|  |  |
| --- | --- |
| **СОГЛАСОВАНО**  Доцент кафедры  ИАНИ ННГУ, к.ф.-м.н.  \_\_\_\_\_\_\_\_\_\_\_\_\_ Д.А. Яшунин  «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_2019 г. | **УТВЕРЖДАЮ**  Профессор кафедры  ИАНИ ННГУ, д.т.н.  \_\_\_\_\_\_\_\_\_\_\_\_\_ Н.В. Старостин  «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_2019 г. |

**Пояснительная записка № 2**

**«ПЗ по методам детекции лиц (ПЗ2)»**

**Этап 1. Подготовка обзоров на существующие подходы к решению   
задачи и построение технологического стека**

**НИР «Разработка и реализация программного обеспечения   
для распознавания лиц на групповых фотографиях»**

**(Шифр ПО «AFR»)**

Ответственный исполнитель

\_\_\_\_\_\_\_\_\_\_\_\_ М.М. Годовицын

«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_2019 г.

**Н. Новгород 2019**

Оглавление

[1 Введение 3](#_Toc4616921)

[2 Regional Convolution Neural Network (RCNN) 3](#_Toc4616922)

[3 Fast Regional Convolution Neural Network (Fast RCNN) 5](#_Toc4616923)

[4 Faster Regional Convolution Neural Network (Faster RCNN) 6](#_Toc4616924)

[5 You Only Look Once (YOLO) 7](#_Toc4616925)

[6 Single-Shot Detector (SSD) 8](#_Toc4616926)

[7 Выводы 9](#_Toc4616927)

[Список используемых источников 11](#_Toc4616928)

# 

# Введение

Задача детекции лица человека на изображении является первым шагом в решении стоящей перед нами задачи его распознавания. Необходимо выбрать алгоритм её решения, а для этого провести анализ существующих методов, осветив слабые и сильные их места, ограничения, производительность. Рассмотрены не все, но многие методы детекции, удовлетворяющие ограничениям разрабатываемого ПО «AFR».

# Regional Convolution Neural Network (RCNN)

Рисунок 1. Схема работы RCNN.

**Входные данные.**

Область изображения, представленная вектором размерности 3x227x227, найденная селективным поиском.

**Выходные данные.**

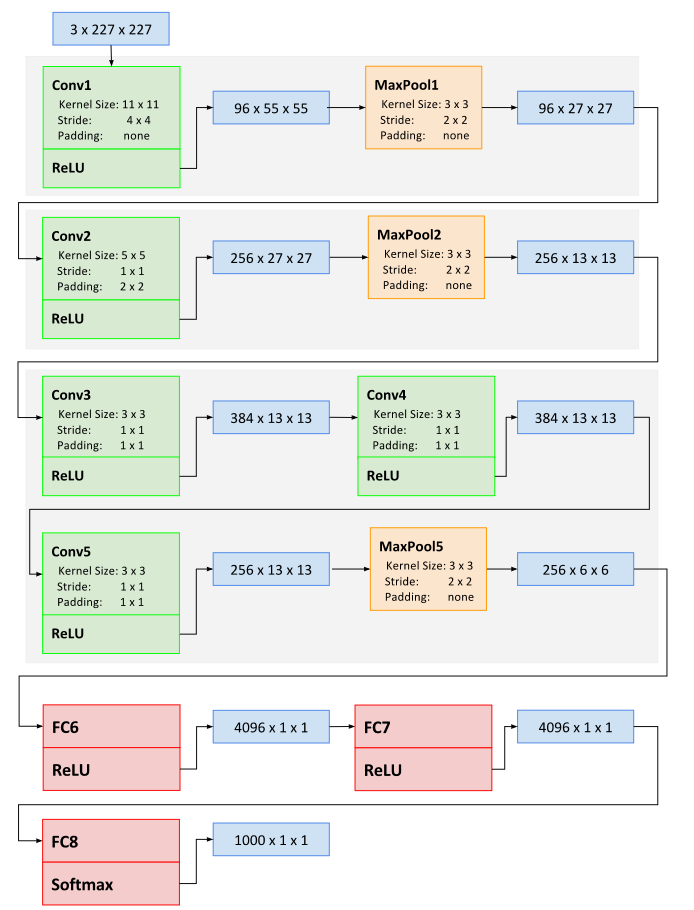
Координаты найденных ограничительных прямоугольников, содержащих искомый объект, и результаты их классификации.

