

МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
им. М.В. ЛОМОНОСОВА

факультет вычислительной математики и кибернетики
кафедра математических методов прогнозирования

ЭССЕ

«Детектирование объектов на изображениях
(часть 1)»

КАЧУРА АЛЕКСАНДР
Глубокое обучение
3 курс, группа 317

Москва, 2021 г.

Содержание

1 Аннотация	3
2 Введение	3
3 Описание сопутствующих задач	4
3.1 Классификация	4
3.1.1 Проблемы задачи классификации	4
3.2 Локализация	5
4 Постановка задачи детектирования объектов	5
5 Метрики качества	5
5.1 Intersection over union + Precision / Recall	6
5.2 Average Precision	9
5.3 Mean Average Precision	10
5.4 Panoptic Quality	10
6 Основная проблема	11
6.1 Selective Search	11
7 Основные подходы к детектированию объектов	13
7.1 Regions With CNNs (R-CNN)	13
7.1.1 Недостатки подхода	15
7.2 Non Maximum Suppression (NMS)	15
7.3 Spatial Pyramid Pooling (SPP-net)	16
7.4 Fast R-CNN	17
7.4.1 RoI Pooling Layer	19
7.5 Faster R-CNN	20
7.5.1 Region Proposal Network (RPN)	20
7.6 Сравнение скорости работы описанных R-CNN сетей	22
7.7 You Only Look Once (YOLO)	22
7.7.1 Недостатки YOLO	25
7.8 YOLO9000: иерархия в классах	26
7.9 YOLOv3	28
7.10 Single Shot MultiBox Detector (SSD)	29
7.10.1 MultiBox	30
7.10.2 Описание SSD	31
7.10.3 Замечания по работе SSD	34
7.11 Сравнение подходов к детектированию объектов	34
8 Альтернативные подходы к учёту масштаба изображений	35
8.1 MatrixNets	35
9 Подходы к улучшению качества детектирования объектов	35
9.1 Многошаговая сеть с калибровочной моделью	35
9.2 Region Decomposition and Assembly Detector (R-DAD)	37

10 Альтернативные подходы к выбору якорных прямоугольников	38
10.1 Anchor pruning	38
10.2 FreeAnchor	39
11 Заключение	41
Список используемой литературы	42

1 Аннотация

В данной работе рассматривается задача детектирования объектов на изображении. В машинном обучении указанная проблема представляет собой совокупность двух подзадач (локализации и классификации), поэтому отдельное внимание уделяется каждой из них. Описываются метрики, используемые для измерения качества работы методов, основные нейросетевые подходы к решению задачи детектирования, приводятся данные по сравнению точности их работы. Кроме того, рассматриваются некоторые альтернативные архитектуры сетей, направленные на улучшение основных подходов к детекции объектов, а также альтернативные подходы для выбора якорных прямоугольников.

2 Введение

Работа с изображениями в настоящее время является одной из центральных задач, связанных с компьютерным зрением. Существует множество различных вариаций и постановок данной задачи, которые исследуются и успешно решаются во многом с использованием нейронных сетей. К примеру, сейчас нейросети умеют описывать изображения текстом, определять, что и где на них расположено, а также делить картинки на смысловые части.

Более конкретно, на данный момент выделено несколько основных задач для работы с изображениями:

- Классификация - целью которой является определение, какой конкретно объект изображен на картинке (Что?);
- Локализация - целью которой является определение места, где располагается объект (Где?);
- Детектирование = Локализация + Классификация - целью которой является определение места расположения объекта и его конкретного типа (Где? + Что?).
- Семантическая сегментация - сопоставление каждому пикселю изображения метки класса.
- Сегментация объектов - обнаружение объектов и выделение относящихся к ним пикселей изображения.
- Преобразование изображений, которая включает такие подзадачи, как удаление фона, удаление шума, стилизация (генерация изображения с контентом, взятым с одного изображения, и стилем, взятым с другого изображения), интерполяция в пространстве изображений (генерация "среднего" изображения по 2 входным изображениям в некотором пространстве), контролируемое изменение изображений (например, изменение внешних черт и выражения лица человека, изображённого на фотографии), супер-разрешение (генерация новой версии изображения, имеющей более высокое разрешение и детализацию), дорисовка изображений и др.
- Восстановление объектов, одной из подзадач которой является реконструкция трёхмерного объекта по двумерным изображениям.

3 Описание сопутствующих задач

3.1 Классификация

Постановка задачи классификации может быть дана следующим образом: будем рассматривать изображения, на которых запечатлен один объект (классы «кот», «собака»), либо само изображение целиком может быть причислено к определенному классу (классы «природа», «море»). Тогда задача классификации изображений включает в себя либо ответ на вопрос: «Присутствует ли на изображении объект заданного класса?» - либо «Относится ли изображение к заданному классу?».

3.1.1 Проблемы задачи классификации

Описанная задача классификации является не такой тривиальной, какой кажется на первый взгляд. Причиной этому является изменение восприятия объекта на изображении вследствие смены освещения (см. рис. 1), ракурса или позы, а также возникновения некоторой деформации предмета (см. рис. 2). Кроме того, объект может слиться с фоном (см. рис. 3), либо подвергнуться окклюзии, то есть оказаться не полностью на изображении, что не должно привести к ошибке классификации (см. рис. 4). Кроме того, смена масштаба, а также вращение изображения могут существенно повлиять на качество определения типа изображенного объекта.



Рис. 1: Примеры изображений класса «собака» с измененным освещением



Рис. 2: Примеры изображений класса «собака» с изменением ракурса / позы



Рис. 3: Примеры слияния с фоном для изображений класса «собака»



Рис. 4: Примеры окклюзии для изображений класса «собака»

Несмотря на описанные трудности, задача классификации формально является решенной, причем существующие подходы позволяют классифицировать изображения точнее, чем человек.

3.2 Локализация

Постановку задачи локализации можно дать следующим образом: пусть имеется некоторое изображение. Требуется перебрать все возможные области (чаще всего используются прямоугольники) и определить, в каких регионах расположены объекты. Тем самым следует максимально точно выделить места, в которых на изображении расположены объекты, без уточнения, о каких конкретно объектах идет речь.

4 Постановка задачи детектирования объектов

Формально задача детектирования объектов на изображении может быть поставлены следующим образом: требуется перебрать все возможные локализации (чаще всего используются прямоугольники), чтобы определить, в какой части изображения расположен тот или иной объект. Затем следует для каждого выделенного прямоугольника произвести классификацию, чтобы выяснить, какой конкретно объект был выделен.

5 Метрики качества

Перед началом решения задачи требуется определить, какой же функционал будет являться мерой качества. Для задачи детектирования объектов требуется определить не только, какой объект представлен на изображении, но также и в каком конкретном месте расположен этот объект.

Пусть объект на изображении расположен в позиции (рамке), представленной на рис. 5 слева (это истинное расположение), а предсказание говорит о том, что объект расположен в розовой рамке на рис. 5 справа.

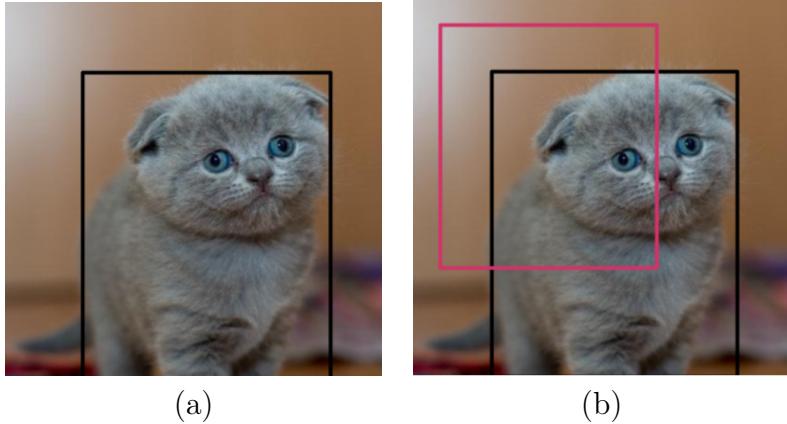


Рис. 5: Примеры истинного и предсказанного расположения объекта ([2])

Рассмотрим возможные варианты метрик для решения задачи детектирования объекта.

5.1 Intersection over union + Precision / Recall

В данном подходе разделяются задачи локализации и классификации. Для решения первой применяется метрика «пересечение над объединением», а для решения второй – метрики, порожденные матрицей несоответствий.

Поскольку для удовлетворения постановке задачи нужно найти конкретный прямоугольник, в котором расположен объект, логично будет рассмотреть, какая же часть из верного прямоугольника была угадана. Для этого подойдет следующая метрика, называемая мерой Жаккара, либо «пересечение над объединением».

$$IoU = \frac{|A \cap B|}{|A \cup B|}$$

Таким образом, если $IoU = 0$, то предсказанное расположение объекта полностью не совпадает с истинным, а если $IoU = 1$, то предсказание абсолютно точно совпало с реальностью.

Задача детектирования объекта на изображении считается решенной (в данном случае речь идет именно о части локализации), если $IoU \geq \alpha$, то есть значение метрики пре-восходит некоторый наперед заданный порог α (зачастую $\alpha = 0.5$). Пояснение к работе меры представлено на рис. 6.

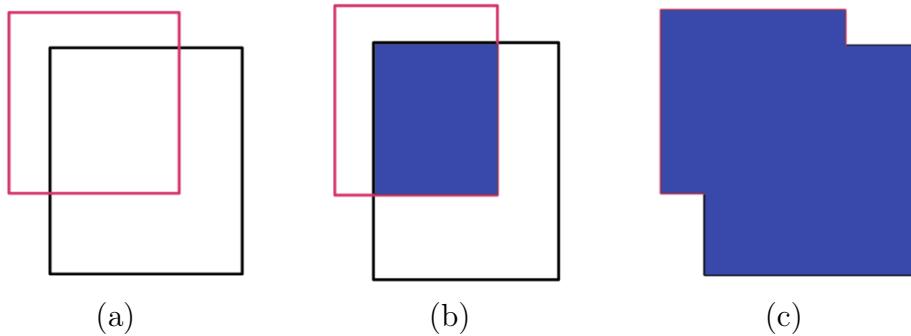


Рис. 6: Пояснение к метрике IoU. (а) – истинный объект (черная рамка) + предсказанный (розовая); (б) – пересечение предсказаний (числитель формулы); (с) – объединение (знаменатель) ([2])

Для второй составляющей задачи (а именно части классификации) используются следующие традиционные метрики (FPR, TPR, точность, полнота). В случае бинарной классификации для их определения используется матрица несоответствий (см. таб. 1).

	Принадлежит классу 0	Принадлежит классу 1
Предсказан класс 0	TP	FP
Предсказан класс 1	FN	TN

Таблица 1: Матрица несоответствий для бинарной классификации (0 и 1)

Тогда в указанной терминологии значения метрик вычисляются следующим образом:

- Точность (accuracy) – показывает долю правильных классификаций:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

- Полнота (recall) – показывает долю найденных объектов класса к общему числу объектов класса:

$$Recall(TPR) = \frac{TP}{TP + FN}$$

- Точность (precision) – показывает долю объектов класса среди объектов выделенных классификатором:

$$Precision = \frac{TP}{TP + FP}$$

- FPR – показывает долю неверных срабатываний классификатора к общему числу объектов за пределами класса:

$$FPR = \frac{FP}{FP + TN}$$

Для задачи многоклассовой классификации обобщение происходит следующим образом (см. таб. 2).

Предсказание	Класс 1 (C_1)	Класс 2 (C_2)	Класс 3 (C_3)
1(P_1)	T_1	F_{12}	F_{13}
2(P_2)	F_{21}	T_2	F_{23}
3(P_3)	F_{31}	F_{32}	T_3

Таблица 2: Матрица несоответствий для многоклассовой классификации (C_1, C_2, C_3)

В этом случае TP, TN, FP и FN считаются относительно некоторого класса (i) следующим образом (а формулы метрик остаются такими же):

$$TP_i = T_i$$

$$FP_i = \sum_{c \in Classes} F_{i,c}$$

$$FN_i = \sum_{c \in Classes} F_{c,i}$$

$$TN_i = All - TP_i - FP_i - FN_i,$$

где All – общее число объектов.

В указанных обозначениях при получении $IoU < \alpha$ возможно следующие ошибки, представленные на рис. 7. Кроме того, на указанном рисунке визуализированы понятия, введенные ранее (TP, FP, FN). Для изображения слева $IoU = 0.8$ (то есть локация установлена верна), а что касается правого – возможны 2 варианта:

- Был сделан прогноз, который не содержал объект (ложно-положительный результат, FP).
- Не удалось предсказать наличие объекта (ложно-отрицательный результат, FN).

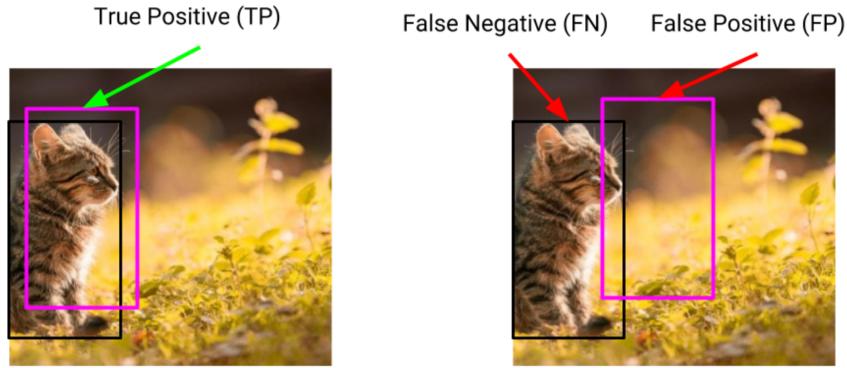


Рис. 7: Пояснение к введенным понятиям TP, FP, FN ([2])

5.2 Average Precision

Для вычисления средней точности (Average Precision), следует ранжировать прогнозы, основываясь на оценках вероятности класса алгоритмом. Примеры вероятностей представлены на рис. 8.

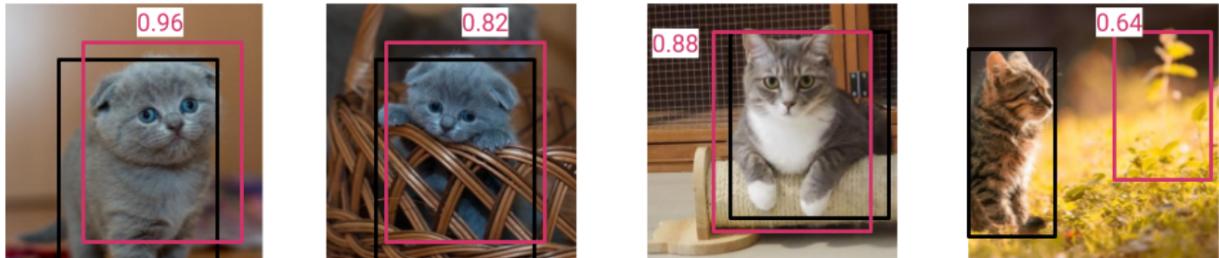


Рис. 8: Примеры уверенности модели в предсказании ([2])

Ранжированные прогнозы см. на рис. 9.

