

Московский государственный университет имени М. В. Ломоносова



Факультет Вычислительной Математики и Кибернетики  
Кафедра Математических Методов Прогнозирования

## **ЭССЕ СТУДЕНТА 317 ГРУППЫ**

### **«Детектирование объектов на изображениях (часть 2)»**

Выполнил:

студент 3 курса 317 группы  
*Бербер Кирилл Андреевич*

Москва, 2021

# Содержание

|  |           |
|--|-----------|
| <b>1 Введение</b>  | <b>3</b>  |
| <b>2 Генерация регионов</b>  | <b>5</b>  |
| 2.1 Selective Search . . . . .                                       | 5         |
| <b>3 Основные методы детектирования объектов</b>                     | <b>7</b>  |
| 3.1 Non Maximum Suppression (NMS) и Soft-NMS . . . . .               | 7         |
| 3.2 Deconvolutional Single Shot MultiBox Detector (DSSD) . . . . .   | 10        |
| 3.2.1 Deconvolution модуль . . . . .                                 | 10        |
| 3.2.2 Prediction модуль . . . . .                                    | 11        |
| 3.3 Feature Pyramid Networks (FPN) . . . . .                         | 12        |
| 3.3.1 FPN + RPN (bounding-box proposal generation) . . . . .         | 15        |
| 3.3.2 FPN + Fast R-CNN . . . . .                                     | 15        |
| 3.4 Region-based Fully Convolutional Networks (R-FCN) . . . . .      | 16        |
| 3.5 Fully Convolutional One-Stage Object Detection (FCOS) . . . . .  | 21        |
| 3.6 CornerNet . . . . .  | 23        |
| 3.7 ExtremeNet . . . . .   | 25        |
| 3.8 CenterNet . . . . .  | 27        |
| 3.9 EfficientDet . . . . .   | 29        |
| 3.10 Приёмы детектирования объектов: аугментация Copy-Paste. . . . . | 32        |
| <b>4 Выводы</b>  | <b>34</b> |
| <b>5 Список литературы</b>   | <b>35</b> |

## **Аннотация**

Проблема распознавания образа приобрела выдающееся значение в условиях информационных перегрузок, когда человек не справляется с линейно-последовательным пониманием поступающих к нему сообщений, в результате чего его мозг переключается на режим одновременности восприятия и мышления, которому свойственно такое распознавание .

Неслучайно, таким образом, проблема распознавания образа оказалась в поле междисциплинарных исследований — в том числе в связи с работой по созданию искусственного интеллекта, а создание технических систем распознавания образа привлекает к себе всё большее внимание.

В этой работе рассмотрятся некоторые методы детектирования изображений и связанные с ними темы. Работа является продолжением первого эссе по теме детектирования изображений, поэтому некоторые понятия, связанные с распознаванием объектов, будут считаться известными.

# 1 Введение

Автоматическое детектирование и распознавание объектов на изображениях и видео - одна из основных задач компьютерного зрения. Классификация изображений (то есть определение того, кто или что изображено на фотографии) является подзадачей детектирования. Второй подзадачей является локализация объекта: необходимо указать области изображения, которой каждый из классифицированных объектов принадлежит. Правильная работа каждой из этих частей является необходимым условием корректной работы методов детектирования.

Тема детектирования объектов на изображениях набирает все большую популярность с течением времени. Рост числа публикаций можно наблюдать на (Рис. 1). С начала двухтысячных годов количество публикаций по данной теме возросло более чем в 10 раз. Закономерно этому, в последнее время было придумано множество различных методов детектирования.

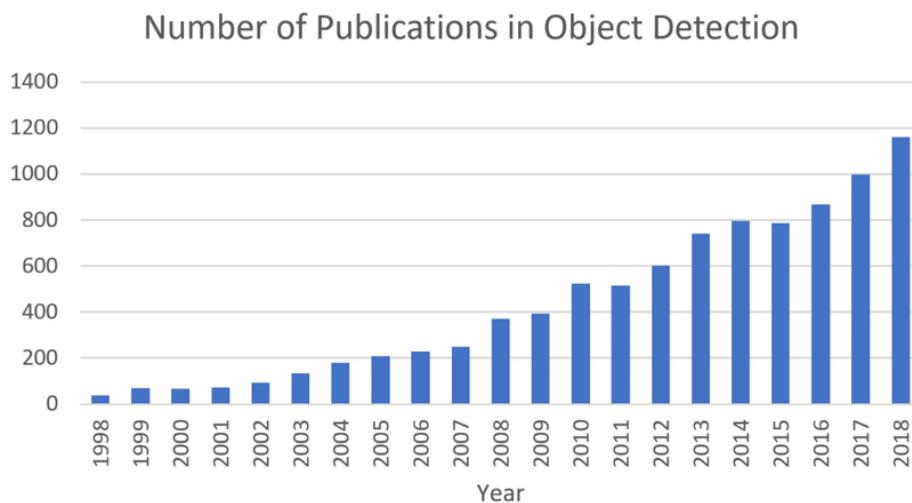


Рис. 1: Рост числа публикаций о детектировании объектов 1998 по 2018 гг. [1]

Можно выделить несколько классов, на которые делятся все методы детектирования:

1. One-stage detectors – одностадийные детекторы. Более быстрые, по сравнению с two-stage.
2. Two-stage detectors – двустадийные, с процедурой пораждения регионов. Отличаются большей точностью по сравнению с первым классом методов, однако проиризывают в скорости.
3. Proposal-free detectors – в этих методах отсутствует часть генерации регионов.

Примеры методов из каждого класса мы рассмотрим далее в этом эссе.

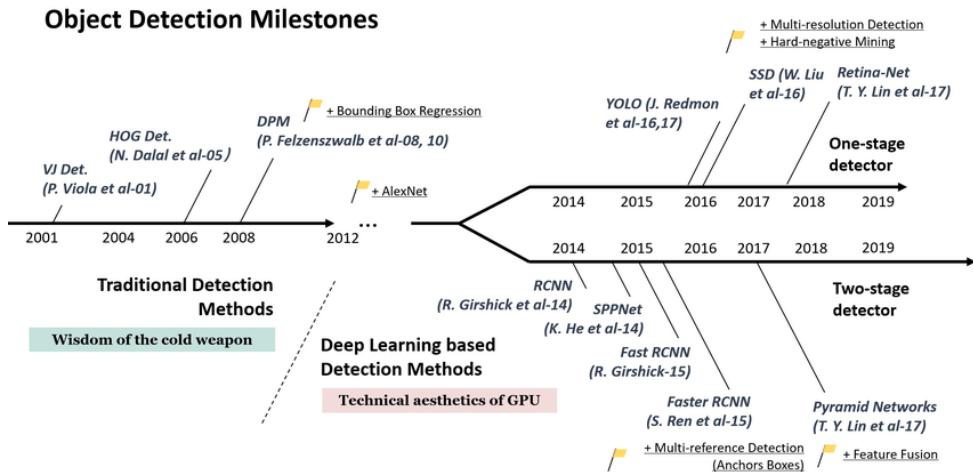


Рис. 2: Карта методов детектирования [1]

## 2 Генерация регионов

Одним из возможных подходов решения задачи детектирования объектов является следующая последовательность. Если у нас уже имеется корректно работающий классификатор, то ему на вход можно подавать всевозможные области изображений, в которых может потенциально находиться нужные объекты, а сам классификатор дальше будет решать, что именно в этих областях находится.

Стоит отметить, что эта задача не является тривиальной, потому что многие классификаторы не могут работать с изображениями разных размеров, они в себе содержат полно связные слои, которые работают только на заданных размерах. Однако эту проблему можно решить, например, масштабированием найденного региона растягиванием и сжатием. То есть приведению к фиксированному размеру.

### 2.1 Selective Search

Selective Search (или выборочный поиск) является один из методов генерации регионов на изображении.

Описываемый метод сочетает в себе как приемы исчерпывающего поиска (exhaustive search), так и сегментации. Как и в случае сегментации, в процессе отбора областей для выборочного поиска используется структура изображения.

Алгоритм выделяет все области на изображении, пиксели которых имеют примерно один цвет, и соединяет их в компоненты. Это происходит при помощи алгоритма кластеризации, учитывая графовую структуру изображения: соседние пиксели считаются соединенными ребром, а несоседние по стороне - нет. Это позволяет при сегментации генерировать непрерывные регионы пикселей, схожих по цвету.

Работу алгоритма можно наблюдать на (Рис. 3) и (Рис. 4).

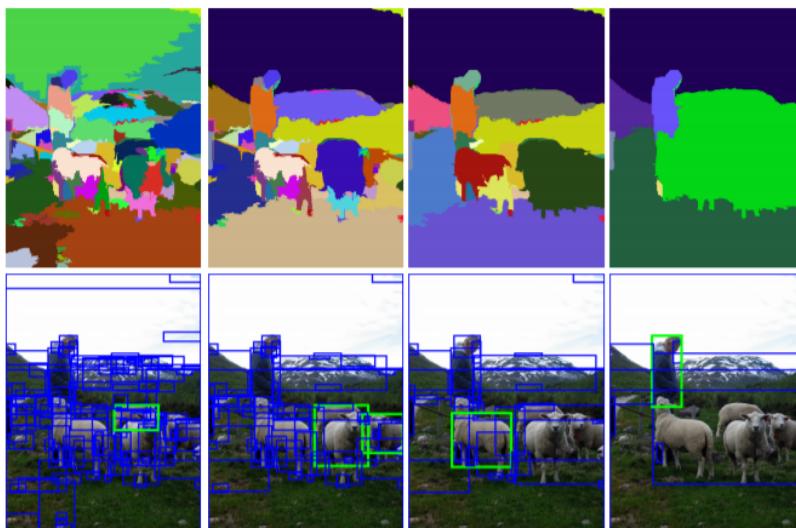


Рис. 3: Пример работы Selective Search [2]

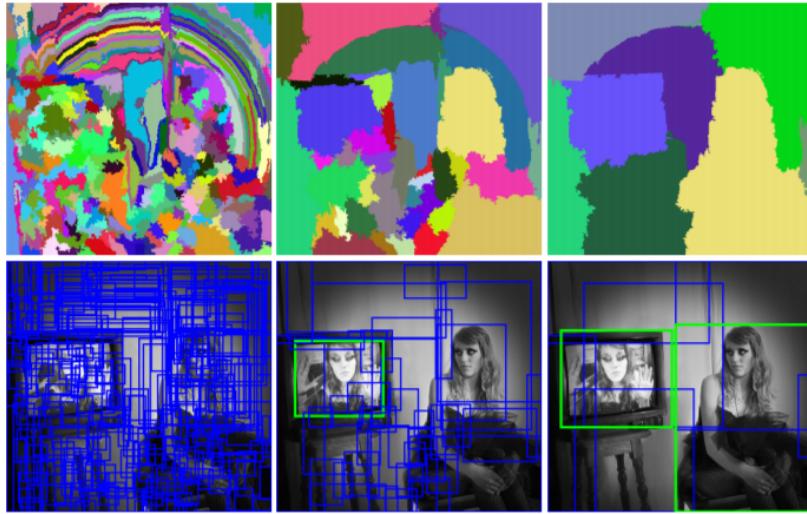


Рис. 4: Пример работы Selective Search [2]

Проведем более формальное описание работы алгоритма: На вход подается изображение RGB формата

1. Первым шагом создается пустой список  $S$ .
2. Изображение разбивается на сегменты (регионы), используя цветовые параметры пикселей и графовую структуру изображения  $R = \{r_1, \dots, r_k\}$ .
3. Для каждого сегмента в список  $S$  помещается его обрамляющая рамка. Для каждой пары сегментов ( $r_i$  и  $r_j$ ) вычисляется мера их схожести, и она добавляется в список  $S$ .
4. Если мера сходства большая – сегменты объединяются: вычисляется максимальное значение списка  $S(\max(S))$ , и запоминаются значения  $r_i$  и  $r_j$ , при которых это значение достигается. Информация о новом сегменте запоминается, а о объединенных старых удаляется.
5. Если изображение еще не единая связная область (число сегментов  $> 1$ ), происходит переход к П.2
6. Для каждой рамки из  $R$  оценивается наличие в ней объекта: и области, в которых объект есть записываются в итоговый список  $L$  - множество потенциальных расположений объекта (Set of object location hypotheses).[2]

На основе Selective Search были разработаны первые алгоритмы детектирования объектов.

### 3 Основные методы детектирования объектов

Большинство алгоритмов, после своей работы выдают множество областей, в которых с какими-то вероятностями предсказываются известные классы и, возможно, какие-либо уточнения относительно положения объектов внутри предсказанных областей. Однако множество регионов могут накладываться друг на друга и предсказывать одни и те же объекты. Необходимо среди всего множества предсказаний выбрать и оставить наиболее важные. За это отвечает модуль Non Maximum Suppression о котором речь пойдет ниже.

#### 3.1 Non Maximum Suppression (NMS) и Soft-NMS

Алгоритм NMS отвечает за слияние рамок, классифицирующих один и тот же объект.

На вход алгоритм получает список предположительных рамок –  $B$ , в которых расположены объекты, значения уверенности предсказания  $S$  (confidence score), для каждой рамки соответственно, и потог перекрытия  $N$ . На выход подается список отобранных предположений  $D$ .

Порог перекрытия рамок между собой считается, как IoU – отношение пересечения областей к объединению (Рис. 5).