Рисунок 2. Архитектура RCNN.

**Производительность.**

50 секунд на изображение (GPU)

**Алгоритм работы.**

С помощью селективного поиска на входном изображении выделяются области (ограничительные рамки), предположительно содержащие искомый объект (гипотезы). Похожие гипотезы объединяются вместе жадным алгоритмом. Далее трансформируются в квадрат (3x227x227) и подаются на вход свёрточной нейронной сети, которая генерирует карту признаков, представленную вектором (4096x1x1). Далее производится классификация либо с использованием метода опорных векторов, либо с помощью softmax слоя (в таком случае получим ускорение, но точность может снизиться в среднем на 4%). Потом с помощью линейного регрессора уточняется местоположение ограничительной рамки с помощью прямой регрессии координат.

**Комментарий.**

Занимает много времени: алгоритм поиска областей является фиксированным (нет элемента обучения).



Рисунок 3. Селективный поиск.

# Fast Regional Convolution Neural Network (Fast RCNN)

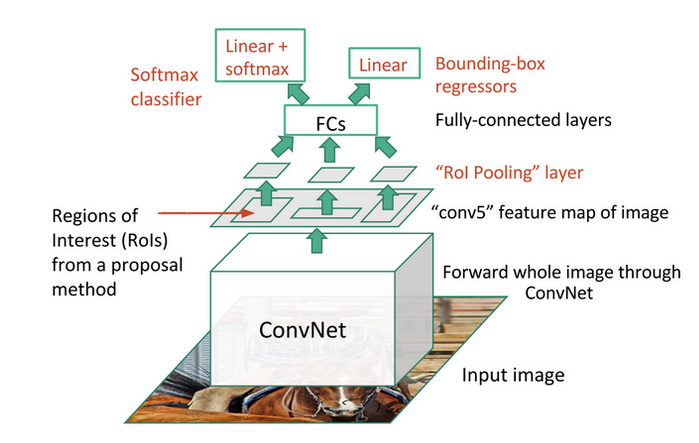
****

Рисунок 4. Схема работы Fast RCNN.

**Входные данные.**

Целое изображение (сеть самостоятельно формирует карту признаков).

**Выходные данные.**

Координаты найденных ограничительных прямоугольников, содержащих искомый объект, и результаты их классификации.

**Производительность.**

2 секунды на изображение (GPU)

Быстрее, потому что операция свёртки выполняется только один раз для каждого изображения, а не каждый раз для каждой области.

**Алгоритм работы.**

В свёрточную сеть подаются не гипотезы, а входное изображение для создания свёрточной карты признаков. Далее гипотезы проецируется на эту карту признаков и перемасштабируются (3x227x227). Далее следуют полносвязные слои, которые для каждой гипотезы вычисляют класс и смещение с помощью прямой регрессии координат ограничительных прямоугольников.

