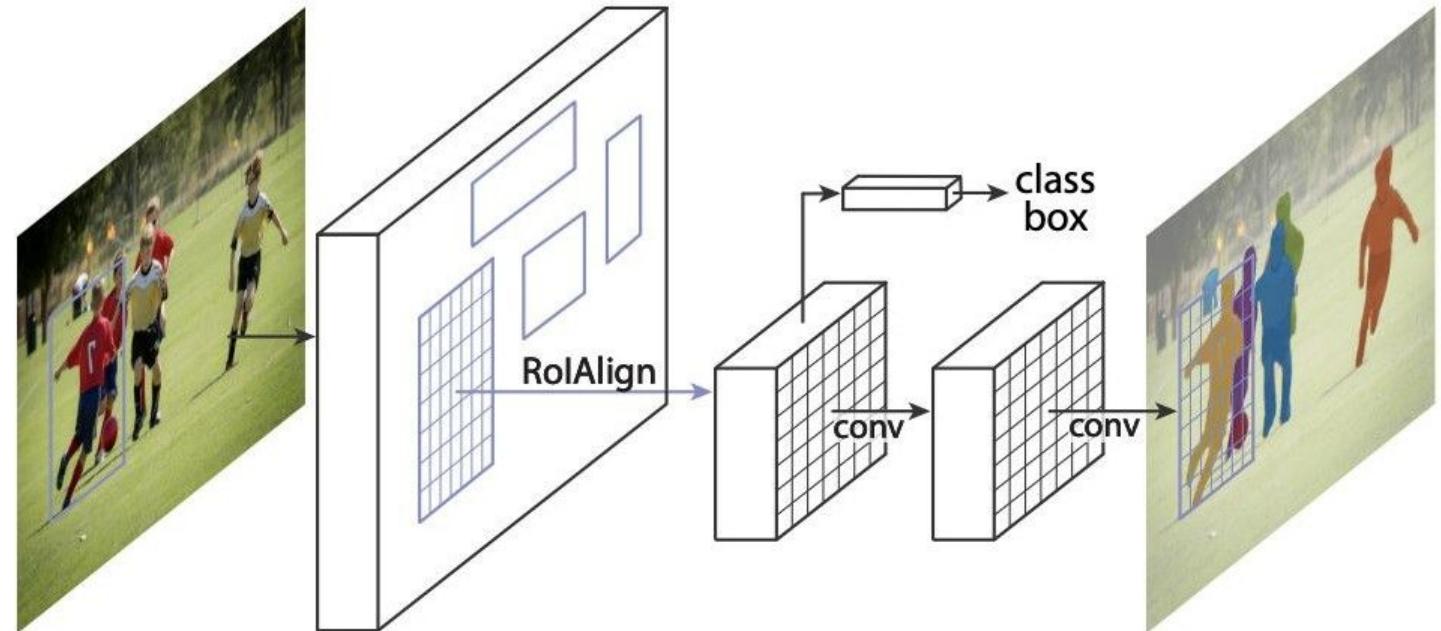


Объектная сегментация

- Идея: встроить в детектор объектов семантическую сегментацию для бокса вокруг каждого объекта

Mask R-CNN (2017)

- Mask R-CNN (2017)
- В основе - Faster R-CNN
- Дополнительная ветвь для предсказания бинарной маски во всех proposals
 - Маски не зависят от классов (т.е. сегментация на 1 класс)
- Вместо RoIPool – RoIAvg



Mask R-CNN (2017) - RoIAlign

- В Mask R-CNN используется операция RoIAlign: вместо грубого округления границ и пулинга значений используется интерполяция значений по сетке

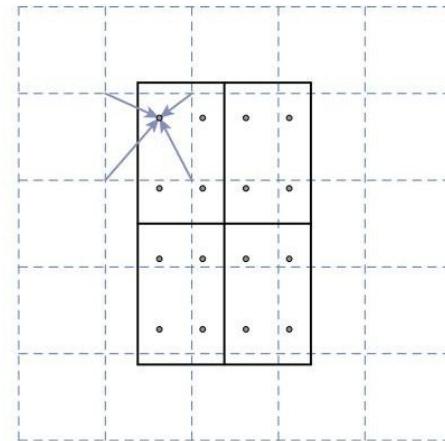


Figure 3. RoIAlign: The dashed grid represents a feature map, the solid lines an ROI (with 2×2 bins in this example), and the dots the 4 sampling points in each bin. RoIAlign computes the value of each sampling point by bilinear interpolation from the nearby grid points on the feature map. No quantization is performed on any coordinates involved in the ROI, its bins, or the sampling points.

