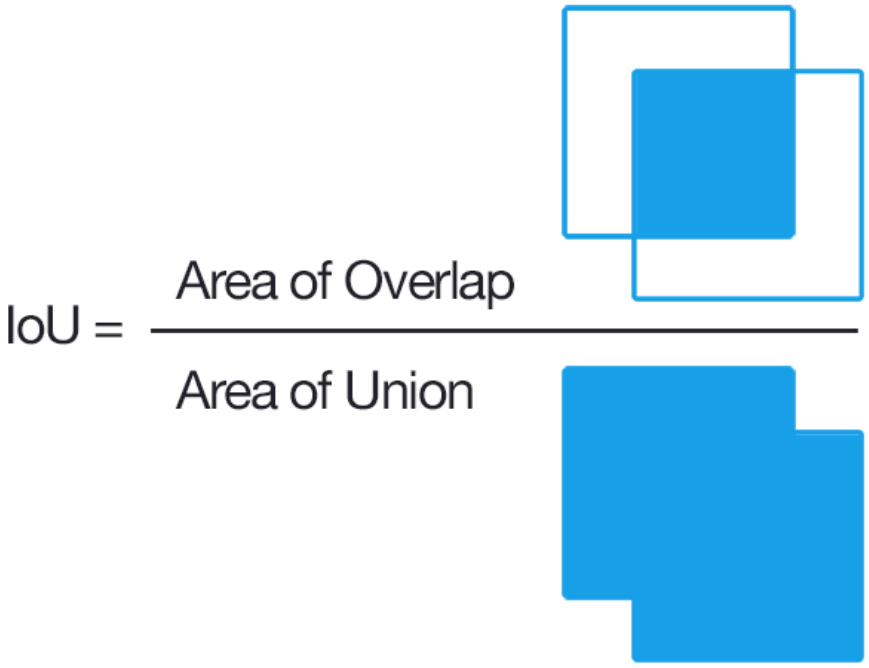
TDT4265 - Computer Vision and Deep Learning Assignment 4

Thomas Aven

March 17, 2019

1 - Object Detection Metrics

a) Intersection over union for object detection is a metric used to evaluate the accuracy of an object detector (e.g. YOLO). Given an image, an object detector will produce predictions for bounding boxes that contain objects. These predictions can be compared to so-called *ground truth* bounding boxes from a test set by means of the intersection over union. The intersection over union metric gives the area of overlap between the two bounding boxes *per area of union*. It is best illustrated with a drawing, which is taken from `https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/`:



b) A true positive is a prediction made for a class when there actually is a an object of that class. A false positive is a prediction made for a class but there is no object of that class.

c) Given true positives (TP), false positives (FP) and false negatives (FN), we have:

$$Precision = \frac{TP}{TP + FP},$$

$$Recall = \frac{TP}{TP + FN}.$$

d) We have two classes, thus we have to get the average AP (average precision) over these two classes. To calculate the AP, we iterate over recall levels in the range 0.0, 0.1, .., 1.0, and sum the largest precisions p with recall value greater than the current recall

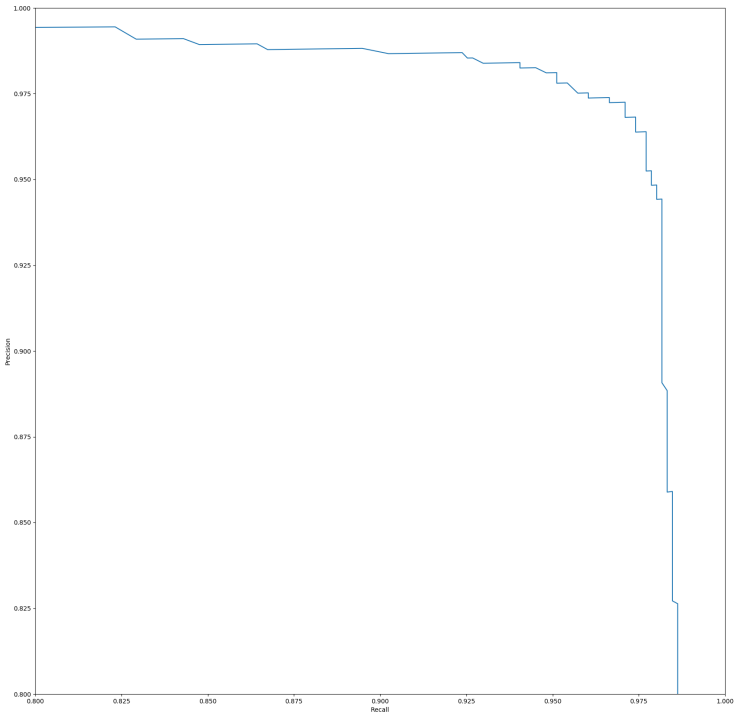
value. For the two classes, we get (where each cell is the largest p with recall bigger than or equal to the given r-value):