Confidence	Cat Number	isCorrect?	Precision	Recall
0.96	Cat 1	1	1	0.25
0.88	Cat 3	1	1	0.5
0.82	Cat 2	1	1	0.75
0.64	Cat 4	0	0.75	0.75

Рис. 9: Пример вычисления метрики AP ([2])

С использованием ранжированных прогнозов затем создается накопительный график (точность – полнота). Результирующий график см. на рис. 10.



Рис. 10: График для вычисления AP ([2])

5.3 Mean Average Precision

Вычисление средней точности, проведенное выше, применяется для одного класса. Что применить в случае нескольких классов? Для этого используется MAP (Mean Average Precision). Предположим, что имеется набор данных о кошках, собаках, автомобилях, велосипедах и воздушных шарах. Тогда следует вычислить AP для каждого из классов отдельно, а затем вычислить среднее значение рейтингов AP. (см. пояснение в таблице 3).

ap_cat	0.75
ap_dog	0.6
ap_car	0.8
ap_bicycle	0.9
ap_balloon	0.5
MAP	0.71

Таблица 3: Пример вычисления MAP для многоклассового случая ([2])

5.4 Panoptic Quality

Выражение для получения метрики выглядит следующим образом (с использованием предыдущих обозначений):

$$PQ = \frac{\sum_{(p,g) \in TP} IoU(p, g)}{|TP| + \frac{1}{2}|FP| + |FN|}$$

Его смысл легче объяснить, разделив на 2 части.

$$PQ = \underbrace{\frac{\sum_{(p,g) \in TP} IoU(p, g)}{|TP|}}_{segmentation\ quality(SQ)} \cdot \underbrace{\frac{|TP|}{|TP| + \frac{1}{2}|FP| + |FN|}}_{recognition\ quality(RQ)}$$

Качество сегментации (SQ) оценивает, насколько предсказанные сегменты соответствуют своему истинному расположению. Когда данное значение приближается к 1, это означает, что предсказанные сегменты (TP) лучше соответствуют настоящим. Тем не менее, указанная часть не принимает во внимание плохие прогнозы.

Именно для корректировки этого недостатка используется качество распознавания (RQ). Этот показатель представляет собой сочетание точности и полноты и позволяет определить, насколько эффективна модель.

6 Основная проблема

Основная проблема при детектировании объектов на изображении заключается в том, что генерируемые области изображений, в которых находится предмет, имеют разный размер. Сверточные слои умеют работать с изображениями разного размера, в то время как полно связные слои – не могут. Это является проблемой, поскольку большинство нейросетей, решающих задачу классификации, представляют собой именно совокупность сверточных слоев, а затем – несколько полно связных. Для решения данной проблемы применяются различные средства, описанные ниже.

Рассмотрим сначала методы генерации тех областей (region proposals), в которых потенциально есть объекты, подлежащие классификации.

6.1 Selective Search

Выборочный поиск (Selective Search) применяется для нахождения возможных местоположений объектов (локализации) и используется при распознавании объектов. Описываемый метод сочетает в себе черты исчерпывающего поиска (exhaustive search) и сегментации. Как и в случае сегментации, в процессе отбора областей для выборочного поиска используется структура изображения. Как и в случае исчерпывающего поиска, алгоритм стремится рассмотреть все возможные местоположения объекта.

Пример работы выборочного поиска в разных масштабах представлен на рис. 11.



Рис. 11: Пример работы выборочного поиска в разных масштабах ([3])

Описанный алгоритм получает на вход цветное изображение, а на выходе предоставляет множество всевозможных местоположений объектов (L). Даный метод работает следующим образом:

- Создается пустой список: $S = []$;
- Происходит разбиение изображения на сегменты: $R = \{r_1, \dots, r_n\}$;
- Для каждого сегмента в список добавляется его обрамляющая рамка следующим образом: для каждой пары сегментов (r_i и r_j) вычисляется мера их схожести, а затем добавляется в список S ;
- Происходит объединение соседних похожих сегментов: вычисляется максимальное значение списка S ($\max(S)$), а также запоминаются r_i и r_j , при которых оно достигается. Полученные сегменты объединяются в новый сегмент (r_t); из списка удаляется вся информация о мере схожести r_i и r_j с соответствующими соседями, а затем добавляется вся информация о соседях нового сегмента r_t ;
- Если число сегментов > 1 (нужно добиться того, чтобы изображение стало одной цельной областью), следует перейти к пункту 2 и продолжить объединение;
- Для каждой рамки из R нужно оценить наличие в ней объекта: извлечь соответствующие области и записать их в итоговый список L .

В качестве меры сходства регионов используется линейная комбинация 4 функций, с коэффициентами $a_i \in \{0, 1\}$, $i = \overline{1, 4}$, каждая из которых оценивает, насколько 2 региона похожи по какому-то одному признаку:

$$s(r_i, r_j) = a_1 s_{colour}(r_i, r_j) + a_2 s_{texture}(r_i, r_j) + a_3 s_{size}(r_i, r_j) + a_4 s_{fill}(r_i, r_j)$$

Данные функции определяются следующим образом:

- s_{colour} оценивает сходство регионов по цвету. Для каждого региона строится по одномерной гистограмме с 25 интервалами для каждого цветового канала. Полученные для региона r_i гистограммы конкатенируются в вектор $C_i = (c_i^1, \dots, c_i^n)$ ($n = 75$ для трёхканального изображения), который нормализуется по норме L_1 . Мера сходства задаётся формулой $s_{colour} = \sum_{k=1}^n \min(c_i^k, c_j^k)$.
- $s_{texture}$ оценивает сходство регионов по текстуре. Для каждого региона вычисляется производная Гаусса для каждого канала с параметром $\sigma = 1$ (то есть разностой производной от фильтра Гаусса, задаваемого формулой $\frac{1}{6} \begin{pmatrix} 0 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 0 \end{pmatrix}$) в 8 направлениях. Для каждого направления вычисляется гистограмма с 10 интервалами. Полученные для региона r_i гистограммы конкатенируются в вектор $T_i = (t_i^1, \dots, t_i^n)$ ($n = 240$ в случае трёхканального изображения), к которому применяется нормализация по норме L_1 . Мера сходства вычисляется по формуле $s_{texture}(r_i, r_j) = \sum_{k=1}^n \min(t_i^k, t_j^k)$.
- s_{size} выдаёт большее значение для регионов небольшого размера с целью объединить их в первую очередь и вычисляется по формуле $s_{size}(r_i, r_j) = 1 - \frac{\text{size}(r_i) + \text{size}(r_j)}{\text{size}(im)}$, где $\text{size}(im)$ - размктер изображения в пикселях.

- s_{fill} выдаёт наибольшее значение для регионов r_i и r_j , среди которых один вложен в другой, и вычисляется по формуле $s_{fill}(r_i, r_j) = 1 - \frac{\text{size}((BB)_{ij}) - \text{size}(r_i) - \text{size}(r_j)}{\text{size}(im)}$, где BB_{ij} - минимальный прямоугольник, содержащий r_i и r_j .

7 Основные подходы к детектированию объектов

7.1 Regions With CNNs (R-CNN)

Данная система обнаружения объектов состоит из трех-четырех модулей. Описание схемы работы метода см. на рис. 12.

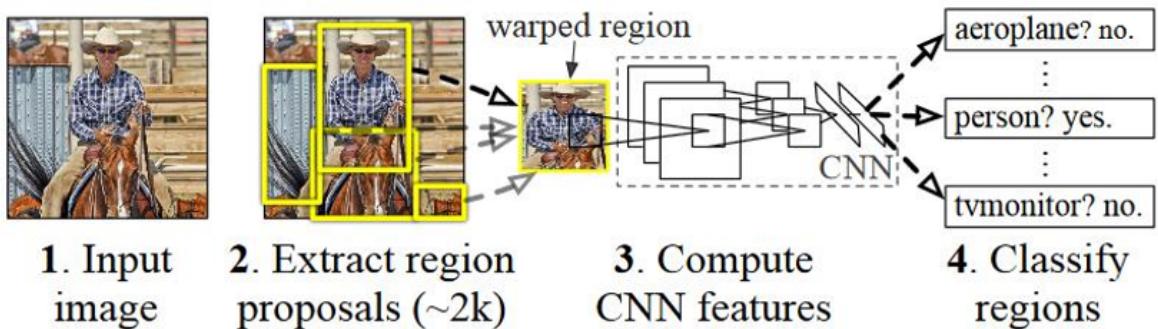


Рис. 12: Пример работы R-CNN ([4])

Первый генерирует независимые от категории предложения области с применением описанного выше метода Selective Search (извлекается около 2000 возможных локализаций). Эти предложения определяют множества месторасположений кандидатов, доступных для детектора.

Второй модуль (детектор) представляет собой глубокую сверточную нейронную сеть, которая извлекает вектор фиксированной длины из каждого региона: данная часть нужна в частности для решения описанной выше проблемы разной размерности входящих изображений. Все они приводятся (с помощью заключения области в наименьший охватывающий квадрат, а затем применение resize) к размеру 277×277 .

Далее полученное изображение с использованием глубокой нейросети, натренированной для классификации изображений на ImageNet (Caffe Net), переводится в вектор размера 4096. Поскольку в изначальной интерпретации число классов равняется $N = 1000$, а в описываемом методе $N=20$ или 200, то последний классификационный слой данной сети заменяется на слой с количеством выходов $N + 1$, где дополнительный класс отвечает за фон. На измененном последнем слое для упомянутой обученной нейронной сети происходит дообучение на текущих тренировочных данных.

Третий модуль (классификатор) – набор классических линейных SVM (по количеству классов) для классификации объектов в выделенных регионах: каждая из частей SVM является бинарным классификатором и отвечает на вопрос, является ли объект из области представителем рассматриваемого класса или нет.

Четвертый модуль (опционально) отвечает за оптимизацию полученных областей с использованием ограничительной линейной регрессии (Bounding-box regression), что позволяет получить более точный охват объекта (см. пример на рис. 13). Работу данно-

го модуля можно представить в виде 4 задач Ridge-регрессии: R-CNN bounding box regression:

1. В качестве обучающей выборки в задаче 1 выступают пары $\{(\phi_5(P_i), \frac{x_i - \bar{x}_i}{\bar{w}_i})\}_{i=1}^N$, где N - количество объектов, распознанных на всех изображениях исходной обучающей выборки, (x_i, y_i) - координаты центра истинного ограничивающего прямоугольника, (\bar{x}_i, \bar{y}_i) - координаты центра построенного ограничивающего прямоугольника, (w_i, h_i) - ширина и высота истинного ограничивающего прямоугольника, (\bar{w}_i, \bar{h}_i) - ширина и высота построенного ограничивающего прямоугольника, $P_i = (x_i, y_i, w_i, h_i)$, $\phi_5(P)$ - отображение части изображения, ограниченного прямоугольником P в выход слоя $pool_5$ при подаче этой части изображения на вход части сети, отвечающей за извлечение признаков. Задача 1 формулируется следующим образом:

$$\sum_{i=1}^N \left(\frac{x_i - \bar{x}_i}{\bar{w}_i} - \langle \beta_x, \phi_5(P_i) \rangle \right)^2 + \lambda_x \|\beta_x\|^2 \rightarrow \min_{\beta_x}$$

2. В качестве обучающей выборки в задаче 2 выступают пары $\{(\phi_5(P_i), \frac{y_i - \bar{y}_i}{\bar{h}_i})\}_{i=1}^N$. Задача 2 формулируется следующим образом:

$$\sum_{i=1}^N \left(\frac{y_i - \bar{y}_i}{\bar{h}_i} - \langle \beta_y, \phi_5(P_i) \rangle \right)^2 + \lambda_y \|\beta_y\|^2 \rightarrow \min_{\beta_y}$$

3. В качестве обучающей выборки в задаче 3 выступают пары $\{(\phi_5(P_i), \log(\frac{w_i}{\bar{w}_i}))\}_{i=1}^N$. Задача 3 формулируется следующим образом:

$$\sum_{i=1}^N \left(\log\left(\frac{w_i}{\bar{w}_i}\right) - \langle \beta_w, \phi_5(P_i) \rangle \right)^2 + \lambda_w \|\beta_w\|^2 \rightarrow \min_{\beta_w}$$

4. В качестве обучающей выборки в задаче 4 выступают пары $\{(\phi_5(P_i), \log(\frac{h_i}{\bar{h}_i}))\}_{i=1}^N$. Задача 4 формулируется следующим образом:

$$\sum_{i=1}^N \left(\log\left(\frac{h_i}{\bar{h}_i}\right) - \langle \beta_h, \phi_5(P_i) \rangle \right)^2 + \lambda_h \|\beta_h\|^2 \rightarrow \min_{\beta_h}$$

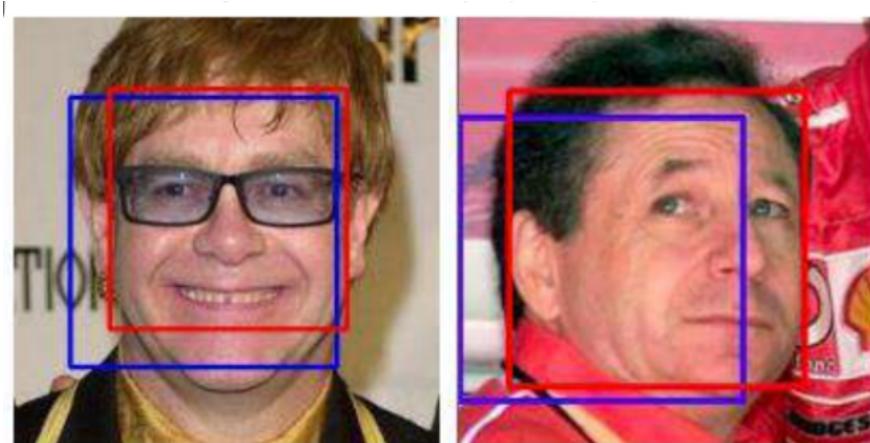


Рис. 13: Пояснение понятия bounding-box ([6])

7.1.1 Недостатки подхода

1. Вычисление признаков происходит для каждого региона-гипотезы. Поскольку они пересекаются, подобные действия ведут к избыточным вычислениям. Кроме того, в методе дообучается сначала CNN, затем работают несколько SVM, затем опционально появляется еще и bounding-box регрессия, что приводит к очень большому времени обучения.
2. Масштабирование фрагментов до нужного разрешения ведет к возможному их искажению.
3. Работа метода зависит от алгоритма генерации регионов (напомним, что в данном случае используется Selective Search).

7.2 Non Maximum Suppression (NMS)

В привычном представлении метод подавления не-максимумов (Non Maximum Suppression) был введен John F. Canny. Этот принцип гласит, что пикселями границ являются точки, в которых достигается локальный максимум градиента в направлении вектора градиента.

Используемый же в данном случае алгоритм отличается от исходной формулировки и используется при решении конкретной задачи фильтрации предположений.

