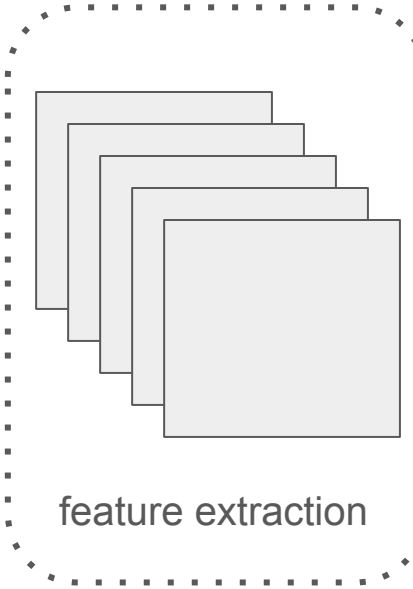
Детекция объектов

1. проблема постановки задачи
2. R-CNN
3. Fast-RCNN
    a. RoI
    b. Loss
4. RPN
    a. loss
5. Mask RCNN
    a. roi align
6. yolo
7. современные достижения

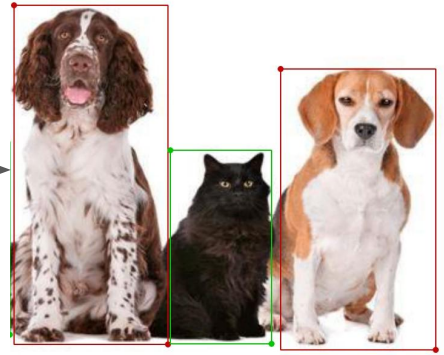
Детекция объектов. В чем сложности?



?

как учесть любое кол-во объектов?

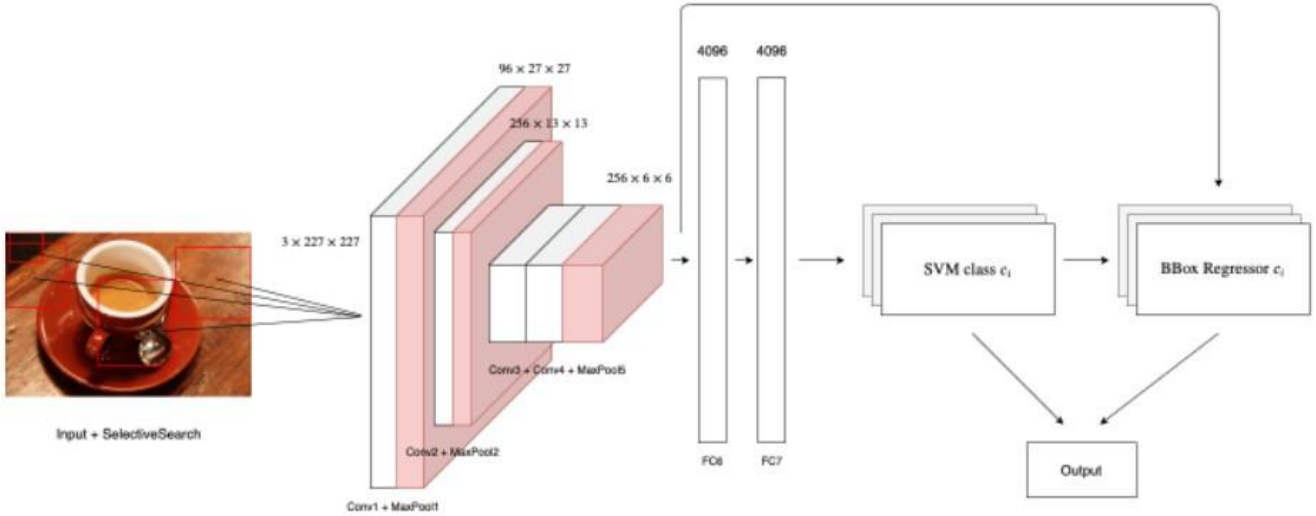
какой выход у сети?



как разные размеры бибоксов?

RCNN

1. Определение набора гипотез.
2. Извлечение из предполагаемых регионов признаков с помощью сверточной нейронной сети и их кодирование в вектор.
3. Классификация объекта внутри гипотезы на основе вектора из шага 2.
4. Улучшение (корректировка) координат гипотезы.
5. Все повторяется, начиная с шага 2, пока не будут обработаны все гипотезы с шага 1.



