TDT4265 Data Vision and Deep Learning

Assignment 4

Taheera Ahmed
Ingrid Kolderup

March 22, 2022

Question 1: Object Detection metrics

1a

Intersection over Union is a metric used for evaulation of the accuracy in object detection given a particular dataset. In Figure 1 we see a grounding-truth bounding boxes and predicted bounding boxes from a model. As long as these two boxes overlap the Intersection over Union can be applied.

The Intersction over Union can be found by taking the area of overlapping divded by the whole union of the overlaps.

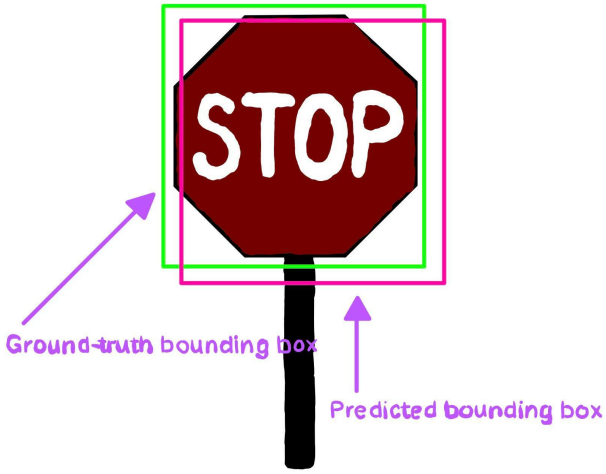


Figure 1: Intersection over Union

1b

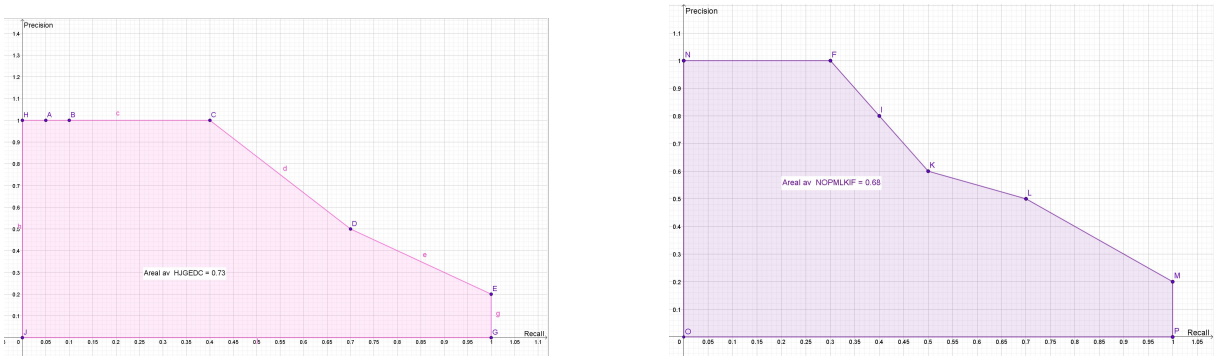
Precision is defined as seen in Equation 1, while recall is defined in Equation 2. A true positive is when the model predicts the correct class for a given input. A false positive is when the model predict the incorrect class for a given input, i.e gives a false alarm.

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{1}$$

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{2}$$

1c

The Mean Average Precision (mAP) can be found by first calculating the Average Precision (AP) for each class and then take the mean of the APs. In order to find mAP one must therefore find the APs first, which can be found by first creating a Precision-Recall graph for each class followed by taking the area beneath the curves. We used Geogebra to both plot the graphs and find the areas beneath them for our classes. The graphs and areas are represented in Figure 2.



(a) Area of the first class

(b) Area of the second class

Figure 2: The AP for our classes

$$\text{mAP} = \frac{1}{N} \sum_{i=1}^{N} \text{AP}_i \tag{3}$$

$$\text{mAP} = \frac{0.73 + 0.68}{2} = 0.705 \tag{4}$$

The resulting areas for the first and second class is 0.73 and 0.68 respectively. By using the formula in Equation 3 to calculate the mAP, we finally get 0.705 as expressed in Equation 4.

Question 2: Implementing Mean Average Precision

2f

In Figure 3 our final precision recall curve can be seen.

