|  |  |
| --- | --- |
| **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 № 1**

**«Метод детекции лиц на фотографии с использованием OpenVINO Deep Learning Framework и сверточной нейронной сети Faster R-CNN»**

**Этап 2. Подготовка обзора выбранных подходов к решению   
задачи и построение технологического стека**

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

**(Шифр ПО «FaceRecognition»)**

Ответственный исполнитель

\_\_\_\_\_\_\_\_\_\_\_\_ A.C. Филатова

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

**Н. Новгород 2019**

В пояснительной записке в рамках этапа обзора выбранных подходов к решению задачи «FaceRecognition» и построения технологического стека представлено описание сверточной нейронной сети Faster R-CNN – ее архитектуры, процесса тренировки и описание функций потерь; демонстрация процесса детекции объектов на фотографии с использованием фреймворка OpenVINO; подробное описание этапа детекции лиц в рамках ПО «FaceRecognition».

Оглавление

[1 Описание сети Faster R-CNN 4](#_Toc4359330)

[1.1 Архитектура сети Faster R-CNN 4](#_Toc4359331)

[1.2 Процесс тренировки сети 6](#_Toc4359332)

[1.3 Процесс тренировки RPN 7](#_Toc4359333)

[1.4 Функции потерь для сети 9](#_Toc4359334)

[2 Демонстрация примера обнаружения предметов на фотографии с помощью Faster R-CNN и OpenVINO Toolkit 9](#_Toc4359335)

[2.1 Загрузка и преобразование Caffe\* Model 9](#_Toc4359336)

[2.2 Запуск 10](#_Toc4359337)

[2.3 Выходные данные 10](#_Toc4359338)

[Список используемых источников 11](#_Toc4359339)

# Описание сети Faster R-CNN

## Архитектура сети Faster R-CNN

Сеть Faster R-CNN (рис.1,2) состоит из двух модулей. Первый модуль представляет собой глубокую полностью сверточную сеть, которая предлагает регионы (замена процедуры генерации претендентов selective search в других моделях R-CNN, так как это занимало достаточно много времени), а второй модуль представляет собой Fast R-CNN детектор, который использует предложенные регионы. Вся система представляет собой единую унифицированную сеть для обнаружения объектов.

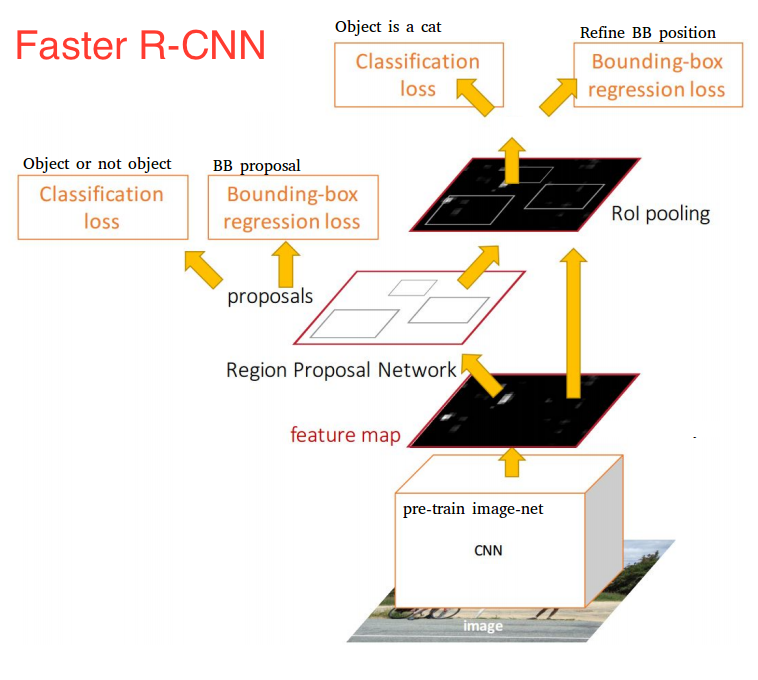
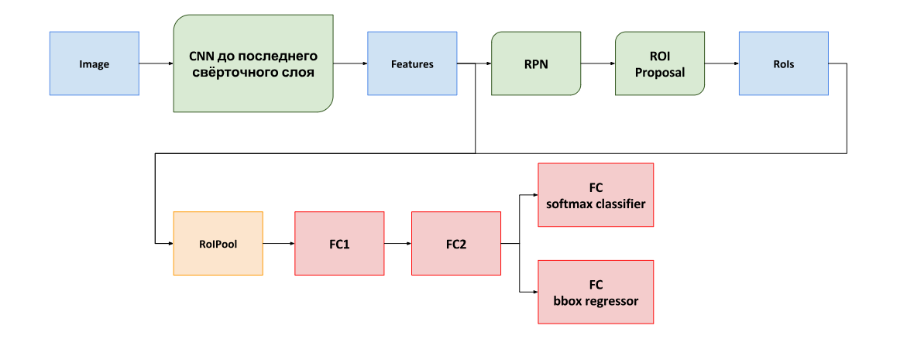


