TDT4265 - Computer Vision and Deep Learning
Assignment 4

Jakob Vahlin
Kristian Stensgård

Contents

1 Task 1: Object Detection Metrics

1.1 Task a: Intersection over union

The intersection over union is a measurement of the overlap between two boundaries. This is used to determine how much of the predicted boundary overlaps with the actual boundary. The measurement is defined as the ratio between the area of the intersection between the two boundaries, and the area of the union over the two boundaries. An illustration of intersection of union is given in Figure 1

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

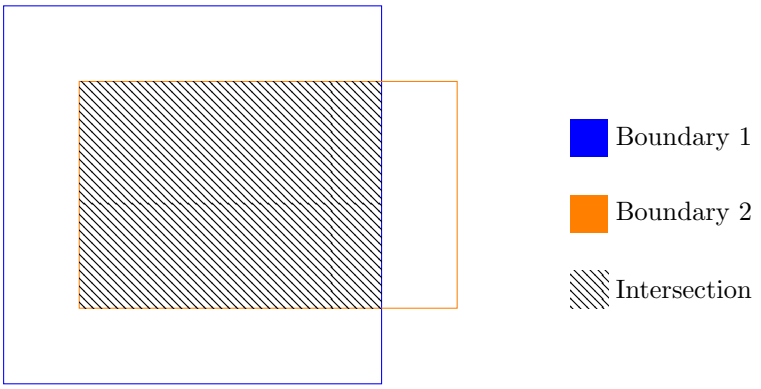


Figure 1: An illustration of intersection over union.

1.2 Task b: Precision & Recall

A *true positive* is a prediction that was predicted as positive, and was correct. A *false positive* is a prediction that was predicted as positive, but was not correct.

Precision is a measurement of the accuracy of the predictions. It measures to what extent the predictions were correct. It is defined as the fraction between the true positives and the sum of the true and false positives, defined in Equation 2.

$$\text{Precision} = \frac{\text{Number of True Positives}}{\text{Number of True Positives} + \text{Number of False Positives}} \tag{2}$$

Recall is a measurement of the extent the model is able to determine the positives. It is defined as the ratio between the True Positive predictions and the sum of the True Positive predictions and False negative predictions, where a *false negative* is a negative prediction that was wrong. Mathematically, recall is defined in Equation 3.

$$\text{Recall} = \frac{\text{Number of True Positives}}{\text{Number of True Positives} + \text{Number of false positives}} \tag{3}$$

1.3 Task c:

Given the precision and recall curves defined in Equation 4 and Equation 5, the mean average precision (mAP) is defined as the mean of the average precision for each of the curves. Thus, the first step is to calculate the average precision (AP) for each of the precision and recall curves.

$$P_1 = [1.0, 1.0, 1.0, 0.5, 0.2], \quad R_1 = [0.05, 0.1, 0.4, 0.7, 1.0] \tag{4}$$

$$P_2 = [1.0, 0.8, 0.6, 0.5, 0.2], \quad R_2 = [0.3, 0.4, 0.5, 0.7, 1.0] \tag{5}$$

To calculate the AP for each of the curves, interpolation is used to fit the precision values on the recall interval $R_{1,2\text{interpol}} = [0.0, 0.1, \ldots, 1.0]$. For a given recall, r_i, the precision value is replaced with the maximum precision value for any recall $r \geq r_i$. The interpolated precision values with the corresponding recall interval, are given in Equation 6 and (7) for curves 1 and 2 respectively.

$$\begin{aligned} P_{1\text{interpol}} &= [1.0, 1.0, 1.0, 1.0, 1.0, 0.5, 0.5, 0.5, 0.2, 0.2, 0.2] \\ R_{1\text{interpol}} &= [0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0] \end{aligned} \tag{6}$$

$$\begin{aligned} P_{2\text{interpol}} &= [1.0, 1.0, 1.0, 1.0, 0.8, 0.6, 0.5, 0.5, 0.2, 0.2, 0.2] \\ R_{2\text{interpol}} &= [0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0] \end{aligned} \tag{7}$$