Алгоритм получает на вход список предположительных рамок, в которых расположены объекты (B), соответствующие значения уверенности (confidence score – S), а также значение порога N; выводит список отфильтрованных предположений (D) и состоит из следующих шагов:

- Регионы упорядочиваются по значению S (0.9, 0.8, 0.7), причем числа ниже определенного порога сразу подлежат занулению.
- Выбираем область (R) с наибольшим S, удаляем ее из B и добавляем в D.
- Сравниваем текущую область R с другими областями из B путем вычисления метрики IoU. В случае если $\text{IoU} > N$ – удаляем данное предположение из B.
- Повторяем описанный выше процесс до тех пор, пока множество B не станет пустым.

Работа метода проиллюстрирована на рис. 14 и рис. 15.

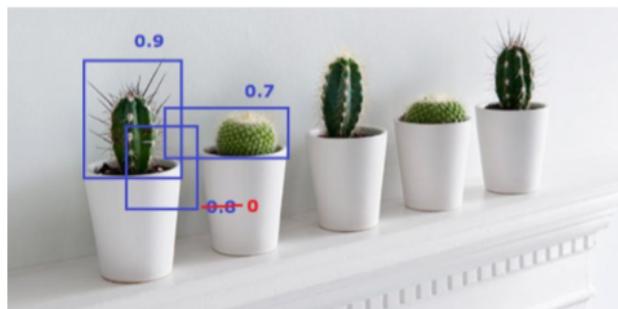


Рис. 14: Пример работы NMS ([1])

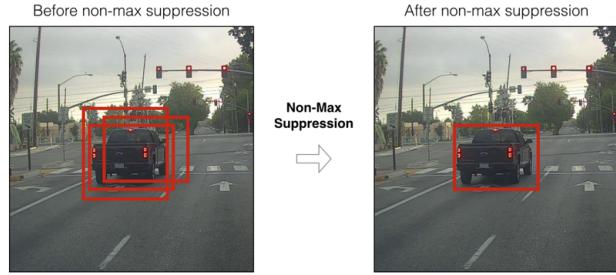


Рис. 15: Пример работы NMS ([7])

7.3 Spatial Pyramid Pooling (SPP-net)

Поскольку полносвязной нейросети нужен вход определенного размера, а чисто сверточная нейросеть способна работать со входом любого размера, идея заключается в том, чтобы добавить к сверточной сети некоторый слой, способный приводить изображение к фиксированному размеру.

В качестве такого слоя используется Spatial Pyramid Pooling (SPP), применяющий операцию Max Pooling к различным частям изображения для приведения его к фиксированному размеру. Преимущество данного подхода в сравнении с использованием операции обрезания изображения (crop) представлено на рис. 16.

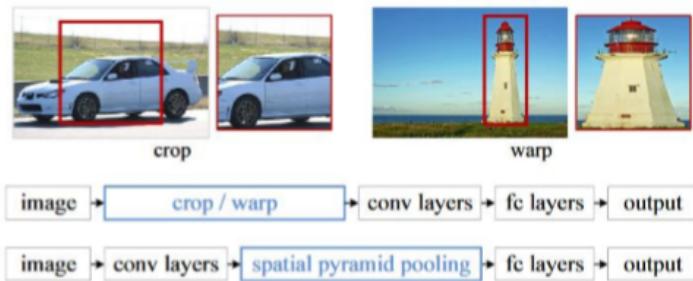


Рис. 16: Пример работы SPP ([9])

Данное средство, как и следует из названия, представляет собой «пирамиду» из обобщающих слоев, каждый из которых задается следующим образом:

Выбираются параметры размер окна (W_{win}, H_{win}), величина сдвига (stride) по вертикали и горизонтали (W_{str}, H_{str}), а также величина расширения исходного изображения (padding) (W_{pad}, H_{pad}). Тогда, подавая на вход такого слоя матрицу размера (W_{inp}, H_{inp}), на выходе мы получим матрицу с размерами:

$$(W_{out}, H_{out}) = ((W_{inp} - W_{win} + 2 \cdot W_{pad})/W_{str} + 1, (H_{inp} - H_{win} + 2 \cdot H_{pad})/H_{str} + 1)$$

Каждый элемент выходной матрицы равен значению обобщающей функции (pooling: MaxPooling или AveragePooling), вычисленному на соответствующем прямоугольном окне размера (W_{win}, H_{win}) входной матрицы. Структура нейронной сети со слоем Spatial Pyramid Pooling представлена на рис. 17

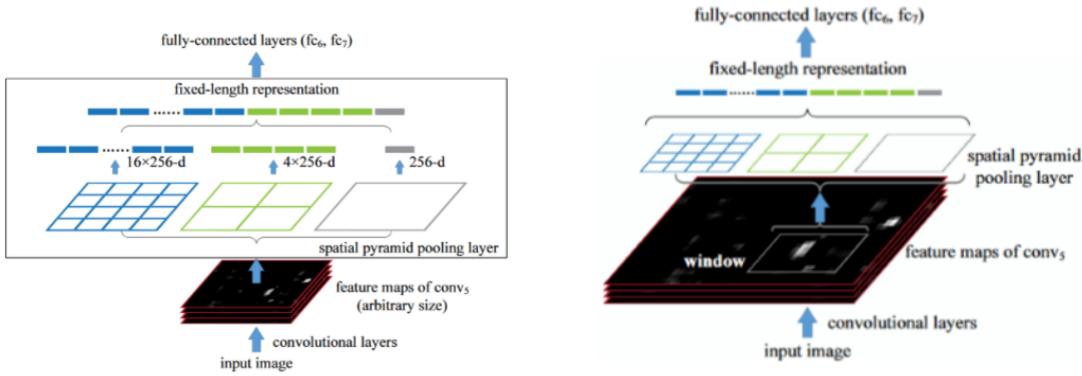


Рис. 17: Структура нейронной сети со слоем SPP ([9])

На данном рисунке 256 – число фильтров на сверточном слое *conv₅*, который является последним сверточным слоем. Для каждого изображения его представление в результате работы CNN вычисляется ровно 1 раз, а затем для каждой области указанным выше способом получается признаковое представление, имеющее фиксированный размер. Скорость работы при таком подходе увеличивается в 10-100 раз! Рассмотрим процесс обучения сети. Первый метод обучения модели на изображениях разных размеров, предложенный авторами, состоит в следующем:

1. Для обучения используются изображения размера 180x180 и 224x224. Изображения размера 224x224 сжимаются до размера 180x180.
2. Одну полную эпоху сеть учится на изображениях с исходным размером 180x180, следующую - на изображениях с исходным размером 224x224, затем - снова на изображениях меньшего размера.

Во втором методе обучения на изображениях разного размера авторы статьи на каждой эпохе не чередовали размер входных изображений, а выбирали их размер s из равномерного распределения на отрезке [180, 224].

Также производилось обучение на изображениях одинакового размера, но такой подход показал качество хуже, чем подход с обучением на изображениях разного размера.

7.4 Fast R-CNN

Авторы данного подхода предложили произвести ускорение процесса с использованием следующих нововведений:

- Производить вычисление нейросетевых признаков не по каждой из 2000 предсказанных областей, а по всему изображению целиком. Затем регионы, полученные, как и в R-CNN, с помощью Selective Search, накладывались на общую карту полученных признаков;
- Поскольку после работы предыдущего пункта получаются изображения разного размера, был добавлен слой SPP (если быть точнее, то ROI-pooling, который будет описан далее) для их приведения к одному фиксированному размеру;

- Вместо независимого обучения трех моделей ($CNN \rightarrow SVM \rightarrow$ bounding-box regression) обучение было объединено в одну стадию (регрессия bounding-box была встроена в нейросеть).

Работа метода проиллюстрирована на рис. 18.

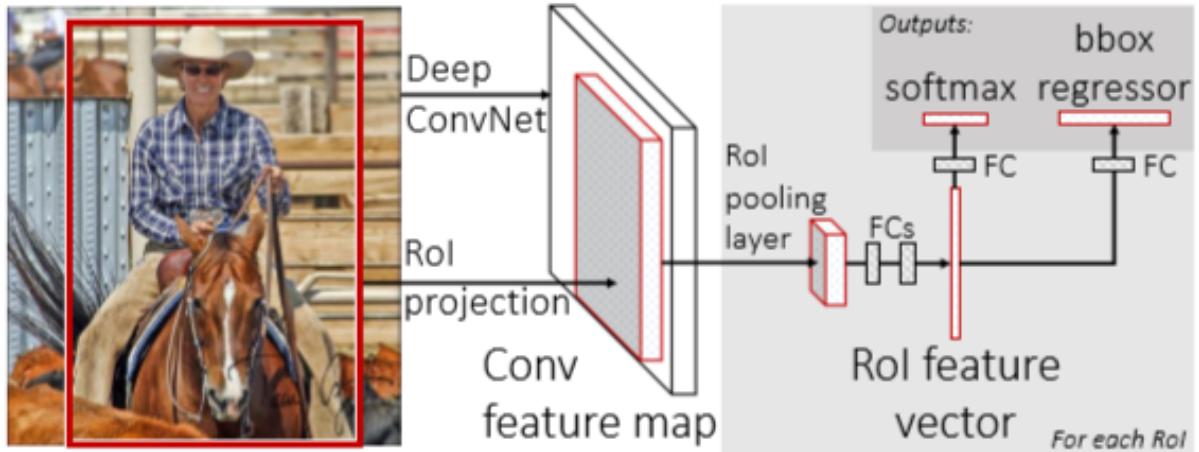


Рис. 18: Пример работы Fast R-CNN ([10])

Кроме того, вместо совокупности бинарных SVM (по числу классов для классификации) все признаки передавались на полносвязный слой, а затем на 2 параллельных слоя: softmax с $N + 1$ выходами (для каждого класса, а также для фона) и bounding-box regressor. Для совместного обучения упомянутых параллельных слоев функция ошибок была представлена как сумма функций ошибок классификатора и регрессора:

$$L(p, u, t^u, v) = L_{cls}(p, u) + \lambda[u \geq 1]L_{loc}(t^u, v),$$

где:

- u – класс объекта, который действительно расположен в окне-гипотезе;
- $L_{cls}(p, u) = -\log(p_u)$ – значение log-loss для класса u ;
- $v = (v_x, v_y, v_w, v_h)$ – реальные изменения рамки области для более точного охвата объекта;
- $t^u = (t_x^u, t_y^u, t_w^u, t_h^u)$ – предсказанные изменения рамки области;
- $L_{loc}(t^u, v) = \sum_{i \in \{x, y, w, h\}} smooth_{L_1}(t_i^u - v_i)$ – функция потерь между предсказанными и реальными изменениями рамки;

$$smooth_{L_1}(x) = \begin{cases} 0.5x^2 & \text{если } |x| < 1 \\ |x| - 0.5 & \text{иначе} \end{cases}$$

- $[u \geq 1]$ – индикаторная функция;
- $u = 0$ – обозначает фон (то есть в некоторой области отсутствуют объекты);

- λ – коэффициент для балансирования вклада обеих функций потерь в общий результат (в оригинальной статье [10] $\lambda = 1$).

smooth_{L_1} представляет собою робастную версию функции потерь L_1 . Она менее чувствительна к выбросам, чем функция потерь L_2 , которая была использована в R-CNN и SPPnet. В случае больших значений целевой переменной может понадобиться аккуратная настройка темпа обучения регрессии с функцией потерь L_2 для того, чтобы избежать "взрыва" градиентов. Использование smooth_{L_1} помогает побороть эту сложность обучения модели.

7.4.1 RoI Pooling Layer

Признаки из разных областей приводились к одной размерности с использованием «модификации SPP» – RoIPooling. Он представляет из себя формально SPP-слой в SPP-net с одной пирамидой (см. иллюстрацию к работе метода на рис. 19).

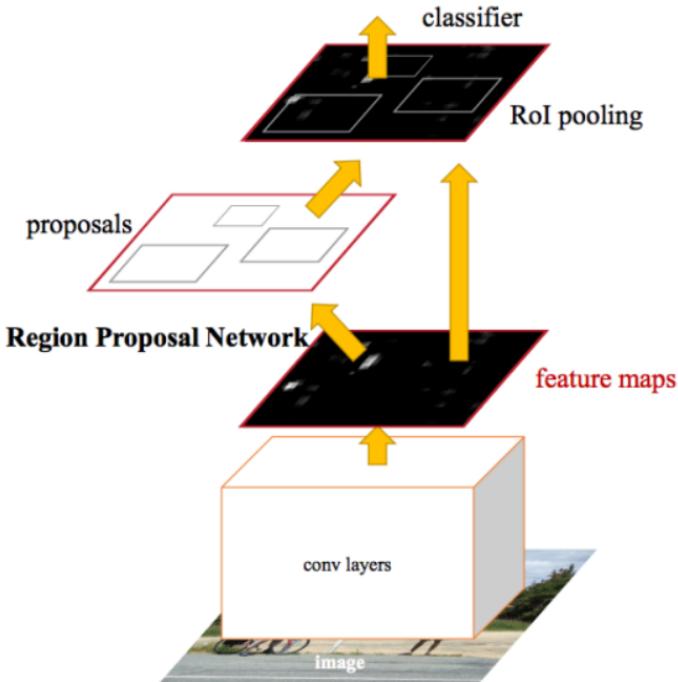


Рис. 19: Иллюстрация к работе RoI ([5])

В интересующей области выделяется окно шириной h и высотой w , которое далее делится на сетку с $H \times W$ ячейками, каждая из которых имеет размер $h/H \times w/W$ (в оригинальной статье [10] используется $W = H = 7$). По каждой ячейке производился Max Pooling для выбора только одного значения. Таким образом получалась результирующая матрица признаков $H \times W$. Пояснение к работе ROI-Pooling Layer см. на рис. 20.

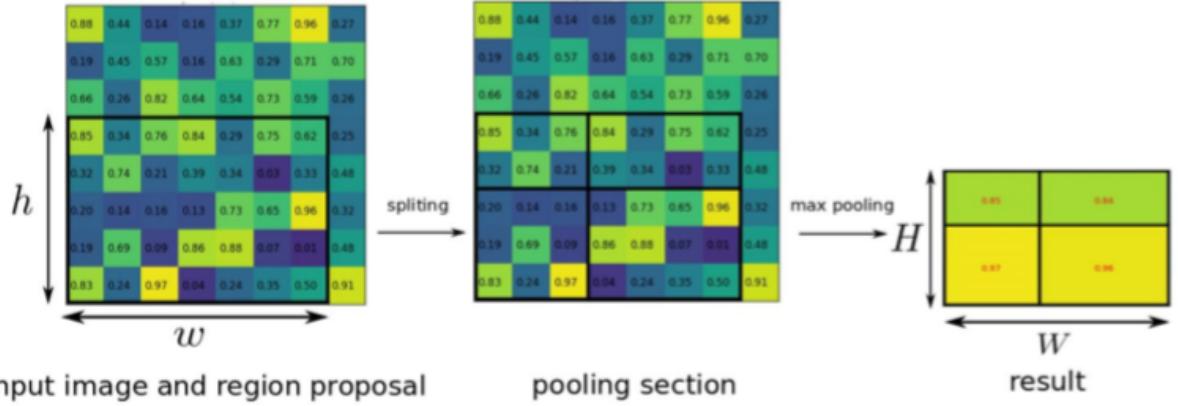


Рис. 20: Пояснение к работе RoI ([5])

7.5 Faster R-CNN

Даже после серии улучшений, проведенных от R-CNN к Fast R-CNN, все равно оставалась проблема, связанная с изначальной генерацией регионов (поскольку Selective Search работал медленно). Поэтому данный этап был заменен на следующий блок под названием Region Proposal Network (RPN), что продемонстрировано на рис. 21.