# Faster Regional Convolution Neural Network (Faster RCNN)

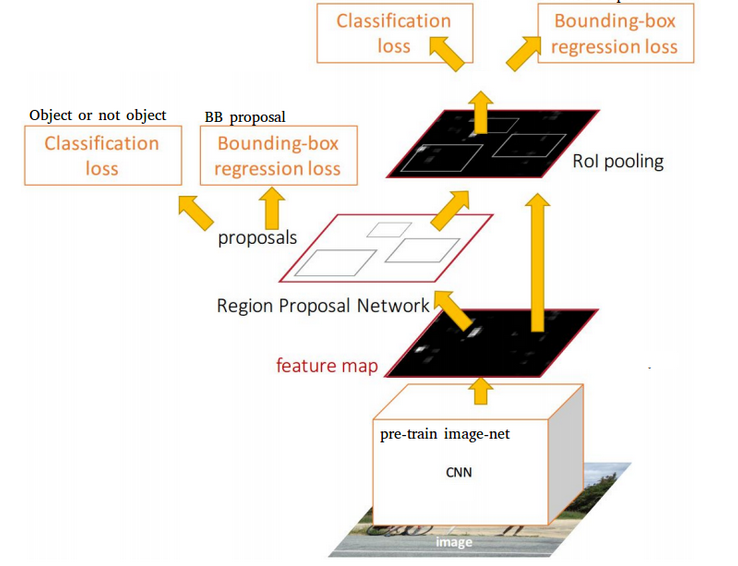


Рисунок 5. Схема работы Faster RCNN.

**Входные данные.**

Целое изображение.

**Выходные данные.**

Координаты найденных ограничительных прямоугольников, содержащих искомый объект, и результаты их классификации.

**Производительность.**

0.2 секунды на изображение.

**Алгоритм.**

В свёрточную сеть подаётся изображение, затем создаётся свёрточная карта признаков. Затем к этой карте применяются сверточный слой и два полносвязанных слоя для генерации гипотез (Regions Proposal Network). После этого полученные с её помощью ограничивающие прямоугольники используются для вырезания из исходной свёрточной карты соответствующих найденному объекту признаков.

Выходы же с этих слоёв базируются на так называемых *«якорях»*, представляющих собой k рамок для каждого положения скользящего окна, которое просматривает карту признаков. Якоря имеют разные размеры и соотношения сторон.

Слой RoI (Region of Interest) pooling принимает на вход карту признаков и гипотезы в координатах изображения. RoI преобразует координаты изображения в координаты на карте признаков. Полученный прямоугольник разбивается на сетку с заданными параметрами ( например, для VGG16 сетка 7x7), и затем производится max pooling по каждой из ячеек. Затем данные подаюся на softmax слой для оценки принадлежности преендента одному из классов объектов и слой регрессии для уточнения координат ограничительных рамок.

# You Only Look Once (YOLO)

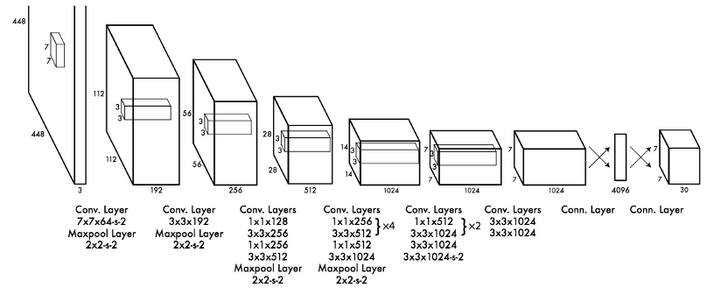


Рисунок 6. Архитектура YOLO.

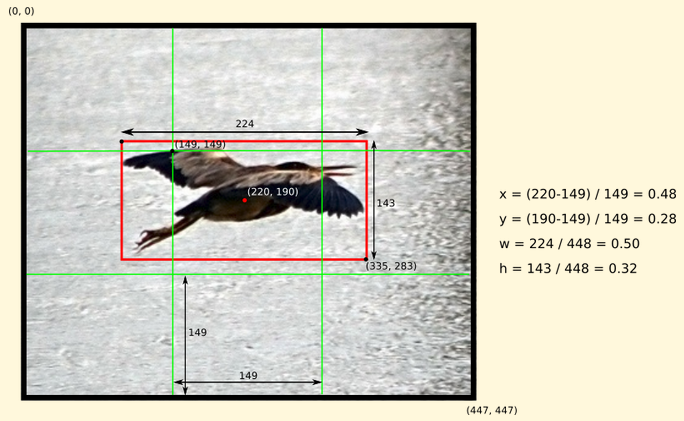
**Входные данные.**

Целое изображение.