Thus the AP for the respective curves is found by computing the average of both sets of precision values.

$$\begin{aligned} AP_1 &= \frac{1}{11} \sum_{i=1}^{11} P_{1i} \\ &= \frac{(5 \cdot 1) + (3 \cdot 0.5) + (3 \cdot 0.2)}{11} \\ &= 0.645 \end{aligned} \tag{8}$$

$$\begin{aligned} AP_2 &= \frac{1}{11} \sum_{i=1}^{11} P_{2i} \\ &= \frac{(4 \cdot 1) + (1 \cdot 0.8) + (1 \cdot 0.6) + (2 \cdot 0.5) + (3 \cdot 0.2)}{11} \\ &= 0.636 \end{aligned} \tag{9}$$

Finally, the mean average precision is found by computing the average of the two values found in Equation 8 and Equation 9.

$$\text{mAP} = \frac{0.636 + 0.645}{2} = 0.641 \tag{10}$$

2 Task 2: Implementing Mean Average Precision

2.1 Task f

When running the task2.py script the mean average precision was 0.9835. The precision-recall curve for this task is plotted in Figure 2.

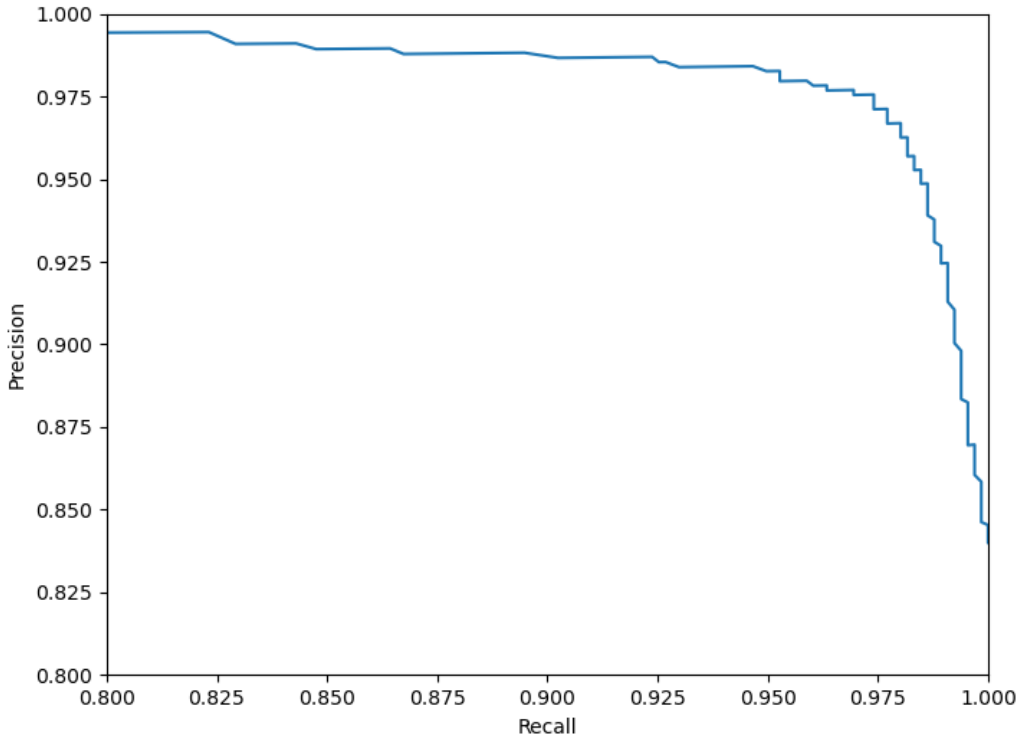


Figure 2: Precision-recall curve for task 2

3 Task 3: Theory

3.1 Task a

SSD uses non-maximum suppression to remove overlapping boxes. If these boxes were not removed, then SSD would have duplicate predictions pointing to the same object.