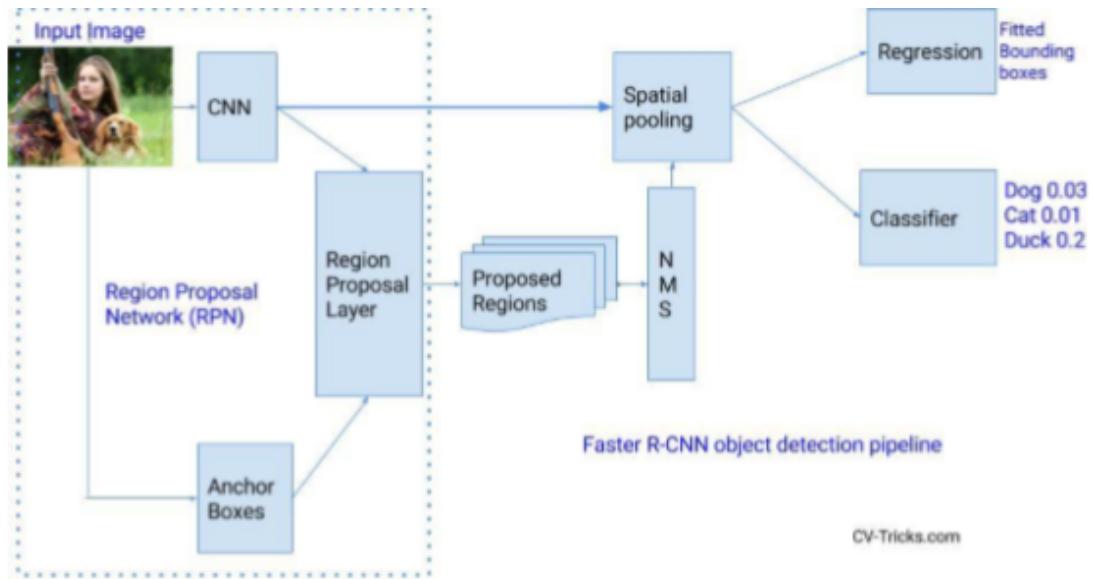


Рис. 21: Архитектура Faster R-CNN ([11])

7.5.1 Region Proposal Network (RPN)

В данном случае процедура генерации регионов стала нейросетевой (и как следствие – обучаемой) и работает следующим образом (см. рис. 21): по извлеченным с помощью CNN признакам совершается проход небольшой нейросетью со сверточным окном 3×3 . Полученные в результате этих действий значения передаются в 2 параллельных полно связных слоя (здесь видна аналогия с Fast R-CNN): box-regression layer (reg) – 4k выход-

дов и box-classification layer (cls) – 2k выходов. Выходы указанных слоев основываются на anchorах (то есть якорных прямоугольниках): к рамкам для каждого положения скользящего окна, имеющих разные размеры и соотношения сторон (с использованием $W \times H$ – карты признаков получается $W \cdot H \cdot k$ регионов – anchors). Reg-слой для каждого якорного прямоугольника выдает по 4 координаты, корректирующие положение охватывающей рамки; cls-слой выдает по 2 числа – вероятности того, что рамка содержит хоть какой-то объект или что не содержит. Также, как и в предыдущем случае, обучение слоев происходит совместно, то есть функция потерь представляет собой сумму двух функций потерь с балансирующим коэффициентом:

$$L(\{p_i\}, \{t_i\}) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) + \lambda \frac{1}{N_{reg}} \sum_i L_{reg}(t_i, t_i^*)$$

В данной формуле i - индекс якорного прямоугольника в мини-батче, p_i - предсказанная вероятность того, что рассматриваемый якорный прямоугольник содержит объект, p_i^* - истинная метка якорного прямоугольника ($p_i^* = 1$, если существует истинный ограничивающий прямоугольник b такой, что рассматриваемый якорный прямоугольник входит в список якорных прямоугольников, имеющих с b наибольшее значение IoU или больше 0.7; $p_i^* = 0$, если рассматриваемый якорный прямоугольник имеет IoU меньше 0.3 со всеми истинными ограничивающими прямоугольниками; если рассматривающий ограничивающий прямоугольник не удовлетворяет описанным 2 условиям, то он не вносит вклад в обучение); $t_i = (x_i, y_i, w_i, h_i)$ - вектор, содержащий положение, ширину и высоту предсказанного прямоугольника; t_i^* - истинный ограничивающий прямоугольник, соответствующий якорному прямоугольнику положительного класса; N_{cls} - размер мини-батча (например, 256); N_{reg} - количество различных положений якорных прямоугольников (например, около 2400); L_{cls} - логарифмическая функция потерь; $L_{reg} = \text{smooth}L_1(t_i - t_i^*)$; λ - балансирующий коэффициент.

Оба указанных слоя выдают предположения для регионов, поэтому области с большей вероятностью содержания объекта передаются далее для уточнения региона и детектирования. Общую схему работы см. на рис. 22.

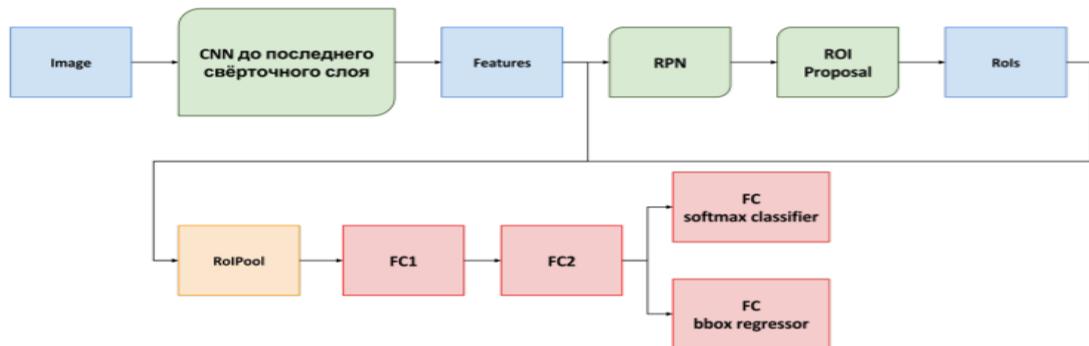


Рис. 22: Схема работы Faster R-CNN ([11])

Таким образом, описанный нетривиальный алгоритм обучения можно сформулировать, например, следующим образом (итерационно):

- Для начала происходит инициализация и обучение RPN – части для поиска областей;
- С использованием полученных RPN-областей заново обучается часть Fast R-CNN;
- Обученная сеть детектирования используется для инициализации весов для RPN. Общие сверточные слои при этом фиксируются и происходит донастройка только тех слоев, которые присутствуют только в RPN;
- С зафиксированными сверточными слоями окончательно настраивается Fast R-CNN.

Важная особенность RPN заключается в том, что данный метод является инвариантным к сдвигу как с точки зрения якорных прямоугольников, так и с точки зрения функций, задающих преобразование положения, высоты и ширины этих прямоугольников: при сдвиге объекта на изображении предлагаемый ограничивающий прямоугольник тоже должен сдвинуться и те же функции преобразования положения, высоты и ширины прямоугольника должны быть способны предсказать соответствующие изменения.

7.6 Сравнение скорости работы описанных R-CNN сетей

На рис. 23 изображена диаграмма, показывающая разницу в скорости методов, рассмотренных выше. Из нее видно, что скорость работы увеличилась в 250 раз от первого к последнему алгоритму.

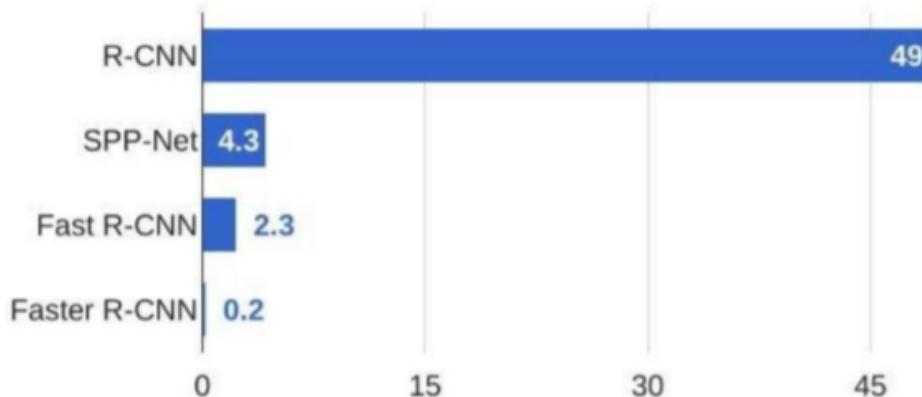


Рис. 23: Сравнение времени работы методов ([1])

7.7 You Only Look Once (YOLO)

Ко всем описанным выше решениям задачи детектирования объектов на изображении добавляется еще и система обнаружения объектов в реальном времени – YOLO. Она способна обрабатывать около 45 изображений в секунду (а ее уменьшенная, хотя и менее точная, но все же сохраняющая почти удвоенное преимущество в сравнении с остальными онлайн-системами обнаружения объектов, версия YOLO-tiny – до 155 изображений).

Результаты работы см. на рис. 24.

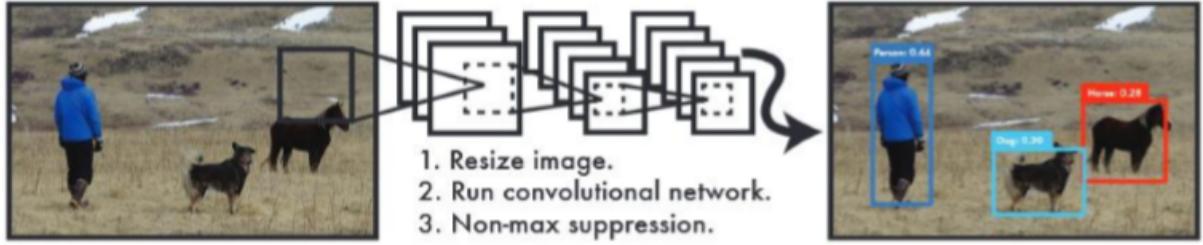


Рис. 24: Результаты работы YOLO ([12])

В сравнении с описанными ранее подходами, YOLO больше напоминает полносверточную нейросеть и пропускает заранее отмасштабированное до размера 448×448 (а не 224×224 , как было ранее, поскольку в случае детектирования объектов увеличение размера повлечет за собой улучшение результата) изображение всего лишь через одну сверточную сеть (архитектуру см. на рис. 25).

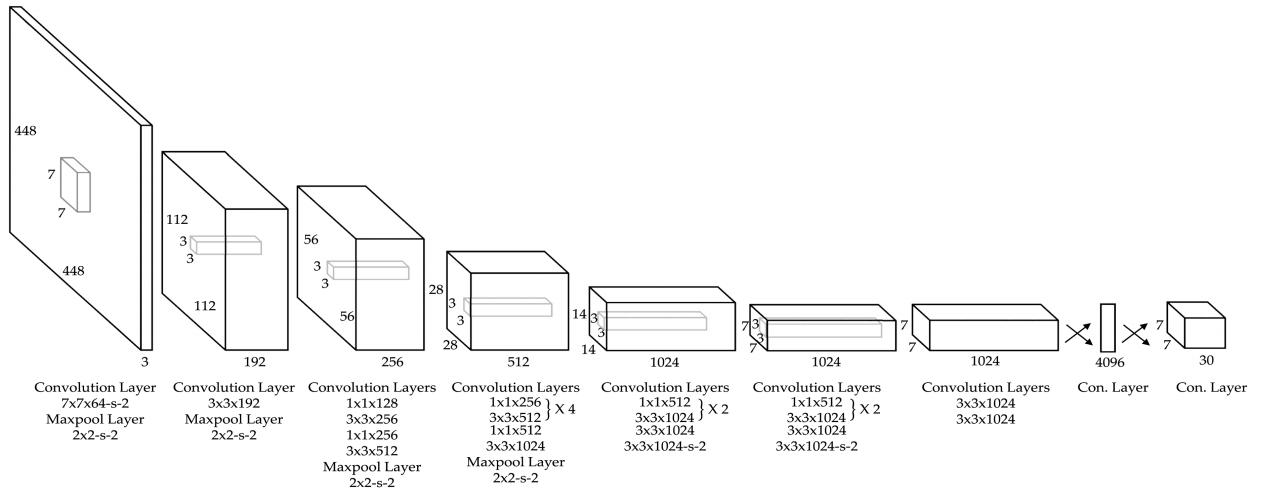


Рис. 25: Архитектура YOLO ([12])

На выходе из данной компоненты получаются координаты рамки, а также вероятности принадлежности объекта к тому или иному классу (так же, как и ранее, координаты – с помощью bound box regression, а вероятности – с использованием softmax). Затем происходит обработка и фильтрация с помощью NMS (со всеми шагами можно ознакомиться на рис. 26).