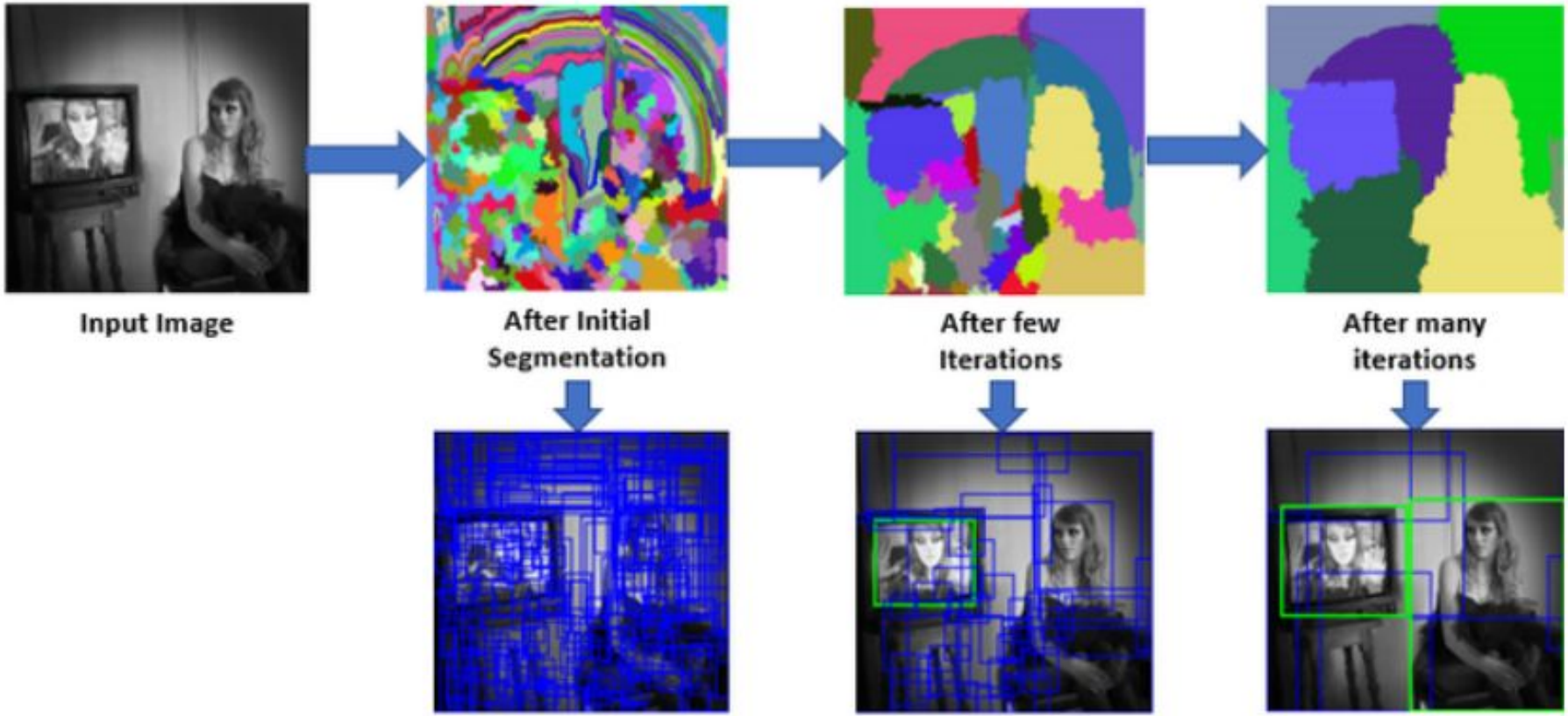
RCNN: отбор гипотез

генерируем из исходного изображения набор его частей (как мозаику). Для более осознанного выделения (чтобы в нарезанных частях были объекты) можно использовать Selective Search – верхнеуровнево, он позволяет составить набор гипотез (класс объекта пока не имеет значения), на основе сегментации определить границы объектов по интенсивности пикселей, перепаду цветов, контраста и текстур. Для более точной последующей обработки каждая гипотеза дополнительно расширяется на 16 пикселей во всех 4 направлениях – как бы добавляя контекст.

Итог:

Вход: исходное изображение.
Выход: набор гипотез разного размера и соотношения сторон.

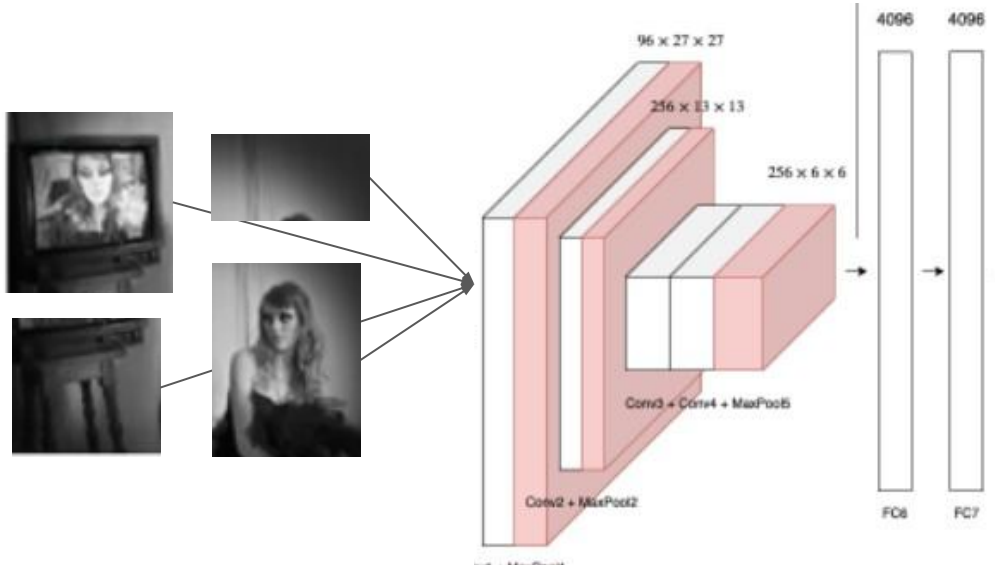
RCNN: selective search



RCCN: извлечение признаков

Извлечение из предполагаемых регионов признаков с помощью сверточной нейронной сети и их кодирование в вектор.