**Выходные данные.**

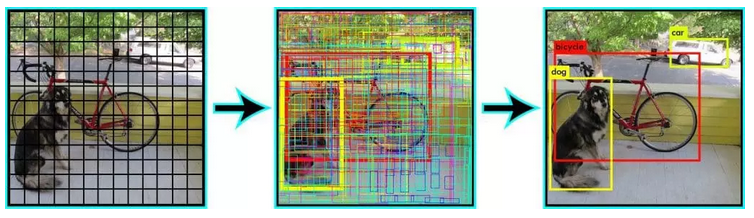
Тензор N\*N(A+B\*5), определяющий прогноз для каждой ячейки сетки, где A — оценка вероятности соответствия каждому классу, B — число рамок в ячейке, каждой рамке соответствуют 5 значений: координаты центра, ширина и высота (нормализованы в интервале [0,1]), а также её точность (=Pr(Object) \* IOU(pred, truth); если в рамке нет объектов, точность будет равна нулю). Подробнее об этом можно узнать, прочитав раздел «алгоритм».

Рисунок 7.Пример вычисления координат рамки для изображения 448x448 пикселей, N=3



**Производительность.**

55 кадров/с.



**Алгоритм.**

Делит изображение на сетку NxN, каждая ячейка сетки формирует B ограничительных рамок и значение её точности, т.е. насколько вероятно содержание в ней объекта (независимо от класса). Из сформированных NxNxB рамок часть может быть убрана из рассмотрения, если их точность не соответствует установленному порогу. Далее с помощью прямой регрессии координат уточняется положение ограничительных прямоугольников.

**Комментарий.**

YOLO Сталкивается с трудностями при детектировании объектов малого размера

Сеть имеет 24 свёрточных слоя и два полносвязных. Существует облегчённая её версия, Fast YOLO, с 9 свёрточными слоями и меньшим количеством фильтров. Работает она ещё быстрее (150 кадров/с), однако точность её значительно ниже (примерно на 8%).

**Модификации.**

1. YOLO v2 (YOLO9000)  
   Рассчитывается смещение *«якорей»* (описаны выше для Faster RCNN) с целью уточнения положения ограничительных прямоугольников.  
   Решена проблема детекции небольших объектов: в YOLOv2 изображение делится на сетку размером 13\*13.  
   Сеть YOLOv2 тренируется на изображениях с различным разрешением от 320\*320 до 608\*608, что позволяет находить объекты на изображениях с различными разрешениями с более высокой точностью.
2. YOLOv3  
   Используются логистические классификаторы вместо softmax слоя, что позволяет использовать мульти-метки (например, если для одного ограничительного прямоугольника вероятность, что внутри него содержится животное, будет равна 0.89, а вероятность того, что в нём содержится собака, равна 0.83, то для этого прямоугольника объект будет обозначен одновременно и как животное, и как собака)

# Single-Shot Detector (SSD)

**Входные данные.**

Изображение фиксированного размера (300x300 для SSD300; максимальное разрешение — 512x512 для SSD512)

**Выходные данные.**

Карта объектов: каждое место этой карты хранит информацию об объекте (класс), который здесь содержится, и об ограничивающей рамке. Например, SSD512 выдаёт 7 карт объектов с разрешениями 64x64, 32x32, 16x16, 8x8, 4x4, 2x2, и 1x1. Для SSD300 информацию о выходных данных можно увидеть на рисунке ниже.

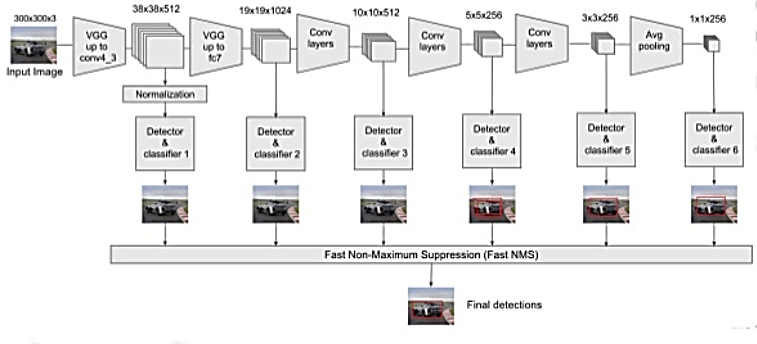


Рисунок 8. Схема работы SSD.