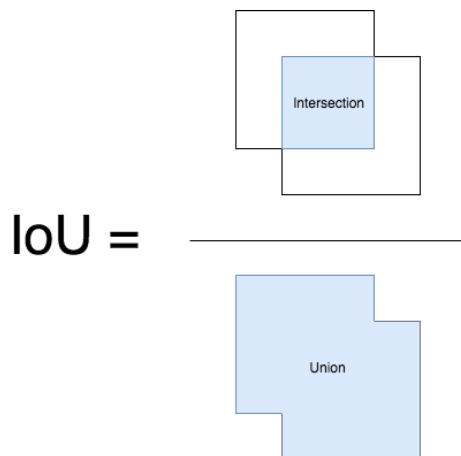


Рис. 5: Intersection over Union [3]

Сам алгоритм состоит из следующих шагов:

1. Выделенные области упорядочиваются по значению уверенности  $S$ . (Берутся только разумные показатели  $S$ , устанавливается определенный порог для отсечения)
2. Выбирается область  $R$  с наибольшим значением уверенности  $S$ . Эта область удаляется из  $B$ .

3. Выбранная область помещается в конечный список D.
4. Считается IoU выбранной области R со всеми другими выделенными областями из B. При IoU больше выбранного порога N – происходит удаление области из B. Она ссылается на тот же самый объект, что и текущая выбранная рамка
5. Алгоритм повторяется сначала до того момента, пока множество B не станет пустым.

Пример корректной работы алгоритма можно видеть на (Рис. 6).

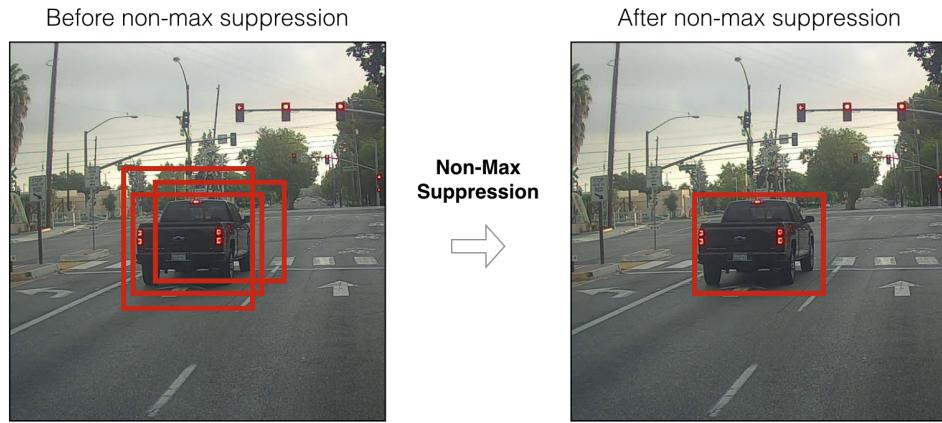


Рис. 6: Пример работы алгоритма NMS [3]

Однако такой подход не является идеальным. Результат алгоритма зависит от выбора значения порога N и при неудачном значении результат может быть неудовлетворительным.

Предположим, что  $N = 0.5$  и, при значении IoU больше заданного числа, рамки будут удаляться. Тогда, даже очень уверенные предсказания объектов, расположенных близко друг к другу будут удаляться. При данном выборе N на (Рис. 7) слева из трех рамок с изображениями лошадей останется только одна. Это снижает точность модели.

Однако и с этими в какой-то степени вырожденными случаями можно бороться. Эффективным способом будет использование Soft-NMS.

Идея заключается в том, чтобы вместо того, чтобы удалять объекты с высокой степенью уверенности S и большим IoU с текущей выбранной областью, можно снижать степень уверенности пропорционально IoU. Тогда, как видно на (Рис. 7) справа, внешние рамки не будут удалены, а их значение S будет снижено.

Записанное выше можно выразить следующей формулой:

$$S_i = \begin{cases} S_i & , \text{IoU}(S_i, R) < N \\ S_i(1 - IoU(S_i, R)) & , \text{IoU}(S_i, R) > N \end{cases}$$

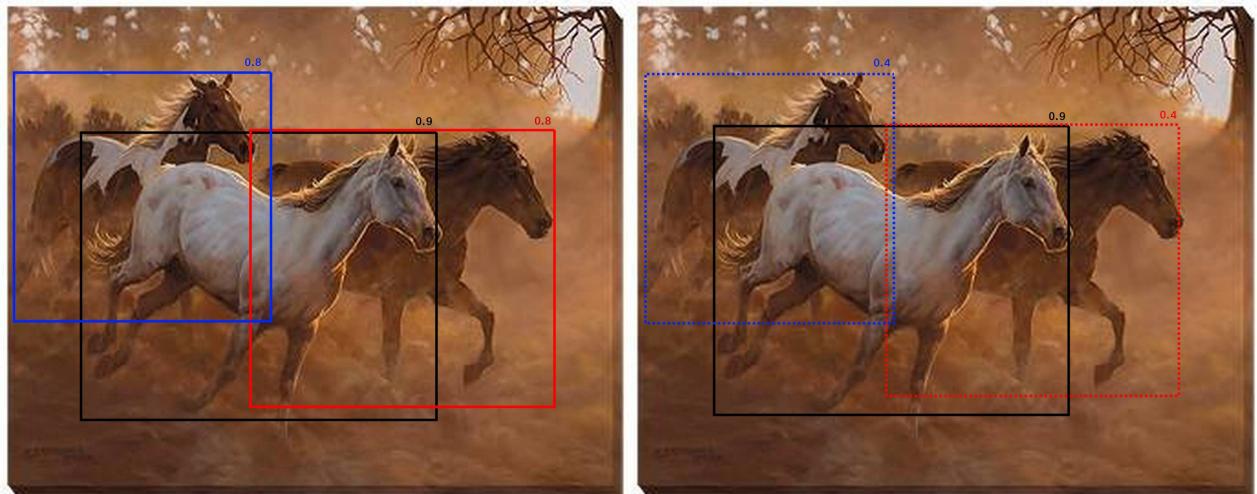


Рис. 7: Пример снижения точности NMS [3]

Таким образом, всего лишь изменение одной строки в реализации алгоритма NMS в значительной степени увеличивает точность.

## 3.2 Deconvolutional Single Shot MultiBox Detector (DSSD)

DSSD является продолжением SSD, описаной в первой части статьи. Сравним принцип работы этих сетей.

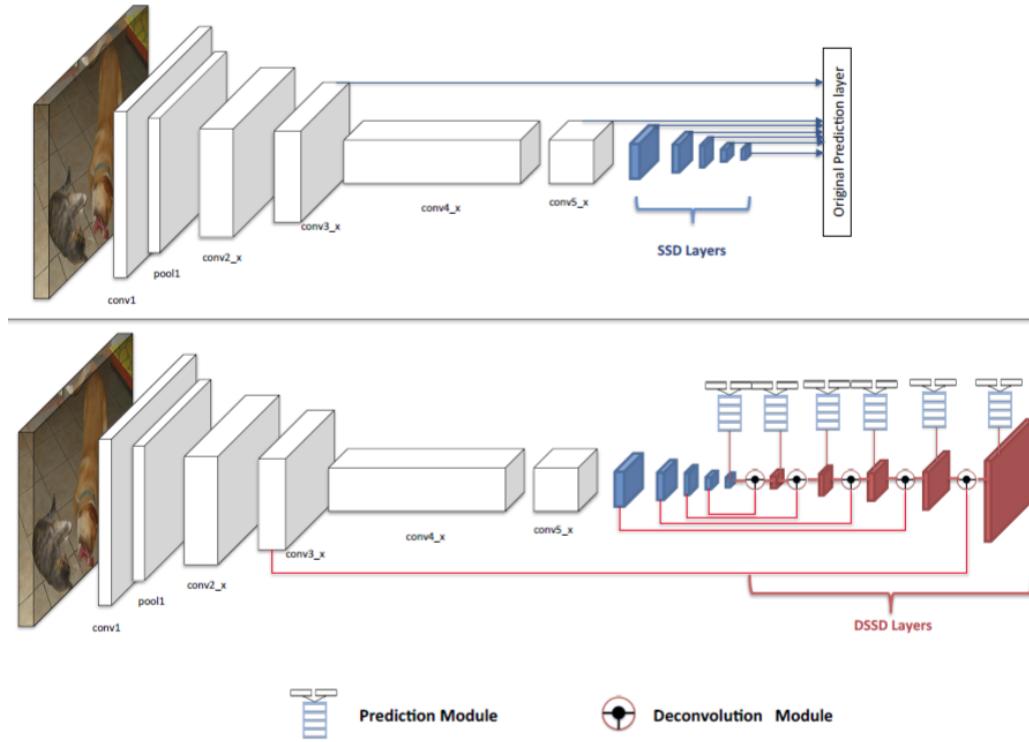


Рис. 8: Сравнение SSD и DSSD [4]

Остановимся на (Рис. 8) и посмотрим на различия методов. Слои, изображенные белым цветом, берутся из ResNet для составления карты характеристик. Добавляется только красная часть в DSSD, остальная часть схожа с SSD. Рассмотрим подробнее части Prediction и Deconvolution которые появляются в новых, выделенным красным частях.

### 3.2.1 Deconvolution модуль

Выходы из BN (Рис. 9) поэлементно умножаются. Deconvolution слои используются для увеличения разрешения карт характеристик для достижения большей точности распознаваний. Кроме того, для интеграции информации из более ранних карт используются Deconvolution модули. Формально, слои деконволюции не должны использоваться вообще. Повышение частоты дискретизации с последующим сверточным слоем также поможет достичь желаемого результата. Тем не менее, поскольку слои с повышающей дискретизацией не имеют каких-либо доступных для изучения параметров, это необязательно приводит к оптимальным результатам.

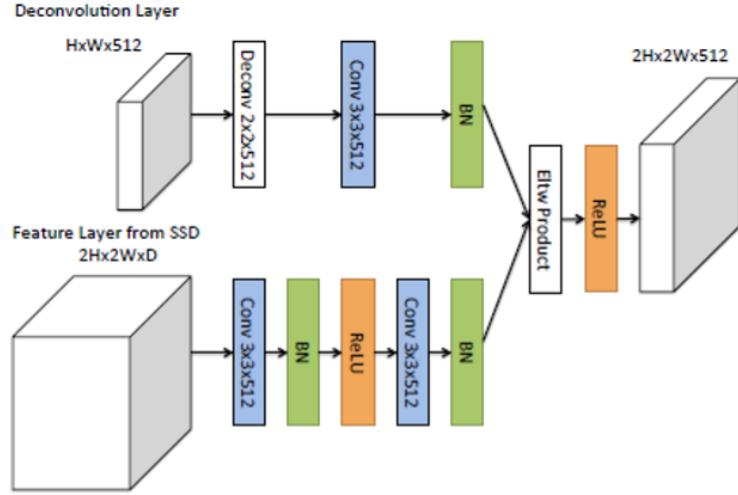


Рис. 9: Deconvolution модуль [4]

### 3.2.2 Prediction модуль

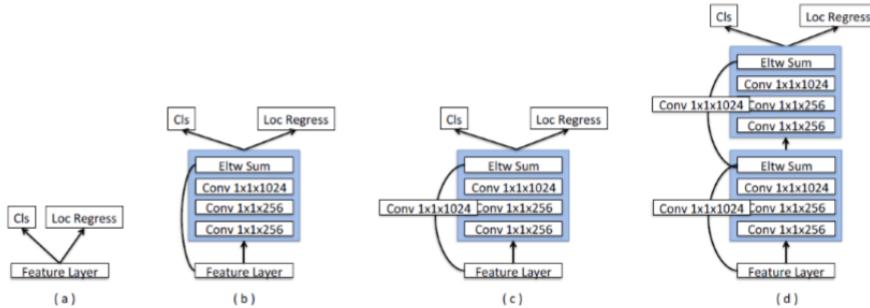


Рис. 10: Prediction модуль [4]

В исходной архитектуре SSD карты характеристик практически не обрабатываются до применения функций потерь. Поэтому слои нейросети должны научиться генерировать карты, которые переносят как пространственную и семантическую информацию из предыдущих слоев, так и преобразования, которые приводят к хорошей классификации. Поскольку разные ветви в SSD соответствуют различным масштабам, может потребоваться отменить последнее преобразования изображения, прежде чем применять то, которое лучше всего подходит для его масштаба. Чтобы решить эту проблему, DSSD использует модули Prediction, которые будут выполнять необходимую обработку карт характеристик, что приведет к увеличению качества классификации. Таким образом, нейросети (используемой для получения карты характеристик) нужно научиться лучше представлять информацию из изображения. Стоит отметить, что подобные модули для разных масштабов независимы друг от друга, поэтому можно изучать преобразования, специфичные только для одного масштаба.

### 3.3 Feature Pyramid Networks (FPN)

Поиск объектов разного размера на изображениях – достаточно трудоемкая задача. Для нейросетей может быть сложно находить маленькие объекты среди больших. Иначе говоря, необходимо решить проблему масштабирования объектов на изображении. Решением этой проблемы может стать Feature Pyramid. Можно подавать одно и то же изображение разного масштаба несколько раз на вход сети. (Рис. 11a). Образуется пирамида изображений. Недостатком такого метода можно назвать скорость. Несмотря на довольно точные прогнозы он работает довольно медленно.

