

Машинное обучение (Machine Learning)

Сегментация и детекция изображений

Уткин Л.В.

Санкт-Петербургский политехнический университет Петра Великого



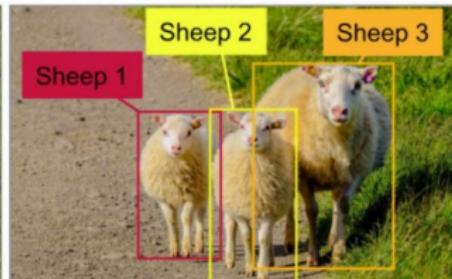
Типы сегментации

- Semantic segmentation
- Classification and localization
- Object detection
- Instance segmentation

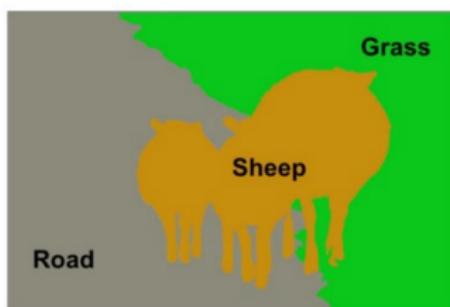
Типы сегментации



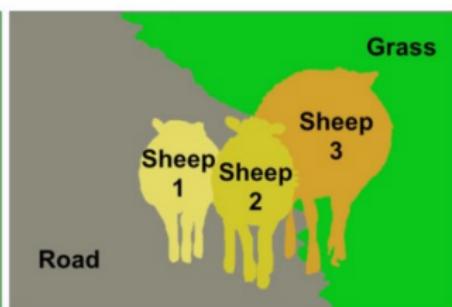
Classification + Localization



Object Detection



Semantic Segmentation



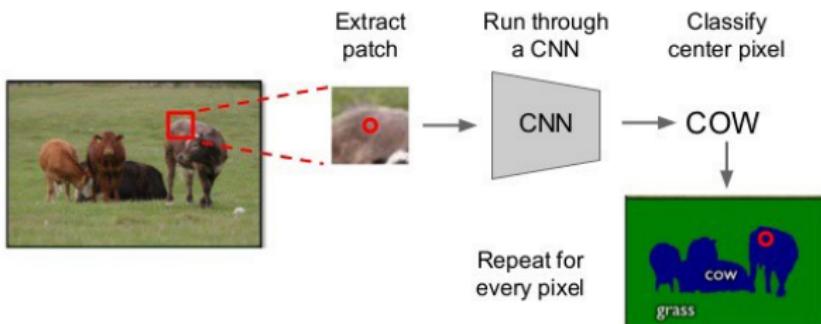
Instance Segmentation

Semantic segmentation (1)

- Есть изображение и выход - классификация каждого отдельного пикселя
- Например, все пиксели с изображением овец будут отнесены к одной категории, как и пиксели с травой и дорогой
- Один из возможных подходов к решению - рассматривать как задачу классификации со скользящим окном
- Исходное изображение разбивается на несколько фрагментов одинакового размера
- Каждый фрагмент – в CNN, чтобы классифицировать

Semantic segmentation (2)

Semantic Segmentation



Slide Credit: [CS231n](#)

8

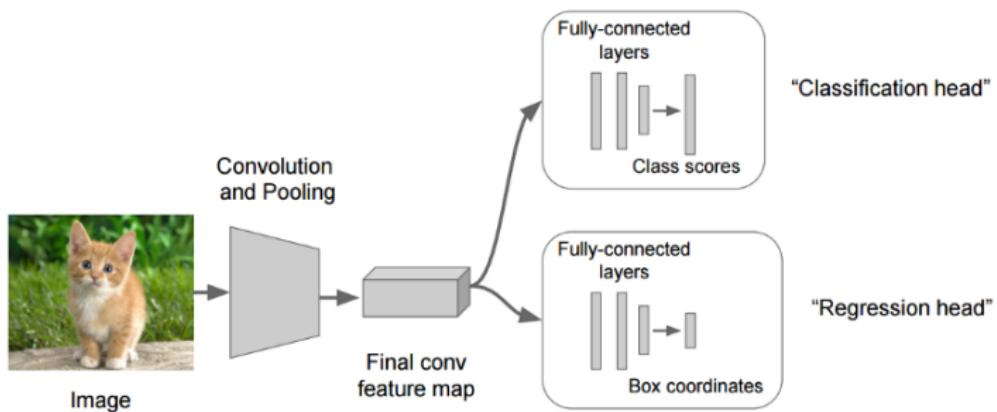
Классификация и локализация (Classification and localization)

- Классификация ставит метку, но иногда, интересует расположение этого объекта на изображении.
- Можно нарисовать ограничивающую рамку вокруг объекта на изображении, чтобы локализовать его
- Как?

Классификация и локализация (1)

- Сначала можно "скормить" входное изображение какой-нибудь гигантской ConvNet, которая выдаст веса по каждой категории
- Теперь есть еще один полносвязный слой, который предсказывает координаты ограничивающей рамки для объекта (координаты центра x, y, а также высоту и ширину) из карты объектов, созданной более ранними слоями
- Таким образом, сетка выдаст два вывода: один соответствует классу изображения, а другой — ограничивающей рамке
- Для обучения сети две функции потерь: потери (энтропия) для классификации и потери L1/L2 для предсказаний ограничительной рамки

Классификация и локализация (2)



Object Detection (1)

- Идея обнаружения объектов: начинаем с некоторого фиксир. набора интересующих нас категорий
- Каждый раз, когда любая из этих категорий появляется на входном изображении, мы рисуем ограничивающую рамку вокруг этого изображения вместе с предсказанием метки класса
- Отличие от классиф-ии и локал-ии в том, что в detection классифицируют и рисуют рамку только вокруг **одного** объекта
- В класс. и локализ. не известно заранее, сколько объектов ожидать на изображении.

Object Detection - Region Proposals Based Algorithms (RPBA)

- RPBA даст много рамок, в которых может присутствовать объект. На выходе возможен шум в виде рамок, в которых нет объектов. Но объект будет выбран в качестве рамки-кандидата.
- Чтобы сделать все рамки-кандидаты одинакового размера, нам нужно деформировать их до некоторого фиксированного размера квадрата, который можно в конечном итоге передать в сеть.
- Затем применяют большую ConvNet к каждой рамке-кандидату для классификации.

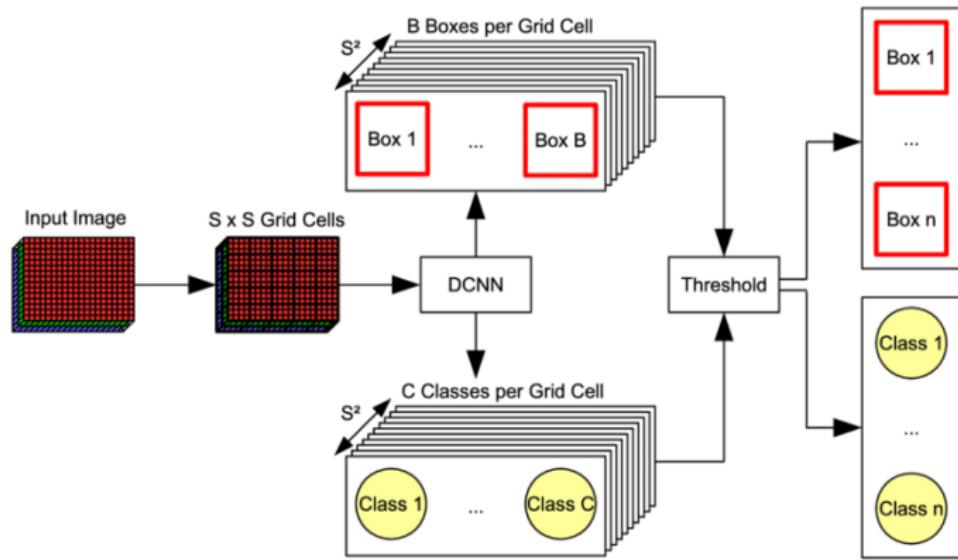
Object Detection - Region Proposals Based Algorithms (RPBA)

- В этом вся идея R-CNN. Для снижения сложности используются быстрые R-CNN: сначала получить карту объектов с высоким разрешением, передав входное изображение через ConvNet, а затем наложить эти Region Proposals на карту объектов вместо фактического изображения. Это позволяет повторно использовать вычисления свертки по всему изображению, когда много обрезков.

Object Detection - Region Proposals Based Algorithms (RPBA)

Object Detection - YOLO

YOLO (You only look once)



Object Detection - YOLO (You only look once)

- Идея YOLO: вместо независимой обработки предложенных областей делать все предсказания одновременно, переформулируя их как единую задачу регрессии, прямо от пикселей изображения до координат ограничивающей рамки и вероятностей классов.
- Сначала делим все входное изображение на сетку $S \times S$.
- Каждая ячейка сетки предсказывает C вероятностей условного класса ($\text{Pr}(\text{Class} \mid \text{Object})$) вместе с B ограничивающими рамками (x, y, w, h), каждая из которых имеет показатель достоверности.
- Координаты (x, y) представляют центр прямоугольника относительно границ ячейки сетки, тогда как ширина и высота предсказываются относительно всего изображения.

Object Detection - YOLO (You only look once)

- Показатели достоверности отражают, насколько модель уверена в том, что рамка содержит объект. Если в рамке нет объектов, то показатель достоверности должен быть равен нулю. С другой стороны, показатель достоверности должен быть таким же, как пересечение по объединению (IOU) между предсказанной рамкой и меткой класса.
- Confidence score = $\text{Pr}(\text{Object}) * \text{IOU}$
- Во время тестирования условные вероятности класса умножаются на предсказанные достоверности отдельных рамок, что дает оценки достоверности для каждого класса для каждой рамки. Эти оценки кодируют как вероятность того, что этот класс появится в рамке, так и то, насколько хорошо предсказанная рамка соответствует объекту.