Рис.2

Рис.1

В качестве свёрточной сети авторы предлагают использовать либо [VGG16](https://vbystricky.github.io/images/2017-06/vgg16.svg) (Simonyan and Zisserman model), либо ZF (Zeiler and Fergus model). В дальнейшем при рассмотрении для определенности будем рассматривать случай VGG16 (для ZF ситуация будет аналогична). Для изображения размера W1×H1 на выходе последнего свёрточного слоя (conv5\_3) сеть VGG16 выдает карту особенностей с пространственными размерами W1/16×H1/16, вектор особенностей для каждой точки будет размерности 512. При этом в вектор особенностей в точке (xf,yf) вносят вклад точки изображения лежащие внутри квадрата с центром в (16xf,16yf) и размера 196×196.

Для каждой точки карты особенностей (xf,yf) будем проверять k претендетов разных размеров на изображении в регионах с центром в (16xf,16yf). В статье авторы предлагают рассматривать 9 претендентов, варьируя три масштаба и три отношения сторон (1:1, 1:2, 2:1). Для этой задачи используется Region Proposal Network (RPN). Модуль RPN сообщает модулю Fast R-CNN, где искать.

В рамках RPN по извлечённым CNN признакам скользят «мини-нейросетью» с небольшим (3х3) окном. Полученные с её помощью значения передаются в два параллельных полносвязанных слоя: box-regression layer (reg) и box-classification layer (cls).

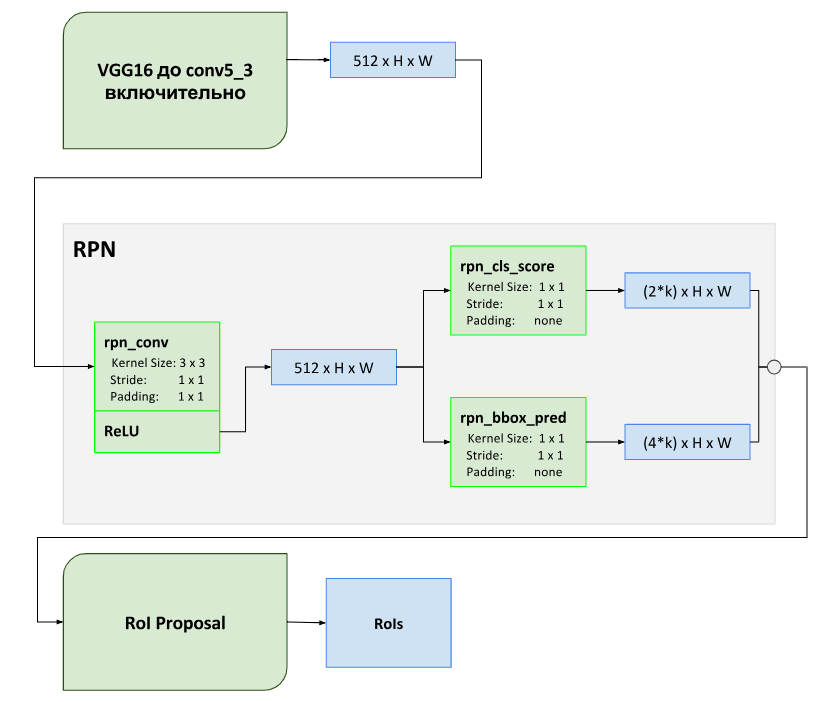
Как видно из схемы (рис.3), карта особенностей, полученная от свёрточной сети, подается на свёрточный слой с ядром размера 3×3. А выход этого свёрточного слоя параллельно подается на два свёрточных слоя с ядром размера 1×1. Первый слой rpn\_cls\_score выдает k пар - вероятности наличия или отсутствия объекта в соответствующем региона (фактически мы имеем обычный полносвязный классификатор, который применяется для каждого вектора особенностей сформированного слоем rpn\_conv).