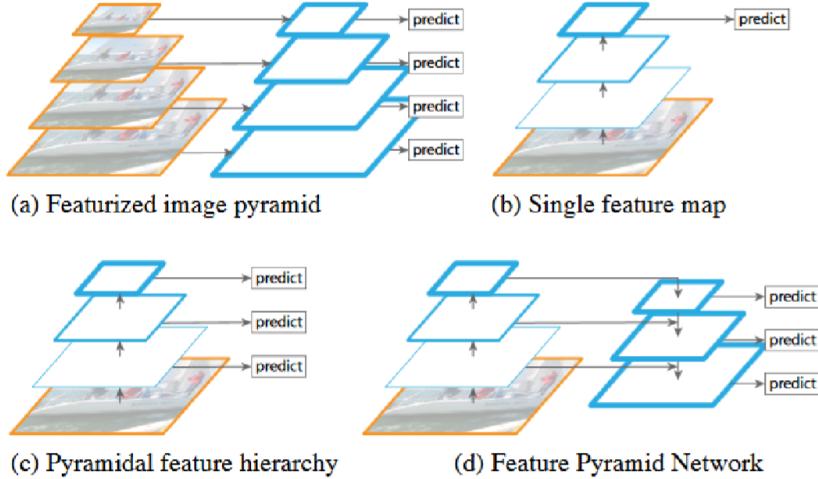


Рис. 11: Сравнение структур пирамид [6]

Несколько другой подход изображен на (Рис. 11b). Можно свести изображение до нужного размера и уже преобразованный формат подавать в нейросеть. Такой метод характерен для сверточных нейронных сетей, в частности, по такому принципу работает метод детектирования YOLO.

Возможно некоторое улучшение метода (Рис. 11c). В данном случае, как и в предыдущем, идет понижение масштаба изображения, но при этом на каждом масштабе делается предсказание того, что изображено. Этот подход реализован в методе SSD. Заметим, что предсказания выполняются во время прямого прохода и не требуют дополнительных вычислительных ресурсов. Однако в этом, как и в предыдущем методе, не учитывается контекст на более низких уровнях и, как следствие, маленькие объекты могут предсказываться плохо.

Стоит отметить, что все эти методы можно использовать с любыми нейронными сетями. Последний метод (Рис. 11d) является более современным решением. Feature Pyramid Network – точный и быстрый алгоритм. В этом случае мы не только делаем предсказание на каждом уровне, но и собираем информацию, которая поступает с более низких уровней. Это преобразование применяется в том числе и в Fast-RNN методе.

На (Рис. 12) можно наблюдать альтернативный вариант пирамиды. В данном случае делается только одно предсказание, что является преимуществом, однако к недостаткам можно отнести некоторую потерю информации.

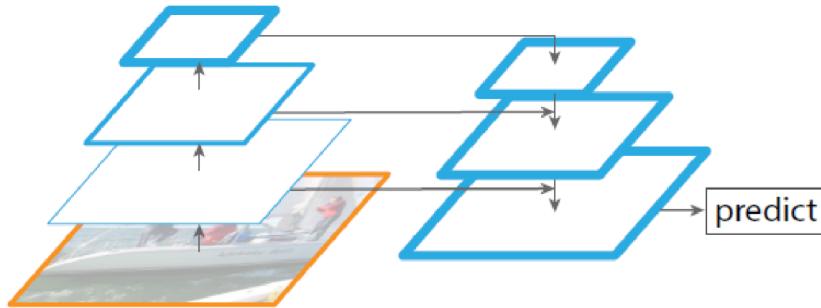


Рис. 12: Альтернативная структура пирамид изображений [6]

Остановимся подробнее на структуре, изображенной на (Рис. 11d). Ниже изображена более подробная интерпритация метода FPN. Есть три канала передачи информации: восходящий, нисходящий и боковой.

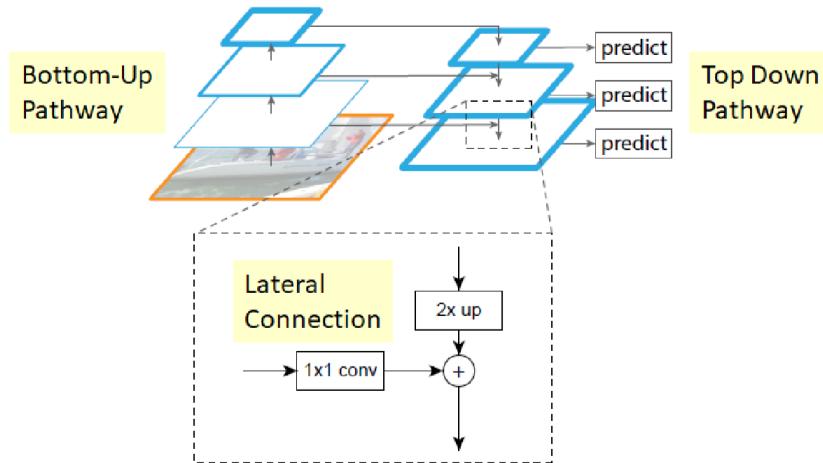


Рис. 13: Структура FPN [7]

Нисходящий путь (Top Down) реализуется с помощью upsampling, проиллюстрированного на (Рис. 14). Каждый пиксель масштабируется в N раз и таким образом получается карта характеристик большего размера. Literal connection в свою очередь складывает признаковые карты одинаковых размеров, соединяя восходящий на нисходящий пути.

Важно отметить, каким образом соединяются Literal connection и Top Down pathway. Это схематично изображено на (Рис. 13). Потенциально, тензоры, которые необходимо сложить, могут иметь разное число каналов, и для того, чтобы выровнить число каналов в изображениях, используется свертка размера  $1 \times 1$ .

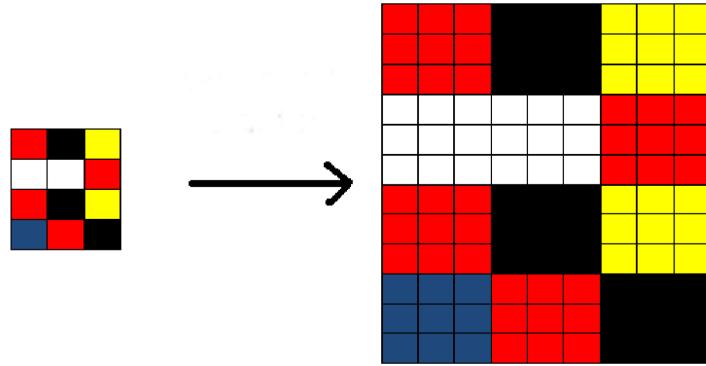


Рис. 14: Upsampling-правило [7]

Восходящий путь в данном случае – это просто посдедовательные переводы на вычисление в другую сверточную сеть. Тем самым создается некоторое множество карт объектов разных размеров. Выходные данные последнего слоя выбираются в качестве основного набора карт характеристик, которые будут использованы в процессе построения. Такой выбор основан на том, что этот слой хранит в себе информацию всех предыдущих.

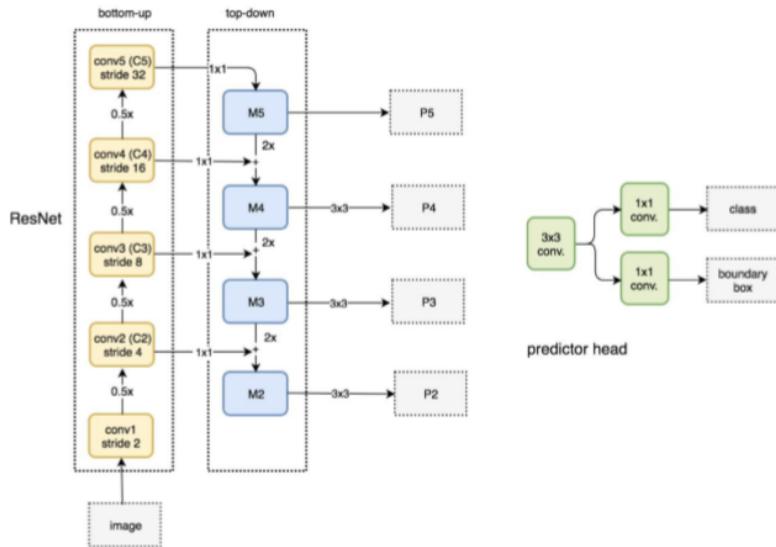


Рис. 15: Архитектура FPN [7]

Формально, архитектура изображенная на (Рис. 15) слева, это архитектура, занимающаяся подготовкой признаков, которые в свою очередь дальше подаются на вход детекторам P2-P5. Детектора P1 нет из-за большой пространственной размерности. Подразумевается, что объекты крупные и детекция идет на более высоких уровнях. В силу того, что левая часть (Рис. 15) с непосредственным детектированием объектов не связана, ее используют и в других областях, например, сегментации изображений. Стоит сказать, что подобная конструкция применяется и в нейросети, разработанной нашим соотечественником, YOLOv4.

### 3.3.1 FPN + RPN (bounding-box proposal generation)

В данной модификации в RPN одноразмерная карта характеристик заменяется на FPN. Часть конструкции на (Рис.15) справа (свертка  $3 \times 3$  и 2 свертки  $1 \times 1$ ) прикрепляется к каждой уровневой пирамиде. Нет необходимости иметь несколько масштабов якоря на определенном уровне. Вместо этого назначаются привязки определенной шкалы к каждому уровню. Формально определяется, что якоря должны иметь области  $\{32^2, 64^2, 128^2, 256^2, 512^2\}$  пикселей. Якоря используются с несколькими пропорциями:  $\{1 : 2, 1 : 1, 2 : 1\}$  на каждом уровне. Метки обучения якорям назначаются на основе значений IoU с реальной рамкой объекта. Якорю присваивается положительная метка, если он имеет IoU более 0.7 или самое высокое IoU для данного блока истинности с любым истинным регионом и отрицательная – если он имеет IoU ниже 0.3 для всех реальных областей.

### 3.3.2 FPN + Fast R-CNN

Чтобы объединить Fast R-CNN с FPN, нужно назначить RoI разных масштабов для всех уровней пирамиды. Формально сопоставим RoI размера  $w \times h$  для входного изображения нейросети уровню  $P_k$  нашей пирамиды:

$$k = \lfloor k_0 + \log_2(\sqrt{wh}/224) \rfloor$$

224 – канонический размер предварительного обучения ImageNet, а  $k_0$  – целевой уровень, на который должен быть отображен RoI с  $w \times h = 224^2$ . Согласно формуле, если масштаб RoI уменьшится, то он должен отобразиться в уровень с более высоким разрешением. Все части RPN head прикрепляются ко всем RoI на всех уровнях. Все части разделяют параметры в независимости от своих уровней.

Архитектуру см. на (Рис. 16).

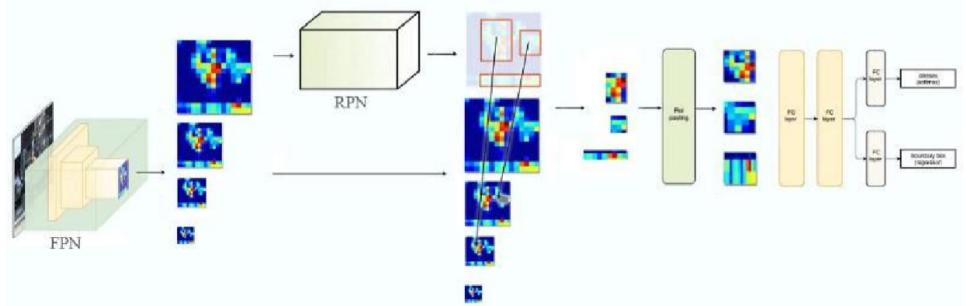


Рис. 16: Архитектура FPN [7]

### 3.4 Region-based Fully Convolutional Networks (R-FCN)

Интуиция данного метода состоит в следующем: допустим, у нас есть только карта характеристик для правого глаза. Эту информацию мы можем использовать, чтобы найти все лицо целиком. Это возможно сделать, так как мы знаем, что правый глаз должен быть на верхнем левом углу изображения лица. Однако с помощью карты характеристик такой ответ предсказать сложно.

