TDT4265 Computer Vision and Deep Learning

Assignment 4

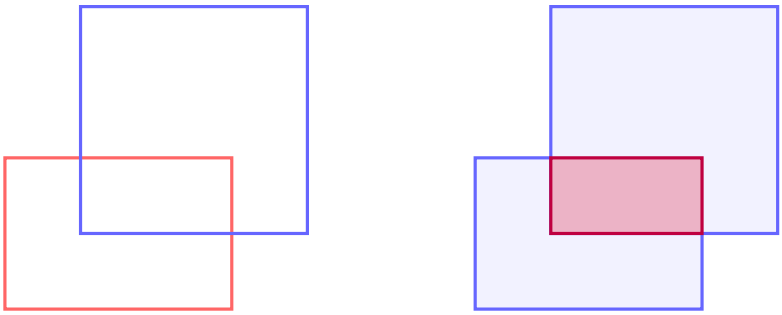
Written March 2019 By
Fredrik Veidahl Aaagaard

1 Object Detection Metrics

a)

Intersection over Union is a ratio of intersection between two bounding boxes and the union of those boxes. We can use this concept to find a measure of how good our prediction is compared to the ground truth. Figure 1 shows this concept, and we can calculate the IoU by using (1).

$$\text{IoU} = \frac{\text{Intersection}}{\text{Union}} \tag{1}$$



(a) Blue box: Ground truth
Red box: Prediction

(b) Blue area: Union
Red area: Intersection

Figure 1: Intersection over union (IoU) visualization

b)

A true positive (TP) is when the program detect the ground truth and tells us that it is the ground truth. A false positive (FP) is when the program detect something that isn't the ground truth, but tells us that it is.

c)

To define Precision and Recall we also use the definition of false negative (FN). A false negative is when the program detect something that is the ground truth and tells us that it isn't. The defintions of Precision and Recall are given in (2) and (3) respectively.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{2}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{3}$$

In (2) and (3), TP, FP, and FN means the total number of true positives, false positives, and false negatives respectively.

d)

First, the average precision (AP) for the two classes was found to be $\text{AP}_1 = 0.645$ and $\text{AP}_2 = 0.636$ using (4). The mean average precision (mAP) is then found to be $\text{mAP} = 0.641$ using (5).

$$\text{AP} = \frac{1}{11} \sum_{r} \max_{\hat{r};\hat{r}>r} p(\hat{r}) \tag{4}$$

$$\text{mAP} = \frac{1}{K} \sum_{c}^{K} \text{AP}_c \tag{5}$$

(4) and (5) were implemented in Python to calculate these numbers.

Note: When I first implemented this function, I set the recall levels according to listing 1. I then saw that the recall levels was set in Task 2 according to listing 2. Although these should be the same, I got two different results (probably due to floating point errors). I chose to stick with my original implementation using listing 1.

```
recall_levels =
np.array([0.0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1.0])
```

Listing 1: First implementation

```
recall_levels = np.linspace(0, 1.0, 11)
```

Listing 2: Implementation from task2.py

2 Implementing Mean Average Precision

a-e)

The functions given in task 2 a-e) was implemented using the starting code such that all tests was passed when running `task2_tests.py`. Running `task2.py` gave the expected result of 0.9066 mAP using an IoU threshold of 0.5.

f)

Figure 2 shows the final precision-recall curve.

