TDT4265 - A4

Martin Madsen, Sondre A Bergum

March 13, 2019

1 Object Detection Metrics

1.1 a)

The Intersection over Union is a measure to quantify how good a predicted bounded box is compared to the "ground truth". The formula is straightforward; you take the intersection of the area of the two bounding boxes and divide by their union. So, in essence it is a measure of how well two bounding boxes match. Jonathan Hui has a nice picture explaining this on his blogpost[1], seen in figure 1. The IoU measure for two bounding boxes is in the range $0 \leq IoU \leq 1$ where a value of 0.5 or higher is usually a match.

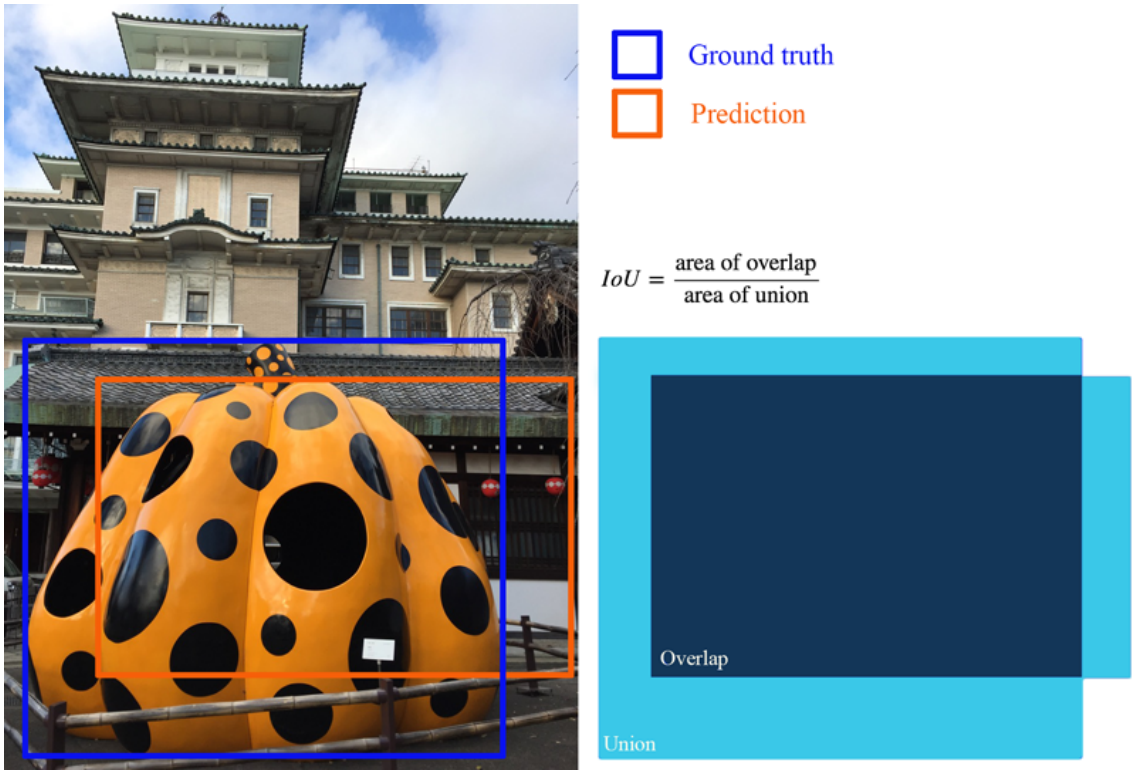

Figure 1: Illustration explaining IoU.

[1]Link to Hui's blog: mAP (mean Average Precision) for Object Detection

1.2 b)

True positive: A true positive is when a proposal is made that an image contains an object of class "C" in box "A" and the image actually contains an object of class "C" in box "A", e.g saying that an image contains a dog in the left-hand lower corner when it does.

False positive: A false positive is when a proposal is made that an image contains an object of Class "C" in box "A" but there is no object of class "C" in box "A", e.g proposing that there is a dog, but the there is no dog in the predicted bounding box. It is worth noting that if several bounding boxes classifies the same unique object as class "C", only the prediction with the highest score will count as a true positive, all the duplicates with lower scores will be false positives.