Рис.3

Слой rpn\_bbox\_pred выдает k четверок - поправки для координат центра и размеров соответствующего региона претендента (получается полносвязный регрессор, который опять применяется к векторам особенностей). Данный генератор претендентов, в силу применяемой схемы, инвариантен к сдвигам объектов на изображении. Авторы называют четвёрку - две координаты центра, масштаб и отношение сторон - анкер (anchor). Анкер полностью определяет регион на изображении. Количество всевозможных анкеров для изображения размеров W1×H1 будет равно 9⋅⌊W1/ 16⌋⋅⌊H1 / 16⌋.

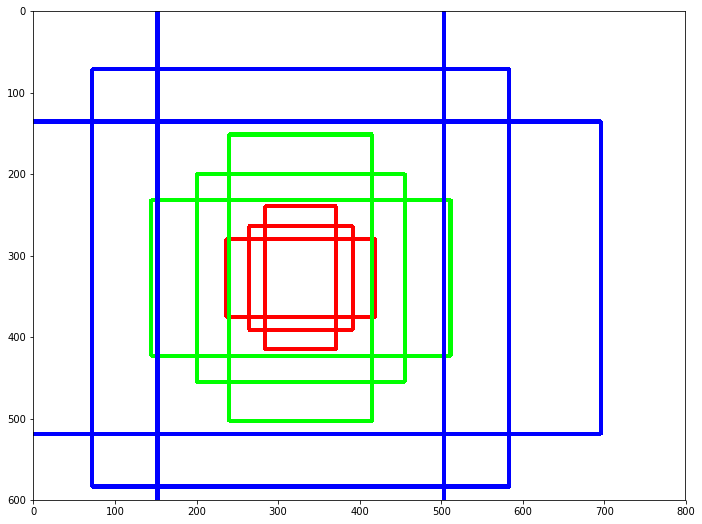
Вкратце, RPN ранжирует области регионов (анкеры) и предлагает те, которые наиболее вероятно содержат объекты. Архитектура выглядит следующим образом. В стандартной конфигурации Faster R-CNN в позиции изображения имеется 9 анкеров. На следующем графике (рис. 4) показано 9 анкеров в положении (320, 320) изображения с размером (600, 800). Три цвета представляют три шкалы или размеры: 128x128, 256x256, 512x512. Каждые три анкера одного цвета имеют соотношения высоты по ширине 1: 1, 1: 2 и 2: 1 соответственно.

Рис.4

## Процесс тренировки сети

Фактически необходимо натренировать две сети: *Fast R-CNN* для классификации и уточнения координат объектов, RPN для генерации претендентов. При этом свёрточные слои должны быть общими для этих двух сетей (собственно, всё и затевалось ради того, чтобы суметь использовать одну карту особенностей для решения обеих задач).

Авторы предлагают несколько возможных подходов к тренировке, однако, результаты, представленные в статье, получены при помощи тренировки по следующей схеме:

1. Тренируем сеть для генерации претендентов. Для этого инициализируем свёрточные слои весами натренированными на ImageNet и доуточняем веса свёрточных слоёв и слоёв RPN части (раздел 1.3.).
2. Тренируем сеть *Fast R-CNN* сеть. Свёрточные слои инициализируем весами от сети, натренированной на ImageNet и дотренировываем (процесс тренировки *Fast R-CNN* мы уже разбирали в соотвествующей части данного текста). При этом претенденты для изображения генерируем при помощи сети полученной в п.1.

Важно, что после выполнения п.2 имеем две нейронных сети с разными весами для свёрточных слоёв.