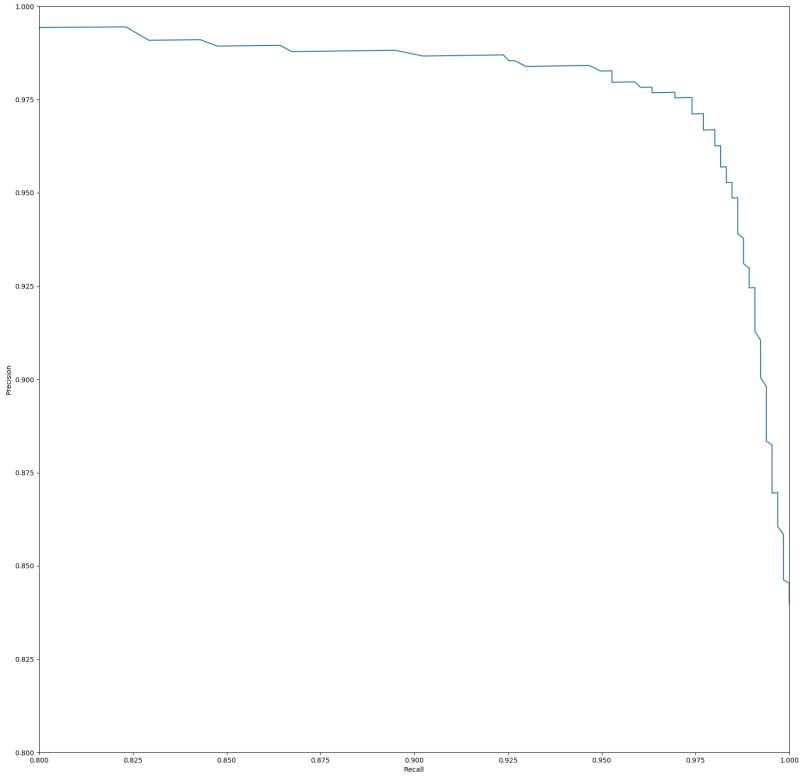


Figure 3: Precision-recall curve for task 2

Question 3: Theory

3a

When performing inference with SSD, we need to filter out a set of overlapping boxes. What is this filtering operation called?

This filtering operation is called non-maximum suppression aka. nms

3b

Is the following true or false: Predictions from the deeper layers in SSD are responsible to detect small objects.

`True`

3c

Why do they use different bounding box aspect ratios at the same spatial location? We do this to make the model better prepared for detecting different type of objects without training. If we have different type of sizes of the default boxes, the model can capture a wider specter of objects from the start. This will give the model training a head start, because it is now capable of detecting a various number of objects types and are not limited to certain types.

For instance: A car might go into a box with a small height and large width, while a person fit better into a box with a high height and a low width. If both of these objects are capture in the default boxes, the training would start of much easier compared to a situation where the objects did not fit into the the default box. The chances of a head start is obviously better with different box sizes and that is why the SSD model have different bounding boxes at the same spatial location.

3d

What is the main difference between SSD and YOLOv1/v2 (The YOLO version they refer to in the SSD paper)?

The main difference between SSD and YOLO is that the SSD model will add several feature layers at the end of the base network. This predict the offset belonging to the default boxes of both different scales and ratios. Meaning the usage of convolutional default boxes given multiple feature maps is different. Also the matching strategy used during training is also different from YOLO.

3e

How many anchors boxes do we have in total for this feature map? For each cell in the feature map we have 6 different anchors. This gives a total of $38 \cdot 38 \cdot 6 = 8664$

3f

How many anchors boxes do we have in total for the entire network? $(38 \cdot 38 \cdot 6) + (19 \cdot 19 \cdot 6) + (10 \cdot 10 \cdot 6) + (5 \cdot 5 \cdot 6) + (3 \cdot 3 \cdot 1) + (1 \cdot 1 \cdot 6) = 11640$

Question 4: Implementing Single Shot Detector

4a

4b

Our final result after training can be seen in Listing 1, Figure 4 and Figure 5. Our mean average precision was 0.789.

```
2022-03-17 15:12:36,285 [INFO ] metrics/mAP: 0.583, metrics/mAP@0.5: 0.789,
    metrics/mAP@0.75: 0.707, metrics/mAP_small: 0.403, metrics/mAP_medium: 0.720,
    metrics/mAP_large: -1.000, metrics/average_recall@1: 0.413,
    metrics/average_recall@10: 0.622, metrics/average_recall@100: 0.622,
    metrics/average_recall@100_small: 0.434, metrics/average_recall@100_medium:
    0.764, metrics/average_recall@100_large: -1.000,
2022-03-17 15:12:38,100 [INFO ] Saved model to: outputs/ssd300/checkpoints/16224.ckpt
```