3.2 Task b

False

The feature maps with highest resolution are responsible for detecting small object. Conversely the lower resolution feature maps are used for detecting large objects. The high resolution feature maps are at the top/beginning of the network and this is where detection of small objects are done. The deep layers towards the end of the network are responsible for detection of the large objects.

3.3 Task c

If the bounding box aspect ratios were all the same, then the model will become good at predicting one category corresponding to a default box with that shape. However, predictions for objects with others shapes corresponding to other box aspect ratios would be poor. This will make initial training poor.

By using different aspect ratios one can improve training and the models ability to detect object of different sizes.

3.4 Task d

The SSD model adds several feature layers to the end of a base network, which predict the offsets to default boxes of different scales and aspect ratios and their associated confidences. The YOLO model uses a CNN feature extractor, much like the SSD model.

The YOLO however does regression using two convolutional layers at the end to make boundary box predictions.

3.5 Task e

Given a feature map with spatial dimensions 38×38 and 6 anchors with different aspect ratios at each anchor location we get:

$$38 \cdot 38 \cdot 6 = 8,664 \tag{11}$$

So in total we 8,664 anchor boxes for this feature map.

3.6 Task f

Considering a newtwork that predicts at the resolutions 38×38, 19×19, 10, 5×5, 3×3 and 1×1 with 6 different aspect ratios at every anchor point, the total number of anchor boxes is:

$$6 \cdot (38^2 + 19^2 + 10^2 + 5^2 + 3^2 + 1^2) = 11,640 \tag{12}$$

4 Task 4: Implementing Single Shot Detector

4.1 Task b

The network mentioned outlined in Table 1 was implemented as the backbone for the SSD described in the assignment description.Each convolutional layer has a kernel size of 3×3 and padding of 1. Each MaxPool2D layer has a kernel size of 2×2. The goal was to achieve a mAP value in the range $[75\%, 77\%]$ within 6 000 iterations. The goal was accomplished, with the model reaching a mAP of **75.7**% after 6 000 iterations.

The total loss and mAP is plotted against the training iterations in Figure 4 and Figure 3 respectively.

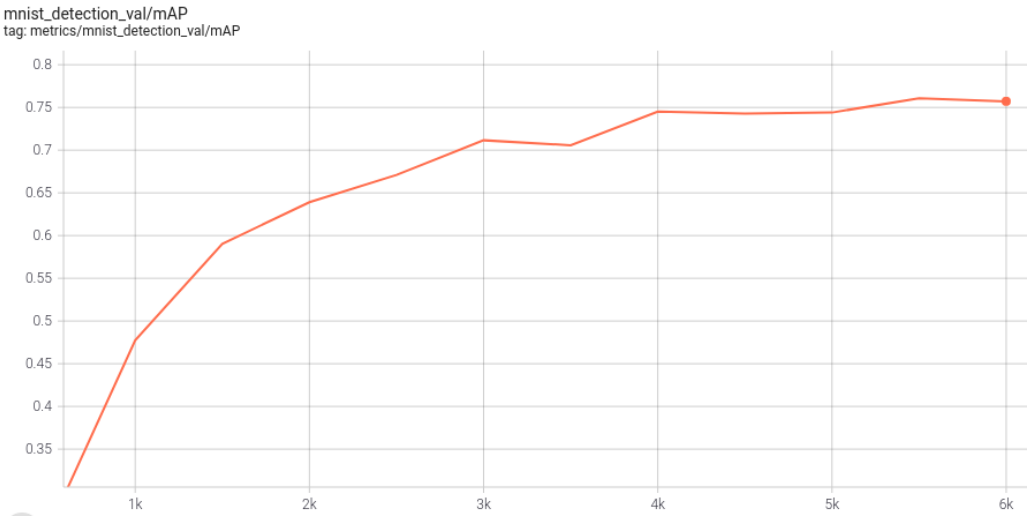


Figure 3: mAP for the SSD using the backbone outlined in ...

