Introduction

Рассмотрим некоторые из задач компьютерного зрения. У нас есть задача классификации. Она говорит о том, что находится на изображении, но часто нам этого не достаточно. Мы хотим знать сколько объектов на изображении, где они находятся, потому что с этой информацией можно неплохо работать. Поэтому еще рассматриваются object detection, instance segmentation, вот те различные вариации того, как они работают. Какие плюсы и минусы. На семинаре мы обсудим то, как решать задачу детекции объектов.

Object detection

Мотивация связанная с локализацией, что мы хотим больше чем говорить, что на картинке.

- Бывает несколько объектов на картинке, причем различных классов.
- Информации, что на картинке есть кот мало. Это может быть большой кот или маленький. Это можетбыть только часть кота или целый кот. Кот может быть и внизу. и вверху, и слева. и справа. Говоря, что на картинке кот мы никак не решаем эту проблему

Problem

Как ставиться задача обучения этой модели?

У нас есть данные: картинка на вход, из которой мы должны выделить объекты и классифицировать их. Объекты мы выделяем как bounding box (коробка, внутри которой находится).

Loss function

$$\mathcal{L} = \mathcal{L}_{cls} + \mathcal{L}_{reg}$$

Обучаем мы как классификацию + регрессию на координаты bounding box (координаты центра + ширина и высота). Loss считаем для коробок, которые имеют IoU > 0.5

Metrics

Метрики качества:

- IoU
- mAP

Non-maximum suppresion

Мотивация алгоритма в следующем. Proposal network, function и так далее производит очень bounding box. И нам бы хотелось научиться фильтровать их. Поэтому используется алгоритм non-maximum suppresion. На вход confidence score, bounding box и порог фильтрации.

- Мы выбираем самый уверенный объект.
- Далее считаем IoU со всеми остальными proposal
- Убираем те proposal, у которых IoU > threshold

- Повторяем процедуру, до тех пока не останется не рассмотренным все элементы из proposal list

А что делать если у нас есть occlusion, то есть объекты накладываются друг на друга. То в этом случае у нас при применении данного алгоритма, будут выкинуты объекты. Что с этим можно сделать. Для решения проблемы предлагается SoftNMS. Вместо того, чтобы полностью выкидывать объекты. Они предлагают снижать confidence score, на величину пересечения. Таким образом, мы частично можем решить проблему

R-CNN

У R-CNN достаточно близкая идея с сегментации, которую я описал в Lecture 7. Если у нас есть какой-то классификатор, то мы можем нагенерировать области, в которых потенциально может находится объект и проверить классификатором их. По факту это работает. Но по времени занимает очень долго + картинки приходится подгонять под одинаковое разрешение. Из интересностей по сравнению с задачей классификации, тут очень много bounding box могут оказаться мусорными (в них нет объекта). Поэтому, чтобы соблюсти хотя бы некоторый баланс классов, используется hard negative sampling: 32 изображения с объектом + 96 без объекта

Fast R-CNN

R-CNN работает хорошо, но очень долго. Поэтому авторы предложили считать conv фичи не для каждого bounding box, но для изображения целиком. А выравнивать изображения под нужные bounding box можно уже внутри сети. Для того чтобы с матчить регионы разных размеров в дальнейшей обработки, нужно как-то их запулить. Тут и используется RoI pooling. Для получения координат сетки мы +- используем интерполяцию, чтобы получить маску внутри сетки.

Faster R-CNN

RoI Pooling

Чтобы еще ускорить модель. Авторы предложили Region Proposal от классических методов на нейросетевой. Для этого в Fast R-CNN была добавлена модель RPN - Region Proposal Network. Region Proposal Network устроена следующим образом: для каждого пикселя смотрим на k различных anchor boxes. Они имеют различные ratio и различные size. Таким образом мы можем выделить различные объекты на различных масштабах. Потом эти bounding boxes классифицируются на наличие или отсутствие объекта и их рамка калибруется. Приписываение градиента происходит к якорем с наибольшем IoU. Приписывание происходит к объекту с IoU > 0.7

Маски фильтруют при помощи NMS.

Ссылки

https://machinethink.net/blog/object-detection/ https://patrick-llgc.github.io/Learning-Deep-Learning/start/first_cnn_papers_notes.html#object-detection

http://vbystricky.ru/2020/08/yolo_ssd_etc.html

https://blog.paperspace.com/mean-average-precision/
https://dudeperf3ct.github.io/object/detection/2019/01/07/Mystery-of-Object-Detection/