Итог:
Вход: каждая из предложенных на предыдущем шаге гипотеза.
Выход: векторное представление для каждой гипотезы.



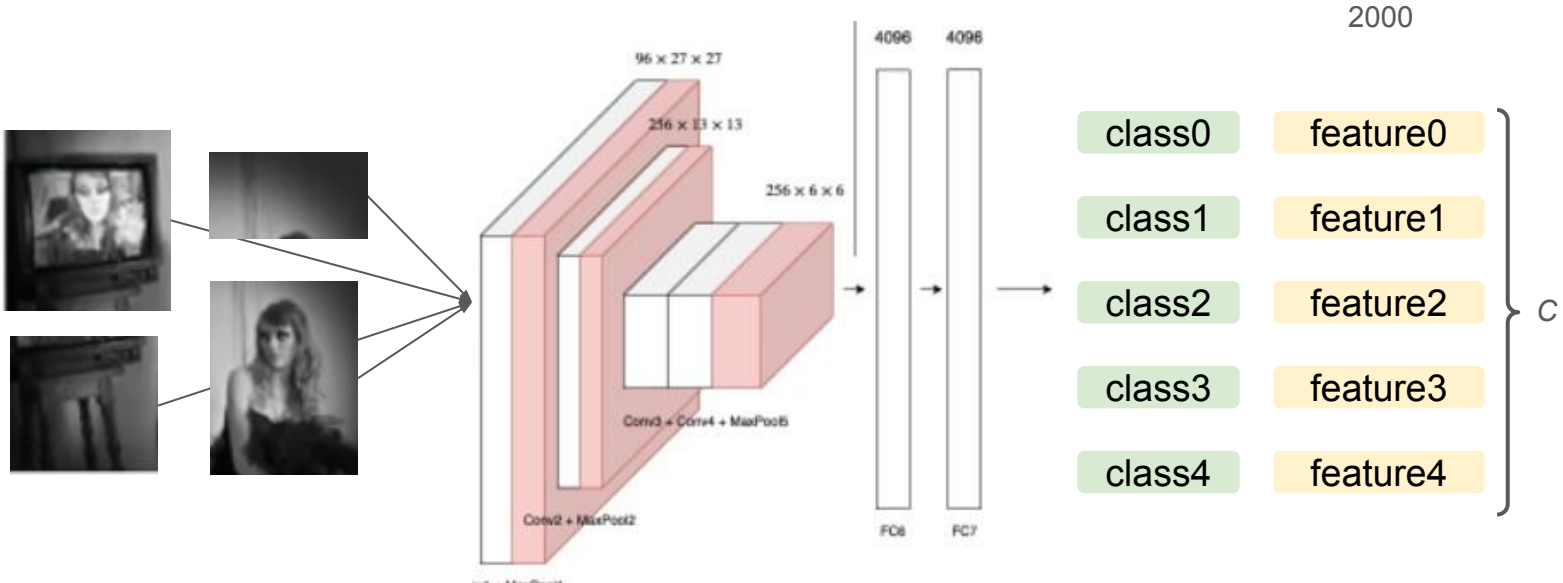
RCNN: классификация

Классификация объекта внутри гипотезы на основе вектора из шага 2.

Итог:

Вход: вектор каждой из предложенных гипотез из предпоследнего слоя сети (в случае AlexNet это FC7).

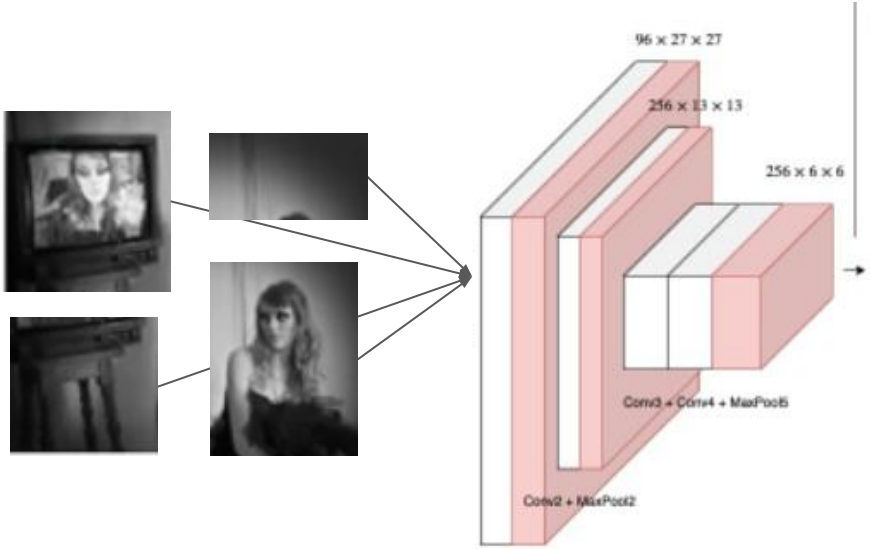
Выход: после последовательного запуска каждой гипотезы, получаем матрицу размерности $2000×N_{c}$, отображающую класс объекта для каждой гипотезы.