Listing 1: Terminal output of after training

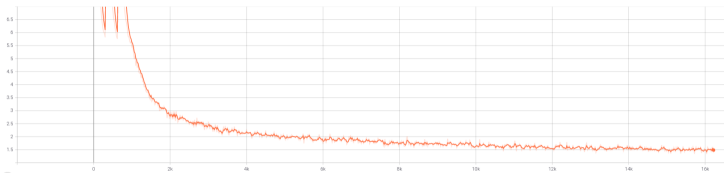


Figure 4: Total loss

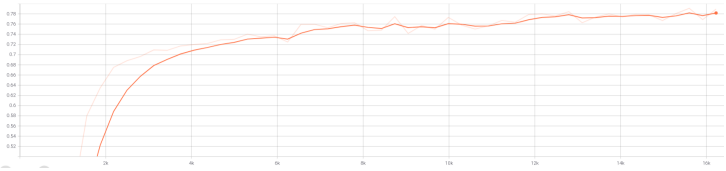


Figure 5: Mean Average Precision

4c

Our final result after training can be seen in Listing 2, Figure 6. Our mean average precision was 0.8585.

```
2022-03-18 10:19:13,224 [INFO ] metrics/mAP: 0.625, metrics/mAP@0.5: 0.858,
    metrics/mAP@0.75: 0.770, metrics/mAP_small: 0.543, metrics/mAP_medium: 0.686,
    metrics/mAP_large: -1.000, metrics/average_recall@1: 0.427,
    metrics/average_recall@10: 0.673, metrics/average_recall@100: 0.673,
    metrics/average_recall@100_small: 0.595, metrics/average_recall@100_medium:
    0.731, metrics/average_recall@100_large: -1.000,
2022-03-18 10:19:16,838 [INFO ] Saved model to: outputs/ssd300/checkpoints/5928.ckpt
```

Listing 2: Terminal output of after training our improved network

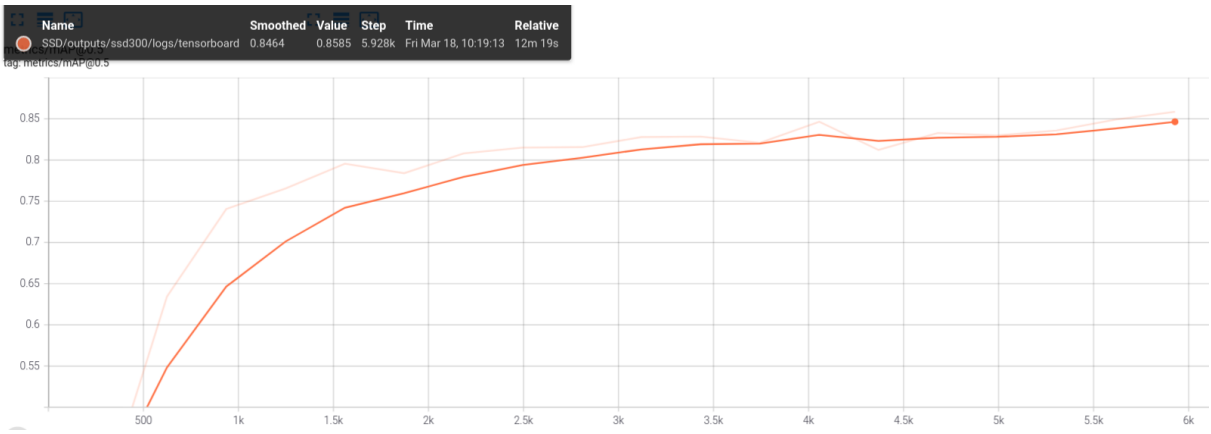


Figure 6: Mean Average Precision

4d

Since the feature map has resolution 5×5 we get strides of 64×64. This can be seen in `configs/ssd300.py` where the $feature_{size}$ list has $[5, 5]$ on index 3. The corresponding tile size for this index is found in the *stride* list and we get $[64, 64]$. This gives us the following 25 center points:

- Row 0: (32, 32), (32, 96), (32, 160), (32, 224), (32, 288)
- Row 1: (96, 32), (96, 96), (96, 160), (96, 224), (96, 288)
- Row 2: (160, 32), (160, 96), (160, 160), (160, 224), (32, 288)
- Row 3: (224, 32), (224, 96), (224, 160), (224, 224), (228, 288)
- Row 4: (288, 32), (288, 96), (288, 160), (288, 224), (288, 288)

We calculated the center point for the tiles by first calculating the center of the first tile with size 64×64 and then applying 64 in x direction and y direction to get all 25 center points. The center point of the first tile is in the middle of the tile, which is $\frac{64}{2} = 32$ in both x and y direction, hence we get $(32, 32)$.

4e

NA

4f

NA