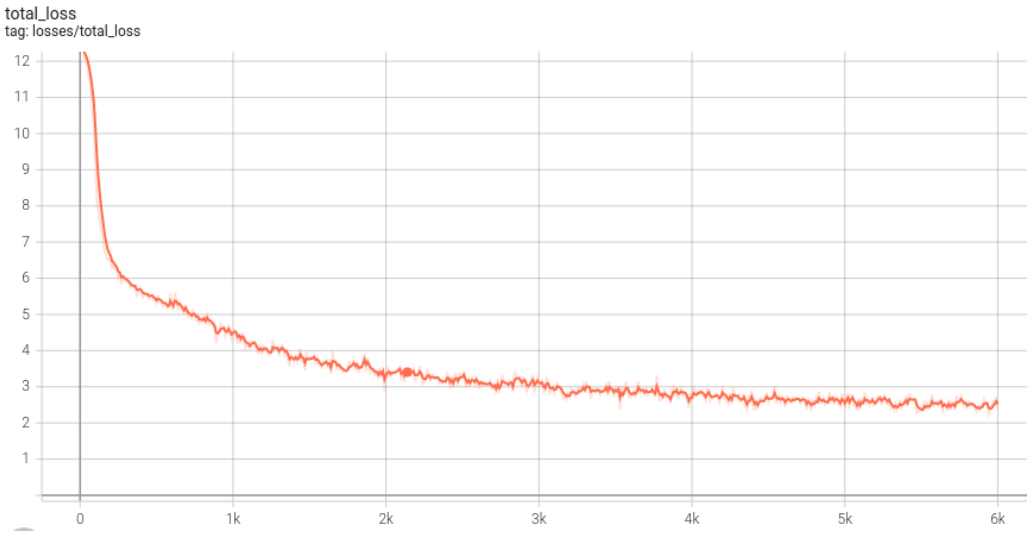


Figure 4: Total loss for the SSD using the backbone outlined in ...

Table 1: The network used as the backbone for the SSD in task 4b. Each convolutional layer has a kernel size of 3×3 and padding of 1. Each MaxPool2D layer has a kernel size of 2×2. ***The last convolution have a stride of 1, padding of 0, and a kernel size of 3 × 3***

Is Output	Layer Type	Number of Filters	Stride
	Conv2D	32	1
	MaxPool2D	-	2
	ReLU	-	-
	Conv2D	64	1
	MaxPool2D	-	2
	ReLU	-	-
	Conv2D	64	1
	ReLU	-	-
Yes - Resolution: 38×38	Conv2D	128	2
	ReLU	-	-
	Conv2D	128	1
	ReLU	-	-
Yes - Resolution: 19×19	Conv2D	256	2
	ReLU	-	-
	Conv2D	256	1
	ReLU	-	-
Yes - Resolution: 9×9	Conv2D	128	2
	ReLU	-	-
	Conv2D	128	1
	ReLU	-	-
Yes - Resolution: 5×5	Conv2D	128	2
	ReLU	-	-
	Conv2D	128	1
	ReLU	-	-
Yes - Resolution: 3×3	Conv2D	64	2
	ReLU	-	-
	Conv2D	128	1
	ReLU	-	-
Yes - Resolution: 1×1 *	Conv2D	64	1

4.2 Task c

In order to increase the mean average precision of the detector from task 4b to reach a mAP value of 85% within 10 000 iterations, modifications and additions to the model have been implemented.

Based on experience from Assignment 3, the first attempt at increasing the mAP involved adding more data augmentation, as this was the modification that resulted in the biggest performance boost in Assignment 3. However, augmenting the dataset by adding flipping and rotation to the training set resulted in poor performance. Thus no data augmentation was used in the final successful model.

The number of filters in the convolutional network was extended, motivated by the success when extending the layers from Assignment 3. More specifically, the number of filters in the third convolutional layer was extended from 128 to 256. This change increased the performance.

Furthermore, reducing the learning rate from $2 \cdot 10^{-3}$ to $1.2 \cdot 10^{-3}$ as well as adding batch normalization at certain layers also slighly improved the performance.

The final network architecture of the improved network is outlined in Table 2.