1. Берем свёрточные слои от *Fast R-CNN*, которую натренировали в п.2, переносим их в *RPN* сеть и дотренировываем её. При этом веса свёрточных слоёв фиксируются, а тренируются только веса слоёв *rpn\_conv*, *rpn\_cls\_score* и *rpn\_bbox\_pred*.
2. Дотренировываем *Fast R-CNN* с учетом изменившегося генератора претендентов. При этом опять не меняем веса свёрточных слоёв, и тренируем только слои специфичные для *Fast R-CNN*.

На этом тренировка заканчивается. Осталось разобраться с тем как тренировать сеть, генерирующую претендентов.

## Процесс тренировки RPN

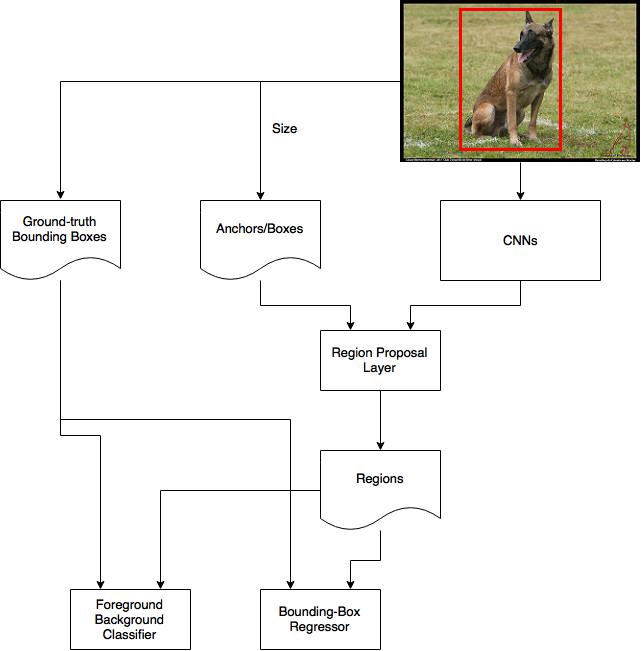
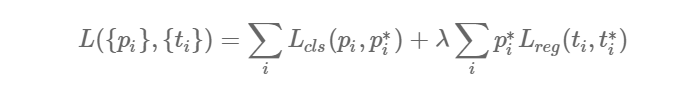
Тренировка осуществляется обычным SGD (Stochastic Gradient Descent) (рис.5). Минибатчи собираются из претендентов (анкеров) случайным образом выбранных на одном изображении (для ускорения тренировки), размер минибатча выбирают равным 256, количество позитивных и негативных претендентов в минибатче пытаются сделать одинаковым, если при этом не удаётся набрать достаточное количество позитивных претендентов, то до 256 экземпляров добираются негативные.

Рис.5

Анкер считается позитивом если метрика IoU (Intersection Over Unit) (рис.6) этого анкера с некоторым объектом из ground truth (это то, что измерили для целевой переменной для примеров обучения и тестирования) данного изображения больше 0.7 или если этот анкер имеет максимальную метрику IoU среди всех анкеров для данного объекта из ground truth. Авторы пишут о том, что крайне редко, но случается, что ни один анкер не имеет с каким-то объектом IoU выше 0.7. Чтобы разрешить эту ситуацию добавлен второй вариант с максимальной метрикой. Заметим, что один объект на изображении может выдать несколько позитивных анкеров.

Рис.6

Негативами считаются анкеры у которых метрика IoU для всех объектов на изображении меньше 0.3. Те анкеры, которые не попали ни в позитивы ни в негативы игнорируются и в тренировке участия не принимают.

Функция потерь для тренировки RPN, как и функция потерь для тренировки Fast R-CNN состоит из двух частей:

здесь pi - вероятность того, что i-ому анкеру соответствует объект, полученная с выхода слоя rpn\_cls\_score. p\*i - равна единице, если i-ый анкер соотвествует позитиву в минибатче, и соотвественно Lcls(pi,p\*i) -логистическая функция потерь на два класса (объект или не объект). Во втором слагаемом сумма считается только по тем анкерам, которые соотвествуют объектам (p\*i=1). L*reg* это сумма функций Хубера, что использовалась при тренировке Fast R-CNN.