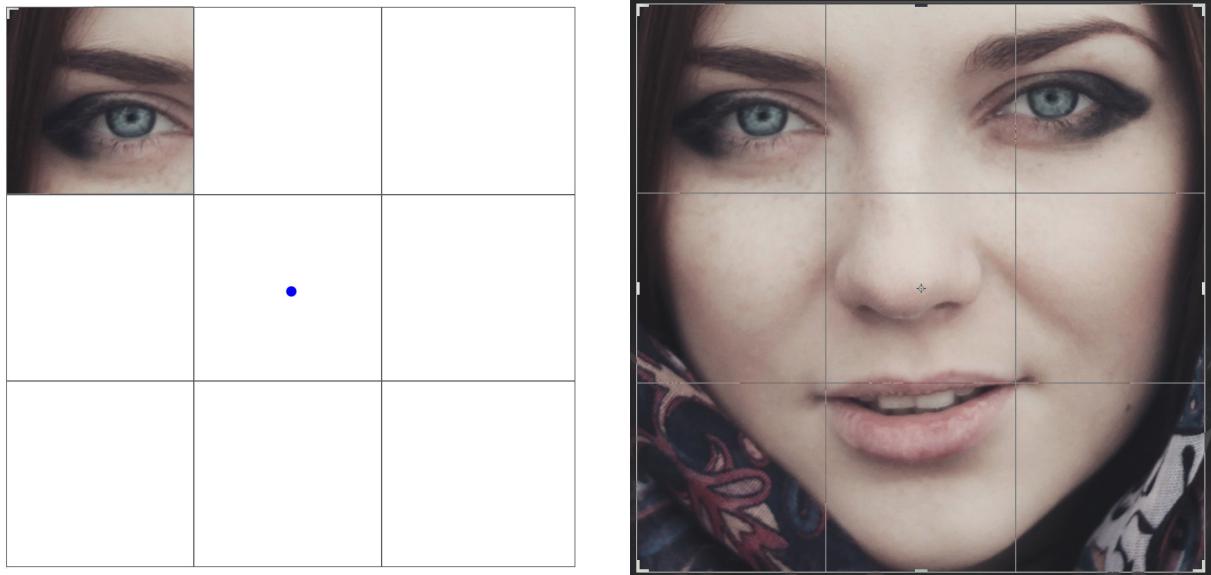


Рис. 17: Пример нахождения центра по фрагменту глаза [8]

Но если у нас есть другие карты характеристик, специализирующиеся на обнаружении левого глаза, носа или рта, мы можем объединить информацию вместе, чтобы сделать обнаружение всего лица более простым и точным. Для нахождения результаты, создается 9 карт объектов на основе регионов, каждая из которых определяет одну из девяти частей изображения. Объединив результаты каждого из фрагментов, можно определить расположение объектов.

Ниже на (Рис. 18) приведено схематическое изображение работы метода R-FCN. На изображении ниже  $k = 3$  и, соответственно,  $k \times k = 3 \times 3$ . Потом для каждой из  $k \times k$  ячеек объединение в ROI идет только по одной из карт характеристик (на картинке отмечены разными цветами). В конце используется softmax для превращения всех  $C + 1$  (добавляется еще класс фон) в вероятности принадлежности каждому из классов.

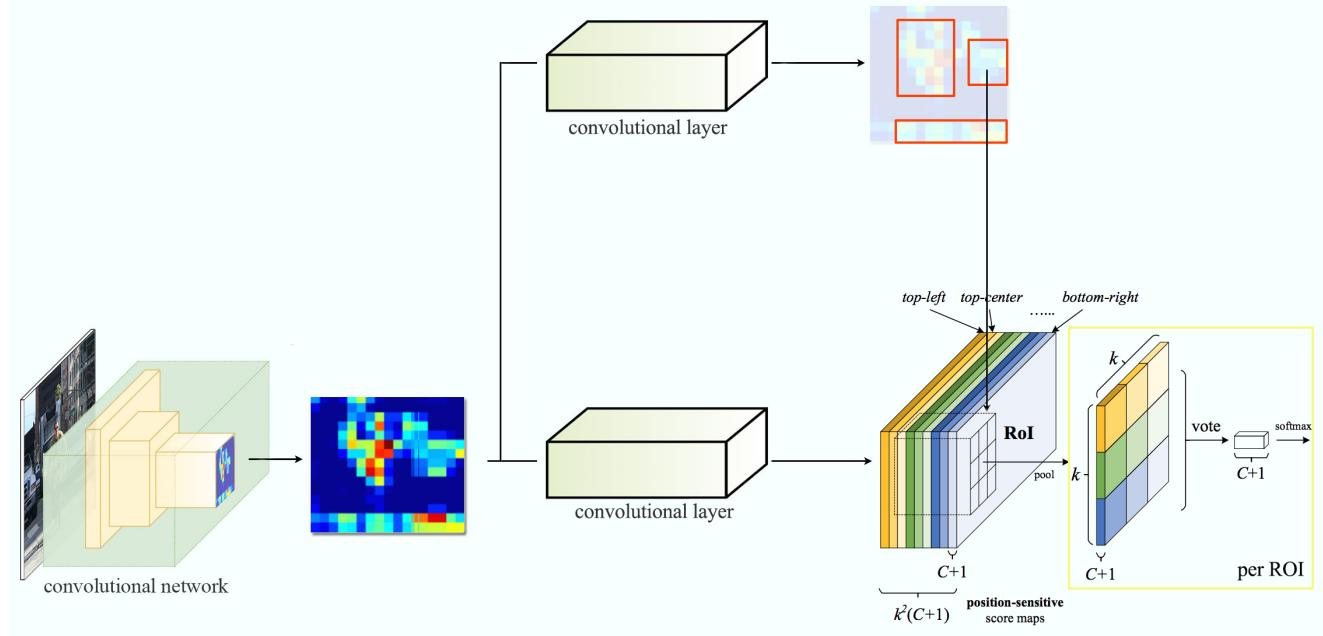


Рис. 18: Основная идея R-FCN [8]

Рассмотрим подробнее работу метода. Рассмотрим карту объектов  $M \times 5$  с квадратным объектом внутри. Разделим наш объект поровну на  $3 \times 3$  области. Теперь создается новая карта объектов из  $M$ , для обнаружения верхний левый (TL) угла квадрата (Рис. 19).

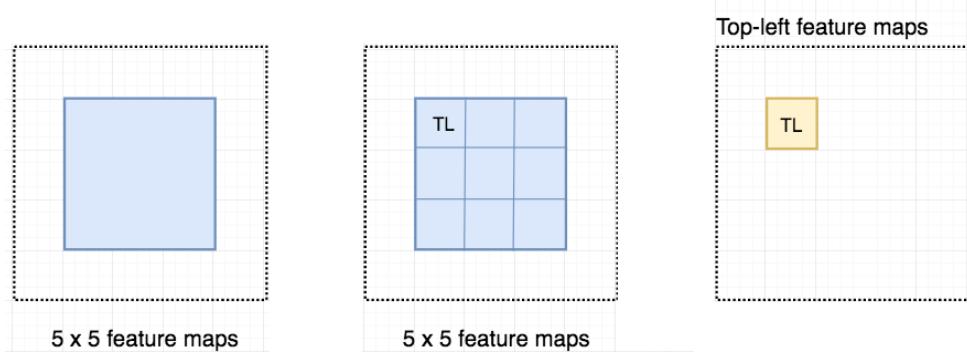


Рис. 19: R-FCN - принцип работы [8]

Так как мы делим квадрат на 9 частей (верхний левый TR, верхний-средний TM, верхний правый TR, центр-левый CF, ..., нижний-правый BR), мы создаем 9 карт объектов, каждая из которых обнаруживает соответствующую область объекта. Эти карты объектов называются картами оценок с учетом позиции, поскольку каждая карта обнаруживает субрегион объекта. (Рис. 20)

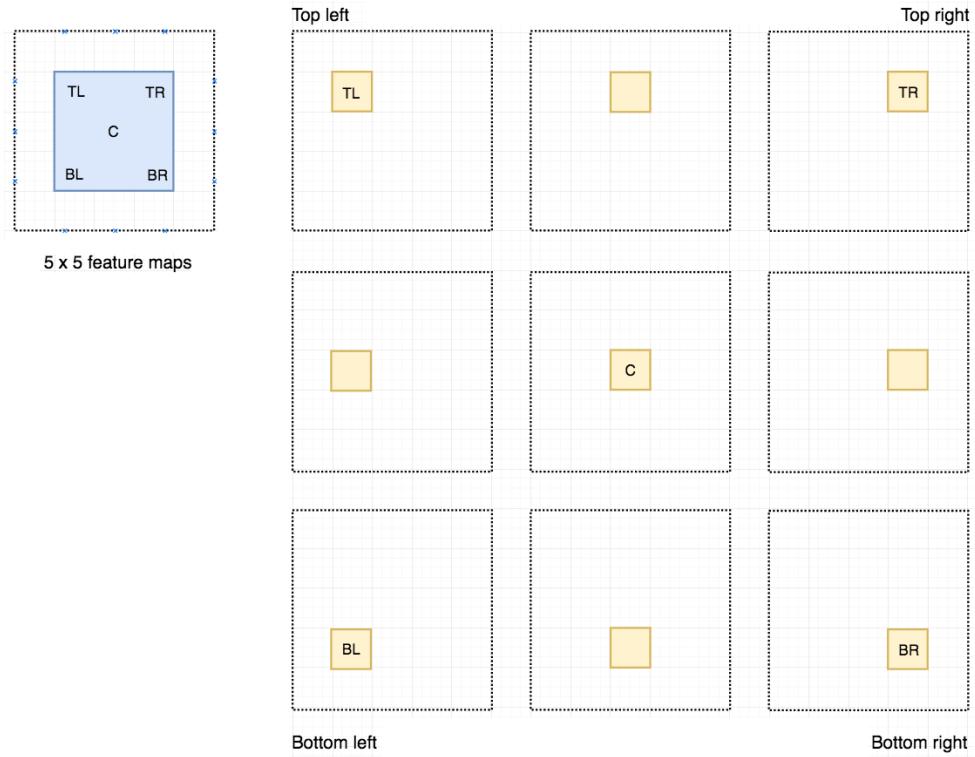


Рис. 20: R-FCN - принцип работы [8]

Допустим, на (Рис. 21) пунктирный красный прямоугольник ниже – предсказанная с помощью RoI область. Мы делим его на  $3 \times 3$  регионов и узнаем, насколько вероятно, что каждый регион содержит соответствующую часть объекта. Например, насколько вероятно, что область RoI содержит левый глаз. Результаты заносятся в массив.

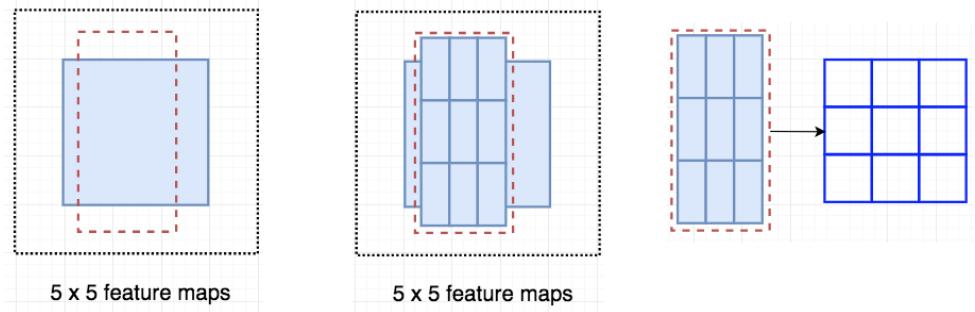


Рис. 21: R-FCN - принцип работы [8]

На диаграмме ниже (Рис. 22) выполняется следующее:

1. Мы берем верхний левый регион ROI и
2. Сопоставляем его с верхней левой ячейкой матрицы результатов (верхняя в центре диаграмма).

3. Вычисляется средний результат левой ячейкой матрицы результатов ограниченной верхней левой ROI. (синий прямоугольник). Около 40% площади внутри синего прямоугольника имеет рейтинг 0 и около 60% - имеет рейтинг 1. То есть процент пересечения – 0.6. Таким образом, вероятность того, что мы обнаружили верхний левый угол составляет 60%.
4. Заносим результат 0.6 в массив.
5. Переходим к центральному-верхнему региону и центральной верхней ячейки матрицы результатов.
6. Заносим результат 0.55 в соответствующее поле.

Фактически, в каждую клетку заносится результат пересечения предсказанной и реальной областей.

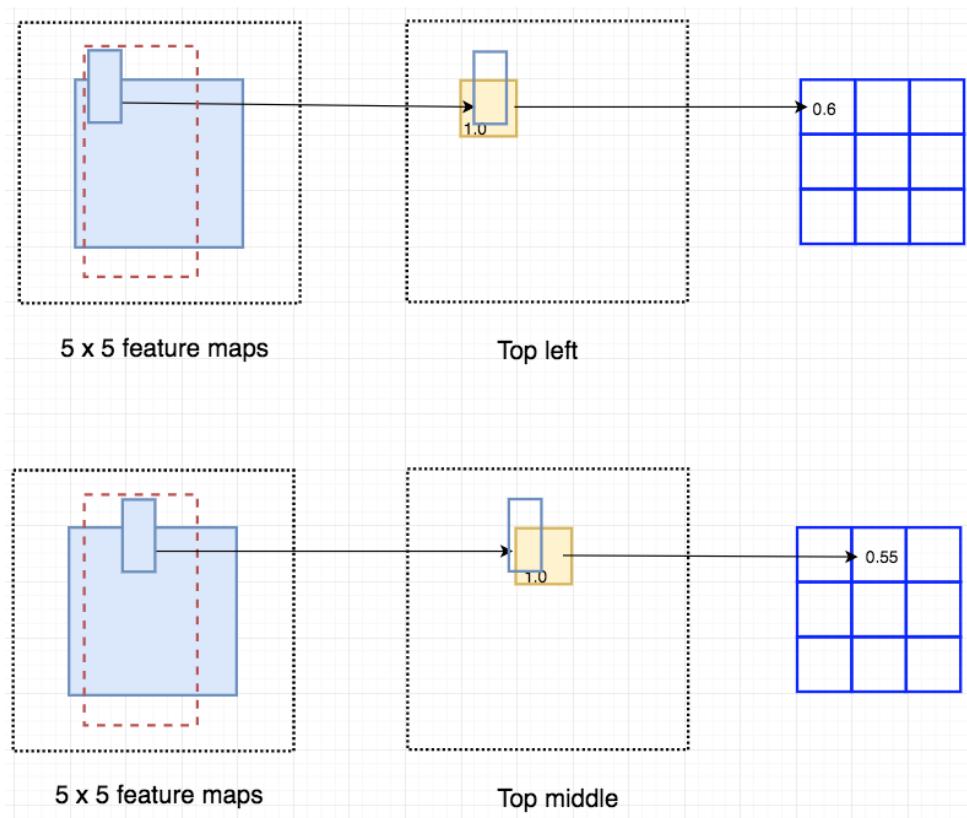


Рис. 22: R-FCN - принцип работы [8]

Полученные оценки суммируются и усредняются по правилу ROI (Рис. 23). Полученное число и будет вероятностью правильного предсказания.