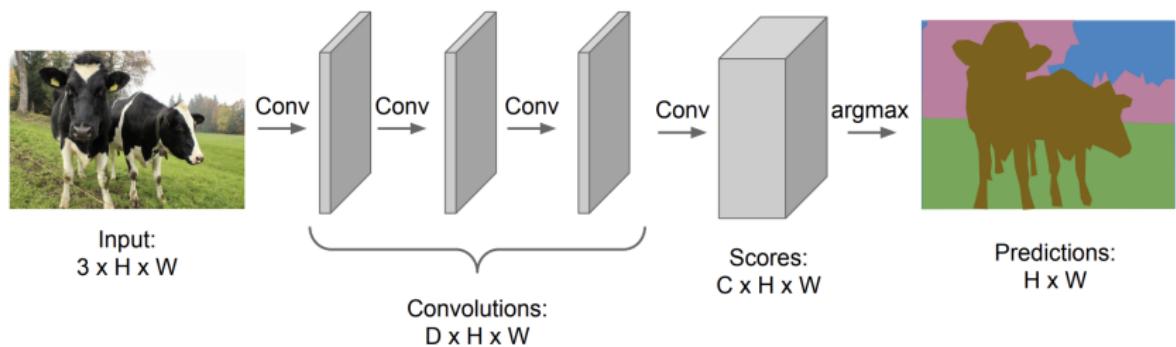
Semantic Segmentation, классификация на уровне пикселов

- Просто соединить набор сверточных слоев, где захватываются локальные элементы в изображениях.
- CNN может кодировать изображение как компактное представление его содержимого
- Отображение между входным изобр. и соответствующим выходом сегментации через иерархическое представление

Patch-by-patch scanning - Основная идея

- Метка класса определяется для каждого пикселя: задача рассматривается как задача “попикセルной” классификации
- Окно (patch) с центром в каждом пикселе подается на вход сверточной сети для генерации его класса
- Точность сегментации повышается с увеличением размера окна, так как оно охватывает больше контекстной информации
- Главный недостаток - большое перекрытие соседних окон и большой объем избыточных вычислений

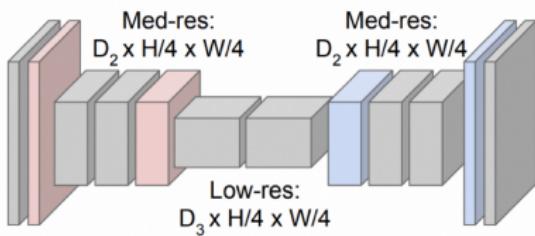
Semantic Segmentation, классификация на уровне пикселов



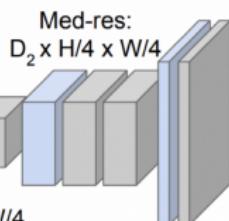
Semantic Segmentation, классификация на уровне пикселов



Input:
 $3 \times H \times W$



High-res:
 $D_1 \times H/2 \times W/2$



High-res:
 $D_1 \times H/2 \times W/2$

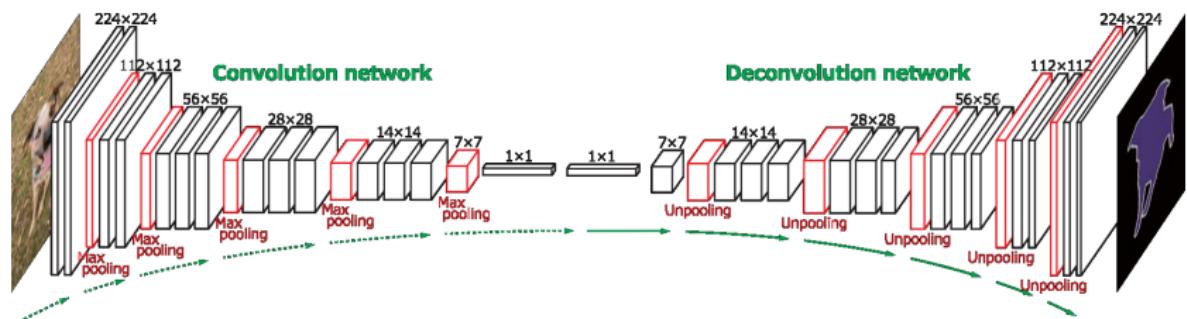


Predictions:
 $H \times W$

Up sampling
Down sampling

Deconvolution network (Deconvnet)

H. Noh S. Hong B. Han, Learning Deconvolution Network for Semantic Segmentation

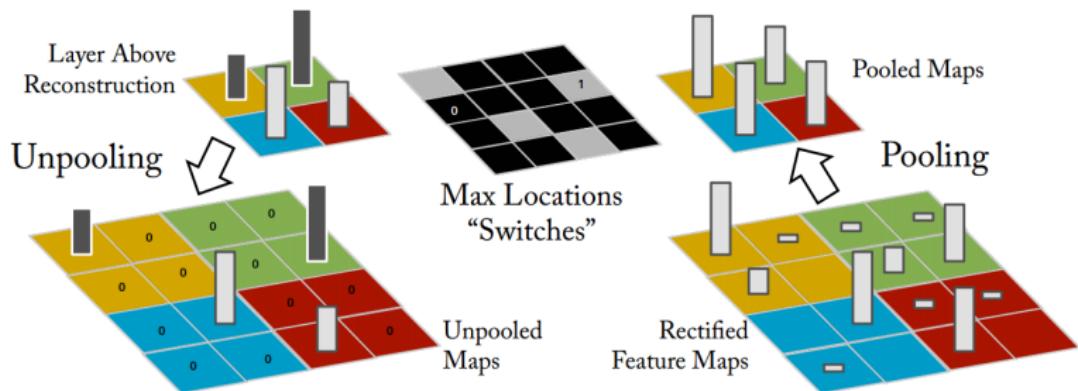


Deconvolution network

Реализует две основные процедуры

- ① Unpooling
- ② Deconvolution

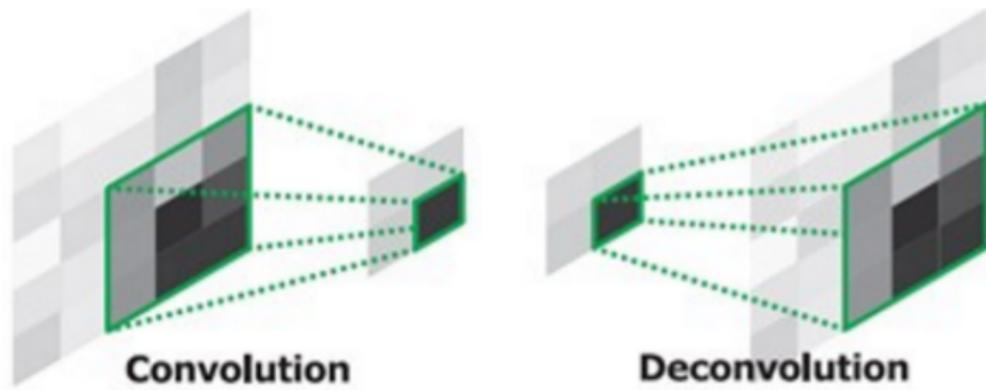
Unpooling (upsampling)



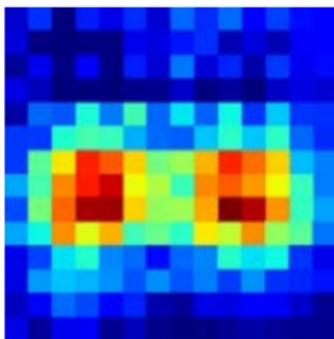
Deconvolution

- Deconvolution слой `layers` уплотняет рареженный слой, полученный в `unpooling`, с использованием операций свертки
- В отличие от сверточных слоев, которые соединяют множественные входные значения активации нейронов внутри окна фильтра в одно значение, обратная операция свртки ассоциирует одиночное входное значение активации с множественным выходом

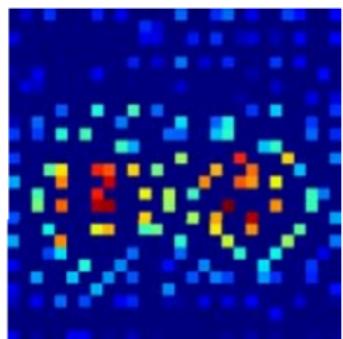
Deconvolution (схема)



Deconvolution and unpooling



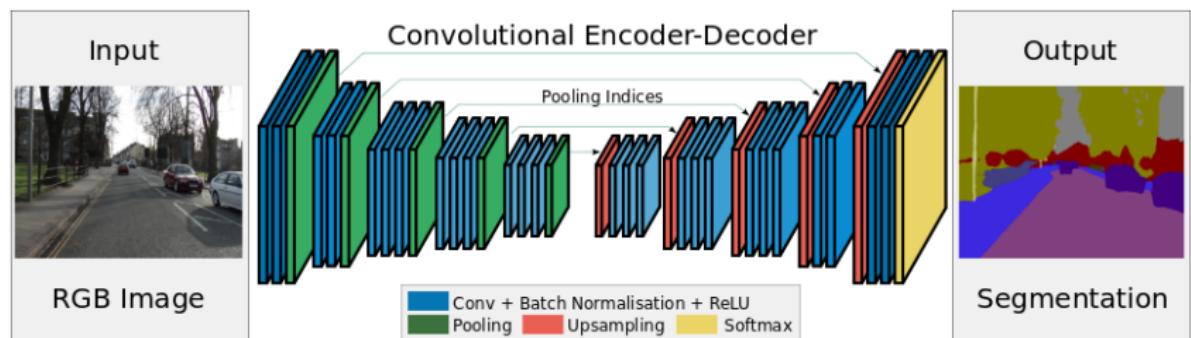
Deconv: 14x14



Unpool: 28x28

SegNet

V. Badrinarayanan, A. Kendall and R. Cipolla "SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation." arXiv preprint arXiv:1511.00561, 2015