The final change to the SSD, that also yielded the biggest performance leap, was altering the sizes of the default boxes. This was done after it was discovered that the detector performed poorly on smaller objects. In an effort to improved the detection on smaller objects, the minimum and maximum dimensions of the first default box was changed from $[30, 30]$ to $[15, 15]$ and $[60, 60]$ to $[30, 30]$ respectively.

With these changes the model reached a mean average precision of **87.93**% after 10 000 iterations, exceeding the desired target value of 85%. The desired target value of 85% was exceed after 5000 gradient descent iterations, with a mean average precision of 85.7%. Plots showing the mean average precision and total loss are shown in Figure 5 and Figure 6 respectively.

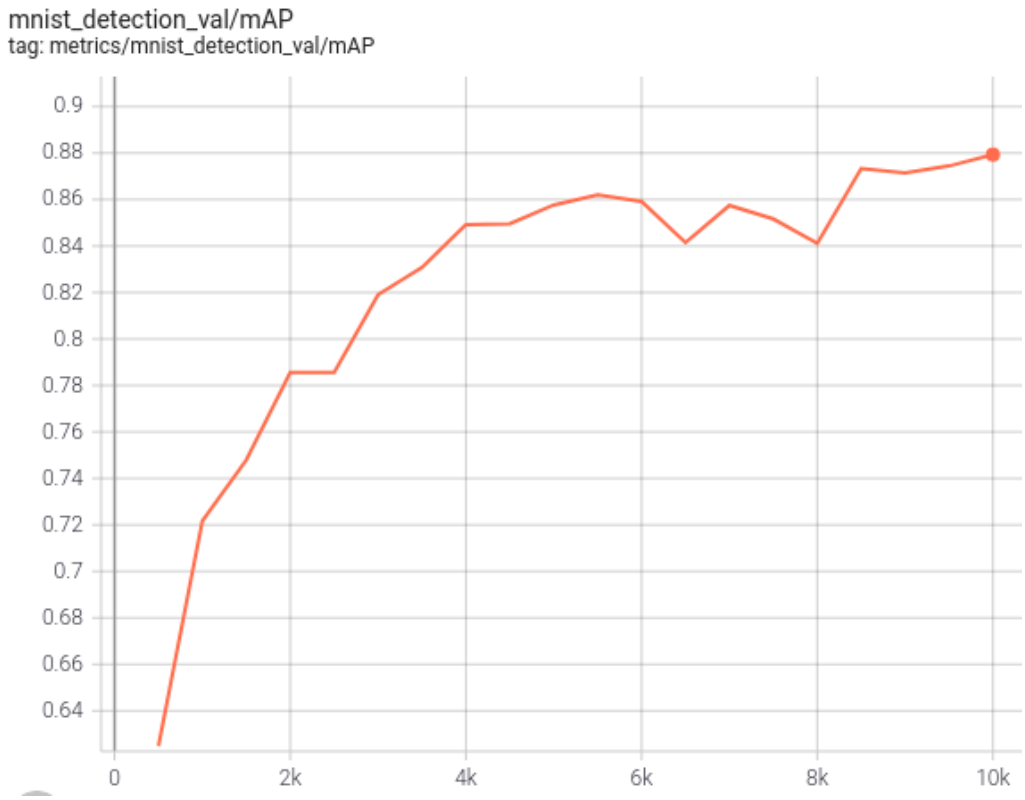


Figure 5: The development of mean average precision values during training.

