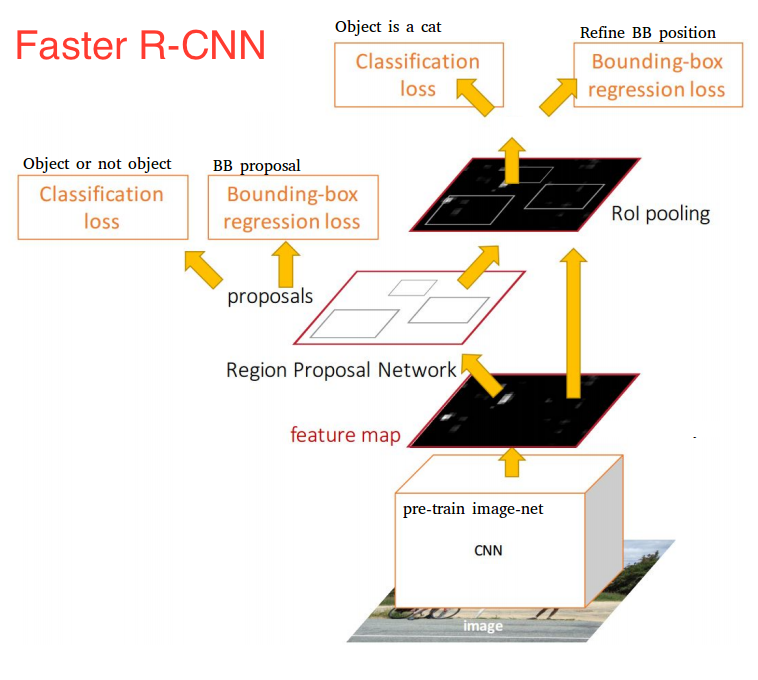
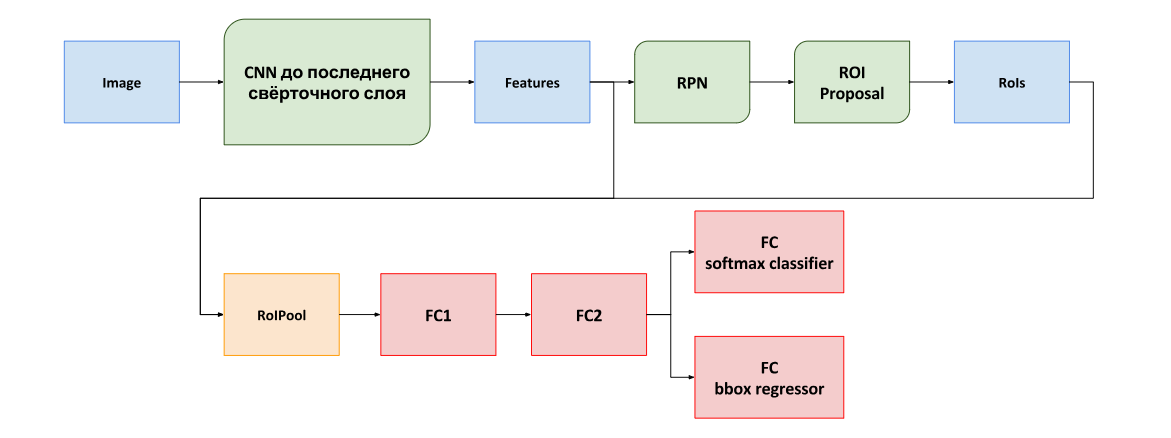
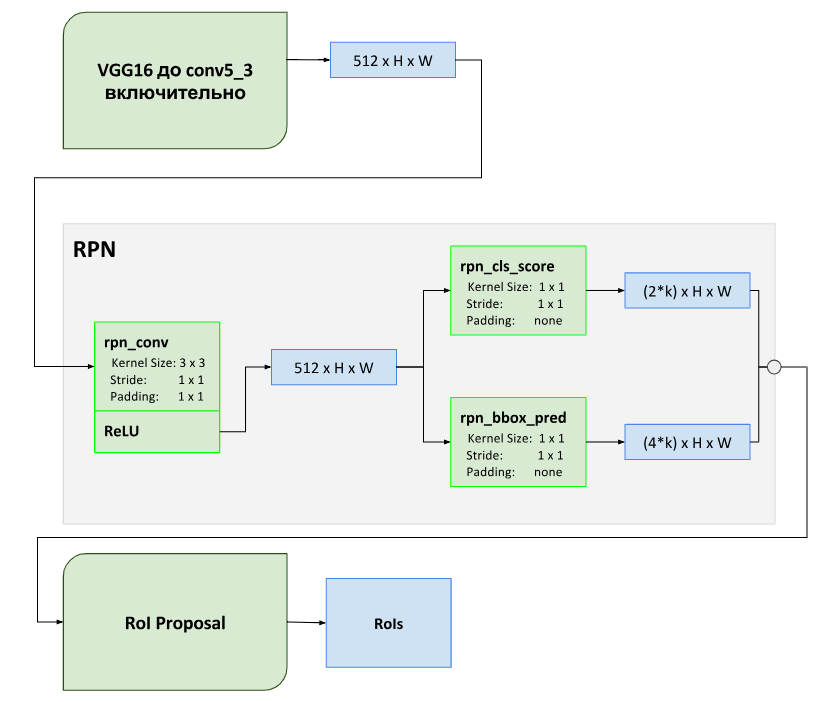
**Пояснительная записка к выбранной сети Faster R-CNN**

Основное назначение данного документа – исследование общих понятий архитектуры нейронной сети Faster R-CNN.