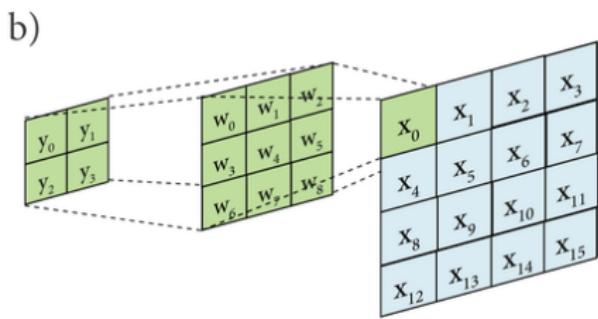
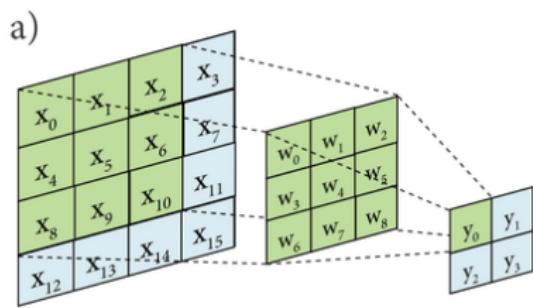
Особенность SegNet

- Сеть состоит из :
 - последовательности слоев нелинейного преобразования (кодер) и
 - соответствующего множества декодеров, за которыми следует “попиксельный” классификатор
- **Ключевой элемент SegNet** - декодер использует индексы, полученные на этапах max-pooling в кодере, для реализации upsampling
- **Индексы** - местоположения признаков с максимальными значениями в каждом окне pooling (запоминаются для каждой карты признаков в кодере)
- Это позволяет избежать обучения проц. upsampling
- Сеть в целом обучается, используя градиентный спуск

Semantic Segmentation, классификация на уровне пикселов

- Сжатие изображения посредством использования пулинга и сверток
- Эта конфигурация кодера для задач классификации, так как она заботилась только о содержании изображения, а не о его местоположении
- Однако для задачи сегментации необходимо иметь маску с полным разрешением для пиксельного прогнозирования

Semantic Segmentation, классификация на уровне пикселов



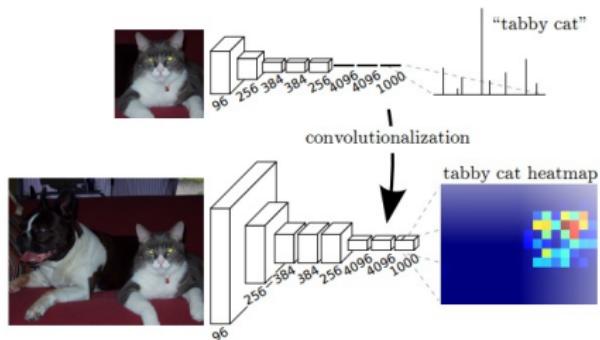
Fully Convolutional Network (FCN) -

Полносверточная сеть

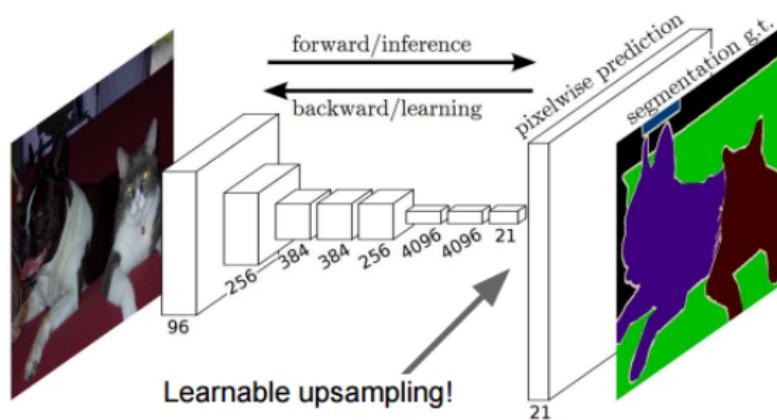
- Полносверточная сеть была предложена для устранения patch-by-patch scanning и для повышения эффективности
- FCN заменяет полносвязанные слои в сверточной сети на 1×1 сверточные ядра.
- FCN в качестве входа использует всю картинку и получает сегментационную карту на выходе одним проходом прямого распространения

Fully Convolutional Network (FCN)

- Преобразование полносвязанных слоев в сверточные, охватывающих всю входную область
- Каждая из этих сверток будет выводить coarse heatmap меток

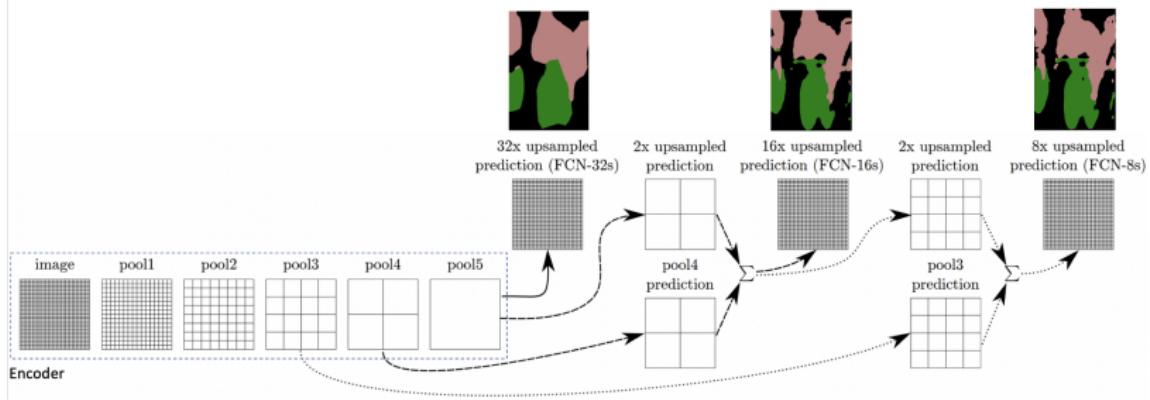


FCN



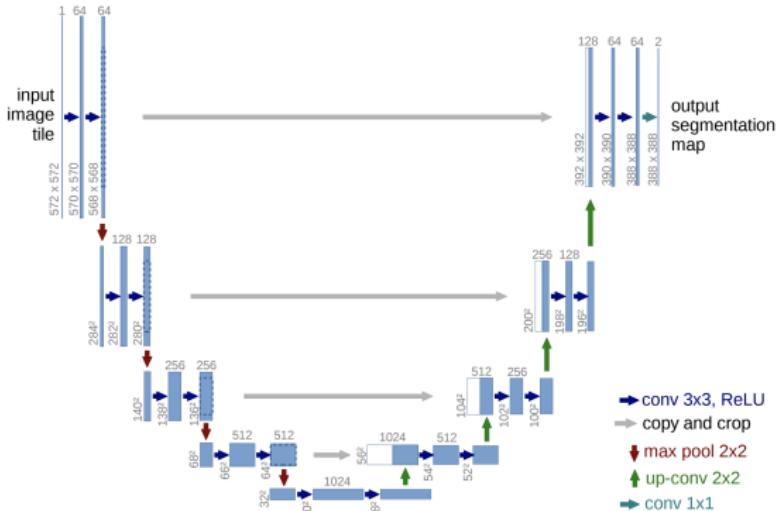
Кодер сжимает изображение в изображение с более низким разрешением, что приводит к грубой сегментации после операции повышения дискретизации (upsampling)

FCN



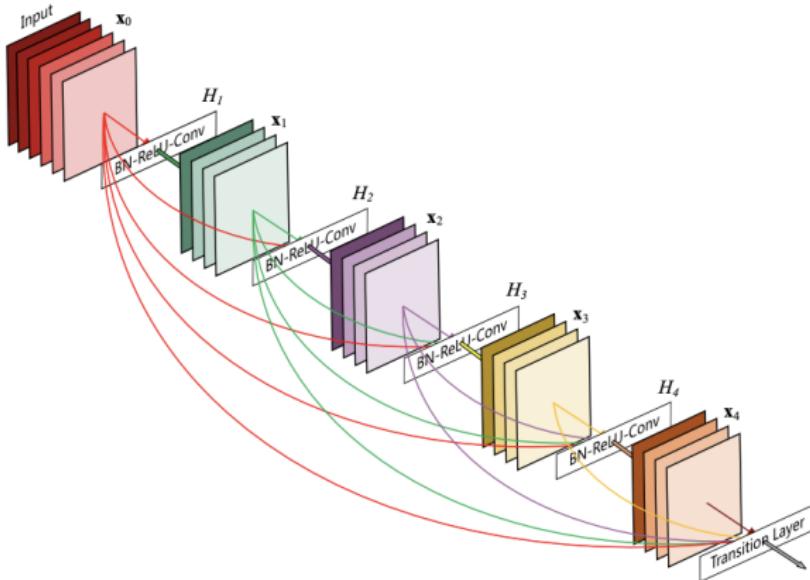
Положение на слоях более высокого уровня соответствуют положениям на изображении, с которыми они связаны путями, называемыми *рецептивными полями*

U-Net



- Первая часть состоит из обычных сверток, ReLU и пулинга 2×2 с шагом 2, который приводит к понижающей дискретизации (downsampling)
- Вторая часть состоит из последовательности слоев повышающей дискретизации (upsampling), конкатенации соответствующей карты признаков, которая необходима из-за потери границ пикселей после каждой свертки
- Как и FCN, признаки высокого разрешения из пути сжатия объединяются с upsampled выходом, который затем подается на ряд сверточных слоев