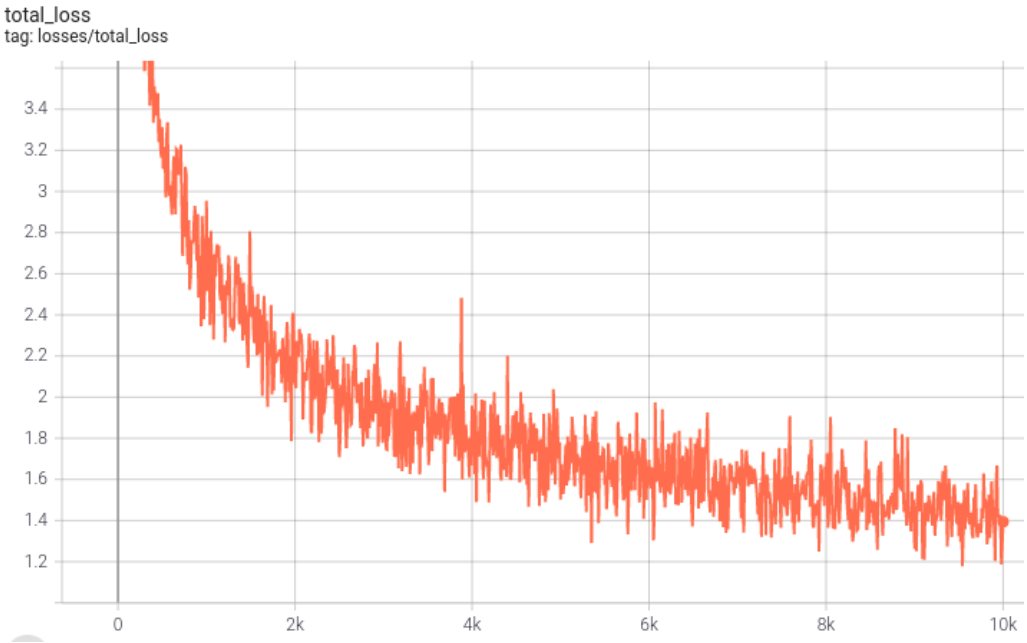


Figure 6: The development of the training loss values during training.

Table 2: The modified network used as the backbone for the SSD in task 4c. Each convolutional layer has a kernel size of 3×3 and padding of 1. Each MaxPool2D layer has a kernel size of 2×2. ***The last convolution should have a stride of 1 and padding of 0***

Is Output	Layer Type	Number of Filters	Stride
	Conv2D	32	1
	MaxPool2D	-	2
	ReLU	-	-
	Conv2D	64	1
	MaxPool2D	-	2
	ReLU	-	-
	Conv2D	128	1
	BatchNorm2D	-	-
	ReLU	-	-
Yes - Resolution: 38×38	Conv2D	128	2
	ReLU	-	-
	Conv2D	128	1
	ReLU	-	-
	BatchNorm2D	-	-
Yes - Resolution: 19×19	Conv2D	256	2
	ReLU	-	-
	Conv2D	256	1
	ReLU	-	-
	BatchNorm2D	-	-
Yes - Resolution: 9×9	Conv2D	256	2
	ReLU	-	-
	Conv2D	256	1
	ReLU	-	-
	BatchNorm2D	-	-
Yes - Resolution: 5×5	Conv2D	256	2
	ReLU	-	-
	Conv2D	128	1
	ReLU	-	-
	BatchNorm2D	-	-
Yes - Resolution: 3×3	Conv2D	64	2
	ReLU	-	-
	Conv2D	128	1
	ReLU	-	-
	BatchNorm2D	-	-
Yes - Resolution: 1×1 *	Conv2D	64	1

4.3 Task d

Having reached a mAP of 87.93% the model was further improved in an effort to reach a mean average precision of 90% or greater within 15 000 timesteps.

It was found that the only modification needed to surpass the target value was to further adjust the minimum and maximum values of the default boxes, as discussed in task 4c. Instead of just changing the first default box, the first 3 default boxes was modified. The changes to first 3 default boxes are shown in Table 3. The rest of the network is identical to the one described in task 4c. The network reached 90.2% mAP on the validation set after training was done.

Plots showing the mean average precision and total loss are shown in Figure 7 and Figure 8 respectively.

Table 3: Original and modified minimum and maximum sizes of the first 3 default boxes in the SSD. The other 3 default boxes have not been modified and are left out.