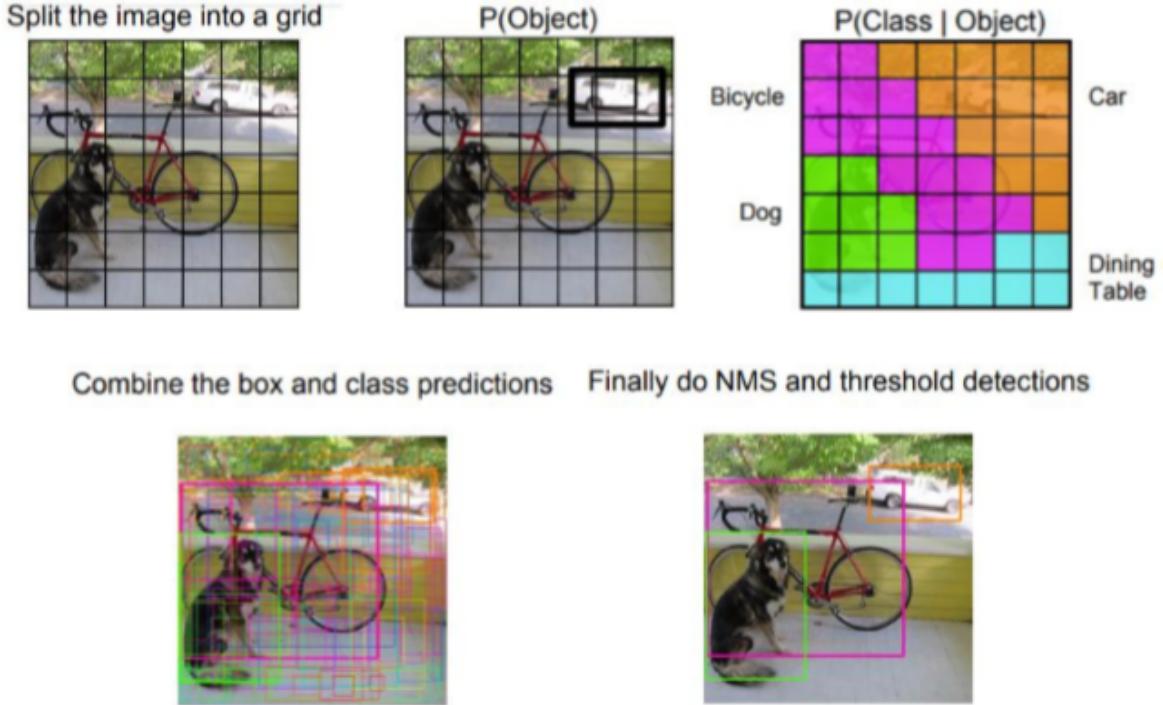


Рис. 26: Шаги работы YOLO ([12])

YOLO работает с сеткой на изображении, показывая вероятность принадлежности каждой ячейки к тому или иному классу. Понятие рамки (в которую заключается объект) включает в себя 5 предсказаний: $\{x, y, w, h, c\}$, где (x, y) – координаты центра соответствующей рамки, $c = Pr(Class_i|Object)$ – мера уверенности в том, что область выделена правильно, w, h – размеры (ширина и высота) рамки. Описанная система рассматривает задачу детектирования объектов как задачу регрессии. Она делит изображение на сетку $S \times S$ и для каждой ячейки сетки предсказывает в ограничивающих рамок, их достоверность, а также вероятности объекта из рамки принадлежать классу С. Эти прогнозы кодируются как тензор $S \times S \times (B \cdot 5 + C)$; в конкретной статье ([12]) выбирается $S = 7, B = 2$ (см. рис. 27).

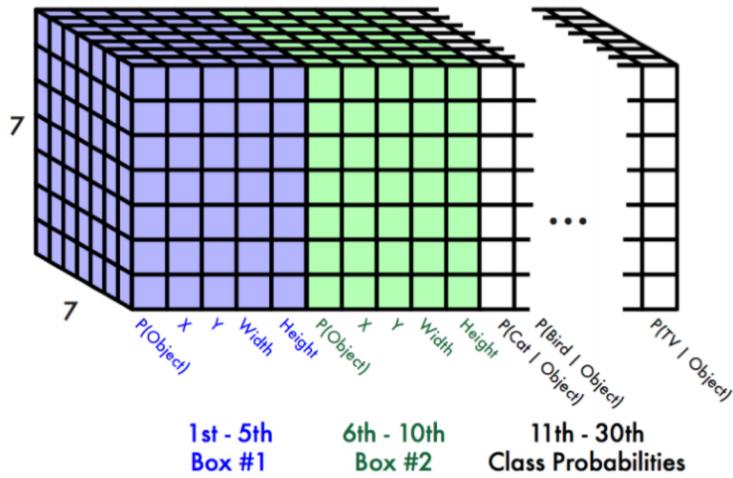


Рис. 27: Вывод YOLO ([1])

Выбор B связан с предположением о том, что в каждом элементе сетки (ячейке) находятся представители не более чем двух классов (кроме того, в задаче делается предположение о том, что каждая ячейка принадлежит не более, чем одному классу). Первые 20 слоев единственной сверточной нейросети, описанной в данном методе, предобучены на датасете ImageNet. В архитектуре нейросети используется функция активации Leaky ReLU, которая выглядит следующим образом:

$$\text{LeakyReLU}(x) = \begin{cases} x & \text{если } x > 0 \\ 0.1x & \text{иначе} \end{cases}$$

В качестве функции потерь выбирается квадратичная функция (и для координат, и для уверенности, и для оценок). Ее легко оптимизировать, однако ее использование не совсем согласовано с идеей максимизации mAP. Кроме того, для любого изображения верно утверждение о том, что во многих его ячейках (при разбиении на сетку) не содержится никаких объектов. Получается, что для таких ячеек $c \rightarrow 0$, что может привести к нестабильности модели. В этой связи вводят новые коэффициенты для их использования в функции потерь: $\lambda_{coord} = 5$ – для слагаемых, содержащих координаты центра, а также размеры рамки и $\lambda_{noobj} = \frac{1}{2}$ – для слагаемых, отвечающих за правильность классификации. Кроме того, для учёта того факта, что небольшие отклонения в размерах для больших ограничивающих прямоугольников значат меньше, чем для маленьких, модель предсказывает не высоту и ширину ограничивающих прямоугольников, а их квадратный корень.

7.7.1 Недостатки YOLO

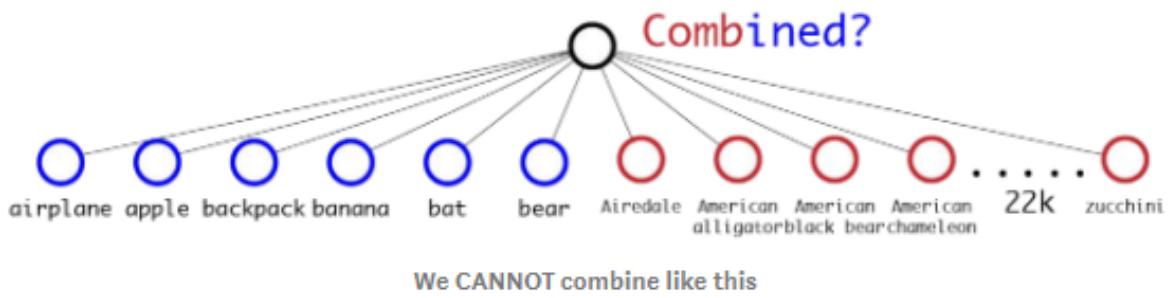
- В работе YOLO есть строгое ограничение на количество предсказываемых ограничивающих прямоугольников: для каждой ячейки сетки предсказывается только 2 прямоугольника и только 1 класса. Подобное условие накладывает ограничение на количество близких объектов, которое модель может предсказать. По этой причине сеть плохо работает с маленькими объектами, близко сгруппированными на изображении.

- Сеть учится предсказывать ограничивающие прямоугольники по данным, поэтому она плохо справляется с обобщением на случай объектов с новым или необычным соотношением сторон.
- Функция потерь одинаково учитывает ошибки, связанные с маленькими и большими ограничивающими прямоугольниками. Небольшая ошибка, связанная с большим прямоугольником, как правило, не сильно влияет на процесс обучения, а небольшая ошибка, связанная с маленьким прямоугольником, может оказаться существенное влияние на значение IoU. Поэтому основным источником ошибок модели является неправильная локализация.

7.8 YOLO9000: иерархия в классах

Так же, как и в случае с переходом от R-CNN к Fast R-CNN новая версия подхода исправляет недостатки предыдущего, при этом улучшая качество и время работы.

Поскольку ранее при обучении нейросетей на различных датасетах использовалось то количество классов, которое содержалось в выбранном для обучения датасете, некоторые обнаруженные объекты могли не попадать ни в один из классов. Для решения этой проблемы были объединены различные классы из разнообразных датасетов, причем сделано это было не с помощью простой операции объединения множеств, а иерархически (с использованием дерева слов – Word Tree – поскольку классы могут входить друг в друга: машина – это транспортное средство; см. пояснение на рис. 28).



To combine, WordTree is used:

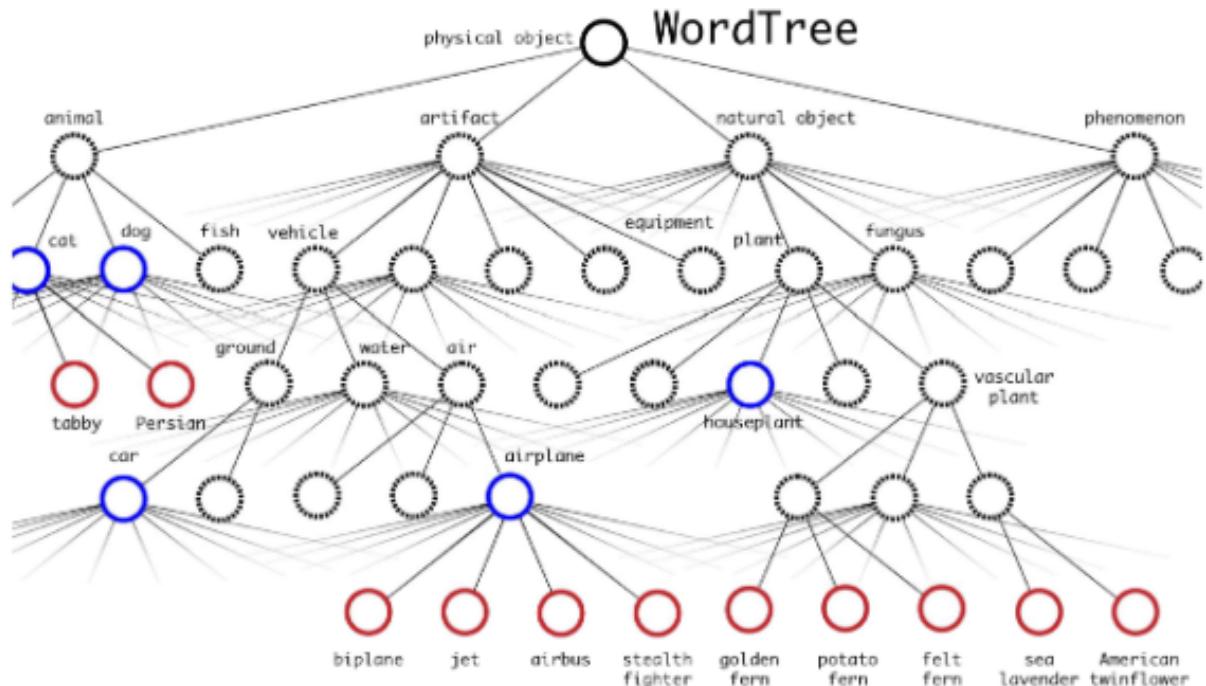


Рис. 28: Пример иерархии классов ([13])

Таким образом, получилось 9418 классов (что и дало название модификации). Помимо данного принципиального нововведения, были рассмотрены и другие поправки. Рассмотрим некоторые из них:

- Использование слоев батч-нормализации (перед всеми сверточными слоями) – что ведет к улучшению mAP на 2%;
- Увеличение разрешения: после обучения на изображениях размера 224×224 , происходит дообучение нейросети на изображениях 448×448 в течение 10 эпох (на датасете ImageNet) – что ведет к улучшению mAP на 4%;
- Замена сверточных слоев на упомянутые ранее якоря (anchor boxes) – немножко уменьшает mAP (0.3%), но значительно увеличивает recall (7%);

- Изменение размера изображений в обучающей выборке через каждые 10 эпох (для достижения разномасштабного обучения);
- Априорные центры якорей выбираются автоматически как центры кластеров, построенных методом KMeans с использованием следующего расстояния:

$$\rho(box, centroid) = 1 - IoU(box, centroid)$$

и $k = 5$ (стандартное Евклидово расстояние не подойдет, поскольку большие рамки заключают в себе большее количество ошибок, чем маленькие);

- Прямое предсказание месторасположения: использование логической активации σ , которое заключит значения в промежуток $[0, 1]$ (см. рис. 29 в обозначениях):
 - (c_x, c_y) – координаты сетки.
 - (b_x, b_y) – координаты ограничивающего прямоугольника (bounding-box): $(c_x, c_y) + \delta$, ограниченная $\sigma(t_x)$ и $\sigma(t_y)$. Увеличивает mAP на 7%.
 - (p_w, p_h) – координаты якоря (предсказанного), полученного в результате кластеризации.
 - (b_w, b_h) – размеры ограничивающего прямоугольника.
 - t_x, t_y, t_w, t_h, t_o – предсказания нейросети для каждого ограничивающего прямоугольника;
- Для лучшего качества при работе с маленькими объектами с более раннего слоя берётся карта признаков с размерами $26 \times 26 \times 512$ и преобразуется в карту признаков с размерами $13 \times 13 \times 2048$, после чего производится конкатенация с исходной картой признаков для дальнейшей детекции;
- Для извлечения признаков используется не часто используемая для этого сеть VGG16, а GoogLeNet, что позволяет снизить количество операций с плавающей точкой на проходе вперёд почти в 4 раза.

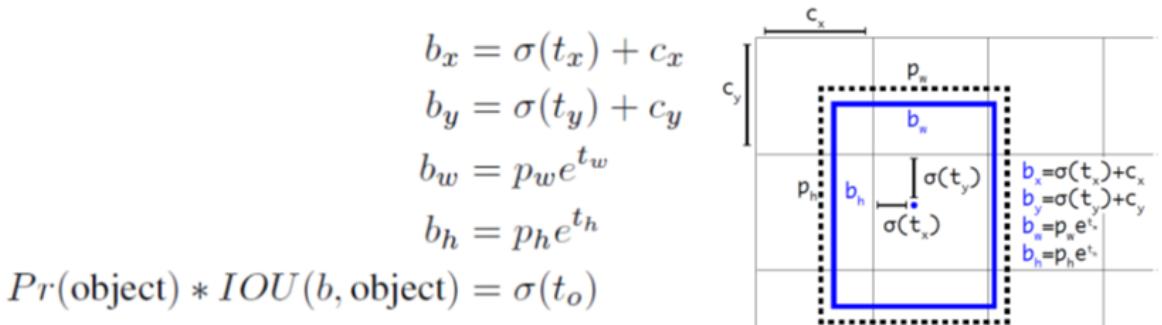


Рис. 29: Обоснование предсказания месторасположения ([13])

7.9 YOLOv3

Нововведение данной более глубокой нейросети по сравнению с YOLO и YOLOv2 заключаются в следующем:

- Было создано 3 выходных слоя для обнаружения объектов разного размера (архитектуру см. на рис. 30).
- В состав YOLOv3 входят уже 53 предобученных на ImageNet слоя. В сравнении с обычной YOLO, данная модификация лучше определяет небольшие объекты. Кроме того, для данной архитектуры так же существует уменьшенная версия, предназначенная для небольших датасетов и содержащая всего 2 выходных слоя вместо 3.
- Детектирование происходит с использованием ядра $1 \times 1 \times (B \times (5+C))$ – в предыдущих обозначениях для YOLO. Предсказание происходит в трех масштабах (scale), уменьшая размерность входного изображения до 32, 16 и 8 соответственно. Для каждого масштаба используется 3 якоря (для их генерации используется метод кластеризации K-Means).
- Квадратичная часть функции ошибок для уверенности в классе (c) и предсказании класса (C) заменяется на кросс-энтропию.
- К вероятностям классов в YOLOv3 не применяется softmax, как это было раньше. Вместо этого объектам присваивается подмножество тех классов вероятность отнесения к которым этих объектов выше заданного порога, то есть YOLOv3 поддерживает многоклассовую классификацию с пересекающимися классами.