RCNN:

Улучшение (корректировка) координат гипотезы.

Итог:
Вход: карта признаков из последнего MaxPooling слоя для каждой гипотезы, которая содержит любой объект, кроме фона.
Выход: поправки к координатам ограничивающей рамки гипотезы.

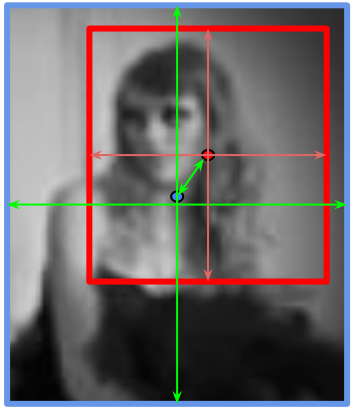


Δx_c

Δy_c

Δw_c

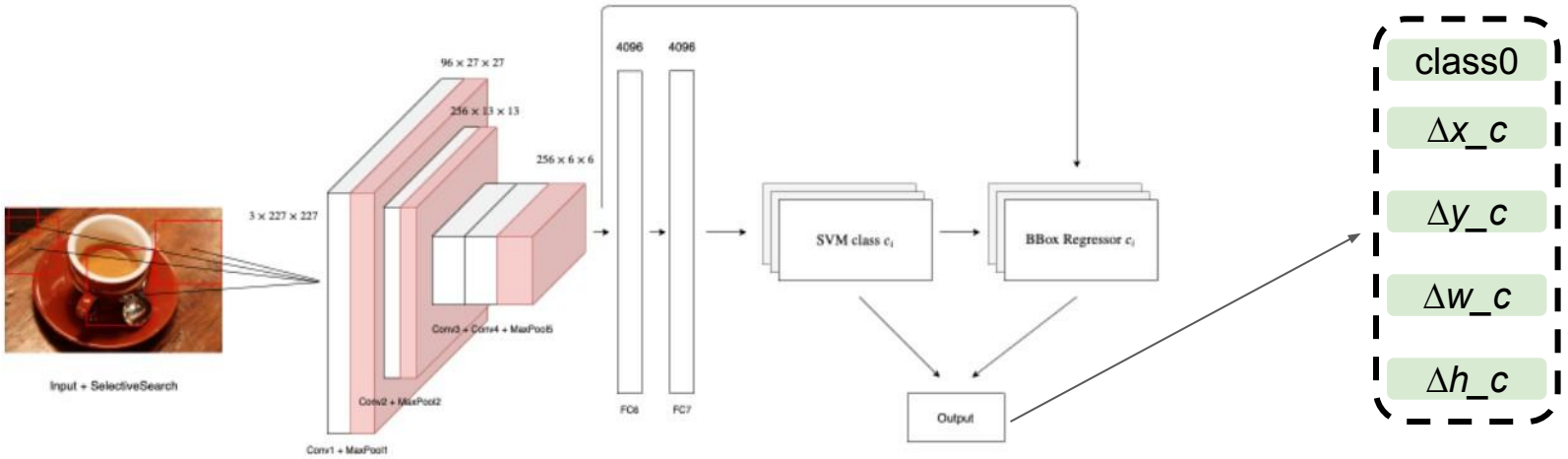
Δh_c



RCNN: Итог

В заключение выделим основные недостатки такого подхода:

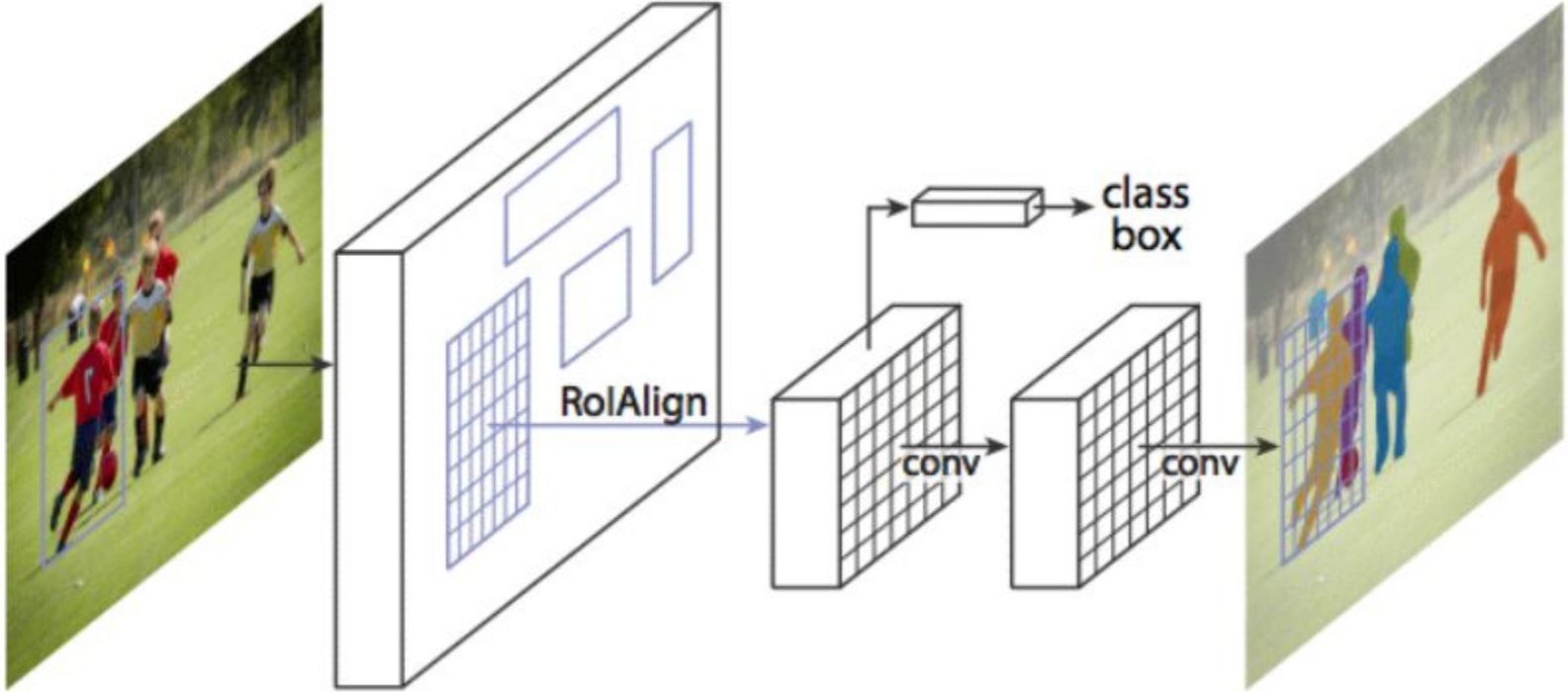
1. Гипотезы, предложенные на шаге 1, могут частично дублировать друг друга – разные гипотезы могут состоять из одинаковых частей, а каждая такая гипотеза отдельно обрабатывалась нейронной сетью. Так получается, что большая часть запусков сети более или менее дублирует друг друга без надобности.
2. Нельзя использовать для real-time работы, поскольку на проход 1 изображения (кадра) тратится ~53 секунды (NVIDIA Titan Black GPU).
3. Алгоритм выделения гипотез никак не обучается, а поэтому дальнейшее улучшение качества почти невозможно (никто не отменял плохие гипотезы).