Box	Original size	Modified size
1	MIN: [30,30] MAX:[60,60]	MIN: [15,15] MAX:[30,30]
2	MIN: [60,60] MAX: [111,111]	MIN: [30,30] MAX: [50,50]
3	MIN: [111,111] MAX: [162,162]	MIN: [80,80] MAX: [130,130]

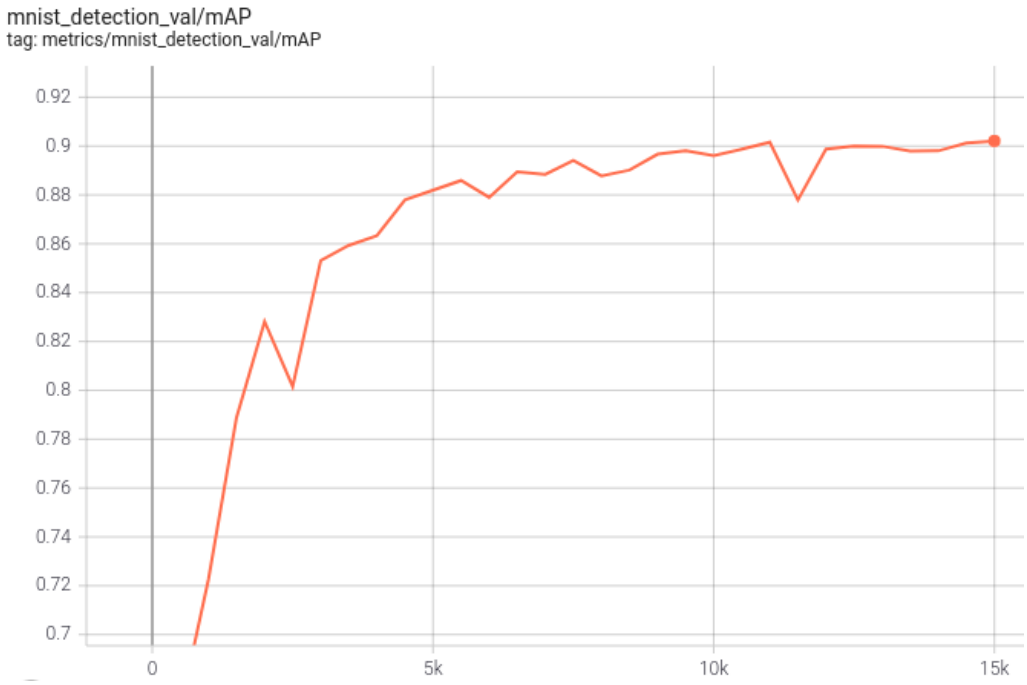


Figure 7: The development of mean average precision values during training for the improved model.

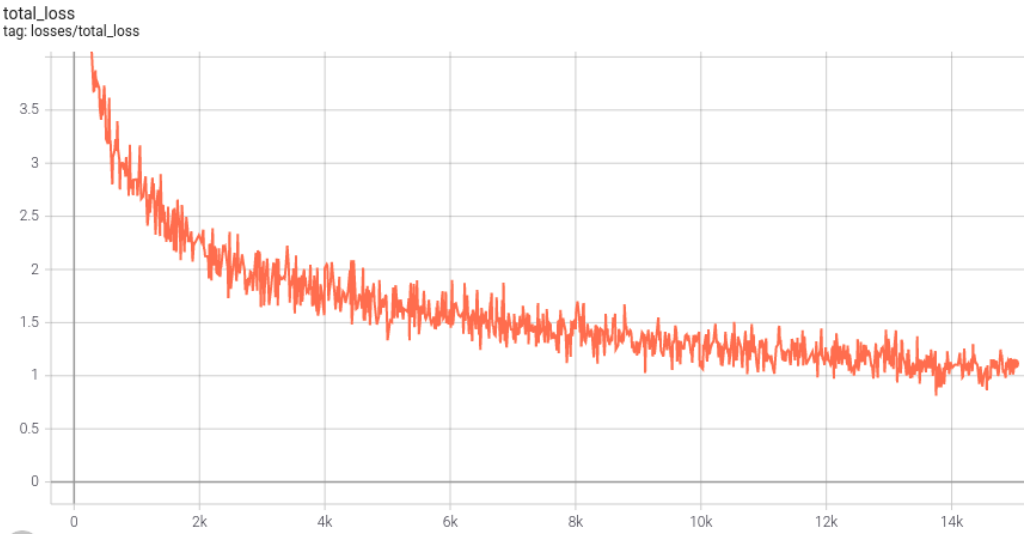


Figure 8: The development of total loss values during training for the improved model.