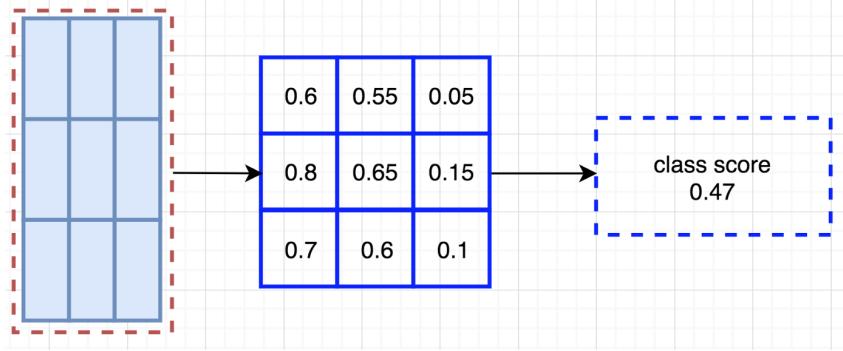
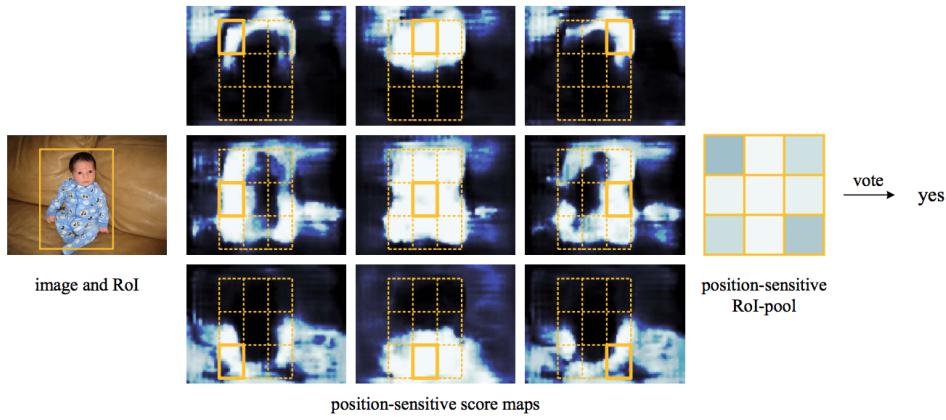


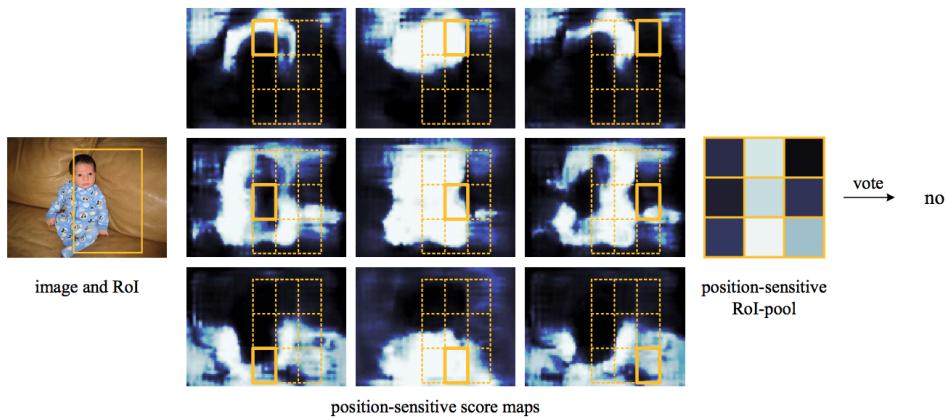
Рис. 23: R-FCN - принцип работы [8]

В данном методе, как и во многих рассмотренных ранее, применяется двухэтапная стратегия обнаружения объектов, которая состоит из локализации и классификации. Реализация R-FCN основана на ResNet-101 обученной на ImageNet. ResNet-101 имеет 100 сверточных уровней, за которыми следует Global Average Pooling и полно связанный слой для распределения объектов по 100 классам. Для вычисления карт характеристик используются только сверточные слои.

Пример работы алгоритма можно наблюдать на (Рис. 24).



Visualization of R-FCN ( $k \times k = 3 \times 3$ ) for the *person* category.



Visualization when an ROI does not correctly overlap the object.

Рис. 24: R-FCN - predict [8]

### 3.5 Fully Convolutional One-Stage Object Detection (FCOS)

На момент написания эссе FCOS является одной из лучших в плане качества сетей. Этот метод является proposal-free методом. В нем отсутствует часть генерации регионов. Отказаться от модуля генерации помогла оригинальная идея, реализованная в FCOS: будем считать, что каждая точка изображения порождает некоторую область. Если, как на правой части (Рис. 25), точка принадлежит сразу нескольким областям, то выбор делается в пользу меньшего из регионов. Данный метод использует пирамиду FPN, описанную ранее.

Как показано на левой части (Рис. 25), FCOS предсказывает четырехмерный вектор  $(l, t, r, b)$ , задающий местоположение ограничивающей рамки в каждом пикселе изображения. В отличие от других моделей обнаружения объектов, цель уточняющей регрессии была изменена на продвижение между краем ограничивающего прямоугольника и точкой  $(l, t, r, b)$  вместо оси  $(x, y, x, y)$ .

Если местоположение точки  $(x, y)$  связано с ограничивающим прямоугольником  $B_i$ , то целевые переменные для обучения регрессии выражаются как:

$$\begin{aligned} l^* &= x - x_0^{(i)}, t^* = y - y_0^{(i)} \\ r^* &= x_1^{(i)} - x, b^* = y_1^{(i)} - y \end{aligned}$$

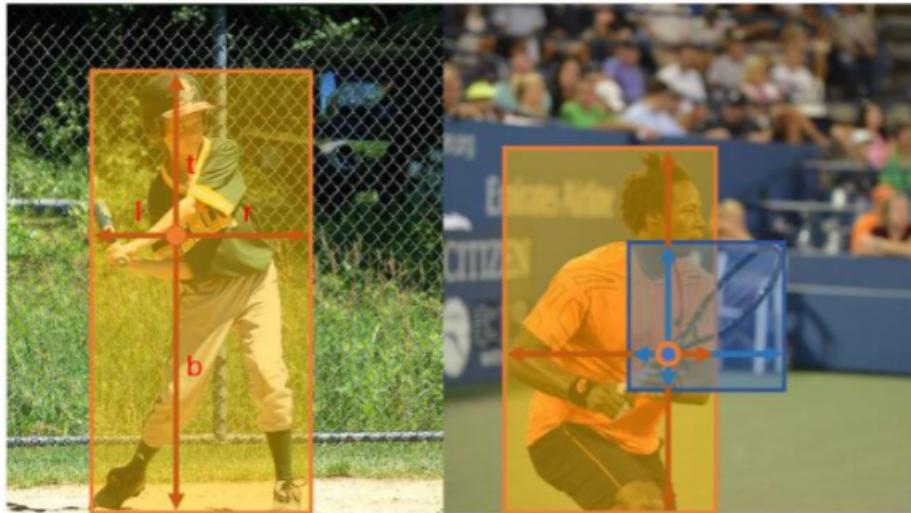


Рис. 25: FCOS [9]

Остановимся чуть подробнее на архитектуре данной сети. Ее архитектура показана на (Рис. 26).

Данная сеть может быть основана на любой известной сверточной сети. В данном случае использован ResNet-50. На схеме  $C3$ ,  $C4$  и  $C5$  обозначают карты характеристик, выдаваемых основной архитектурой, а  $P3 - P7$  – уровни, используемые для финального предсказания.  $H \times W$  – размера карты характеристик,  $s$  ( $s = 8, 16, \dots, 128$ ) –

уменьшающий коэффициент. Последний слой сети предсказывает 80-мерный вектор  $r$  классификационных меток и 4-мерный вектор  $t = (l, t, r, b)$  с координатами рамки. Вместо обучения многоклассового классификатора, происходит обучение с бинарных классификаторов, где  $c$  – количество классов. После карт характеристик основных сетей добавлены 4 сверточных слоя соответственно для ветвей классификации и регрессии.

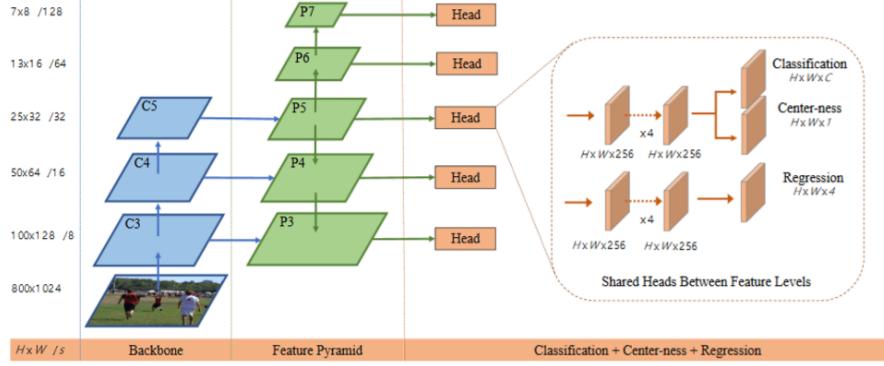


Рис. 26: Архитектура FCOS [9]

FCOS рассматривает каждую точку и, как говорилось выше, спорные случаи относит к меньшему классу. Это приводит к большому количеству предсказанных рамок низкого качества, созданных в местах далеких от центра объекта. Для решения данной проблемы, вводится индекс, описывающий расстояние от точки до центра истинной рамки и добавляется в виде ветви после карт характеристик:

$$\text{centerness}^* = \sqrt{\frac{\min(l^*, r^*)}{\max(l^*, r^*)} \times \frac{\min(t^*, b^*)}{\max(t^*, b^*)}}$$

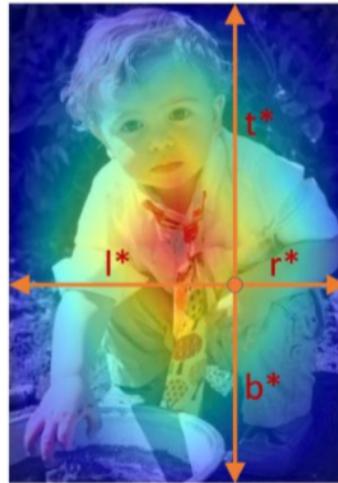


Рис. 27: FCOS [9]

Это не единственная сеть, работающая без предварительной генерации регионов. Далее мы рассмотрим еще несколько методов из данного класса.

### 3.6 CornerNet

CornerNet – один из первых представителей данного класса. Все началось с того, что решили начать предсказывать углы рамок.

На (Рис. 28) можно видеть архитектуру данной сети. Фактически существует два выхода из сети. Первый называется Top-left, а второй Bottom-right. Как следует из названия, мы предсказываем вероятность для точки быть верхним левым углом и правым нижним углом соответственно. Помимо этого, сеть натренирована выдавать еще и меру принадлежности разных углов к одной рамке. Как именно происходит сравнение углов показано ниже.

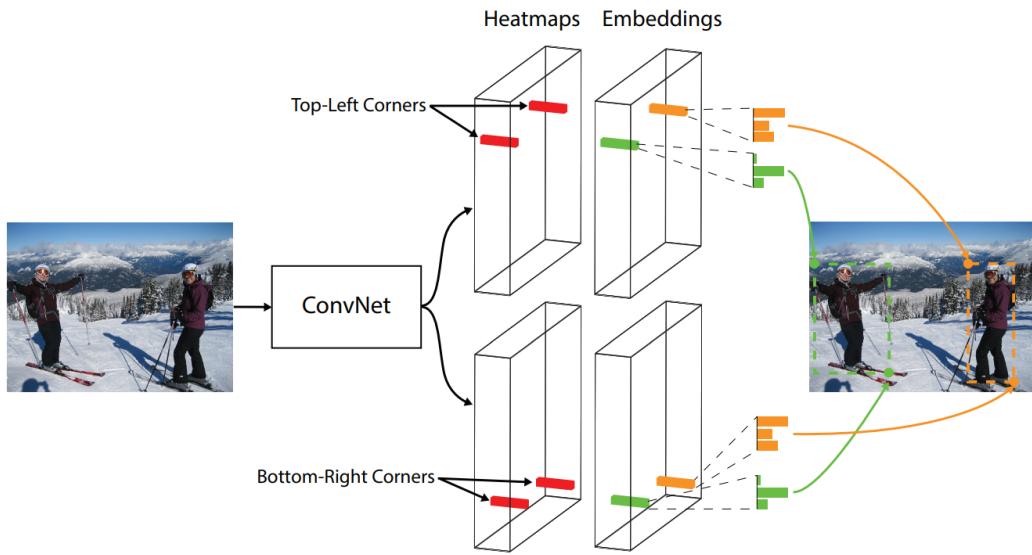


Рис. 28: Архитектура CornerNet [10]

Одно из возможных объяснений использования такого метода связано с тем, что метод обнаружения углов рамки может работать лучше, чем метод нахождения центра. Во-первых, центр сложнее детектировать, потому что его координаты зависят от всех четырех сторон, в то время как каждый из углов зависит только от двух параметров, что может упростить задачу. Во-вторых, способ поиска угловых точек может увеличить количество потенциальных рамок: нам нужно всего  $w \cdot h$  углов, чтобы представлять  $w^2 \cdot h^2$  потенциальных рамок.