Fast-RCNN

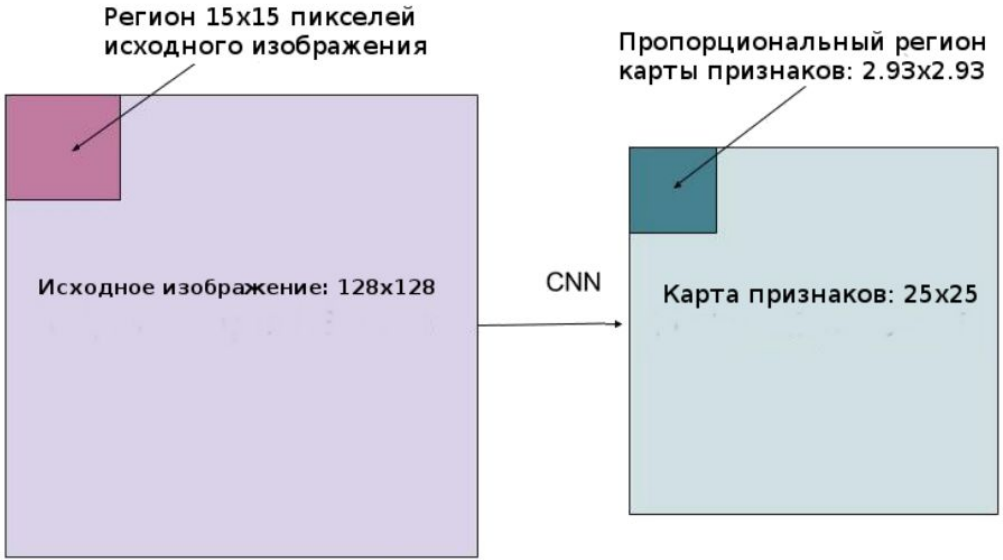
См. ноутбук в репозитории

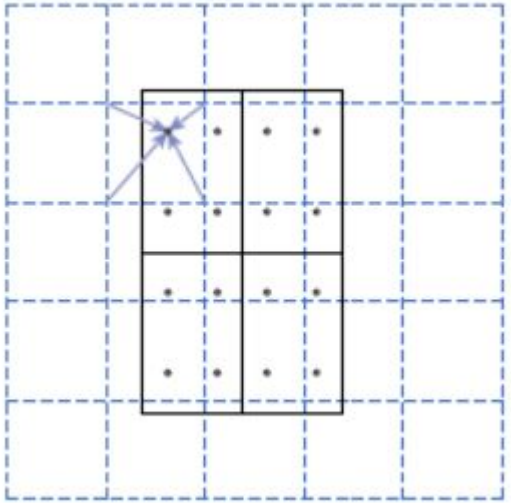
Mask-RCNN



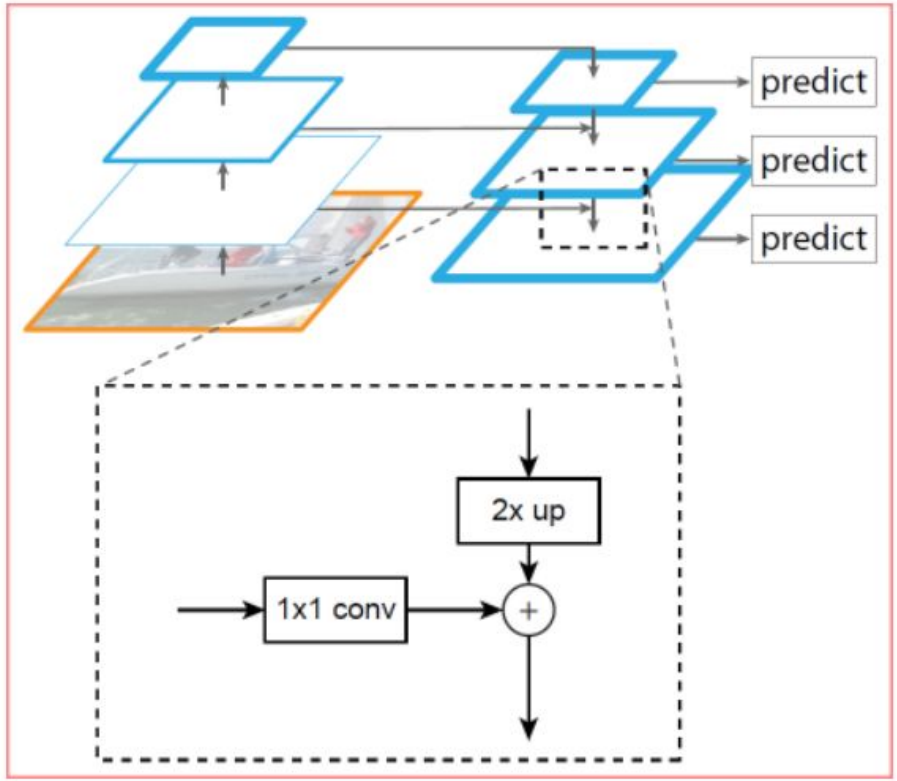
RoI Align

билинейная интерполяция

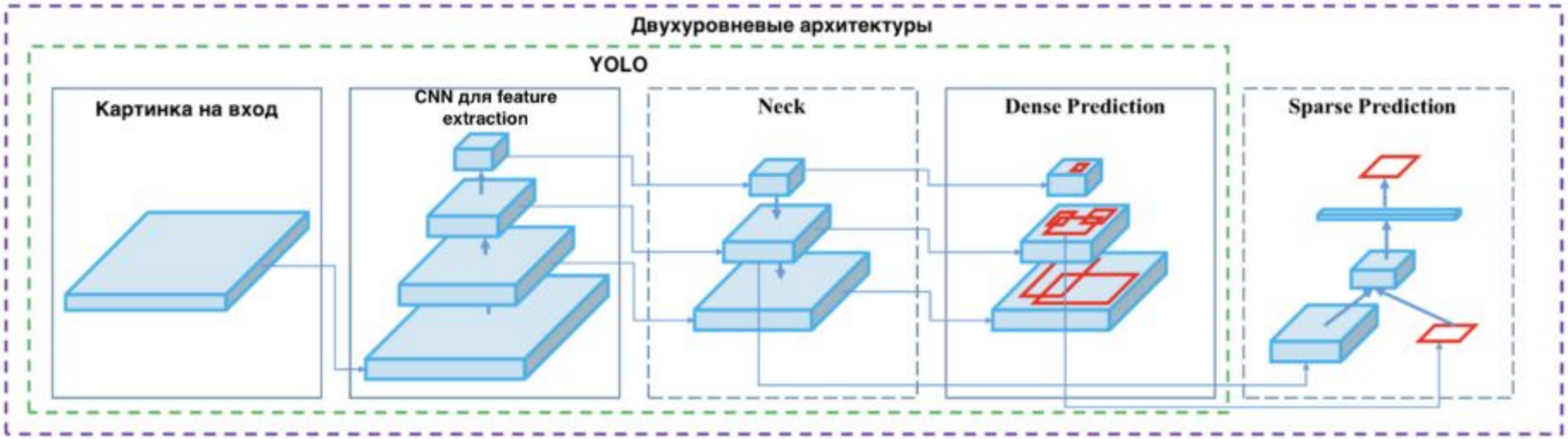