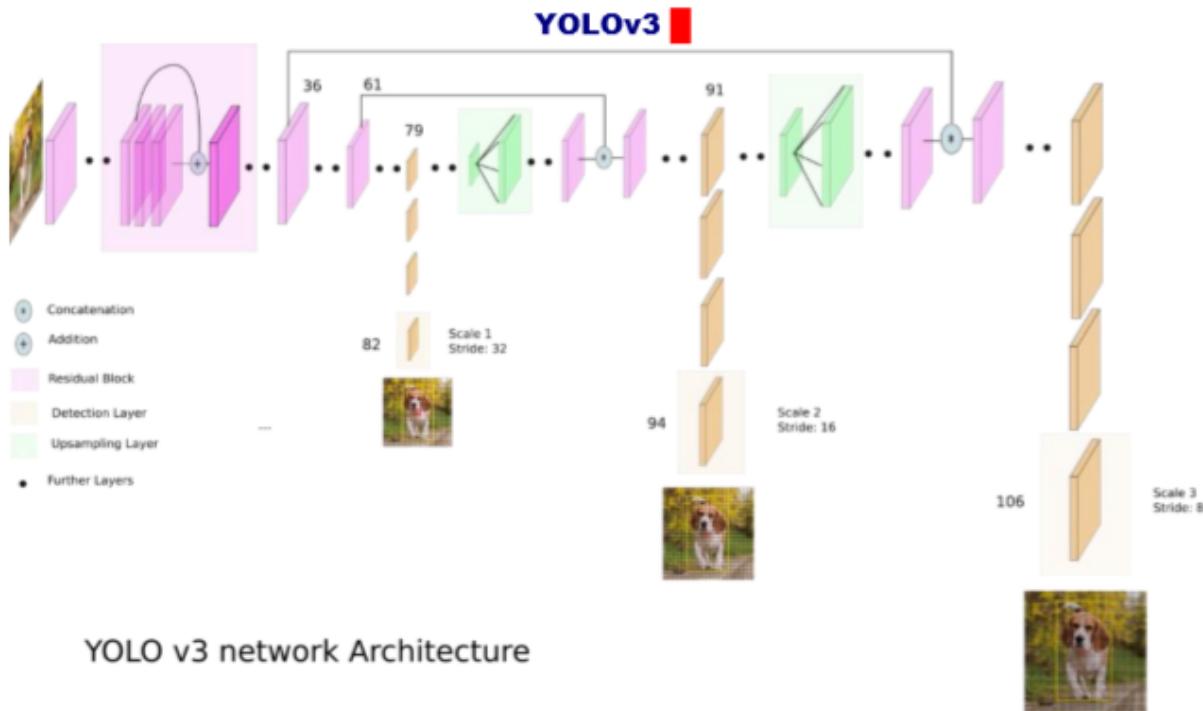


Рис. 30: Архитектура YOLOv3 ([14])

7.10 Single Shot MultiBox Detector (SSD)

Начнем рассмотрение данного метода, также, как и YOLO, используемого в реальном времени, со сравнения архитектур нейросетей данных подходов. (см. рис. 31)

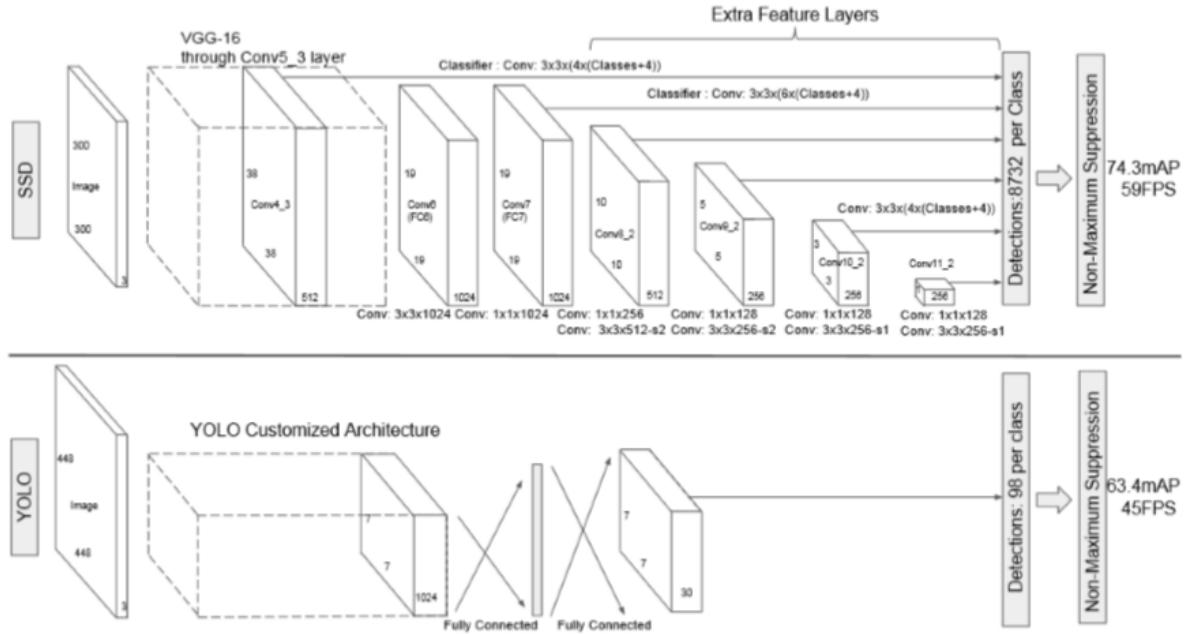


Рис. 31: Сравнение архитектур YOLO и SSD ([16])

Архитектура SSD основана на VGG-16 без использования полносвязных слоев. VGG-16 показывает высокое качество для решения задачи классификации изображений (которая является подзадачей для детектирования), поэтому в данном случае ее выбор обоснован. Вместо полносвязных слоев были добавлены несколько вспомогательных сверточных слоев для извлечения объектов в нескольких масштабах, а также уменьшения размера входных данных для каждого последующего слоя.

Сам термин «single shot multibox detector» состоит из нескольких слов, каждое из которых вносит определенный смысл для понимания работы описываемого метода в целом.

- Single shot – означает, что задача локализации объекта и классификации производится за единственный прямой проход по нейросети (как и в случае с YOLO);
- MultiBox – название определенного метода регрессии, уточняющей координаты рамки (bounding-box);
- Detector – показывает, что нейросеть является детектором объектов, которая не только локализует, но также и классифицирует их.

Для обучения SSD требуется лишь входное изображение, а также истинные позиции рамок для каждого объекта.

7.10.1 MultiBox

Как было описано ранее, MultiBox используется для уточнения координат рамки. Архитектуру сети см. на рис. 32.

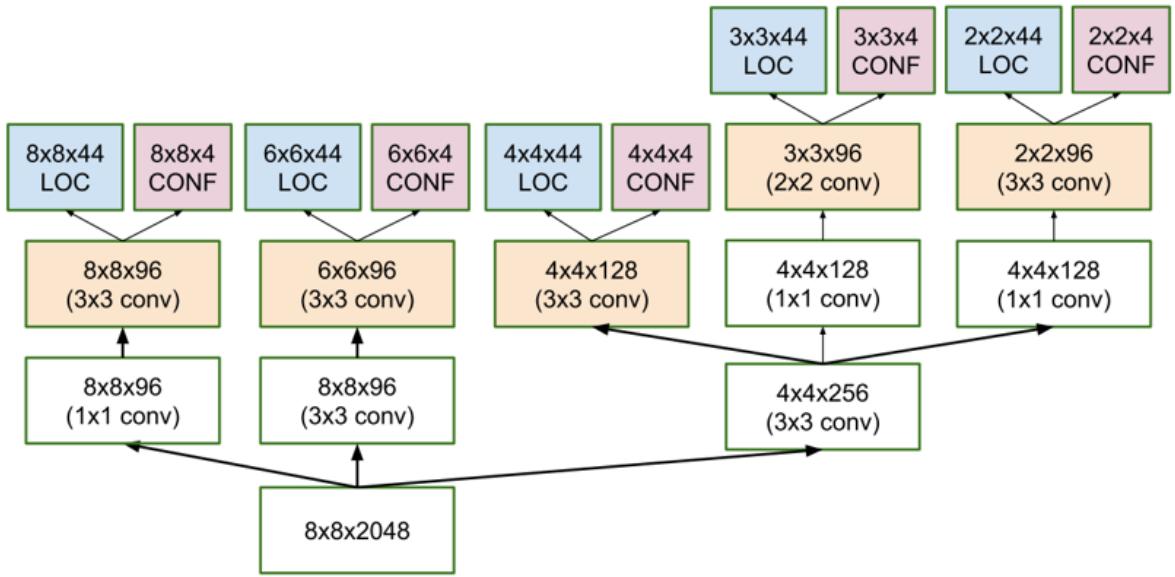


Рис. 32: Архитектура MultiBox ([18])

Присутствующие свертки 1×1 помогают уменьшить размерность с сохранением ширины и высоты изображения.

Функция потерь для MultiBox объединяет 2 важные компоненты: первая отвечает за то, насколько нейросеть уверена в том, какой именно объект расположен в выделенной области (крoss-энтропия), а вторая показывает, насколько предсказанные области расположения объектов близки к заданным (L_2 -норма). Как и в предыдущих методах, перед второй функцией потерь добавляется коэффициент сбалансированности α .

MultiBox снова взаимодействует с якорями (anchors), которые представляют собой заранее подсчитанные ограничивающие блоки фиксированного размера, которые близки к распределению реальных областей с объектами (то есть $IoU > 0.5$). Таким образом, MultiBox начинает предсказание с некоторого начального приближения, а затем пытается максимально приблизить границы рамки к истинным.

7.10.2 Описание SSD

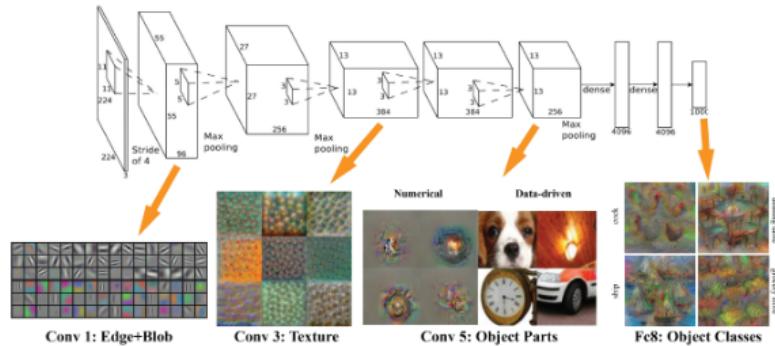


Рис. 33: Визуализация результатов VGG ([18])

В случае с самим SSD было добавлено несколько различных нововведений для улучшения способности сети локализовать и классифицировать объекты. В отличие от традиционного MultiBox, каждая карта характеристик (результат работы сверточных слоев нейросети; визуализацию см. на рис. 33) ассоциируется с набором областей разных размеров и соотношений сторон (8×8 на рис. 34(a), 4×4 на рис. 34(b)), а начальные приближения задаются более точно вручную (см. пример начальных приближений – якорей на рис. 35).

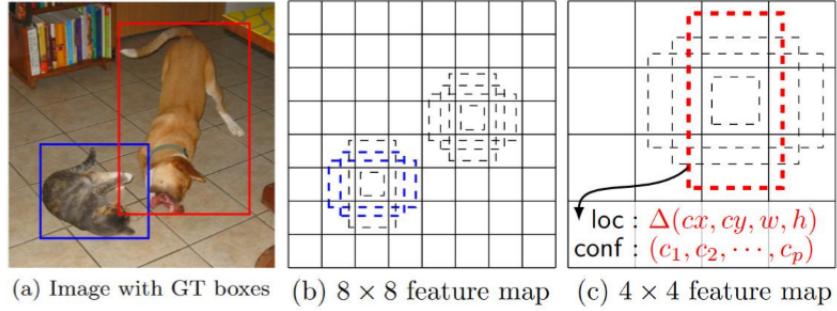


Рис. 34: Пример разделения на области ([16])

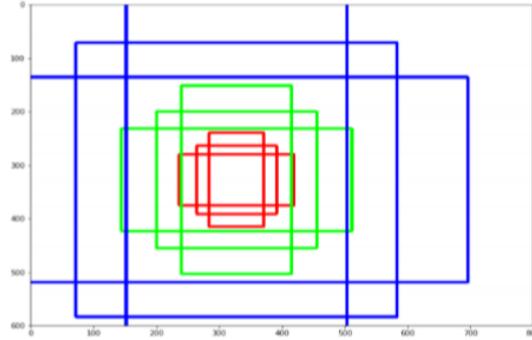


Рис. 35: Пример якорей ([1])

Это позволяет SSD взаимодействовать с входными данными (размера 300×300 – нечто среднее между ранее использовавшимися для R-CNN изображениями с небольшим разрешением 224×224 и форматом 448×448 для YOLO; кроме того, существует версия, принимающая на вход изображения размером 512×512) без предварительной подготовки. К примеру, если настроены расположения пары диагонально противоположных точек (x_1, y_1) и (x_2, y_2) , то для каждого b (максимальное число рамок, покрывающих ту или иную ячейку) и с классов при рассмотрении карты размерами $f = m \cdot n$, SSD будет вычислять $f \cdot b \cdot (4 + c)$ значений, где 4 – указанные сдвиги по отношению к реальной области. (см. рис. 34).

В качестве функции потерь для задачи локализации используется теперь L_1 (а не L_2) норма, поскольку разница в несколько пикселей при выделении рамки незаметна и непринципиальна, поэтому ее не следует штрафовать с квадратом.

В качестве функции потерь для задачи классификации используется SoftMax. Фон выделен в отдельный класс.

Кроме того для фильтрации большого количества созданных рамок используется NMS (см. рис. 36).

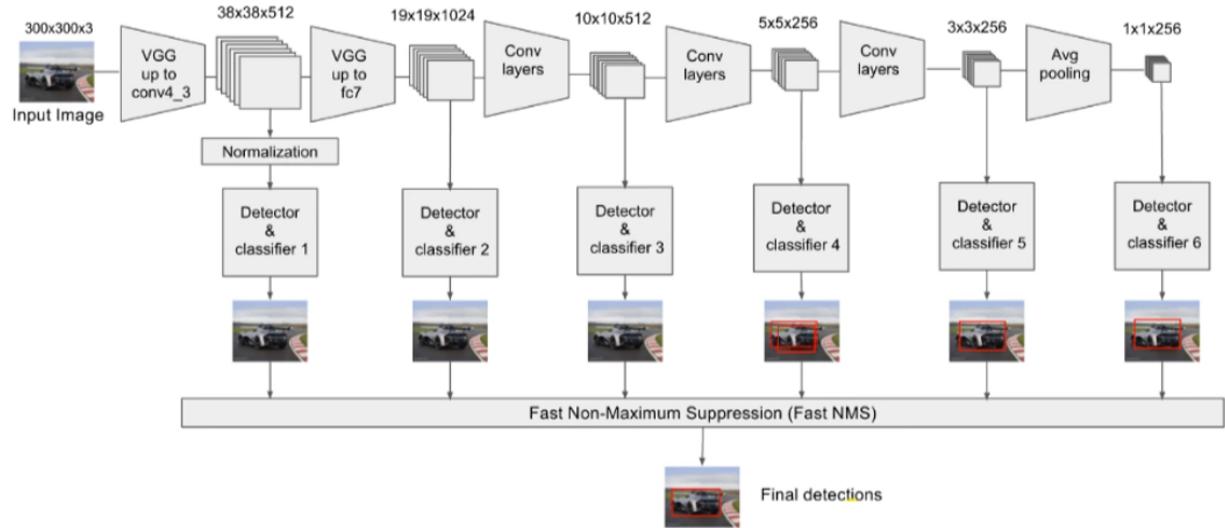


Рис. 36: Общая схема работы SSD ([1])

Общая схема работы повторена на рис. 37.