Mask R-CNN - примеры

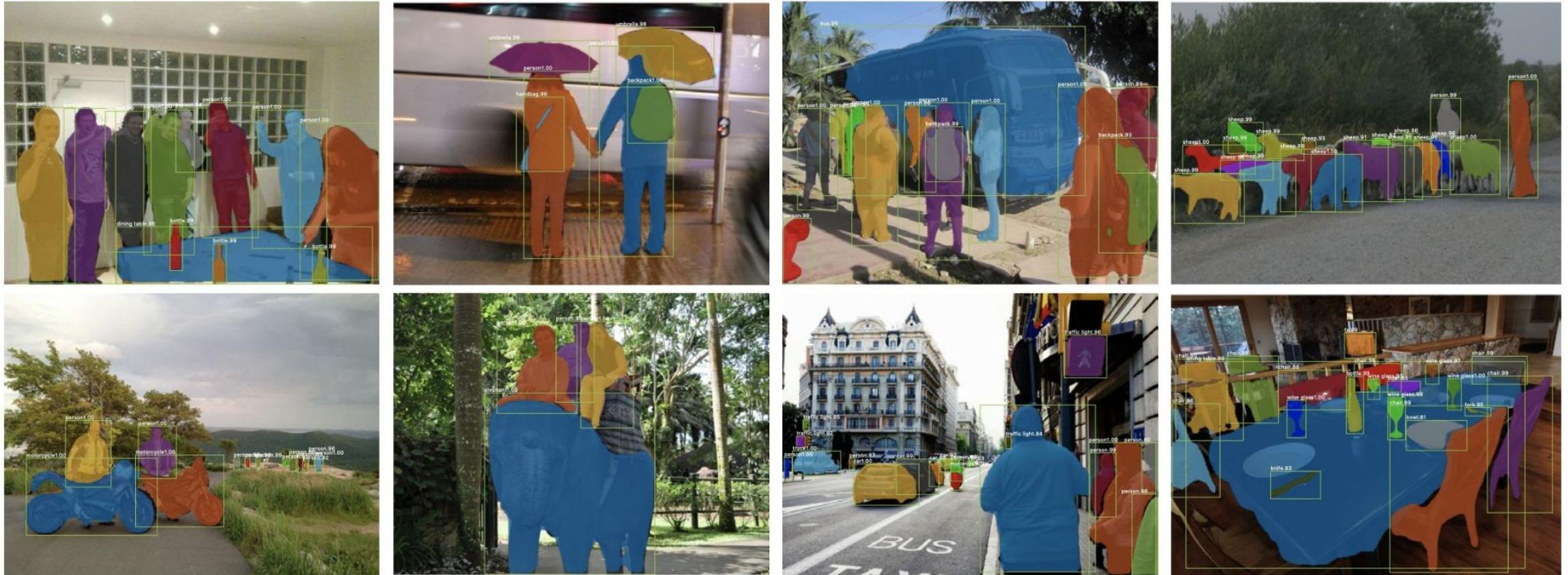


Figure 2. **Mask R-CNN** results on the COCO test set. These results are based on ResNet-101 [19], achieving a *mask AP* of 35.7 and running at 5 fps. Masks are shown in color, and bounding box, category, and confidences are also shown.

Mask R-CNN

- Есть в PyTorch

```
from torchvision.models.detection.mask_rcnn import MaskRCNNPredictor
```

Вопросы?



Спасибо
за внимание!