Feature pyramid network

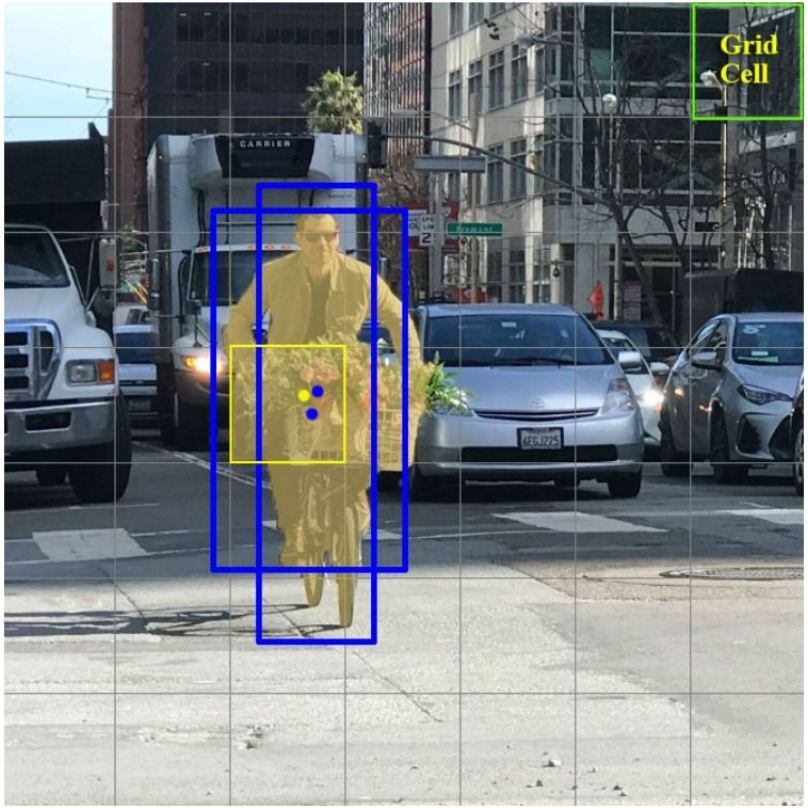


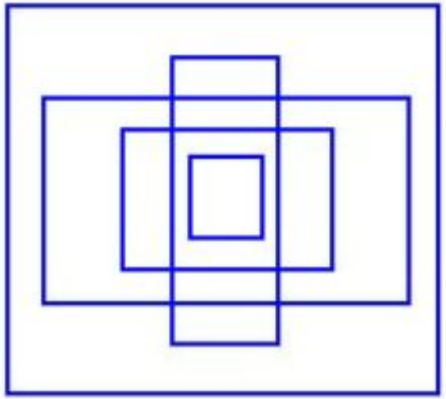
You only look once



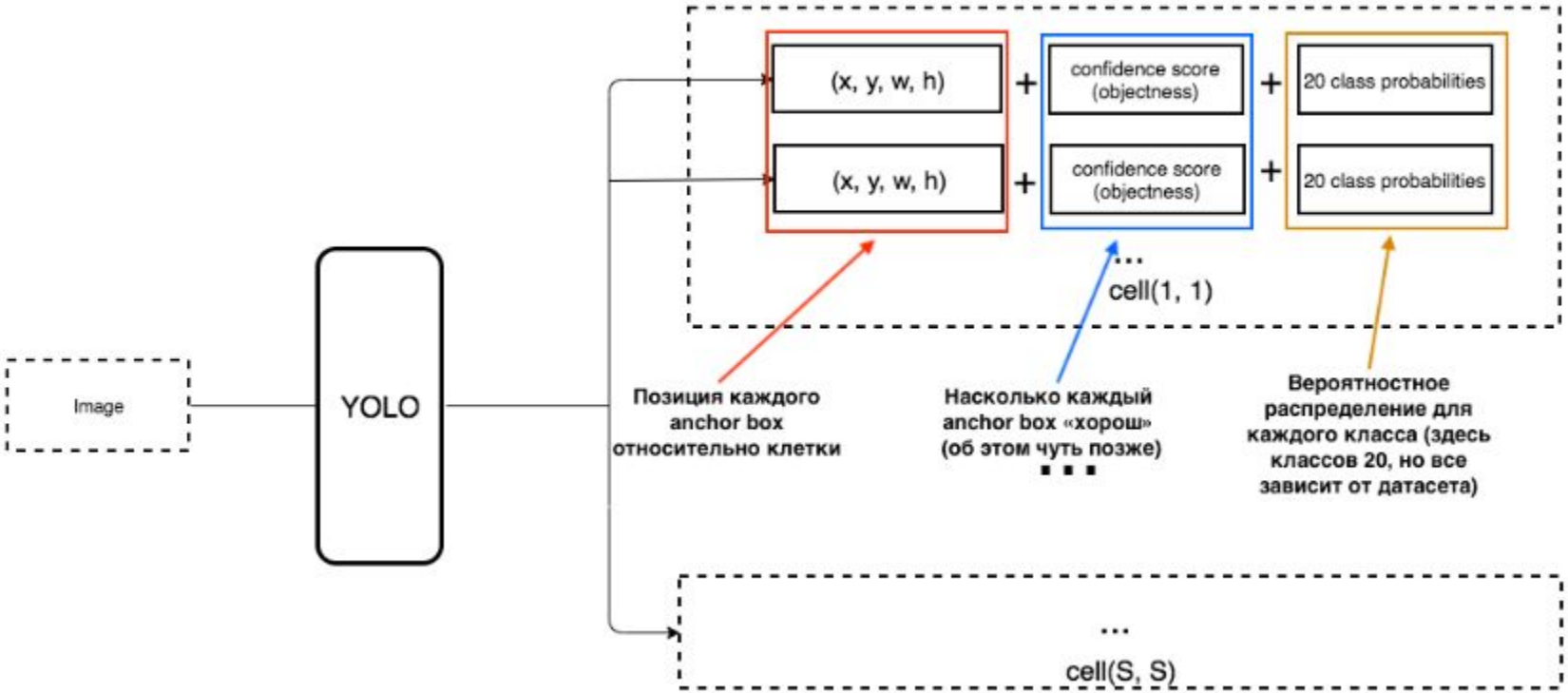
Хотим предсказывать объекты за один прогон картинки через архитектуру