FC-DenseNet

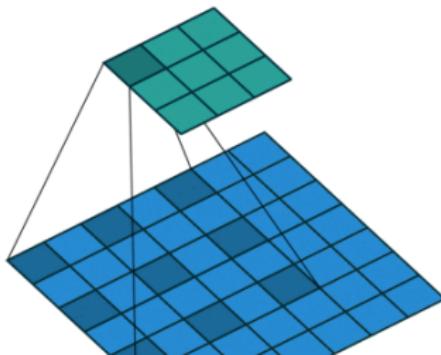


FC-DenseNet

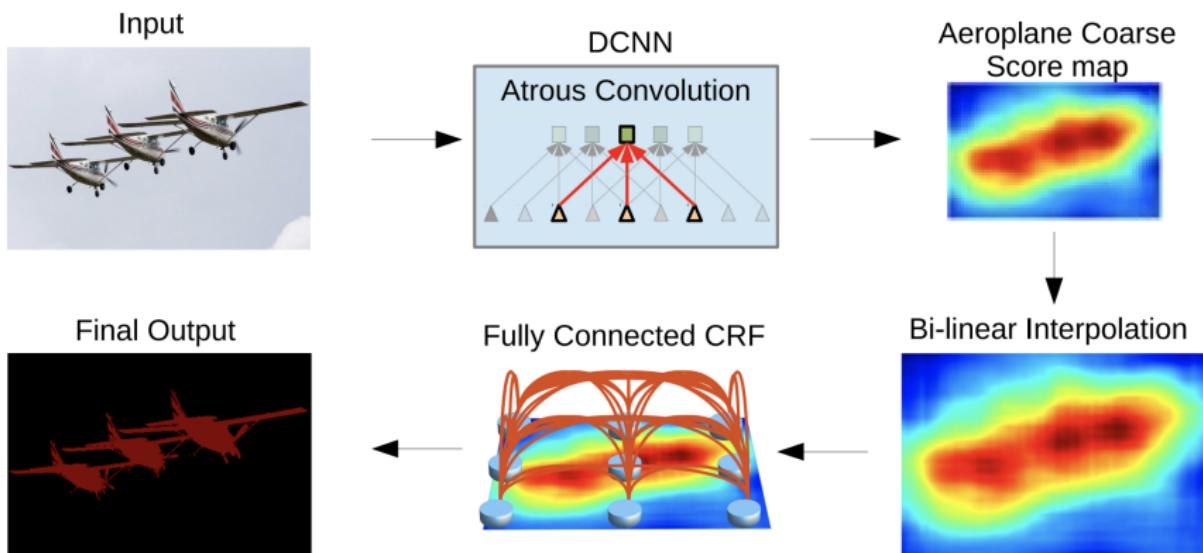
- FC DenseNet или 100 Layers Tiramisu - это метод сегментации, основанный на архитектуре DenseNet
- DenseNet основан на том, что “перемычки” с ранних уровней устанавливаются на поздние уровни и все слои связаны друг с другом
- Каждый слой передает свои карты признаков всем последующим слоям
- Если ResNet использует поэлементное суммирование для объединения признаков, то DenseNets - конкатенацию
- Каждый уровень получает совокупный набор “знаний” от всех предыдущих уровней

DeepLab

- DeepLab v1 основан на двух идеях: Atrous Convolution и полносвязанное условное случайное поле (FC CRF)
- Atrous convolution с французского “a trous” - дыра (hole), также называется “расширяющаяся (dilated) свертка”
- “Расширяющаяся (dilated) свертка” - это стандартная свертка, через которую пропускается некоторое число пикселей в двух измерениях



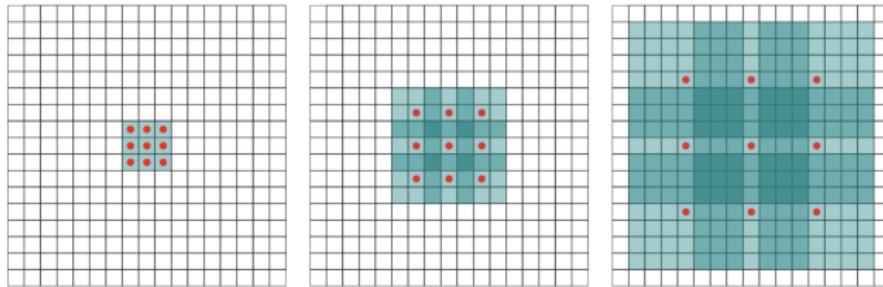
DeepLab



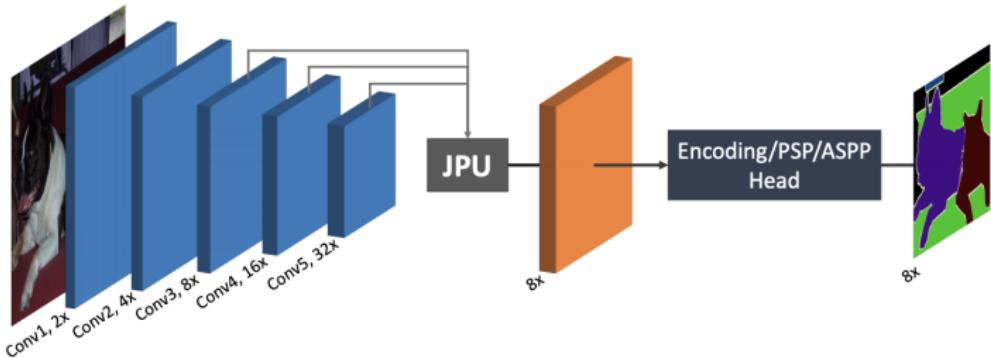
FC CRF is applied to refine the segmentation result

DeepLab

- Вместо типового пулинга DeepLab использует расширенные слои для решения проблемы балансировки
- Контролируя поле зрения в dilated свертках, можно найти лучший компромисс между точной локализацией (малое поле зрения) и ассилияцией контекста (большое поле зрения)

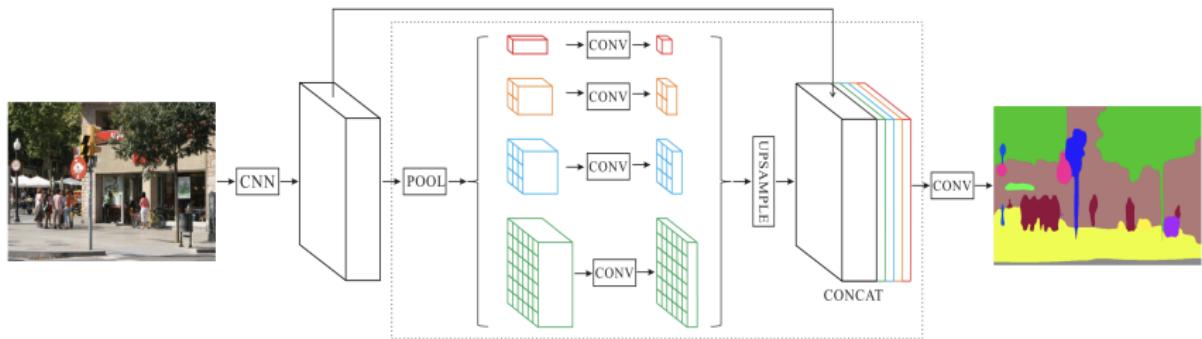


Fast FCN

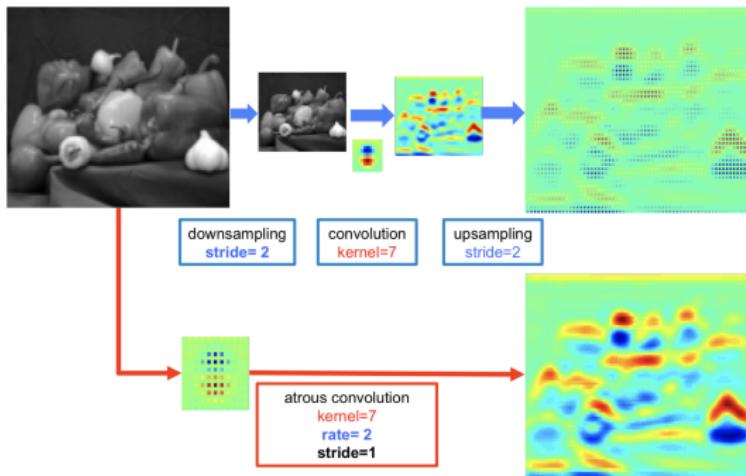


ASPP - atrous spatial pyramid pooling; PSP - pyramid scene parsing network

PSP - pyramid scene parsing network



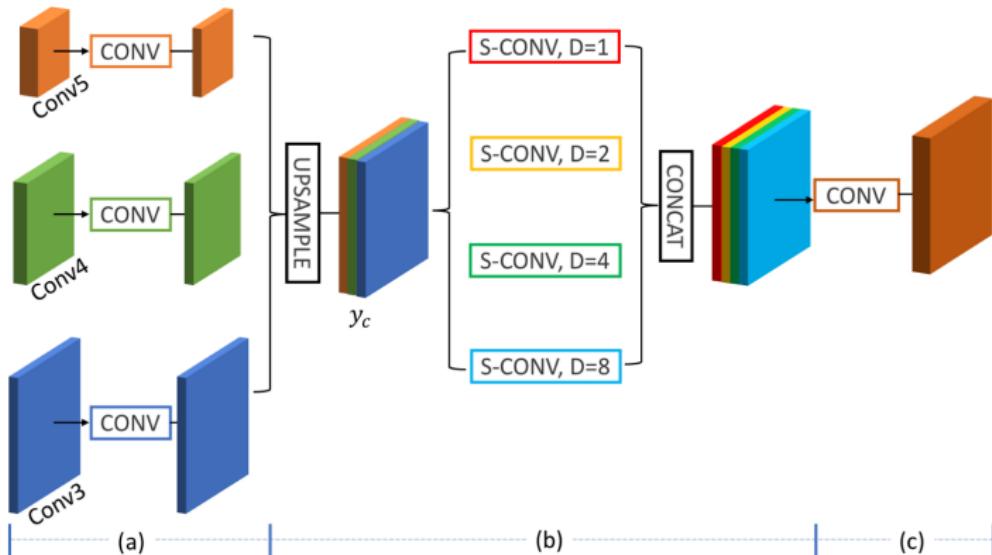
Пример “atrous convolution”



Fast FCN

- Принцип расширенной свертки был заменен совместным пирамидальным upsampling (JPU)
- Различия между Fast FCN и DilatedFCN заключаются в последних двух стадиях свертки
- Как правило, карта входных объектов сначала обрабатывается обычной сверткой, а затем серией расширенных сверток.
- Fast FCN концептуально обрабатывает входную карту признаков сверткой, а затем использует несколько сверток для генерации выходных данных
- Это снижает вычислительную сложность по сравнению с DilatedFCN
- JPU создан, чтобы упростить процесс оптимизации.