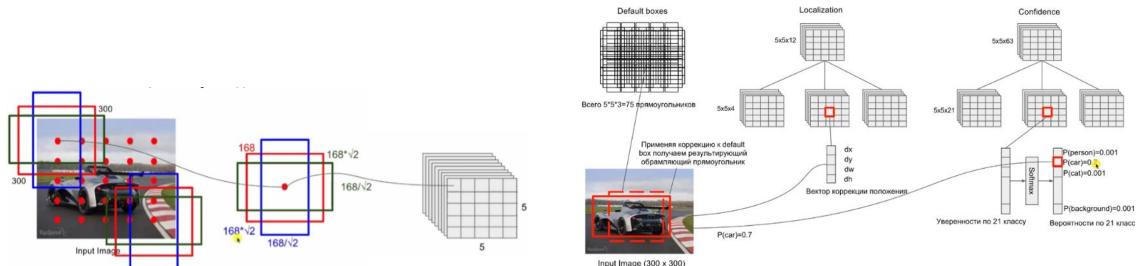


Рис. 37: Общая схема работы SSD ([1])

7.10.3 Замечания по работе SSD

В результате изучения работы данного метода были сделаны следующие промежуточные замечания:

- Большое количество default boxes на разных масштабах ведет к увеличению качества работы, хотя и незначительно сказывается на скорости (но в общем метод работает очень быстро);
- Большое количество времени (80%) тратится именно на работу VGG16, то есть при желании работу сети можно ускорить, заменив данную часть архитектуры на более легкую, либо улучшить – заменив на более точную;
- SSD путает объекты в похожих категориях из-за того, что одна и та же локация может принадлежать представителям разных классов;
- При увеличении масштаба входа SSD лучше распознает небольшие объекты;
- Стоит отметить и тот факт, что во время обучения, для того чтобы сделать модель более устойчивой к разным размерам и формам входных изображений, используются некоторые приемы аугментации данных, а именно: генерация патча таким образом, чтобы IoU составляло 0.1, 0.3, 0.5, 0.7 или 0.9, либо же случайная генерация патча.

7.11 Сравнение подходов к детектированию объектов

Результаты сравнения качества (mAP) работы описанных методов представлены на рис. 38.

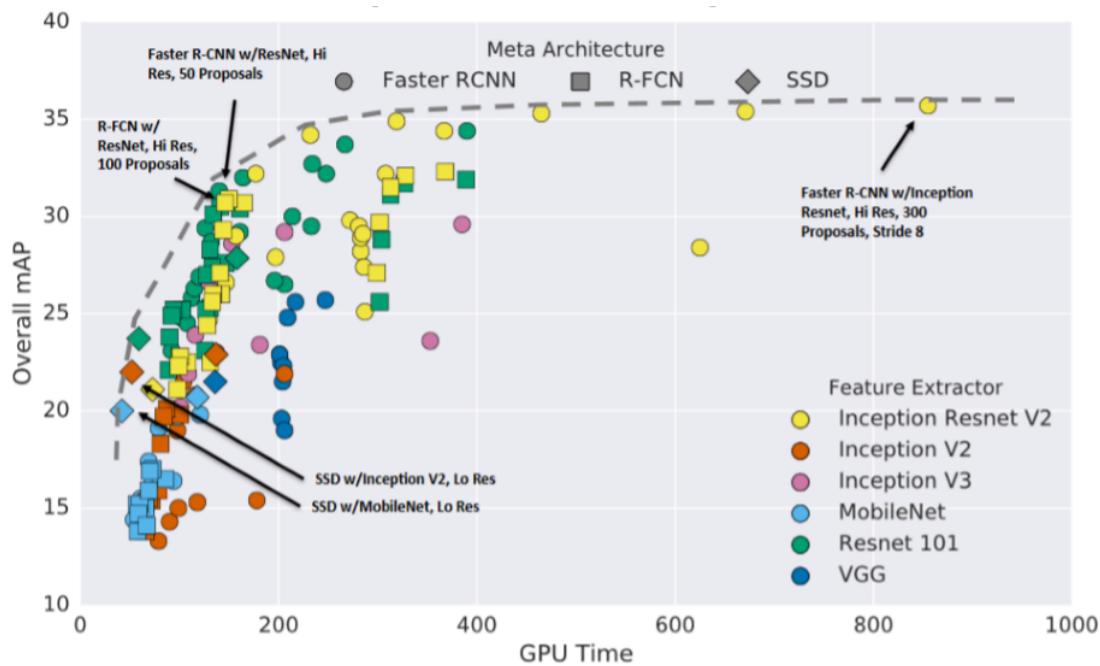


Рис. 38: Сравнения mAP описанных методов ([19])

8 Альтернативные подходы к учёту масштаба изображений

8.1 MatrixNets

Обнаружение объектов разных масштабов является одной из основных проблем задачи детектирования объектов на изображениях. MatrixNets является свёрточной нейронной сетью, которая учитывает масштаб объектов и соотношения сторон ограничивающих прямоугольников. Она состоит из матрицы слоёв (см. рис. 39). Слой $l_{1,1}$ расположен в левом верхнем углу. Слой $l_{i,j}$ имеет ширину в 2^{i-1} раз меньше, чем $l_{1,1}$, и высоту в 2^{j-1} раз меньше, чем $l_{1,1}$. Близкие к верхнему правому или левому нижнему углу матрицы моделируют объекты с ограничивающим прямоугольником, имеющим очень высокое или очень низкое соотношение сторон. Подобные объекты являются очень редкими, поэтому эти слои могут быть удалены.

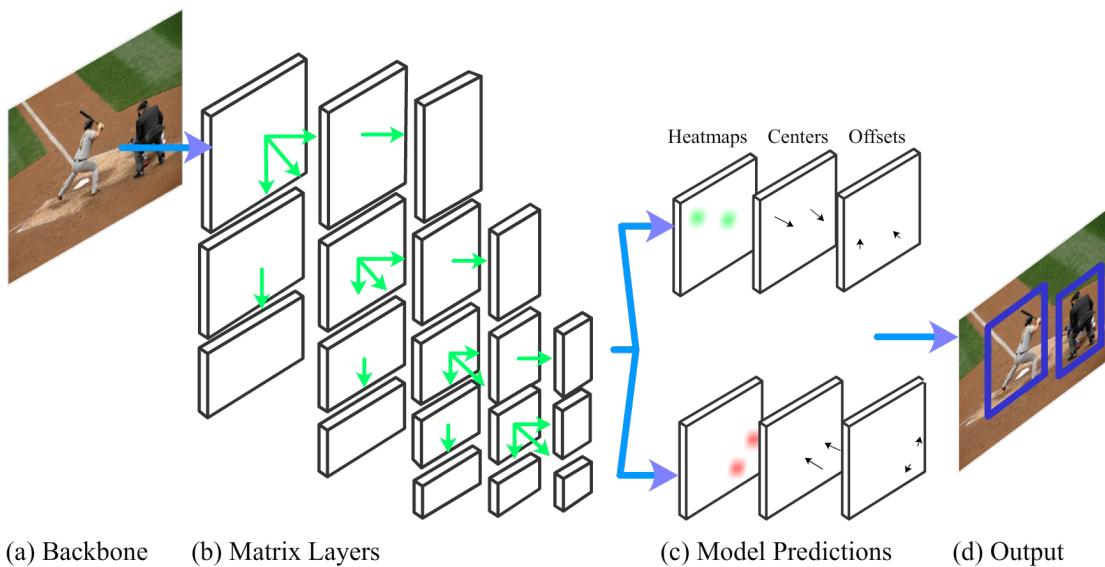


Рис. 39: Архитектура сети MatrixNets ([20])

9 Подходы к улучшению качества детектирования объектов

9.1 Многошаговая сеть с калибровочной моделью

В данном методе предлагается применить к выходу сети-детектора калибровочную модель, улучшающую качество обнаружения объектов за счёт использования контекстной информации (например, сцены на изображении). Предлагаемая модель получает на вход метки объектов, информацию о расположении и размерах ограничивающих прямоугольников, а также оценки уверенности детектора, и выдаёт уточнённый результат в таком же формате (см. рис. 40).

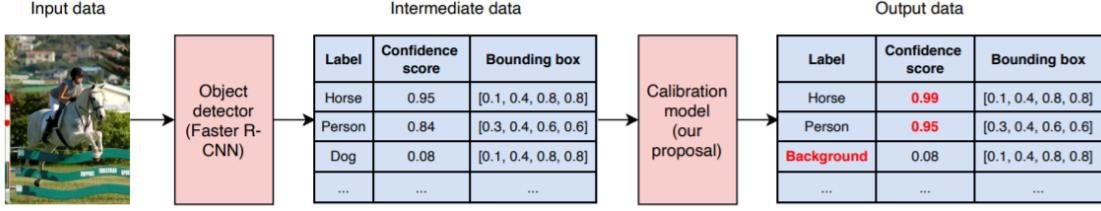


Рис. 40: Схема многошаговой модели, использующей калибровочную модель ([21])

Работа калибровочной модели основана на использовании так называемых "признаковых переменных" ("feature variables"), которые описываются на основе знаний о сфере применения детектора: форме объектов, их взаимном расположении и т.д. Признаковые переменные, отражающие взаимное расположение объектов, являются числовыми характеристиками, связывающими целевой ограничивающий прямоугольник (прямоугольник, значения метки класса которого и уверенности модели для которого необходимо обновить) и опорных ограничивающих прямоугольников, окружающих целевой прямоугольник. В следующей таблице описаны все используемые калибровочной моделью типы признаковых переменных:

$(x, y)^{\text{target, support}}_{\text{min, max, center}}$	Позиции целевого и одного из опорных ограничивающих прямоугольников
$\text{area}^{\text{target, support}}, \text{area}^{\text{ratio}}$	Площадь целевого и одного из опорных прямоугольников и отношение этих площадей к их суммам
$\text{distance}, \text{angle}$	Расстояние и угол между $(x, y)^{\text{target}}_{\text{center}}$ и $(x, y)^{\text{support}}_{\text{center}}$
$\text{iou}, \text{iou}^{\text{target, support}}$	IoU между целевым и одним из опорных прямоугольников
$n^{\text{support}}, n^{\text{overlap}}$	Количество опорных ограничивающих прямоугольников и количество пересекающихся опорных прямоугольников для каждой класса
$\text{score}^{\text{target, support}}$	Оценка уверенности для целевого и одного из опорных прямоугольников
$\text{label}^{\text{target, support}}$	Закодированные методом one-hot метки классов для целевого и одного из опорных прямоугольников
$\text{aspect}^{\text{target, support}}, \text{aspect}^{\text{global}}$	Отношения сторон целевого и одного из опорных прямоугольников и изображения
$\text{edge}^{\text{target, support}}$	Флаг, обозначающий лежит ли ограничивающий (целевой или опорный) прямоугольник на краю изображения
$\text{image}^{\text{target}}, \text{image}^{\text{global}}$	Часть изображения, ограниченная целевым прямоугольником и само изображение

В качестве детектора выбрана сеть Faster R-CNN. Получив на вход результат работы детектора, калибровочная модель вычисляет значения признаковых переменных и представляет их в виде матрицы размера $M \times N$, где M - количество признаковых переменных, N - количество опорных ограничивающих прямоугольников. В статье ([21]) детектор настроен на вывод не более 100 ограничивающих прямоугольников, поэтому

в таком случае $N = 99$, если на некотором изображении было обнаружено меньше 100 объектов, то недостающие позиции в матрице заполняются нулями. Далее матрица подаётся на свёрточный слой с 4 свёртками с размером ядра 1 и количеством выходных каналов, равным 256, 512, 1024 и 2048 соответственно. Слой с операцией max-pooling позволяет избавиться от зависимости от порядка, в котором были указаны опорные прямоугольники. Затем выход слоя подаётся в модуль, решающий задачу классификации. Архитектура модели представлена на рис. 41.

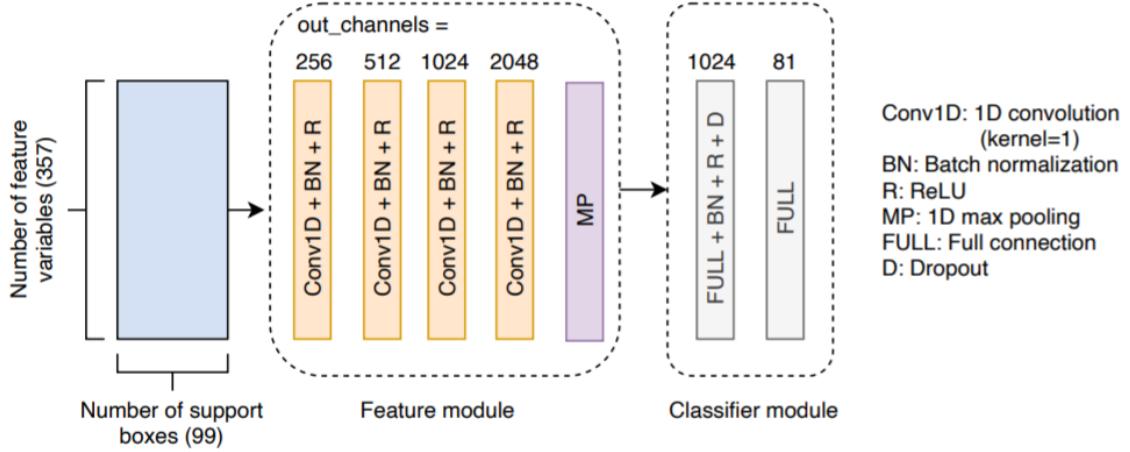


Рис. 41: Архитектура калибровочной модели ([21])

9.2 Region Decomposition and Assembly Detector (R-DAD)

В данном методе предлагается использовать для обучения сети не только на сгенерированных регионах, но и на их частях, что может улучшить качество детекции изображений с окклюзиями. Архитектура R-DAD представлена на рис. 42. Принцип работы данной сети заключается в следующем:

- Для каждого выдаваемого RPN региона $d = (x, y, w, h)$, генерируется регион $d^s = (x, y, s \cdot w, s \cdot h)$ для нескольких s . Используя большие значения s можно добавить в регион контекстную информацию (например, фон или соседние объекты), а, используя маленькие значения s можно лучше изучить детали изображения в высоком разрешении, что важно при окклюзиях.
- С помощью RoI pooling из карты признаков извлекаются регионы, соответствующие сгенерированным на предыдущем шаге ограничивающим прямоугольникам. Обозначим за x_l один из извлечённых регионов. Он делится на 4 части x_l^p , $p \in \{\text{left, right, bottom, upper}\}$.
- Далее производится свёртка $x_{i,l-1}^p$ на слое $(l-1)$ с w_{ij}^l и выход подаётся в функцию активации f , в результате чего получается карта признаков $x_{j,l}^p = f(\sum_{i=1}^{k_l} x_{i,l-1}^p * w_{ij}^l + b_j^l)$, $l = 2, 3, 4$, где $p \in \{\text{left, right, bottom, upper, l/r, b/u, comb}\}$, b_j^l - вектор сдвиг=га, k_l - количество свёрток.
- Полученные на предыдущем шаге для разных регионов x_l^p и x_l^q объединяются в $x_l^r = \max(x_l^p, x_l^q)$.