1.3 c)

Precision measures how big a percentage of the predicted boxes that are correct and is calculated as:

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

Recall is a measure of how big a percentage of the ground truth boxes we are able to detect, and is calculated as:

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

Where TP is the number of true positives, FP is the number of false positives, and FN is the number of false negative.

1.4 d)

To find the mean average precision we first find the average precision for each class, to aid in this we construct a table for the maximum recall at each recall level from the precision-recall curve:

Rank	**Precision**	**Recall**	**Recall levels**	**Max**
1	1.0	0.05	0.0	1.0
2	1.0	0.1	0.1	1.0
3	1.0	0.4	0.2	1.0
4	0.5	0.7	0.3	1.0
5	0.2	1.0	0.4	1.0
-			0.5	0.5
-			0.6	0.5
-			0.7	0.5
-			0.8	0.2
-			0.9	0.2
-			1.0	0.2

Table 1: The max recall at each recall level for class 1.

Rank	Precision	Recall	Recall levels	Max
1	1.0	0.3	0.0	1.0
2	0.8	0.4	0.1	1.0
3	0.6	0.5	0.2	1.0
4	0.5	0.7	0.3	1.0
5	0.2	1.0	0.4	0.8
-			0.5	0.6
-			0.6	0.5
-			0.7	0.5
-			0.8	0.2
-			0.9	0.2
-			1.0	0.2

Table 2: The max recall at each recall level for class 2.

From table 1 we calculate the average precision of class 1 as:

$$AP = \frac{0.2 \times 3 + 0.5 \times 3 + 1.0 \times 5}{11} = 0.645 \tag{1}$$

From table 2 we calculate the average precision of class 2 as:

$$AP = \frac{0.2 \times 3 + 0.5 \times 2 + 0.6 + 0.8 + 1.0 \times 4}{11} = 0.636 \tag{2}$$

To find the mean average precision, (mAP), we simply take the mean over all classes:

$$mAP = \sum_{c}^{K} AP_c = \frac{AP_1 + AP_2}{2} = \frac{0.645 + 0.636}{2} = 0.6405 \tag{3}$$

2 Implementing Mean Average Precision

2.1 a-e)

We implemented the earlier functions as per the assignment text. Our code passed all the tests in the test-script. We got the expected mAP of 0.9066 with an IoU-threshold of 0.5.

2.2 f)

The precision-recall curve from our dataset can be found in figure 2. We used 500 recall-levels as suggested in the code.