В CornerNet задействован особый вид пуллинга. Это связано с тем, что угловые точки, которые алгоритм пытается предсказать, в большинстве случаев не содержат информацию об объекте, рамку которого мы пытаемся предсказывать. На (Рис. 29) показано, как именно эта операция проходит. В зависимости от того, какой угол предсказывается зависит в какую сторону будет проходить распространение.

Для каждой точки берется максимальный элемент в каждом из двух направлений (для top-left – это вниз и вправо, для bottom-right – это вверх и вправо) из сторон

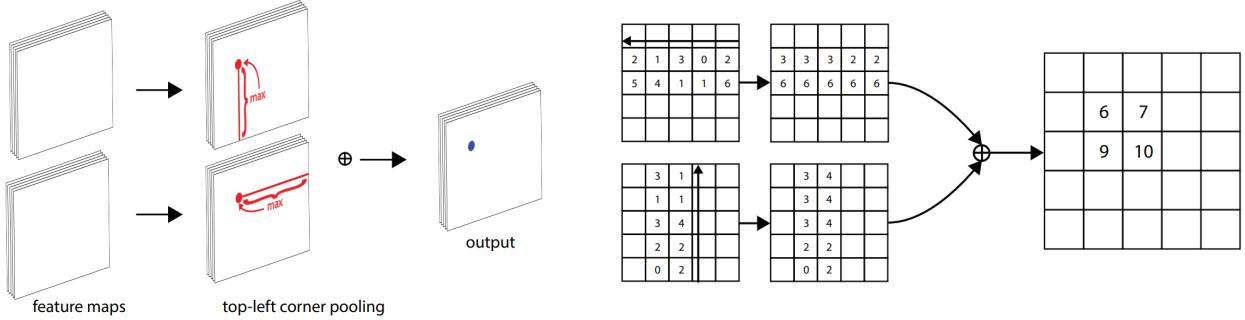


Рис. 29: Polling в CornerNet [10]

относительно самой точки. Затем эти числа складываются. Таким образом, крайняя точка получает информацию о том, что именно изображено на картинке.

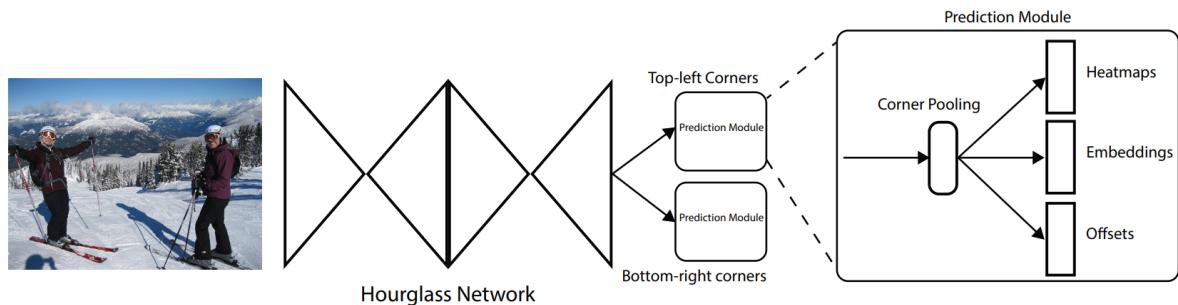


Рис. 30: Выходы CornerNet [10]

В каждом из случаев Top-left и второй Bottom-right сеть выводит 3 параметра (Рис. 30). Первый из них – Heatmaps, вероятность того, что точка является угловой. Помимо этого параметра выводится еще Embeddings – расстояния между представлениями одной рамки минимального размера, и Offsets, который нужен для редактирования координат углов. Координаты могут выдаваться для рамок довольно крупного масштаба. При изменении масштаба при проходе сверточных слоев некоторая информация может потеряться.

Выход Embeddings нужен для того, чтобы соединять только правильные углы рамок. Предсказания Top-left и Bottom-right независят друг от друга. Чтобы не генерировать лишние рамки, если точки принадлежат одной рамке, то расстояние между их ембеддингами должно быть небольшим.

Однако такой способ не всегда является удачным. Следующие рассмотренные сети появились для того, чтобы отойти от сравнения embeddings разных углов.

Примером такой сети может служить ExtremeNet, речь о которой пойдет дальше.

### 3.7 ExtremeNet

ExtremeNet начал использовать несколько другую парадигму, в отличие от CornerNet. В данной сети предсказывается 4 стороны (4 крайних точки) и центр рамки. Если четырем углам определенного класса соответствует центр того же класса, то из них формируется рамка-ответ. За счет согласования вероятностей центров и границ объекта, делаются более увереные предсказания. Таким способом мы можем избавиться от сравнения embeddings. Пример работы можно наблюдать на (Рис. 31).

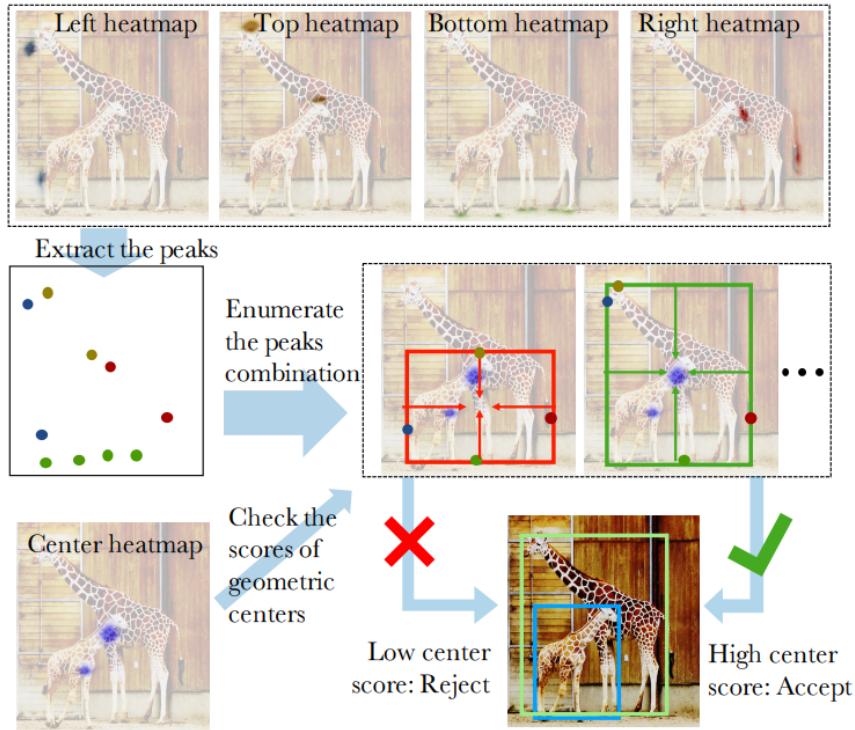


Рис. 31: ExtremeNet. Согласование центров [11]

Для границ, выделенных красным цветом, в их геометрическом центре вероятность обнаружения центра объекта была низкой, в отличие от рамки, обведенной зеленым цветом. Поэтому сеть отклонила красную рамку и правильно детектировала зеленую.

Преимуществом относительно CornerNet можно считать еще тот факт, что в ExtremeNet предсказываются крайние точки самого объекта, тогда как в CornerNet могут возникнуть проблемы с обнаружением угловых точек – они лежат вне объекта и в CornerNet делались специальные преобразования polling чтобы решить данную проблему.

ExtremeNet можно отнести к one-stage детекторам за исключение того, что вместо поиска рамок за  $O(h^2 \cdot w^2)$  здесь согласуются 5 независимых частей, что снижает сложность  $O(h \cdot w)$ .

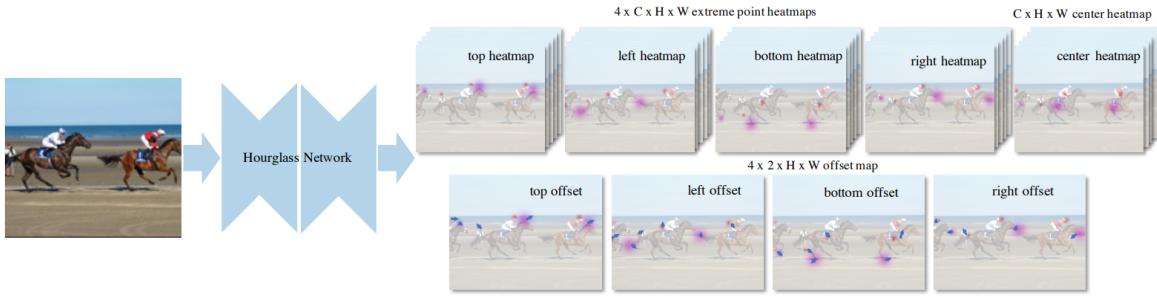


Рис. 32: ExtremeNet. Иллюстрация работы алгоритма [11]

Пример внутренней структуры работы алгоритма можно видеть на (Рис. 32). Сеть принимает изображение в качестве входных данных и создает четыре тепловые карты размера  $C \times H \times W$ , одну тепловую карту центров  $C \times H \times W$ , а также четыре двухканальных карты смещения, не зависящие от категории. Тепловые карты обучаются с помощью взвешенной пиксельной логистической регрессии, где вес используется для уменьшения штрафа за ложные срабатывания около точек, близких к истинным значениям границ.

|                            | Backbone         | Input resolution  | $AP$ | $AP_{50}$ | $AP_{75}$ | $AP_S$ | $AP_M$ | $AP_L$ |
|----------------------------|------------------|-------------------|------|-----------|-----------|--------|--------|--------|
| <b>Two-stage detectors</b> |                  |                   |      |           |           |        |        |        |
| Faster R-CNN w/ FPN [24]   | ResNet-101       | $1000 \times 600$ | 36.2 | 59.1      | 39.0      | 18.2   | 39.0   | 48.2   |
| Deformable-CNN [7]         | Inception-ResNet | $1000 \times 600$ | 37.5 | 58.0      | -         | 19.4   | 40.1   | 52.5   |
| Deep Regionlets [51]       | ResNet-101       | $1000 \times 600$ | 39.3 | 59.8      | -         | 21.7   | 43.7   | 50.9   |
| Mask R-CNN [15]            | ResNeXt-101      | $1333 \times 800$ | 39.8 | 62.3      | 43.4      | 22.1   | 43.2   | 51.2   |
| LH R-CNN [23]              | ResNet-101       | $1000 \times 600$ | 41.5 | -         | -         | 25.2   | 45.3   | 53.1   |
| Cascade R-CNN [2]          | ResNet-101       | $1333 \times 800$ | 42.8 | 62.1      | 46.3      | 23.7   | 45.5   | 55.2   |
| D-RFCN + SNIP [43]         | DPN-98           | $1333 \times 800$ | 45.7 | 67.3      | 51.1      | 29.3   | 48.8   | 57.1   |
| PANet [27]                 | ResNeXt-101      | $1000 \times 600$ | 47.4 | 67.2      | 51.8      | 30.1   | 51.7   | 60.0   |
| <b>One-stage detectors</b> |                  |                   |      |           |           |        |        |        |
| YOLOv2 [39]                | DarkNet-19       | $544 \times 544$  | 21.6 | 44.0      | 19.2      | 5.0    | 22.4   | 35.5   |
| YOLOv3 [40]                | DarkNet-53       | $608 \times 608$  | 33.0 | 57.9      | 34.4      | 18.3   | 35.4   | 41.9   |
| SSD [28]                   | ResNet-101       | $513 \times 513$  | 31.2 | 50.4      | 33.3      | 10.2   | 34.5   | 49.8   |
| DSSD [10]                  | ResNet-101       | $513 \times 513$  | 33.2 | 53.3      | 35.2      | 13.0   | 35.4   | 51.1   |
| RetinaNet [25]             | ResNet-101       | $1333 \times 800$ | 39.1 | 59.1      | 42.3      | 21.8   | 42.7   | 50.2   |
| RefineDet (SS) [52]        | ResNet-101       | $512 \times 512$  | 36.4 | 57.5      | 39.5      | 16.6   | 39.9   | 51.4   |
| RefineDet (MS) [52]        | ResNet-101       | $512 \times 512$  | 41.8 | 62.9      | 45.7      | 25.6   | 45.1   | 54.1   |
| CornerNet (SS) [22]        | Hourglass-104    | $511 \times 511$  | 40.5 | 56.5      | 43.1      | 19.4   | 42.7   | 53.9   |
| CornerNet (MS) [22]        | Hourglass-104    | $511 \times 511$  | 42.1 | 57.8      | 45.3      | 20.8   | 44.8   | 56.7   |
| ExtremeNet (SS)            | Hourglass-104    | $511 \times 511$  | 40.2 | 55.5      | 43.2      | 20.4   | 43.2   | 53.1   |
| ExtremeNet (MS)            | Hourglass-104    | $511 \times 511$  | 43.7 | 60.5      | 47.0      | 24.1   | 46.9   | 57.6   |

Рис. 33: Сравнение COCO test-dev. [11]

SS / MS - это сокращение от одноуровневого / многомасштабного тестирования, соответственно.