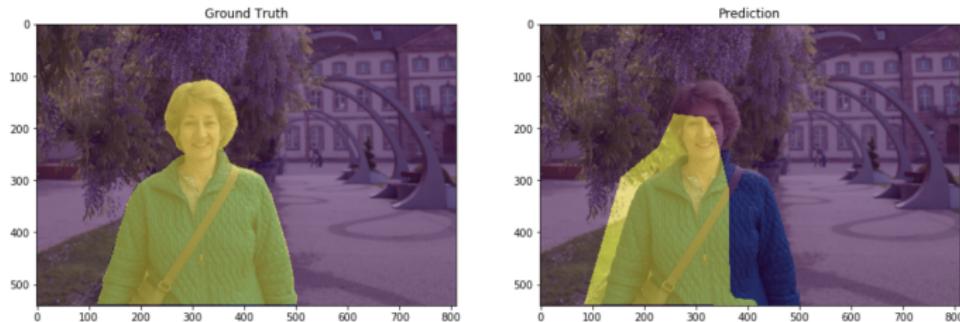
Fast FCN - архитектура JPU



Fast FCN - архитектура JPU

- Каждая карта признаков проходит через обычный сверточный блок
- Затем карты признаков подвергаются дискретизации и конкатенации, которые затем проходят через четыре свертки с различными степенями расширения
- Результаты свертки снова объединяются и проходят через слой окончательной свертки
- Где ASPP использует только информацию из последней карты признаков, JPU извлекает информацию о всем контексте из карт многоуровневых объектов, что приводит к повышению производительности

Оценка качества сегментации



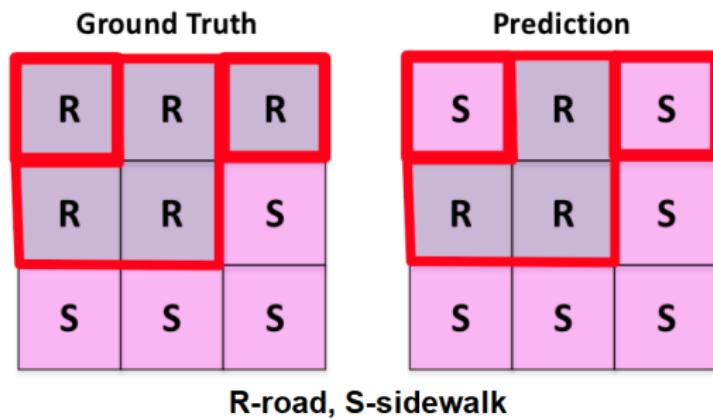
<https://www.jeremyjordan.me/evaluating-image-segmentation-models/>

Оценка качества сегментации - Pixel Accuracy

- Pixel Accuracy - процент пикселей в изображении, которые были правильно классифицированы
- Pixel Accuracy вычисляется для каждого класса отдельно, а также для всех классов в целом
 - TP: пиксель правильно классифицирован как X
 - FP: пиксель не правильно классифицирован как X
 - TN: пиксель правильно классифицирован как не X
 - FN: пиксель не правильно классифицирован как X

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

Pixel Accuracy - пример



$$\text{accuracy} = \frac{TP + TN}{TP + TN + FP + FN} = \frac{3 + 4}{3 + 4 + 0 + 2} = \frac{7}{9}$$

Pixel Accuracy - недостаток



Птицы - positive класс и Небо - negative класс. TN (Небо) - высокий -> Pixel Accuracy - высокая.

$$accuracy = \lim_{TN \rightarrow \infty} \frac{TP + TN}{TP + TN + FP + FN} \rightarrow 1$$

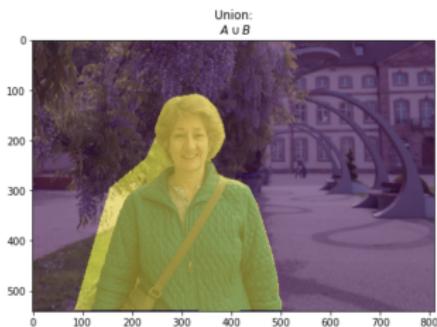
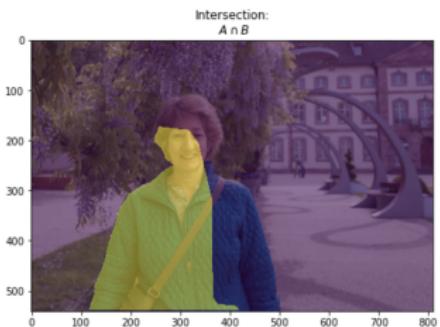
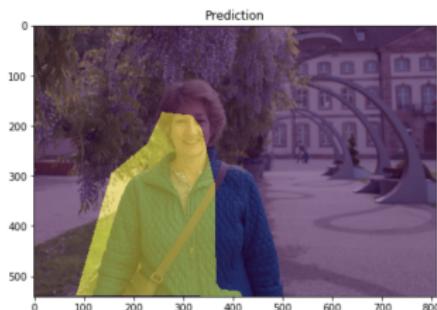
Оценка качества сегментации - IoU

- Intersection over Union (IoU) = Jaccard index

$$IoU = \frac{target \cap prediction}{target \cup prediction}$$

- IoU измеряет количество пикселей, общих для целевой маски и маски после сегментации, деленное на общее количество пикселей, присутствующих в обеих масках
- IoU вычисляется для каждого класса в отдельности, а затем усредняется по всем классам, чтобы получить глобальный средний показатель IoU

Оценка качества сегментации - пример



<https://www.jeremyjordan.me/evaluating-image-segmentation-models/>

Оценка качества сегментации - Dice

- Dice - показатель перекрытия
- Используется больше при определении функции потерь

$$Dice = 2 \frac{\#\{target \cap prediction\}}{\#target + \#prediction}$$

- $\#\{target \cap prediction\}$ заменяют поэлементным умножением сегментационной и целевой масок, а затем суммируют полученную матрицу

Dice index

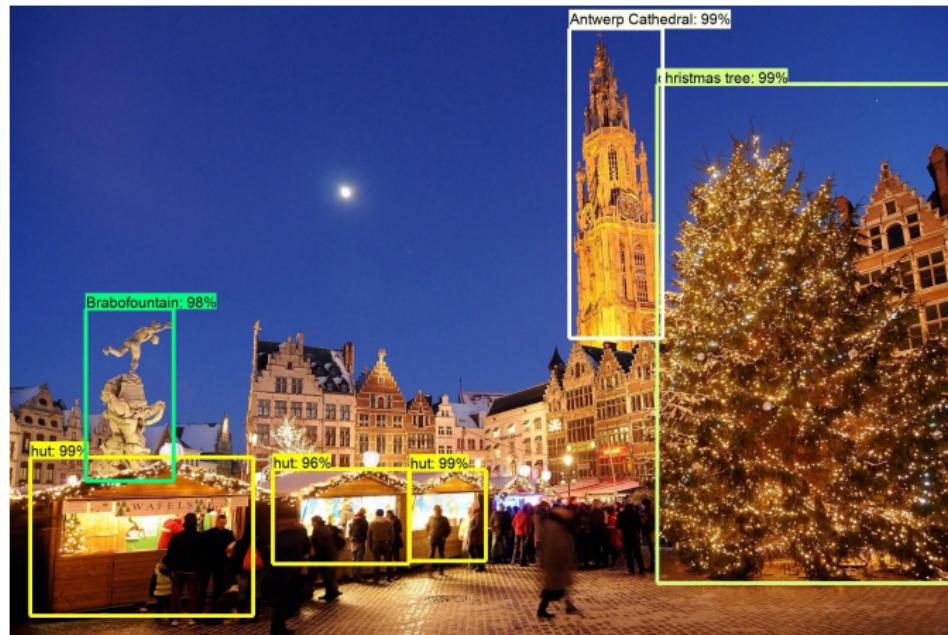
$$|A \cap B| = \begin{bmatrix} 0.01 & 0.03 & 0.02 & 0.02 \\ 0.05 & 0.12 & 0.09 & 0.07 \\ 0.89 & 0.85 & 0.88 & 0.91 \\ 0.99 & 0.97 & 0.95 & 0.97 \end{bmatrix} * \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

prediction target

element-wise multiply \rightarrow

$$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0.89 & 0.85 & 0.88 & 0.91 \\ 0.99 & 0.97 & 0.95 & 0.97 \end{bmatrix} \xrightarrow{\text{sum}} 7.41$$

Detection обнаружение (1)



Detection обнаружение (2)

- При обучении детектора набор данных будет иметь для каждого изображения 0 или более ограничивающих прямоугольников (bounding box) и соответствующих им меток
- Т.е. модель имеет 2 выхода:
 - распределение вероятностей задачи классификации
 - прогнозирование bounding box
- Решение - детектор для обнаружения одного конкретного объекта по всему изображению. Один детектор для кошек, другой - для собак.
- **Другое решение** - использование скользящего окна в сочетании с простым классификатором изображений

Detection обнаружение (3)

- ① Создать небольшое окно, вырезать область внутри окна и передать его в ConvNet.
- ② Сдвинуть окно дальше и продолжить передавать вырезанное содержимое в ConvNet.
- ③ Как только окно переместится по всему изображению, увеличить размер окна и повторите шаг 1 и шаг 2.
- ④ Конечным результатом является набор различных вырезанных изображений, где некоторые содержат метки и bounding box(es) объекта(ов).