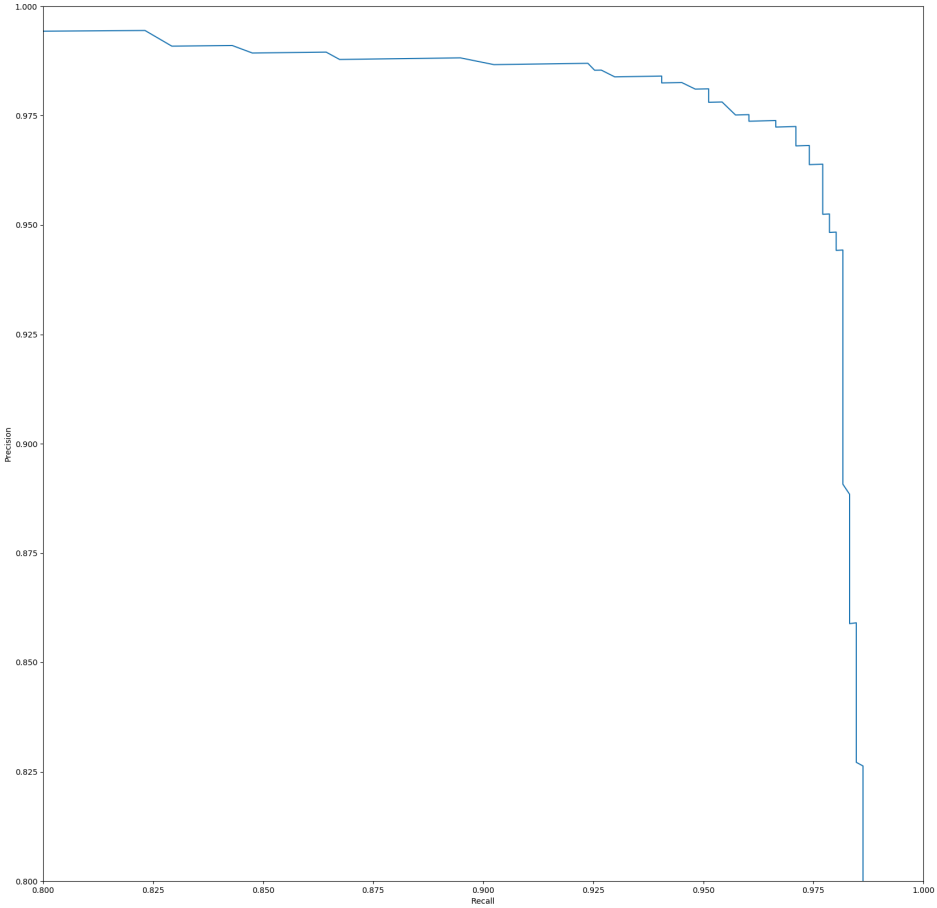

Figure 2: Precision-recall curve for the given images.

3 You Only Look Once

3.1 a)

Spatial constraints: YOLO struggles with detecting groups of small objects such as flocks of birds. The YOLO system divides each picture into an $S \times S$ grid and if the center of an object falls within a grid cell, that grid cell is responsible for detecting that object. A single grid cell can only contain 2 boxes of 1 unique class. So if a grid cell contains the center of multiple instances of an object of the same class, YOLO will only detect a maximum of 2 of them. If a grid cell contains the center of multiple objects of different classes, YOLO will only detect one of the classes.

Generalization: Since YOLO learns to predict bounding boxes from the data set it struggles to generalize this and predict bounding boxes for objects in new or unusual aspect ratios or configurations. The predicted bounding boxes rely on coarse features as the image is heavily downsampled through multiple layers.

3.2 b)

False: YOLO does not utilize a sliding-window approach. YOLO sees the entire image during training and test time so it implicitly encodes contextual information about classes as well as their appearance.

3.3 c)

FastYOLO utilizes a neural network with fewer convolutional layers, 9 instead of 24, and fewer filters in every layer compared to ordinary YOLO. Other than network size all other training and testing parameters are the same. The faster network is able to reach an fps of over 150 compared to 45 on the standard YOLO network when running on an Titan X gpu.

3.4 d)

R-CNN uses selective search for the task of generating bounding boxes, a convolutional network to extract features, an SVM to score boxes, a linear model to score boxes, and non-max suppression to remove duplicate detection. This is a very complex pipeline that needs to be tuned and tweaked individually. Faster R-CNN focus on speeding up the same pipeline and replaces the selective search with a neural network to propose regions. It does however not reduce the complexity of the pipeline, but does give increases in speed and accuracy.

In the paper presenting YOLOv1[2] they compare YOLO to the faster R-CNN VGG-16 system. Faster R-CNN VGG-16 beats YOLOv1 with a mAP of 73.2 and 63.4 respectively. The faster R-CNN VGG-16 network is far from performing in real-time though with and FPS of only 7 compared to YOLO's 45.

The spatial constraints of YOLO is not present in R-CNN. So even though the lesser number of bounding boxes makes it faster, it has the tradeoff that the number of objects it can detect depend upon the number of grid cells.

4 Object detection with YOLO

4.1 d)

After passing the predicted boxes from the test-image through score filtering and non-max suppression we are left with 7 detected objects; 6 cars and 1 bus. The corresponding bounding boxes and scores for each detected object is seen in 3. The figure 3 shows the boxes with their class and score overlayed on the test-image.

Found 7 objects:					
Class	**Score**	**X_min**	**Y_min**	**X_max**	**Y_max**
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

Table 3: The boxes that are left after filtering and non-max suppression.

[2]You Only Look Once: Unified, Real-Time Object Detection [Redmon et al., 2016]

Figure 3: Output image from the YOLO evaluation function on the provided test image.