На (Рис. 33) показана некоторая статистика. Из таблицы видно, что ExtremeNet несколько уступает CornerNet при детектировании больших объектов, однако точность на небольших объектах у ExtremeNet выше.

### 3.8 CenterNet

CenterNet является достаточно популярной сетью. Это связано не только с тем, что она довольно молодая, но и с тем, что с ее помощью можно решать не только задачи детектирования объектов на изображении. С ее помощью также можно решать задачу 3D object detection ((Рис 34) вторая строка) или оценивание позы человека ((Рис 34) третья строка).

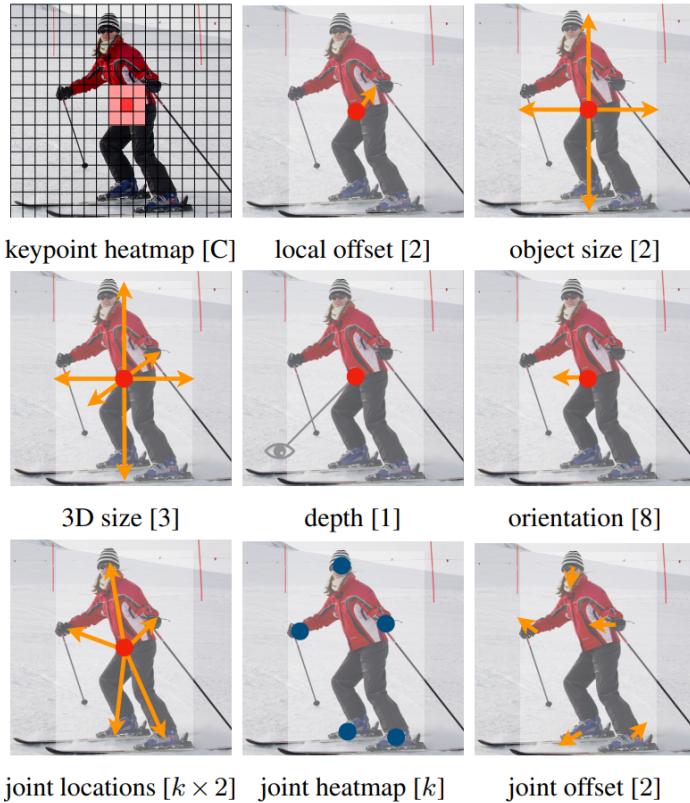


Рис. 34: Возможности CenterNet [12]

CenterNet представляет каждый объект как отдельную точку, ограниченную его рамкой (см. (Рис. 35)) Остальные признаки, такие как поза, размер, ориентация и другие, вытекают из точечного представления объекта. В таком случае именно поиск центральной точки является ключевой задачей.

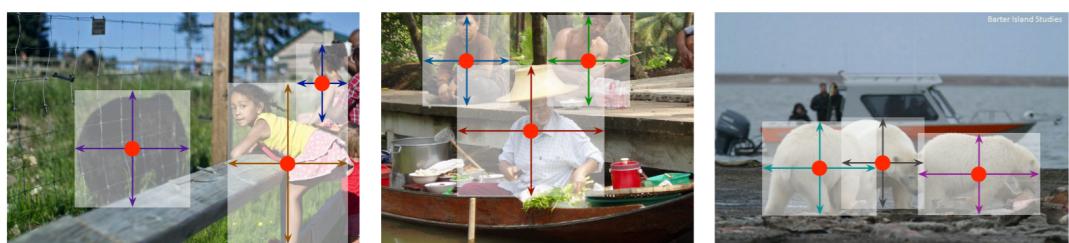


Рис. 35: Моделирование объекта, как центральной точки и рамки [12]

Для решения этой задачи изображение подается на вход полностью сверточной нейронной сети, которая генерирует тепловую карту. Пики на этой тепловой карте соответствуют центрам объектов. Характеристики изображения на каждом пике прогнозируются исходя из веса точки на тепловой карте и ее положением.

В общем случае, сеть на вход получает изображение размера  $W \times H \times 3$  ( $W, H$  – размеры входного изображения), а на выход подается тензор размера  $\frac{W}{k} \times \frac{H}{k} \times c$ . Здесь  $c$  – количество параметров, которое надо определять, а  $k$  – поникающий коэффициент. Помимо прочего, CenterNet показывает хорошие результаты качества относительно известных сетей, что показано на (Рис. 36).

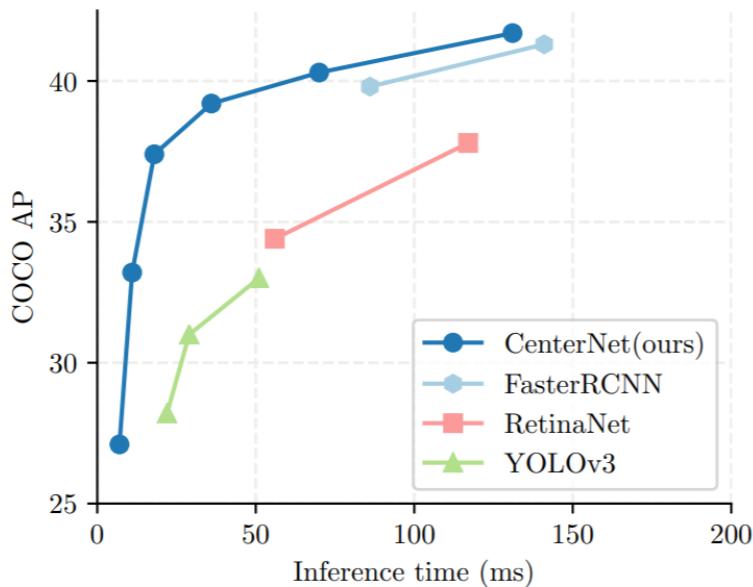


Рис. 36: Сравнение скорости работы и точности на COCO validation [12]

Посмотрим подробнее, как происходит представление объекта как точки. Пусть  $(x_1^{(k)}, y_1^{(k)}, x_2^{(k)}, y_2^{(k)})$  - ограничивающая рамка объекта  $k$  с категорией  $c_k$ . Его центральная точка считается как  $p_k = (\frac{x_1^{(k)}+x_2^{(k)}}{2}, \frac{y_1^{(k)}+y_2^{(k)}}{2})$ . Используется ключевая оценка  $\hat{Y}$  для предсказания всех центральных точек на изображении. Кроме того мы переходим к размеру объектов  $s_k = (x_2^{(k)} - x_1^{(k)}, y_2^{(k)} - y_1^{(k)})$  для каждого  $k$ . Чтобы ограничить вычислительную нагрузку, снижается размерность, как было показано выше ( $\hat{S}$ ). И при этом читаются потери. Используется L1 loss.  $L_{size} = \frac{1}{N} \sum_{i=1}^N |\hat{S}_{p_k} - s_k|$ . Масштаб не нормализуется и напрямую используем необработанные координаты пикселя. Вместо этого масштабируются потери на постоянную  $\lambda_{size}$  (по умолчанию 0.1). В итоге получается  $L_{det} = L_k + \lambda_{size}L_{size} + \lambda_{off}L_{off}$ . Используется единая сеть для прогнозирования ключевых точек  $\hat{Y}$ , смещений  $\hat{O}$  и размера  $\hat{S}$ . Сеть прогнозирует  $C+4$  выходных параметра. Все параметры передаются в полностью сверточную сеть, затем через свертку  $3 \times 3$ , ReLU и другую свертку  $1 \times 1$ .

### 3.9 EfficientDet

EfficientDet состоит из EfficientNet в качестве основы, к которой приделан слой по работе с пирамидой признаков под названием BiFPN (Рис. 37), за которым идет «стандартная» сеть вычисления класс/рамка объекта.

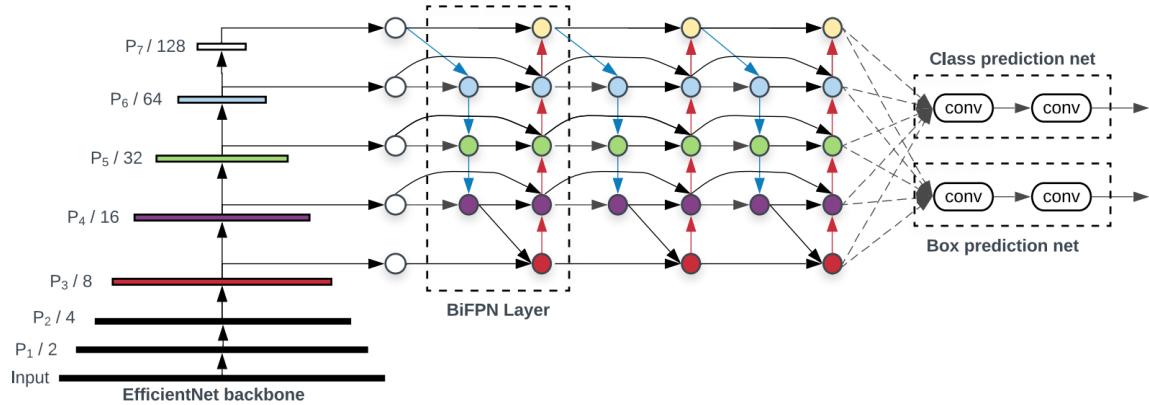


Рис. 37: Top Down проход EfficientDet [14]

Из (Рис.38) видно, что у EfficientDet качество лучше, чем у аналогичных сетей. Постараемся понять, за счет чего удалось добиться такого результата.

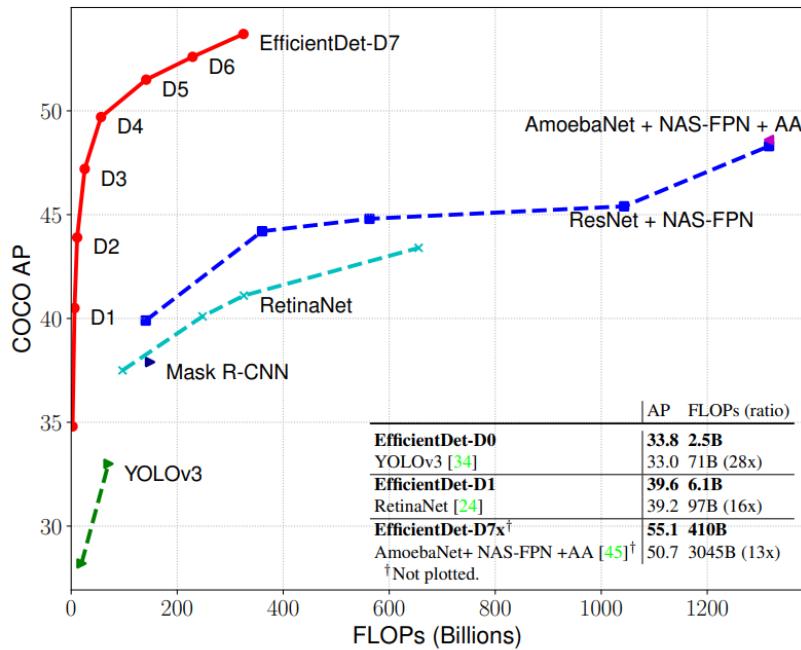


Рис. 38: Сравнение качества [14]

Остановимся подробнее на блоке BiFPN. На (Рис. 39а) показано распространение информации сверху вниз, которая поступает после прохождения первичных слоев. В сети PaNet добавили еще одну ветку, по которой информация распространялась еще и

вверх. Это дополнение в любом случае не повлияет на конечный результат негативно, но, возможно, уменьшится скорость работы и увеличится время обучения.

Результат, полученный на (Рис. 39c) получен перебором разлчных архитектур. Можно сказать, что такое представление найдено экспериментально. Интуитивно сложно понять, за счет чего достигается такое высокое качество. На (Рис. 38) Можно видеть, что NAS-FPN показывает высокие результаты. BiFPN была получена по аналогии с NAS-FPN, но с небольшими модификациями.