Detection обнаружение (4)

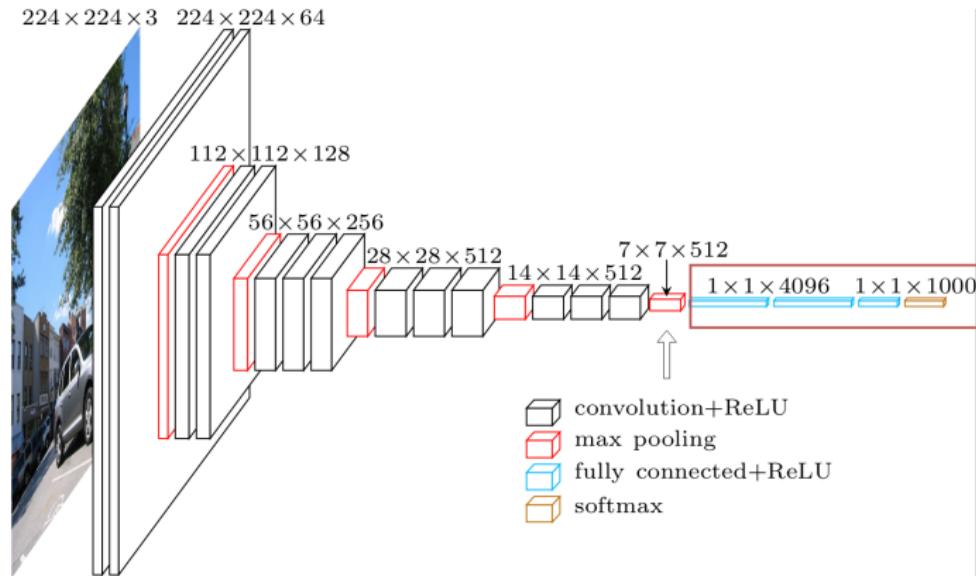
Минусы алгоритма:

- Скользящее окно имеет фиксированную прямоугольную форму - неточное обнаружение bounding box, если ни одно из скользящих окон не соответствует реальному объекту
- Постоянное вырезание изображений и подача их в ConvNet требует больших вычислительных ресурсов

One-stage Object Detection

- One-shot detectors обходят оба минуса, используя фиксированное множество детекторов на сетке (grid detectors)
- Каждый bounding box детектор находится в определенной позиции на изображении
- Подаем изображение в обычную сверточную сеть, которая называется основой детектора объектов.
- Эта сеть - есть не что иное, как классификатор изображений и ее можно предварительно обучать на огромных наборах данных (ImageNet)

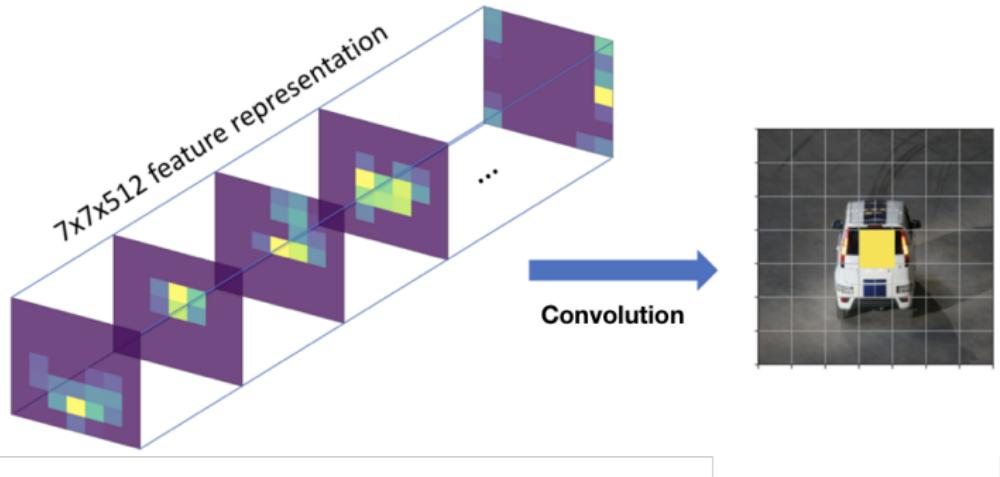
Detection обнаружение - основная сеть



Detection обнаружение (5)

- Удаляем последние слои (красный прямоугольник), так как не надо классифицировать входное изображение
- Выход основной сети - изображение $7 \times 7 \times 512$, которое имеет низкое пространственное разрешение и высокое разрешение признаков
- Располагая эту карту признаков на исходном изображении, можно сопоставить ячейки сетки и входное изображение
- При таком отображении ячейка сетки, которая содержит центр bounding box, может быть аппроксимирована.

Detection обнаружение (6)



Detection обнаружение (7)

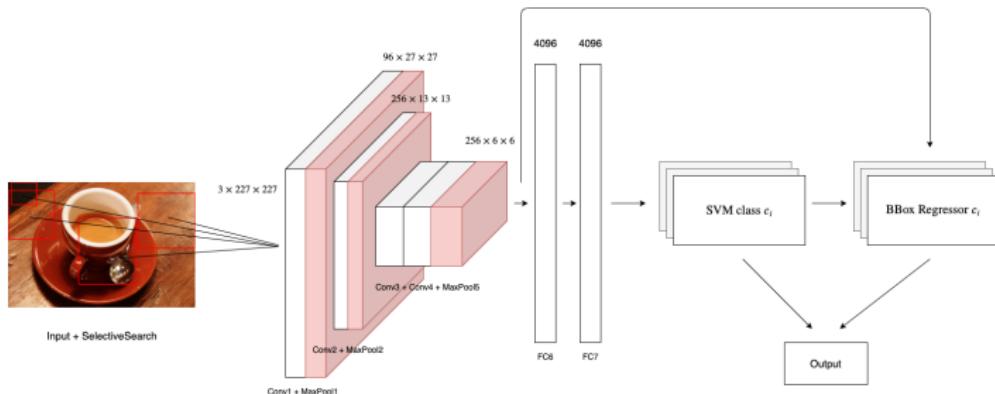
- Сетка объединит все 512 признаков, используя сверточные слои, чтобы обнаружить объект с помощью bounding box.
- Атрибуты, необходимые для описания обнаруженного объекта:
 - координаты x-y и ширина/высота bounding boxes (4)
 - вероятность ячейки сетки, содержащей объект (1)
 - класс из N классов, к которым принадлежит обнаруженный объект
- Каждая ячейка сетки выполняет $4+1+N$ сверток, чтобы обнаружить и нарисовать bounding box для объекта.

R-CNN

Материал заимствован из статьи “Сергей Михайлин, Object Detection. Распознавай и властвуй.

[https://habr.com/ru/company/jetinfosystems/blog/498294/”](https://habr.com/ru/company/jetinfosystems/blog/498294/)

R-CNN - Region Convolution Neural Network



R-CNN (1)

Шаги алгоритма:

- ① Определение набора гипотез (регионов изображения).
- ② Извлечение из предполагаемых регионов признаков с помощью сверточной нейронной сети и их кодирование в вектор.
- ③ Классификация объекта внутри гипотезы на основе вектора из шага 2.
- ④ Улучшение (корректировка) координат гипотезы.
- ⑤ Все повторяется, начиная с шага 2, пока не будут обработаны все гипотезы с шага 1.

R-CNN (2)

1 Определение набора гипотез (регионов изображения).

- составляется набор гипотез и на основе сегментации определяются границы объектов по интенсивности пикселей, перепаду цветов, контраста и т.д.
- регионы частично перекрывают друг друга.
- гипотезы дополнительно расширяются на m пикселей во всех 4 направлениях (добавляется контекст) для уточнения

R-CNN (2)

- 2 Извлечение из предполагаемых регионов признаков с помощью сверточной нейронной сети и их кодирование в вектор.
 - Каждая гипотеза из предыдущего шага по отдельности друг от друга поступает на вход сверточной сети (AlexNet) без последнего softmax-слоя
 - Результат - кодирование изображения в **векторное представление**, которое извлекается из последнего полносвязного FC7 слоя

R-CNN (3)

3 Классификация объекта внутри гипотезы на основе вектора из шага 2.

- SVM (One vs. Rest – один против всех)
- Выход - вектор длины $+1$ по количеству классов плюс нулевой класс - фон

R-CNN (4)

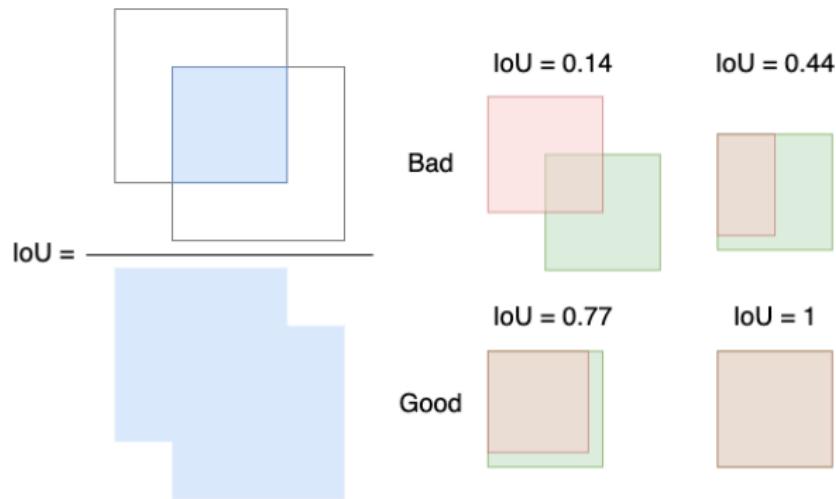
4 Улучшение (корректировка) координат гипотезы.

- Гипотезы, содержащие какой-либо объект, дополнительно обрабатываются линейной регрессией (кроме гипотез с классом «фон», там нет объекта)
- Для каждого класса используется свой регрессор.
- На входе - не вектор из слоя FC7, а карты признаков, извлеченные из последнего MaxPooling слоя, так как вектор сохраняет информацию о наличии объекта с какими-то характеристическими подробностями, а карта признаков наилучшим образом сохраняет информацию о местоположении объектов.

R-CNN - позитив. и негат. гипотезы (1)

- Проблема - несбалансированность гипотез: на картинке с единственным объектом лишь несколько гипотез из множества содержат этот самый объект $c > 0$ (**позитивные гипотезы**), а все остальные являются фоном $c = 0$ (**негативные гипотезы**)
- Метрика пересечения регионов - Intersection over Union (**IoU**) - площадь пересечения двух областей делится на общую площадь регионов

R-CNN - позитив. и негат. гипотезы (2)



R-CNN - позитив. и негат. гипотезы (3)

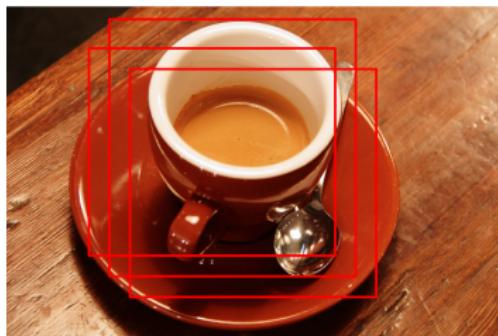
- Позитивные гипотезы – если класс определен неверно, то штраф
- Негативные? Их много. Фон (easy negative) просто классифицировать, чем другие объекты (hard negative)
 - Easy negative и hard negative определяются по пересечению ограничивающей рамки (IoU) с правильным положением объекта на изображении.
 - Если пересечения нет или оно крайне мало – это easy negative ($c = 0$), если большое – это hard negative или positive.
 - Подход Hard Negative Mining: для обучения только hard negative, поскольку, научившись распознавать их, мы автоматический добиваемся наилучшей работы с easy negative

Non-maximum suppression (1)

- Проблема - модель выделяет несколько гипотез с большой уверенностью указывающие на один и тот же объект
- Non-maximum suppression (NMS) позволяет оставить только одну, наилучшую, ограничивающую рамку.
- И на изображении может быть два разных объекта одного класса

Non-maximum suppression (2)

Before non-maximum suppression



After non-maximum suppression



Non-maximum suppression (3)

Алгоритм на одном классе:

- ➊ На вход функция принимает набор гипотез для одного класса и порог, задающий величину максимального IoU между гипотезами.
- ➋ Гипотезы сортируются по их «уверенности» (IoU).
- ➌ В цикле выбирается первая гипотеза (имеет наибольшую величину IoU) и добавляется в результирующий набор.
- ➍ В цикле выбирается вторая гипотеза (среди оставшихся после шага 3).
- ➎ Если между выбранными гипотезами IoU больше, чем выбранный порог, то вторая гипотеза отбрасывается и далее не присутствует в результирующем наборе.
- ➏ Все повторяется, начиная с шага 3, до полного перебора гипотез.