5. После вычисления x_3^{comb} на предыдущем шаге выполняется объединение $X_4^{comb} = \max(x_3^{comb}, x^{whole})$
6. x_4^{comb} подаётся на вход классификатору, а соответствующий региону x_l ограничивающий прямоугольник - на вход регрессору.

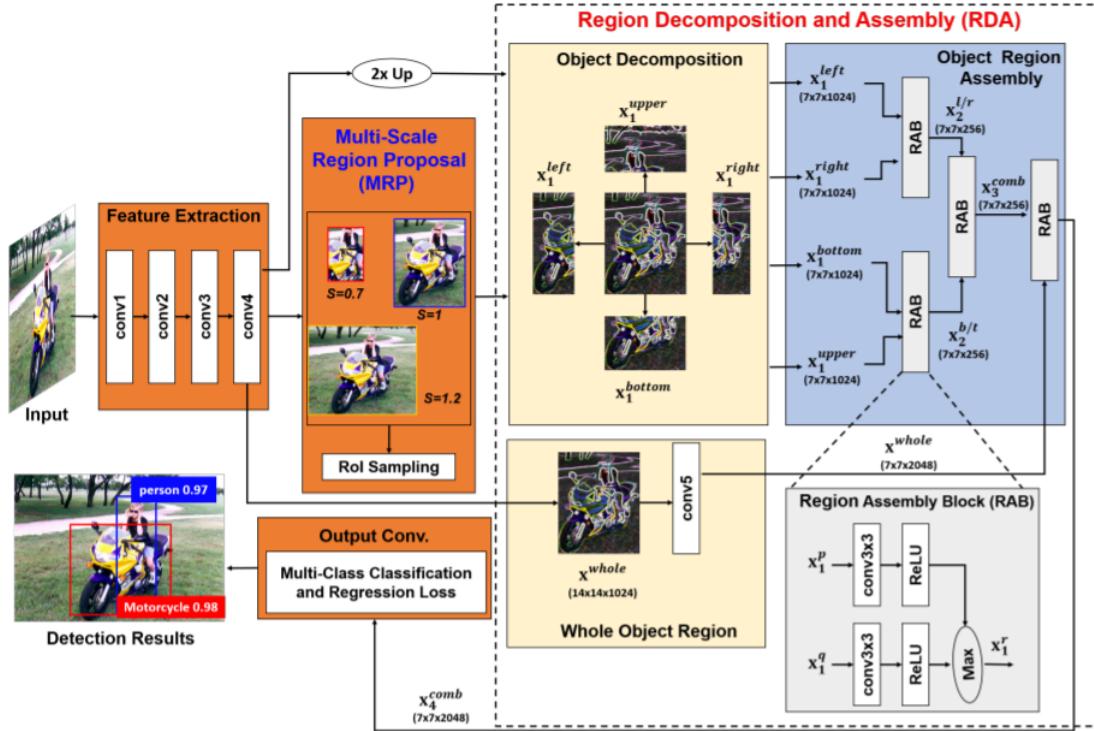


Рис. 42: Архитектура R-DAD ([22])

10 Альтернативные подходы к выбору якорных прямоугольников

10.1 Anchor pruning

В одностадийных детекторах, основанных на построении якорных прямоугольников, количество предсказываемых прямоугольников на одном изображении составляет $\sum_{i=1}^k (A_i \times H_i \times W_i)$, где H_i и W_i - высота и ширина карты признаков, A_i - количество якорных прямоугольников на одну ячейку, k - количество слоёв, генерирующих карты признаков. Пусть A - множество всех якорных прямоугольников, M - модель, обученная на наборе якорных прямоугольников $C = A$. Ниже представлен алгоритм сокращения количества якорных прямоугольников. При его описании были использованы следующие понятия: кривая Парето - множество оптимальных решений, оптимальность по Парето — такое состояние системы, при котором ни один показатель системы не может быть улучшен без ухудшения какого-либо другого показателя:

Algorithm 1 Алгоритм сокращения количества якорных прямоугольников

Input: обученная модель M с якорными прямоугольниками A

Output: кривая Парето P

```
S ← {A}
P ← {A}
while S ≠ ∅ do
    for  $a_i \in C$  do
         $C_i = C \setminus \{a_i\}$ 
        accuracy $_{C_i}$  = точность  $M$ , использующей только предсказания якорных прямо-
        угольников из  $C_i$ 
        cost $_{C_i}$  = ресурсоёмкость работы  $M$  с якорями из  $C_i$ 
        Сравнить accuracy $_{C_i}$  и cost $_{C_i}$  с accuracy $_{C_j}$  и cost $_{C_j}$  для всех  $C_j \in P$ 
        if  $C_i$  оптимальный по Парето набор для  $P$  then
            Добавить  $C_i$  в  $S$  и  $P$ 
        end if
        Удалить из  $P$  все  $C_i$ , которые больше не являются оптимальными по Парето
    end for
end while
return P
```

10.2 FreeAnchor

В данном методе не накладывается ограничение на значение IoU для якорных прямоугольников. Соответствие между объектами и якорными прямоугольниками моделируется путём формулировки процесса обучения детектора как процедуры максимизации правдоподобия.

Для работы метода используется одностадийный детектор. Пусть используемая для его обучения функция потерь задаётся формулой:

$$L(\theta) = \sum_{a_j \in A_+} \sum_{b_i \in B} C_{ij} L_{ij}^{cls}(\theta) + \beta \sum_{a_j \in A_+} \sum_{b_i \in B} C_{ij} L_{ij}^{loc}(\theta) + \sum_{a_j \in A_-} L_j^{bg}(\theta)$$

В данной формуле $A_+ \subset A = \{a_j = (a_j^{cls}, a_j^{loc})\}$ - множество якорных прямоугольников на изображении, которые содержат объект, где A - множество всех якорных прямоугольников на изображении, a_j^{cls} - предсказанный класс для объекта, лежащего в якорном прямоугольнике a_j , $a_j^{loc} = (x, y, w, h)$ - координаты, ширина и высота a_j соответственно; A_- - множество якорных прямоугольников на изображении, которые не содержат объект; $B = \{b_i = (b_i^{cls}, b_i^{loc})\}$ - множество истинных ограничивающих прямоугольников на изображении; $C_{ij} = 1$, если якорный прямоугольник a_j соответствует истинному ограничивающему прямоугольнику b_i , и 0 иначе; θ - обучаемые параметры сети; β - параметр регуляризации; $L_{ij}^{cls} = BCE(a_j^{cls}, b_i^{cls}, \theta)$ - функция потерь в задаче классификации, бинарная кросс-энтропия; $L_{ij}^{loc}(\theta) = \text{smooth}L_1(a_j^{loc}, b_i^{loc}, \theta)$ - функция потерь в задаче локализации; $L_j^{bg}(\theta) = BCE(a_j^{cls}, \vec{0}, \theta)$ - функция потерь в задаче классификации для якорных прямоугольников, не содержащих объектов. Минимизация функции потерь $L(\theta)$ эквивалентна максимизации правдоподобия, заданного формулой:

$$P(\theta) = e^{(-L(\theta))} = \prod_{a_j \in A_+} \sum_{b_i \in B} C_{ij} P_{ij}^{cls}(\theta) \prod_{a_j \in A_+} \sum_{b_i \in B} C_{ij} P_{ij}^{loc}(\theta) \prod_{a_j \in A_-} P_j^{bg}(\theta)$$

В данной формуле $P_{ij}^{cls}(\theta)$ и $P_j^{bg}(\theta)$ выражают уверенность классификации, а $P_{ij}^{loc}(\theta)$ - уверенность локализации.

Чтобы обучить детектор выбирать оптимальные якорные прямоугольники, необходимо модифицировать $P(\theta)$:

$$P'(\theta) = P_{recall}(\theta) \times P_{precision}(\theta) = \prod_i \max_{a_j \in A_i} (P_{ij}^{cls}(\theta) P_{ij}^{loc}(\theta)) \times \prod_j (1 - P(a_j \in A_-)(1 - P_j^{bg}(\theta)))$$

P_{recall} задаётся формулой:

$$P_{recall}(\theta) = \prod_i \max_{a_j \in A_i} (P_{ij}^{cls}(\theta) P_{ij}^{loc}(\theta))$$

Здесь A_i - подмножество якорных прямоугольников, имеющих наибольшее значение IoU с прямоугольником b_i . Смысл формулы для P_{recall} заключается в необходимости гарантировать существование для каждого истинного прямоугольника $b_i \in B$ якорного прямоугольника $a_j \in A_i$ такого, что значения a_j^{cls} и a_j^{loc} довольно близки к b_i^{cls} и b_i^{loc} соответственно.

$P_{precision}$ задаётся формулой:

$$P_{precision}(\theta) = \prod_j (1 - P(a_j \in A_-)(1 - P_j^{bg}(\theta)))$$

Здесь $P(a_j \in A_-) = 1 - \max_i P(a_j \rightarrow b_i) = 1 - \text{Saturated linear}(\text{IoU}(a_j, b_i), t, \max_j(\text{IoU}(a_j, b_i)))$, где t - порог, необходимый для совместимости с NMS, Saturated linear задаётся формулой:

$$\text{Saturated linear}(x, t_1, t_2) = \begin{cases} 0 & \text{if } x \leq t_1 \\ \frac{x-t_1}{t_2-t_1} & , t_1 < x < t_2 \\ 1 & , x \geq t_2 \end{cases}$$

11 Заключение

Итак, в результате детального изучения задачи детектирования объектов на изображении, а также ее всевозможных решений были сделаны следующие выводы:

- В результате развития технологий произошел переход от смешанных решений (частично нейросетевых и частично содержащих методы машинного обучения, такие как SVM) к полностью нейросетевым. Именно такие методы показывали наилучшие результаты для решения поставленной задачи не только в плане точности, но и в смысле скорости решения;
- Локализация объектов сама по себе не обязательно должна быть классориентированной, поскольку за определение, что конкретно находится в той или иной области отвечает классификация;
- Существует большое количество средств, позволяющих работать с изображениями различных размеров;
- В одном регионе могут находиться (следовательно, и должны детектироваться) несколько объектов;
- За время существования и изучения конкретной задачи детектирования произошел переход от оффлайн- к онлайн-методам обнаружения предметов;
- Несмотря на большое количество изобретенных решений, задача детектирования представляет собой перспективную область для изучения, поскольку для ее решения могут подойти различные уже существующие для других задач архитектуры нейронных сетей.

Список литературы

- [1] Александр Дьяконов, "Детектирование объектов на изображениях" (текст лекции)
- [2] "Panoptic Segmentation — The Panoptic Quality Metric <https://medium.com/@danielmechea/panoptic-segmentation-the-panoptic-quality-metric-d69a6c3ace30>
- [3] Uijlings J. R. R. et al., "Selective search for object recognition International journal of computer vision, Т. 104, №2, 2013, pp. 154-171
- [4] Girshick R. et al., "Rich feature hierarchies for accurate object detection and semantic segmentation Proceedings of the IEEE conference on computer vision and pattern recognition, 2014, pp. 580-587.
- [5] "Mask R-CNN: архитектура современной нейронной сети для сегментации объектов на изображениях <https://habr.com/ru/post/421299/>
- [6] Курс лекций по компьютерной графике кафедры ИИТ, <https://yadi.sk/d/T5U9p9xwpoB4qQ>
- [7] "Non-maximum Suppression (NMS) <https://towardsdatascience.com/non-maximum-suppression-nms-93ce178e177c>
- [8] "Canny edge detector https://en.wikipedia.org/wiki/Canny_edge_detector
- [9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, "Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition arXiv preprint arXiv: 1406.4729, 2014
- [10] Girshick R., "Fast R-CNN Proceedings of the IEEE international conference on computer vision, 2015, pp. 1440-1448
- [11] Ren S. et al., "Faster R-CNN: Towards real-time object detection with region proposal networks Advances in neural information processing systems, 2015, pp. 91-99.
- [12] Redmon J. et al., "You only look once: Unified, real-time object detection Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 779-788
- [13] Redmon J., Farhadi A., "YOLO9000: better, faster, stronger Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 7263-7271
- [14] Redmon J., Farhadi A., "YOLOv3: An incremental improvement arXiv preprint arXiv: 1804.02767, 2018
- [15] "Review: YOLOv2 & YOLO9000 — You Only Look Once (Object Detection) <https://towardsdatascience.com/review-yolov2-yolo9000-you-only-look-once-object-detection-7883d2b02a65>
- [16] Liu W. et al., "SSD: Single shot multibox detector European conference on computer vision Springer, Cham, 2016, pp. 21-37
- [17] "SSD — Single Shot Detector <https://towardsdatascience.com/review-ssd-single-shot-detector-object-detection-851a94607d11>

- [18] "Understanding SSD MultiBox — Real-Time Object Detection In Deep Learning <https://towardsdatascience.com/understanding-ssd-multibox-real-time-object-detection-in-deep-learning-495ef744fab>
- [19] Huang J. et al., "Speed/accuracy trade-offs for modern convolutional object detectors Proceedings of the IEEE conference on computer vision and pattern recognition, 2017
- [20] Rashwan A. et al., "Matrix Nets: A New Deep Architecture for Object Detection arXiv preprint arXiv: 1908.04646, 2019
- [21] Tomoe Kishimoto et al., "An Improvement of Object Detection Performance using Multi-Step Machine Learnings arXiv preprint arXiv: 2101.07571, 2021
- [22] Seung-Hwan Bae, "Object Detection based on region Decomposition and Assembly arXiv preprint arXiv: 1901.08225, 2019
- [23] Bonnaerens M., Freiberger M., Dambre J., "Anchor Pruning for Object Detection arXiv preprint arXiv: 2104.00432, 2021
- [24] Xiaosong Zhang et al., "FreeAnchor: Learning to Match Anchors for Visual Object Detection arXiv preprint arXiv:1909.02466, 2019