## Функции потерь для сети

Обычно рассматривают четыре типа функций потерь для Faster R-CNN:

* RPN classification (anchor good/bad)
* RPN regression(anchor -> proposal)
* Fast R-CNN classification (over classes)
* Fast R-CNN regression (proposal ->box)

# Демонстрация примера обнаружения предметов на фотографии с помощью Faster R-CNN и OpenVINO Toolkit

В этом разделе показано, как запустить демонстрационное приложение Object Detection, которое делает вывод, используя сети обнаружения объектов, такие как Faster R-CNN на процессорах Intel® и Intel® HD Graphics.

После запуска демонстрационное приложение считывает параметры командной строки и загружает сеть и изображение в плагин Inference Engine. Когда вывод сделан, приложение создает выходное изображение и выводит данные в стандартный поток вывода.

## Загрузка и преобразование Caffe\* Model

VGG16-Faster-RCNN является общедоступной CNN, которую можно легко взять на GitHub:

1. Скачать test.prototxt из <https://raw.githubusercontent.com/rbgirshick/py-faster-rcnn/master/models/pascal_voc/VGG16/faster_rcnn_end2end/test.prototxt>
2. Скачать предварительно обученные модели сиз <https://dl.dropboxusercontent.com/s/o6ii098bu51d139/faster_rcnn_models.tgz?dl=0>
3. Распакавать архив. Необходим файл VGG16\_faster\_rcnn\_final.caffemodel.

Для правильного преобразования исходной модели необходимо запуститьModel Optimizer. Можно использовать следующую команду для преобразования исходной модели:

python3 ${MO\_ROOT\_PATH}/mo\_caffe.py

--input\_model <path\_to\_model>/VGG16\_faster\_rcnn\_final.caffemodel

--input\_proto <path\_to\_model>/deploy.prototxt

## Запуск

Запуск приложения с параметром -h выдает следующее сообщение об использовании:

./object\_detection\_demo -h

InferenceEngine:

API version ............ <version>

Build .................. <number>

object\_detection\_demo [OPTION]

Options:

-h Print a usage message.

-i "<path>" Required. Path to an .bmp image.

-m "<path>" Required. Path to an .xml file with a trained model.

-l "<absolute\_path>" Required for MKLDNN (CPU)-targeted custom layers. Absolute path to a shared library with the kernels impl.

-c "<absolute\_path>" Required for clDNN (GPU)-targeted custom kernels. Absolute path to the xml file with the kernels desc.

-pp "<path>" Path to a plugin folder.

-d "<device>" Specify the target device to infer on; CPU, GPU, FPGA or MYRIAD is acceptable. The demo will look for a suitable plugin for a specified device.

-pc Enables per-layer performance report

-ni "<integer>" Number of iterations (default 1)

-bbox\_name "<string>" The name of output box prediction layer (default: bbox\_pred)

-proposal\_name "<string>" The name of output proposal layer (default: proposal)

-prob\_name "<string>" The name of output probability layer (default: cls\_prob)

-p\_msg Enables messages from a plugin

Запуск приложения с пустым списком опций выдает приведенное выше сообщение об использовании и сообщение об ошибке. Используйте следующую команду, чтобы сделать вывод о процессорах Intel® для образа, использующего обученную сеть Faster R-CNN:

$ ./object\_detection\_demo -i <path\_to\_image>/inputImage.bmp -m <path\_to\_model>/faster-rcnn.xml -d CPU

## Выходные данные

Приложение выводит изображение с именем out\_0.bmp с обнаруженными объектами, заключенными в прямоугольники. Он выводит список классов обнаруженных объектов вместе с соответствующими значениями достоверности и координатами прямоугольников в стандартный поток вывода.

# Список используемых источников

1. Техническое задание на НИР «Разработка и реализация программного обеспечения для распознавания лиц на групповых фотографиях» (Шифр ПО «FaceRecognition»)
2. Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun - Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks, <https://arxiv.org/pdf/1506.01497.pdf>
3. Rohith Gandhi - R-CNN, Fast R-CNN, Faster R-CNN, YOLO — Object Detection Algorithms

Understanding object detection algorithms, <https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e>

1. Image Classification Sample - https://software.intel.com/en-us/articles/OpenVINO-IE-Samples#object-detection-faster-rcnn