Обучение R-CNN (векторного представления)

- Предобученная на ImageNet сеть дообучается, чтобы работать с нужными классами
- Для этого изменяют размерность выходного слоя на C и обучают модифицированный вариант
- Первые слои можно заблокировать, т.к. они извлекают первичные признаки, а последующие во время обучения адаптируются под признаки нужных классов
- Когда сеть обучилась классифицировать объекты, последний слой с SoftMax активацией отбрасывается и выходом становится слой FC7, выход которого - векторное представление гипотезы
- Позитивные гипотезы - $\text{IoU} > 0.5$, остальные негативные. Мини-батч размерности 128 состоит из 32 позитивных и 96 негативных гипотез

Обучение R-CNN (классификаторы и регрессоры)

- SVMы получают на вход векторное представление гипотезы и обучаются как обычные SVM модели, но негативы берутся с $\text{IoU} > 0.3$
- Регрессии:
 - $G = (g_x, g_y, g_w, g_h)$ – правильные координаты объекта;
 - $\hat{G} = (\hat{g}_x, \hat{g}_y, \hat{g}_w, \hat{g}_h)$ – исправленное положение координат гипотез (должно совпадать с G);
 - $T = (t_x, t_y, t_w, t_h)$ – правильные поправки к координатам;
 - $P = (p_x, p_y, p_w, p_h)$ – координаты гипотезы;
- Необходимо построить преобразование P в G

Обучение R-CNN (регрессоры)

- Регрессоры - это четыре функции:
 - $d_x(P), d_y(P)$ – определяют поправки к координатам центра (x, y)
 - $d_w(P), d_h(P)$ – определяют поправки к ширине и высоте в логарифмическом пространстве
- $\varphi_5(P)$ - карта признаков, полученная из $MaxPool_5$ слоя сети, при подаче в сеть гипотезы, ограниченной координатами P .
- Преобразование P в G как:

$$\begin{aligned}\hat{g}_x &= p_w d_x(P) + p_x, \quad \hat{g}_y = p_h d_y(P) + p_y \\ \hat{g}_w &= p_w e^{d_w(P)}, \quad \hat{g}_h = p_h e^{d_h(P)}\end{aligned}$$

Обучение R-CNN (регрессоры)

- $d_*(P) = w_*^T \varphi_5(P)$ (здесь $*$ $\in (x, y, w, h)$) - линейная функция, а вектор w ищется с помощью задачи оптимизации (гребневая регрессия):

$$w_* = \arg \max_{\hat{w}_*} \sum_i^N (T_{i*} - d_*(P))^2 + \lambda \|\hat{w}_x\|^2$$

- Поправки к координатам - пары между правильным положением гипотез G и их текущем состоянием P :

$$T_x = \frac{g_x - p_x}{p_w}, \quad T_y = \frac{g_y - p_y}{p_h}$$

$$T_w = \log \frac{g_w}{p_w}, \quad T_h = \log \frac{g_h}{p_h}$$

R-CNN - недостатки

- Гипотезы на шаге 1 могут частично дублировать друг друга – разные гипотезы могут состоять из одинаковых частей и отдельно обрабатываться сетью
- Алгоритм выделения гипотез никак не обучается, а поэтому дальнейшее улучшение качества почти невозможно (есть плохие гипотезы).

Fast R-CNN - ускоренный вариант R-CNN

Алгоритм:

- ❶ Извлечение карты признаков изображения (не для каждой гипотезы по отдельности, а для всего изображения целиком)
- ❷ Поиск гипотез (аналогично R-CNN на основе Selective Search)
- ❸ Сопоставление каждой гипотезы с местом на карте признаков
- ❹ Классификация каждой гипотезы и исправление координат ограничивающей рамки

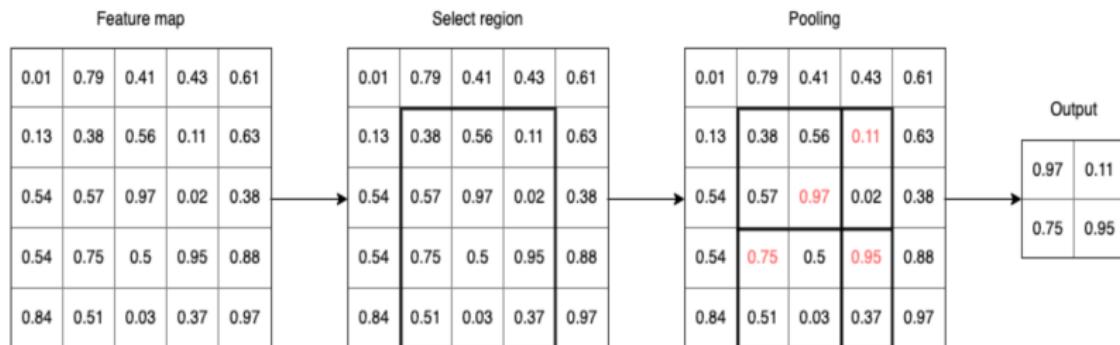
Region of Interest (RoI) слой

- RoI - позволяет один раз обрабатывать изобр. целиком сетью, получая на выходе карту признаков, которая используется для обработки каждой гипотезы
- Основная задача RoI - сопоставление координат ограничивающих рамок с соответствующими координатами карты признаков
- RoI слой подает его на вход полносвязному слою «срез» карты признаков для определения класса и поправок к координатам

RoI слой

- Как подать на вход полносвязному слою гипотезы разного размера и соотношения сторон?
- RoI слой преобразует изображение с размерами $I_h \times I_w$ в размеры $O_h \times O_w$
- Для этого исходное изображение разделяют на сетку размером $O_h \times O_w$ (размер ячейки примерно $\frac{I_h}{O_h} \times \frac{I_w}{O_w}$) и из каждой ячейки выбирают максимальное число

ReLU слой - пример



ReLU слой - пример

- Пусть имеются карта признаков размером 5×5 и нужная гипотеза на этой карте имеет координаты $(1,1,4,5)$.
- Полносвязный слой ожидает размерность 4×1 (вытянутую 2×2 матрицу).
- Поделим гипотезу на неодинаковые блоки разной размерности (Pooling) и возьмем в каждом максимальное число (Pooling и Output).
- Таким образом становится возможным обработать целиком изображение, а потом работать с каждой гипотезой на основе карты признаков.

Полносвязный слой и его выходы

- В предыдущей версии R-CNN использовались SVM-классификаторы, в Fast R-CNN они заменены одним SoftMax выходом размерности $+1$
- Выход регрессоров обрабатывается с помощью NMS (Non-Maximum Suppression)
- Результат - вероятности принадлежности гипотезы к классам и поправки к координатам ограничивающей рамки

Multi-task loss

Для задач регрессии ограничивающих рамок и классификации применяется специальная loss function:

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

u - правильный класс

L_{cls} - функция ошибки для классификации

$$L_{cls}(P, u) = -\log P_u$$

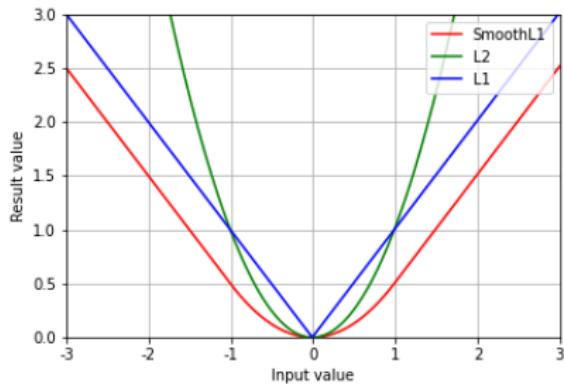
L_{loc} - является SmoothL1-функцией и измеряет разницу между $v = (v_x, v_y, v_w, v_h)$ и $t^u = (t_x^u, t_y^u, t_w^u, t_h^u)$ значениями:

$$\text{SmoothL1} = \begin{cases} \frac{1}{2}x^2, & |x| < 1 \\ |x| - \frac{1}{2}, & \text{otherwise} \end{cases}$$

Здесь x обозначает разность целевого значения и предсказания $t_i^u - v_i$.

SmoothL1

Функция SmoothL1 сочетает в себе преимущества L1 и L2 функции, является устойчивой при больших значениях и не сильно штрафует при малых значениях.