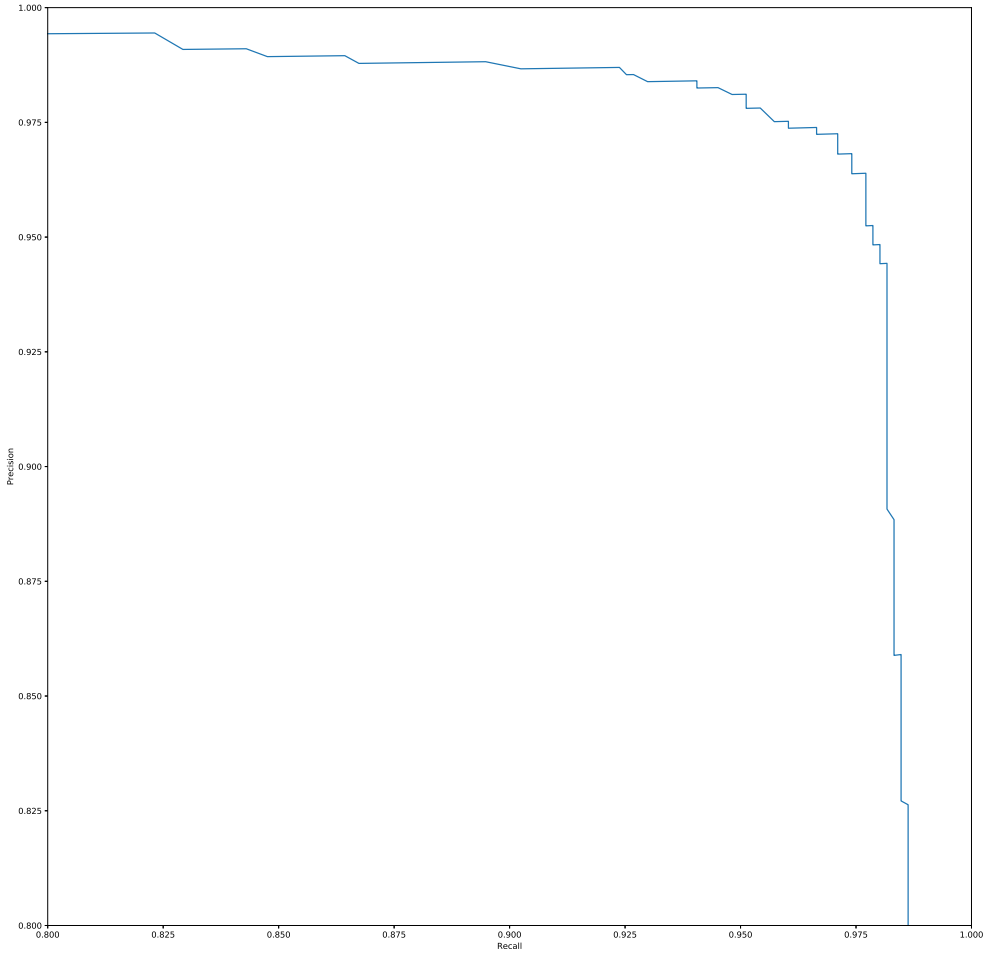


Figure 2: Precision-recall curve

3 You Only Look Once

a)

YOLO imposes spatial constraints by the fact that each grid cell only predicts two boxes and only has one class. This makes the model not suited for predicting objects that appear in groups, since it limits the number of nearby objects that can be predicted. YOLO also has limitations when it comes to generalization. This is because the model learns to predict bounding boxes from data, which makes it harder to adapt to new aspect ratios and configurations.

b)

It is false that YOLO utilizes a sliding-window approach to detect objects in an image. YOLO (You Only Look Once) sees the entire image during training and testing.

c)

Fast YOLO uses a neural network with 9 convolutional layers, instead of 24 which is used in YOLO, and it also uses fewer filters in each of these layers. Except for this, the training and testing parameters are the same for both YOLO and Fast YOLO.

d)

A strong point for Faster R-CNN is that it is more accurate than YOLO. Some weak points are that it is much slower than YOLO, and it has more background errors than YOLO. Since it is so slow, it falls short when it comes to real-time performance. Performance was tested on PASCAL VOC 2007, and a summary of these two detectors are given in Table 1.

Detector	Train	mAP	FPS
YOLO	2007+2012	63.4	45
Faster R-CNN VGG-16	2007+2012	73.2	7

Table 1: Performance of YOLO and Faster R-CNN VGG-16 on PASCAL VOC 2007.

4 Object detection with YOLO

a-c)

The functions given in task 4 a-c) was implemented in `task4.py`. The starter code was extracted from the notebook, and the `iou` function was implemented using [1] as a basis.

d)

After implementing the functions in task 4 a-c), all the tests were passed (all the outputs matched the expected outputs) and after running `yolo_eval`, the algorithm classifies 7 objects. The summary of these classifications are given in Table 2 and Figure 3 shows the final image with bounding boxes, scores and class names.

Object	Confidence	Position	Size
Car	0.60	(925, 285)	(1045, 374)
Car	0.66	(786, 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)

Table 2: Summary of classified objects. The bounding boxes of the classified objects are given by the position and size,

Figure 3: Final image with bounding boxes and class names

References

[1] H. Hukkelås, "Tdt4265_a4_task4a.py," 2019. [Online]. Available: https://gist.github.com/hukkelas/74f0420e64309a46f9751629dda710da/