Сеть Faster R-CNN состоит из двух модулей. Первый модуль представляет собой глубокую полностью сверточную сеть, которая предлагает регионы (это замена процедуры генерации претендентов selective search в других моделях R-CNN, 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

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

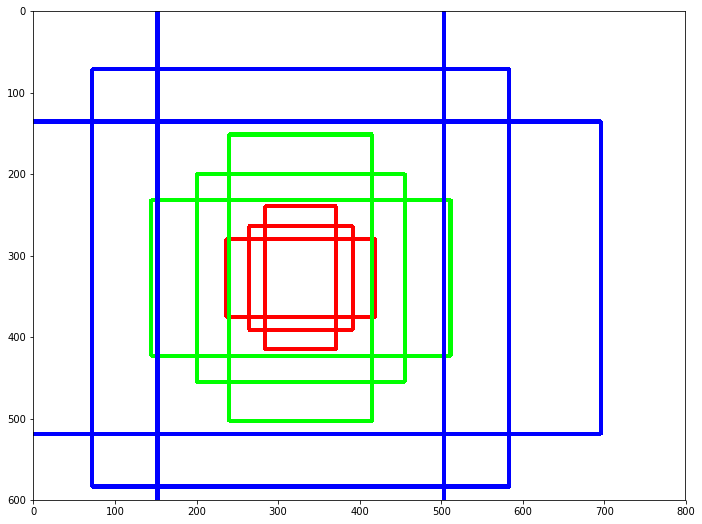
В качестве свёрточной сети авторы предлагают использовать либо [VGG16](https://vbystricky.github.io/images/2017-06/vgg16.svg) (Simonyan and Zisserman model), либо ZF (Zeiler and Fergus model). Для определенности будем рассматривать случай VGG16 (для ZF ситуация будет аналогична). Для изображения размера 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 задачи будем использовать Region Proposal Network (RPN). Модуль RPN сообщает модулю Fast R-CNN, где искать.

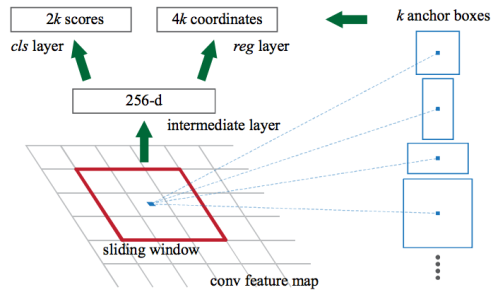
В рамках RPN по извлечённым CNN признакам скользят «мини-нейросетью» с небольшим (3х3) окном. Полученные с её помощью значения передаются в два параллельных полносвязанных слоя: box-regression layer (reg) и box-classification layer (cls).

Как видно из схемы, 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).

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). Отметим, что 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 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.

Выходные данные RPN представляют собой набор предложений, которые будут проанализированы классификатором и регрессором, чтобы в конечном итоге проверить наличие объектов. Т.е. RPN предсказывает возможность анкера быть фоном ил передним планом и уточняет анкер.



**Тренировка**

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).

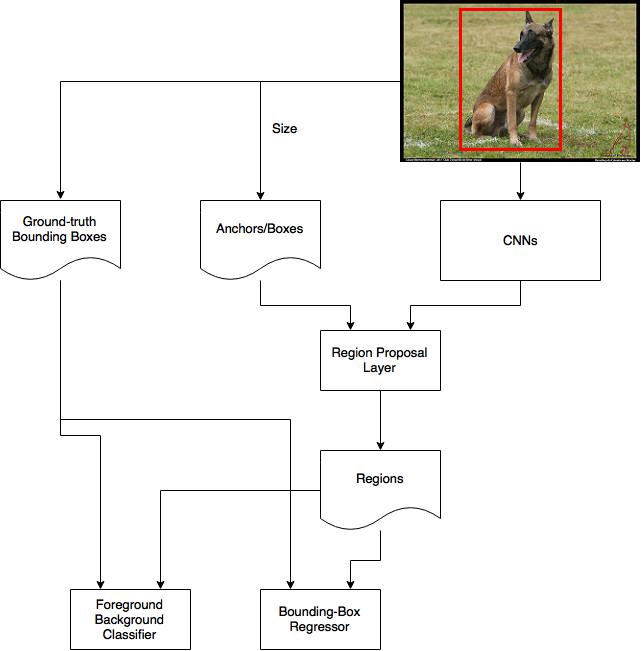
Авторы предлагают несколько возможных подходов к тренировке, однако, результаты, представленные в статье, получены при помощи тренировки по следующей схеме:

1. Тренируем сеть для генерации претендентов. Для этого инициализируем свёрточные слои весами натренированными на ImageNet и доуточняем веса свёрточных слоёв и слоёв RPN части (подробно про процесс тренировки этой сети чуть ниже).
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 мы 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.

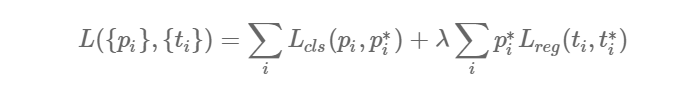
**Тренировка RPN**

Тренировка осуществляется обычным SGD (Stochastic Gradient Descent). 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.

Анкер считается позитивом если метрика IoU (Intersection Over Unit) этого анкера с некоторым объектом из ground truth (это то, что вы измерили для своей 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.

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). Lreg это таже сумма функций Хубера, что использовалась при тренировки Fast R-CNN.

**4 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)

Оригинальная статья авторов Faster R-CNN: <https://arxiv.org/pdf/1506.01497.pdf>

Несколько полезных ссылок:

<https://medium.com/@smallfishbigsea/faster-r-cnn-explained-864d4fb7e3f8>

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