Fast R-CNN - обучение

Формирования батча:

- ① Выбирается количество гипотез в батче R
- ② Выбирается случайно изображений N
- ③ Для каждого из N изображений берется R/N гипотез (равномерно на каждое изображение)
- ④ В R включаются как позитивные (25 % всего батча), так и негативные (75 % всего батча) гипотезы.
Позитивные - $\text{IoU} > 0.5$

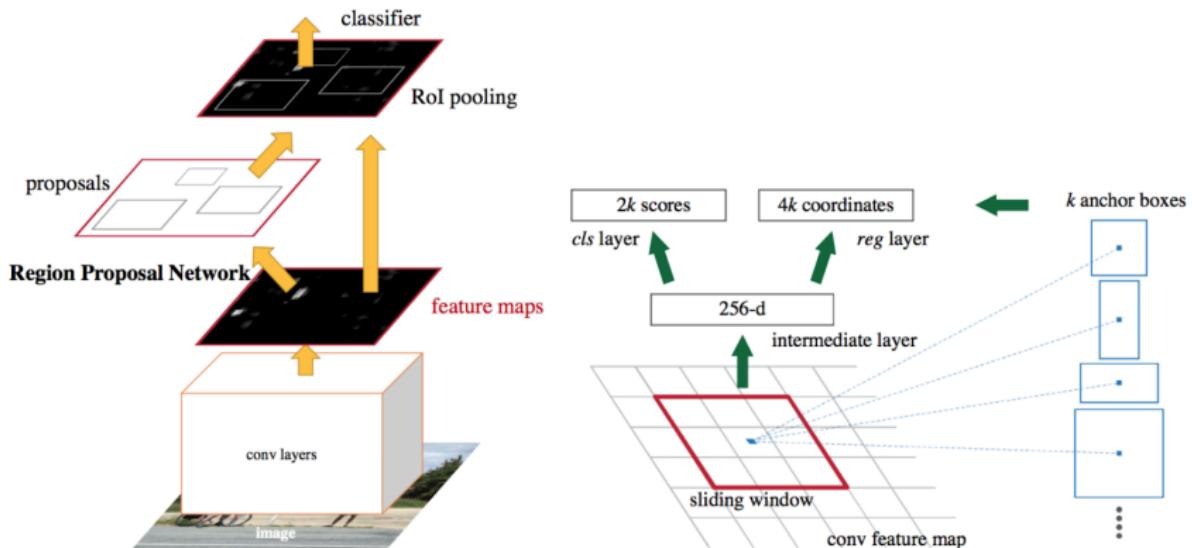
Faster R-CNN

- Улучшение - устранение зависимости от алгоритма Selective Search (в R-CNN)
- Система представляется как композиция двух модулей – определение гипотез и их обработка
- Первый модуль - Region Proposal Network (RPN)
- Второй модуль аналогично Fast R-CNN (начиная с ROI слоя)

Faster R-CNN - алгоритм

- ① Извлечение карты признаков изображения с помощью нейронной сети
- ② Генерация на основе полученной карты признаков гипотез – определение приблизительных координат и наличие объекта любого класса
- ③ Сопоставление координат гипотез с помощью RoI с картой признаков, полученной на первом шаге
- ④ Классификация гипотез (уже на определение конкретного класса) и дополнительное уточнение координат

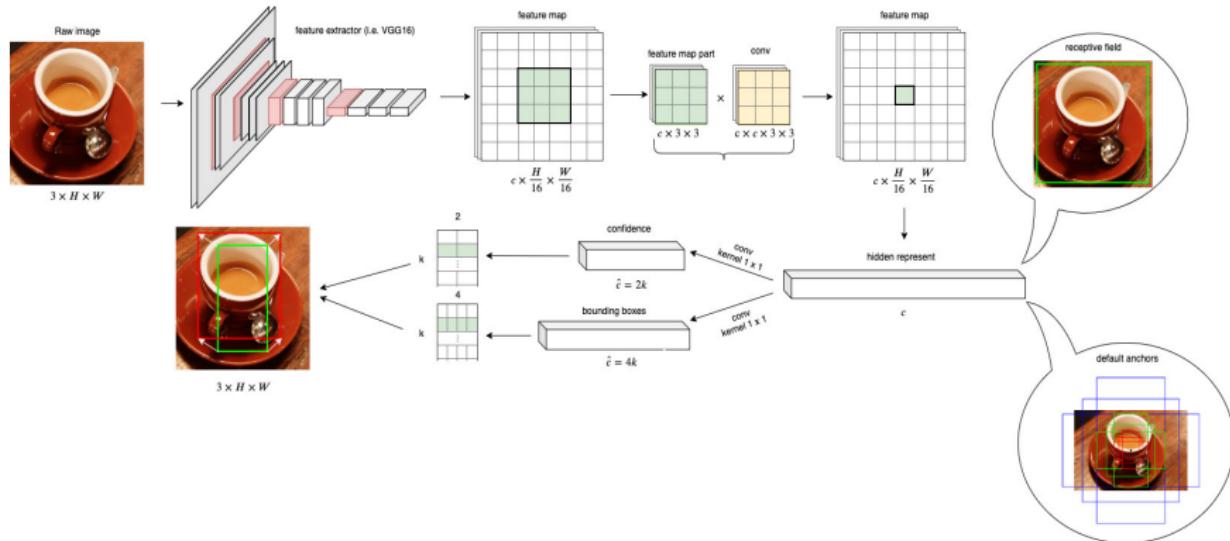
Region Proposal Network (1)



Region Proposal Network (2)

- Признаки извлекаются из сеть VGG16 с выходом - последний сверточный слой – conv5_3.
- Характеристики рецептивного поля:
 - Эффективное сжатие (effective strides,): 16
 - Размер рецептивного поля (receptive field size,): 196
- Карта признаков в 16 раз меньше изначального размера изображения, количество каналов - 512

Region Proposal Network (3)



Region Proposal Network (4)

- ➊ Получим карту признаков $c \times \frac{H}{16} \times \frac{W}{16}$ с пред. шага
- ➋ Применим сверт. слой 3x3 (отступ 1 – итоговая матрица не меняется в размерах) для доп. наращивания рецептивного поля ($P_0 = 106, r_0 = 228$). Ячейке (i, j) карты признаков соответствует вектор разм. c (512).
- ➌ К каждому вектору применимы два сверт. слоя с ядром 1x1 и кол-вом вых. каналов \hat{c} (ядро отображает размерность c в \hat{c}):
 - ➊ Первой слой (cls): параметр $\hat{c} = 2k$ – для определения вер-ти наличия (отсутствия) объекта внутри гипотезы (бинарная классификация)
 - ➋ Второй слой (reg): параметр $\hat{c} = 4k$ – для определения координат гипотез.

Region Proposal Network (5)

- Полученные вектора можно переформировать в матрицы $k \times 2$ и $k \times 4$, где строке i соответствуют значения для конкретной гипотезы
- Для правильного определения координат необходимо использовать якоря (anchors) и поправки к их координатам
- Якорем называют четырехугольник разного соотношения сторон (1:1, 2:1, 1:2) и размеров (128x128, 256x256, 512x512).
- Центром якоря считается центр ячейки ($i.j$) карты признаков

Функция потерь

- Для обучения Region Proposal Network используется следующее обозначение классов:
 - Позитивными являются все якоря, имеющие $IoU > 0.7$ или имеющие наибольшее пересечение среди всех якорей (в случае, если нет $IoU > 0.7$)
 - Негативными являются все якоря, имеющие $IoU < 0.3$
 - Все остальные якоря не участвуют в обучении
- Таким образом класс p_i^* якоря присуждается по следующему правилу:

$$p_i^* = \begin{cases} 1, & IoU > 0.7 \\ 0, & IoU < 0.3 \\ \text{nothing}, & \text{otherwise} \end{cases}$$

Функция потерь

- Функция потерь

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

- i – номер якоря; p_i – вероятность нахождения объекта в якоре;
- p_i^* – правильный номер класса;
- t_i и t_i^* – 4 предсказанные и ожидаемые (ground truth) поправки к координатам
- $L_{cls}(p_i, p_i^*)$ – бинарный log-loss; $L_{reg}(t_i, t_i^*)$ – SmoothL1 лосс
- $\{p_i\}, \{t_i\}$ – выходы классификационной и регрессионной модели
- N_{cls} и N_{loc} - размер мини-батча (256) и количество якорей

Функция потерь

Для регрессии поправок к ограничивающим рамкам:

$$t_x = \frac{(x - x_a)}{w_a}, \quad t_x^* = \frac{(x^* - x_a)}{w^*}, \quad t_y = \frac{(y - y_a)}{h_a}, \quad t_y^* = \frac{(y^* - y_a)}{h_a}$$

$$t_w = \log \frac{w}{w_a}, \quad t_w^* = \log \frac{w^*}{w_a}, \quad t_h = \log \frac{h}{h_a}, \quad t_h^* = \log \frac{h^*}{h_a}$$

x, y, w, h - центр, ширина и высота ограничивающей рамки
 x, x^*, x_a - предсказание, ground truth и значение якорей

Общее обучение сети

- ➊ Тренировка RPN. Инициализация весами на ImageNet. Дообучаем на задаче определения регионов с каким-либо классом
- ➋ Тренировка Fast R-CNN. Инициализация весами на ImageNet. Дообучаем, используя гипотезы с помощью RPN сети. Задачи - уточнение координат и определение конкретного класса объекта
- ➌ Используя веса из ш.2, обучаем только RPN часть (слои до RPN, принадлежащие feature extractor, замораживаются и никак не изменяются)
- ➍ Используя веса из ш.3, обучаем слои для Fast R-CNN (остальные веса – идущие ранее или относящиеся к RPN — заморожены)

Предсказание

- ➊ Изображение поступает на вход нейронной сети, генерируя карту признаков
- ➋ Каждая ячейка карты признаков обрабатывается RPN, результат - поправки к положению якорей и вероятность наличия объекта любого класса
- ➌ Предсказанные рамки на основе карты признаков и RoI слоя поступают на обработку Fast R-CNN
- ➍ На выходе - класс объектов и их точное положение на изображении

- ① Использование Selective Search как генератор гипотез
- ② Использование SVM + Ridge для классификации и регрессии гипотез
- ③ Запуск нейронной сети для обработки каждой гипотезы по отдельности.
- ④ Низкая скорость работы.

Fast R-CNN

- ➊ Нейронная сеть запускается только один раз на изображение – все гипотезы проверяются на основе единой карты признаков
- ➋ «Умная» обработка гипотез разного размера за счет RoI слоя
- ➌ Замена SVN на SoftMax слой
- ➍ Возможность параллельной работы классификации и регрессии.

Faster R-CNN

- ① Генерация гипотез с помощью специального отдельно дифференцируемого модуля
- ② Изменения в процессе обработки изображения, связанные с появлением RPN модуля
- ③ Самая быстрая из этих трех моделей
- ④ Является одной из самых точных и по сей день.

Вопросы

?