**Производительность.**

45 кадров/с.

**Алгоритм.**

Рассчитывается смещение *«якорей»* с целью уточнения положения ограничительных прямоугольников. Изображение обрабатывается сетью с различными фильтрами (10x10, 5x5, 3x3). Карты признаков, получаемые в разных частях сети, обрабатываются фильтрами 3x3 и формируют ограничительные рамки (подобно Faster RCNN). Каждая рамка имеет пять параметров: координаты центра, ширину, высоту, а также вектор соответствия каждому классу (подобно YOLO).

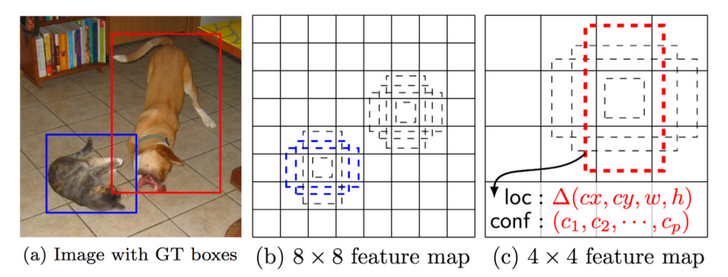


Рисунок 9. Поиск координат ограничительных прямоугольников с помощью якорей в SSD.

# Выводы

YOLO и SSD несомненно демонстрируют хорошее соотношение между скоростью и точностью, но содержат в себе существенный недостаток, который, возможно, не так уж и критичен для нашей задачи: трудности с обнаружением маленьких объектов. Пока не ясно, насколько далеки будут от камеры лица, которые необходимо будет обнаружить.

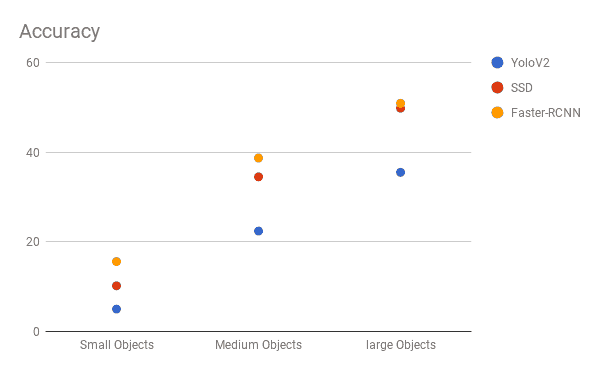
Faster RCNN пусть и работает медленнее, но показывает более высокую точность. Время обработки изображения этим алгоритмом вполне укладывается в ограничения.

Ниже приведены результаты сравнения описанных методов на датасете VOC2007 с помощью метрики mAP (Рис. 10) и на датасете COCO с помощью метрик precision, recall, IoU (Рис. 11).

Рисунок 10. Сравнение методов на датасете VOC2007.



Figure 11. Сравнение методов на датасете COCO.



# Список используемых источников

1. <https://medium.com/@smallfishbigsea/faster-r-cnn-explained-864d4fb7e3f8>
2. <https://medium.com/@smallfishbigsea/understand-ssd-and-implement-your-own-caa3232cd6ad>
3. <https://medium.com/diaryofawannapreneur/yolo-you-only-look-once-for-object-detection-explained-6f80ea7aaa1e>
4. <https://towardsdatascience.com/fasterrcnn-explained-part-1-with-code-599c16568cff>
5. <https://courses.engr.illinois.edu/cs445/fa2015/projects/final/object-detection-video.pdf>
6. <https://medium.com/zylapp/review-of-deep-learning-algorithms-for-object-detection-c1f3d437b852>
7. <https://cv-tricks.com/object-detection/faster-r-cnn-yolo-ssd/>
8. https://arxiv.org/pdf/1512.02325.pdf