Что вместо RoI?

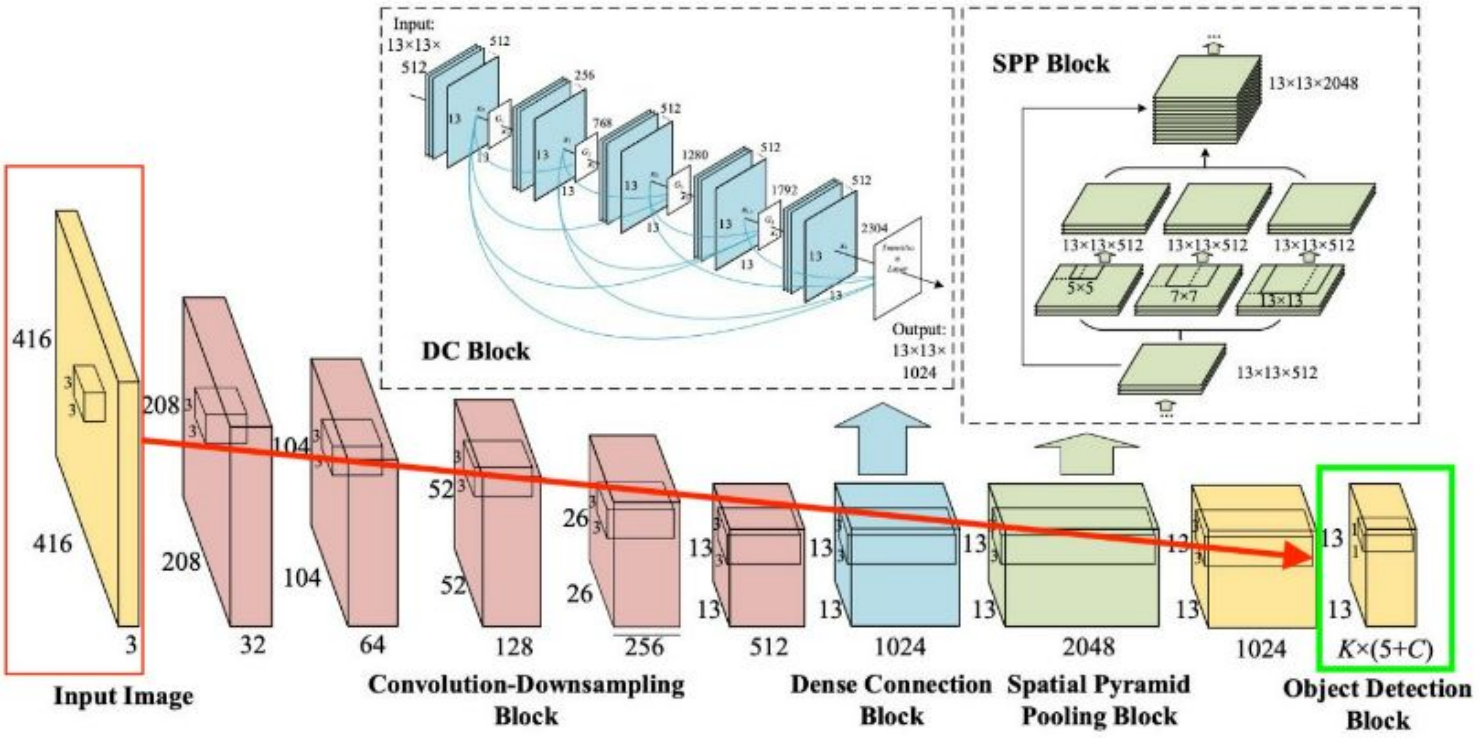




Что на выходе?



Yolo 3. Архитектура



Yolo loss

$$\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} [(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2]$$

$$+ \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} [(\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2]$$

$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} (C_i - \hat{C}_i)^2 + \lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{noobj} (C_i - \hat{C}_i)^2$$

$$+ \sum_{i=0}^{S^2} 1_{i}^{obj} \sum_{c \in classes} (p_i(c) - \hat{p}_i(c))^2$$