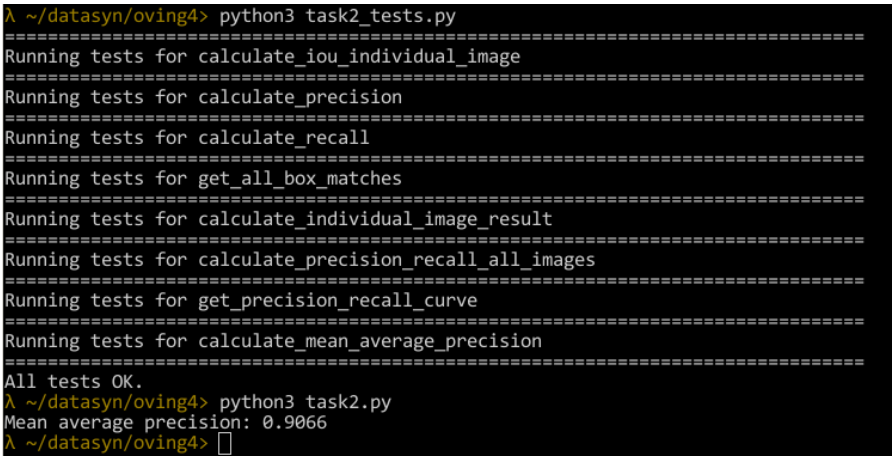
	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0	Sum
Class 1	1.0	1.0	1.0	1.0	1.0	0.5	0.5	0.5	0.20	0.20	0.20	7.1
Class 2	1.0	1.0	1.0	1.0	0.80	0.60	0.5	0.5	0.20	0.20	0.20	7.0

Which gives $AP = 7.1/11 \approx 0.65$ for the first class, and $AP = 7.0/11 \approx 0.64$ for the second class. The mAP thus becomes $mAP = (0.65 + 0.64)/2 = 0.645 \approx 0.65$.

2 - Implementing Mean Average Precision

f) Below is the final precision-recall curve from implementing all the functions (which is equal to the one given in the assignment lecture), as well as a screenshot of running `task2` and all the tests.





3 - You Only Look Once

a) There are spatial constraints on bounding box predictions because of the fact that each grid cell only predicts two boxes, and only one class. And thus, to cite the paper, «this spatial constraint limits the number of nearby objects that our model can predict. Our model struggles with objects that appear in groups, such as flocks of birds.»

The generalization constraint concerns the issue YOLO has with generalizing to objects in «new or unusual aspect ratios or configurations.» This is because of the fact that the model is trained to predict bounding boxes from data, such that images the model is not used to will hinder generalization.

b) False. YOLO does not utilize a sliding window approach to detect objects, but instead sees the entire image in a global manner.

c) Fast YOLO is similar to the standard YOLO, but created to push the boundaries of how fast proper object detection can be done. It uses fewer convolutional layers, and fewer filters in those layers. Besides the size, everything is otherwise the same (training and testing parameters, that is).

d) According to the authors, both Fast and Faster R-CNN fall short of real-time performance. «Instead of trying to optimize individual components of a large detection pipeline, YOLO throws out the pipeline entirely and is fast by design.»

However, as answered above, YOLO has some limitations regarding spacial constraints that can be mitigated with Faster R-CNN, at the cost of speed. Faster R-CNN does not have this same spatial constraints on grid cells, which may give the Faster R-CNN higher accuracy in some cases.

4 - Object detection with YOLO

I was a bit confused on this task by the fact that I thought the images were given as `x, y, width, height`, and that I thus had to translate the points to create `xmin, ymin, xmax, ymax`. When I used this translation, I ended up with only 6 boxes, but looking at `draw_boxes` in `drawing_utils.py`, I realized that the boxes were actually given as `top, left, bottom, right`, which gave the correct 7 boxes. All the other tests within the notebook ended up correct in both cases.

Below are the classified bounding boxes with scores and class names, as well as the original image overlayed with the object detections.

```
Found 7 boxes
car 0.60 (925, 285) (1045, 374)
car 0.66 (706, 279) (786, 350)
bus 0.67 (5, 266) (220, 407)
car 0.70 (947, 324) (1280, 705)
car 0.74 (159, 303) (346, 440)
car 0.80 (761, 282) (942, 412)
car 0.89 (367, 300) (745, 648)
```