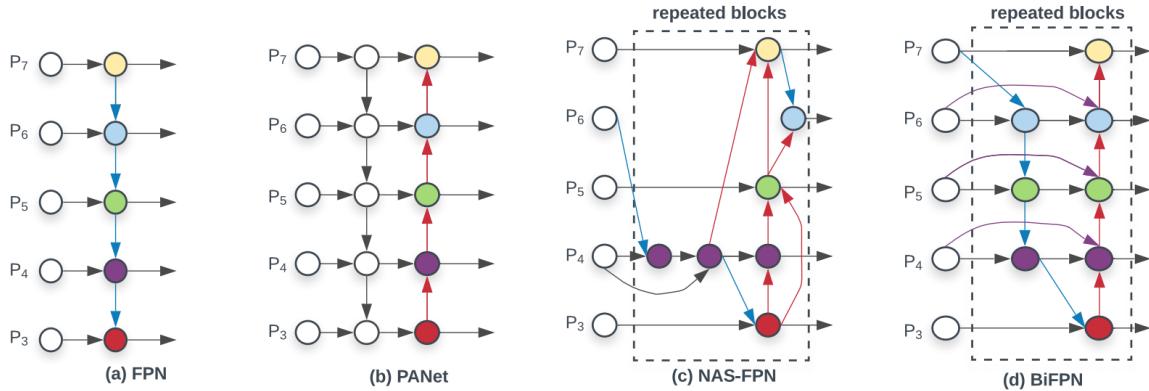


Рис. 39: Top Down проход EfficientDet [14]

Запишем отдельно, какие параметры возникают в промежуточных шагах и на выходе из блока:

$$P_6^{td} = \text{Conv}\left(\frac{w_1 \cdot P_6^{in} + w_2 \cdot \text{Resize}(P_7^{in})}{w_1 + w_2 + \epsilon}\right),$$

$$P_6^{out} = \text{Conv}\left(\frac{w'_1 \cdot P_6^{td} + w'_2 \cdot \text{Resize}(P_7^{in}) + w'_3 \cdot \text{Resize}(P_5^{out})}{w'_1 + w'_2 + w'_3 + \epsilon}\right)$$

Важно, что появляются параметры  $w_i$ , которые влияют, с каким весом брать информацию, полученную с разных сторон, вместо обычного суммирования тензоров, что тоже сказалось на качестве.

Из формул видно, что сеть выучивает важность информации с каждого разрешения изображения.

В качестве альтернативы EfficientNet пробовали брать и Res-Net, но результаты оказались хуже.

Здесь, для улучшения качества используется compound scaling для совместного масштабирования параметров сети. Однако пробовались и другие альтернативные методы, но compound scaling показал лучшее качество, поэтому в конечной версии алгоритма остался именно этот метод. Сравнить показания можно на (Рис. 40).

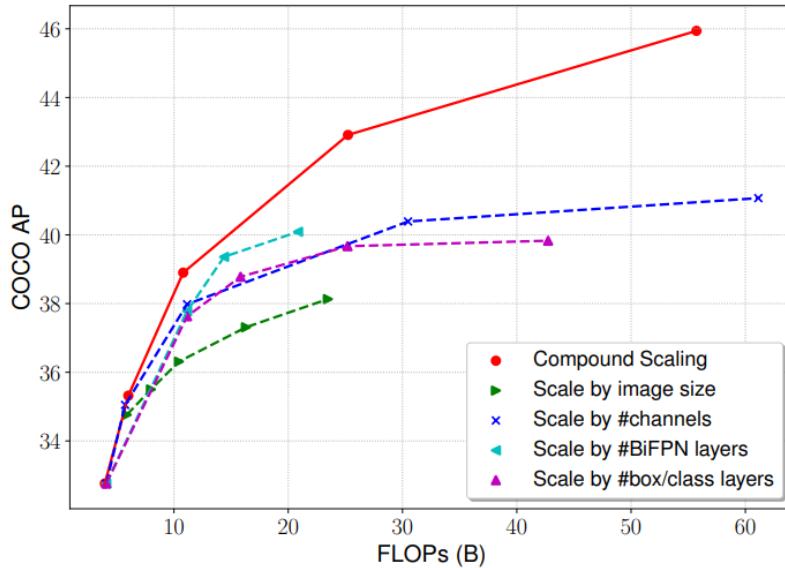


Рис. 40: Сравнение методов масштабирования [14]

В результате систематического изучения различных вариантов проектирования архитектуры сети для эффективного обнаружения объектов предложена взвешенная двунаправленная функциональная сеть и настраиваемый метод составного масштабирования для повышения точности и эффективности. На основе этих оптимизаций было разработано новое семейство детекторов под названием EfficientDet, которые последовательно достигают большей точности и эффективности, чем предшествующий уровень техники, при широком спектре ограничений по ресурсам. В частности, наш EfficientDet-D7 достигает современной точности с меньшим количеством параметров, чем лучший из существующих детекторов. EfficientDet также в 3,2 раза быстрее на GPU и в 8,1 раза быстрее на CPU.

### 3.10 Приёмы детектирования объектов: аугментация Copy-Paste.

Посмотрим на недавние работы, которые написаны менее полугода назад. За основу этого алгоритма была взята NAS-FPN, рассмотренная ранее. Напомним, что ее архитектура несильно интуитивно понятна, потому что фактически сеть была получена перебором. Однако дополнительно применили специальную процедуру аугментации. На (Рис. 41) показан принцип этой аугментации.



Рис. 41: Создание новых изображений для увеличения выборки [15]

Несмотря на то, что есть широкоизвестные и универсальные методы аугментации изображений, было предложено создать новый алгоритм Copy-Paste. К тому же, за счет универсальности стандартные методы не направлены конкретно на задачу сегментации. Copy-Paste в свою очередь хорошо подходит для этой задачи, с его помощью можно генерировать новые сложные данные.

Основная идея алгоритма состоит в том, чтобы добавлять объекты с одного изображения в другое. Причем за счет большого количества источников существует большое количество возможностей для генерации новых изображений. Возможно выбирать пары изображений, с которого и в которое будет вставляться экземпляр, выбор самого объекта на изображении, и выбор места, в которое его вставлять.

За счет такого большого количества вариаций, можно исследовать наиболее удачную комбинацию этих параметров. В отличие от других методов аугментации, здесь не моделируется окружающий контекст, и в результате сгенерированные изображения могут сильно отличаться от реальных изображений с точки зрения совместного появления объектов или связанных масштабов объектов. Так, например, жирафы и футболисты с очень разными размерами могут появляться рядом с друг другом (Рис. 41).

Как показывает практика, такой метод в сочетании с jittering приносит увеличение качества работы модели, что видно из (Рис. 42).

Для компоновки новых объектов в изображение, вычисляется бинарную маску  $\alpha$  вставленных объектов используя достоверные аннотации, и новое изображение вы-

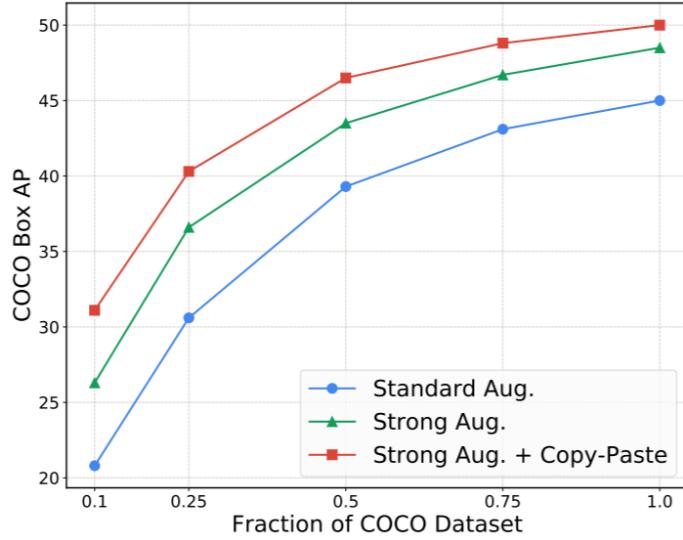


Рис. 42: Сравнения качества при различных аугментациях [15]

числяется как  $I1\alpha + I2(1 - \alpha)$ , где I1 - вставленное изображение, а I2 - это основное изображение. Чтобы сгладить края вставленного объекта, после применяется фильтр Гаусса к  $\alpha$ , аналогичный «смешиванию».

Посмотрим на увеличение качества, которое демонстрируется на NAS-FPN.

| Model                                     | FLOPs | # Params | AP <sub>val</sub> | AP <sub>test-dev</sub> | Mask AP <sub>val</sub> | Mask AP <sub>test-dev</sub> |
|---|-------|----------|-------------------|------------------------|------------------------|-----------------------------|
| SpineNet-190 (1536) [11]                  | 2076B | 176.2M   | 52.2              | 52.5                   | 46.1                   | 46.3                        |
| DetectoRS ResNeXt-101-64x4d [43]          | —     | —        | —                 | 55.7 <sup>†</sup>      | —                      | 48.5 <sup>†</sup>           |
| SpineNet-190 (1280) [11]                  | 1885B | 164M     | 52.6              | 52.8                   | —                      | —                           |
| SpineNet-190 (1280) w/ self-training [71] | 1885B | 164M     | 54.2              | 54.3                   | —                      | —                           |
| EfficientDet-D7x (1536) [56]              | 410B  | 77M      | 54.4              | 55.1                   | —                      | —                           |
| YOLOv4-P7 (1536) [60]                     | —     | —        | —                 | 55.8 <sup>†</sup>      | —                      | —                           |
| Cascade Eff-B7 NAS-FPN (1280)             | 1440B | 185M     | 54.5              | 54.8                   | 46.8                   | 46.9                        |
| w/ Copy-Paste                             | 1440B | 185M     | (+1.4) 55.9       | (+1.2) 56.0            | (+0.4) 47.2            | (+0.5) 47.4                 |
| w/ self-training Copy-Paste               | 1440B | 185M     | (+2.5) 57.0       | (+2.5) 57.3            | (+2.1) 48.9            | (+2.2) 49.1                 |

Рис. 43: Прирост качества при добавлении аугментации[15]

В завершение стоит сказать, что после экспериментов [15] подтверждается, что Copy-Paste эффективен и надежен. Он хорошо работает в нескольких экспериментальных настройках и обеспечивает значительные улучшения помимо сильных базовых показателей для COCO. Стратегия дополнения Copy-Paste проста, легко интегрируется к любой базе кода сегментации экземпляров и не увеличивает время обучения или время вывода. Также показано, что Copy-Paste полезен для включения дополнительных немаркированных изображений во время обучения и дополняет успешные методы самообучения.

## 4 Выводы

Стоит еще раз заострить внимание на ключевой информации по поводу детектирования изображений:

- Локализация объектов не обязательно должна быть классориентированной, возможно определять, что именно определено на изображении уже на этапе классификации.
- В одном регионе возможно детектировать несколько объектов.
- Существуют методы, генерирующие области (якоря) потенциально содержащие объекты, но есть способы обойтись и без генерации регионов.
- Пирамидные сети (FPN) являются способом формирования признакового пространства.
- Идея того, что каждая точка генерирует потенциальную рамку очень хороша.
- Интерес к детектированию объектов с течением времени только возрастает. Появляются все новые методы, улучшающие качество.

## 5 Список литературы

### Список литературы

- [1] Zhengxia Zou: «Object detection in 20 years: A survey»  
[https://www.researchgate.net/figure/The-increasing-number-of-publications-in-object-detection-from-1998-to-2018-Data-from\\_fig1\\_333077580](https://www.researchgate.net/figure/The-increasing-number-of-publications-in-object-detection-from-1998-to-2018-Data-from_fig1_333077580)
- [2] Selective Search for Object Recognition 2012:  
<http://www.huppenen.nl/publications/selectiveSearchDraft.pdf>
- [3] Non-maximum Suppression (NMS). A technique to filter the predictions of object detectors 2019: <https://towardsdatascience.com/non-maximum-suppression-nms-93ce178e177c>
- [4] Dssd: Deconvolutional single shot detector, Fu C. – 2017: <https://arxiv.org/abs/1701.06659>.
- [5] Feature Pyramid Networks for Object Detection: <https://arxiv.org/abs/1612.03144>
- [6] Lin T. Y. et al. Feature pyramid networks for object detection //Proceedings of the IEEE conference on computer vision and pattern recognition. – 2017. – С. 2117-2125
- [7] Александр Дьяконов: лекции весеннего семестра ВМК МГУ 2021: Детектирование объектов на изображениях
- [8] Understanding Region-based Fully Convolutional Networks (R-FCN) for object detection 2018: <https://jonathan-hui.medium.com/understanding-region-based-fully-convolutional-networks-r-fcn-for-object-detection-828316f07c99>
- [9] Tian Z. et al. Fcos: Fully convolutional one-stage object detection //Proceedings of the IEEE International Conference on Computer Vision. – 2019. – С. 9627-9636.
- [10] CornerNet: Detecting Objects as Paired Keypoints, Hei Law, Jia Deng – March 2019: <https://arxiv.org/pdf/1808.01244.pdf>
- [11] Bottom-up Object Detection by Grouping Extreme and Center Points, Xingyi Zhou, Jiacheng Zhuo – April 2019: <https://arxiv.org/pdf/1901.08043.pdf>
- [12] Objects as Points Xingyi Zhou, Dequan Wang, Philipp Krahenbuhl. – April 2019: <https://arxiv.org/pdf/1904.07850.pdf>
- [13] CenterNet: Keypoint Triplets for Object Detection Kaiwen Duan, Song Bai, Lingxi Xie, Honggang Qi – April 2019: <https://arxiv.org/pdf/1904.08189v3.pdf>

- [14] EfficientDet: Scalable and Efficient Object Detection, Mingxing Tan, Ruoming Pang – June 2020: <https://arxiv.org/pdf/1911.09070.pdf>
- [15] Simple Copy-Paste is a Strong Data Augmentation Method for Instance Segmentation. Golnaz Ghiasi, Yin Cui, Aravind Srinivas, Rui Qian – Dec 2020. <https://arxiv.org/pdf/2012.07177v1.pdf>