4.4 Task e

The trained model from the previous task that reached 90.2% mAP was tested on 15 of the images from the yymnist validation set. The score threshold used for this test was 0.7. The resulting prediction boxes and their category was placed on top of these images. The results are shown in Figure 9 to Figure 16. In these images we see that the detector performs quite well, but struggles a bit with numbers that are very close to each other, like in the right image in Figure 14. Some digits are quite obvious for us humans, but apparently the detector fails to detect the 7 and 6 in the right images in Figure 12 and Figure 13. There are also more examples of clearly distinguishable numbers that have not been detected. Lowering the score threshold will most likely give better results.

Figure 9: Two of the test images side by side

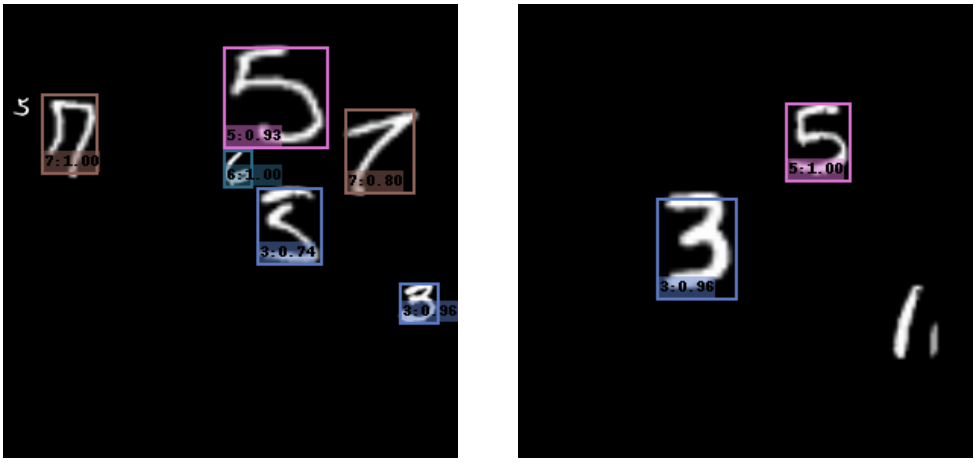


Figure 10: Two of the test images side by side

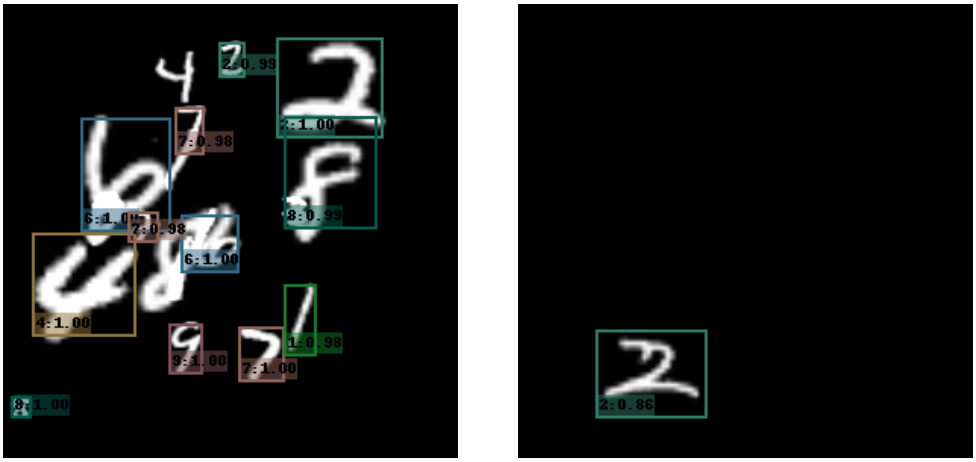


Figure 11: Two of the test images side by side

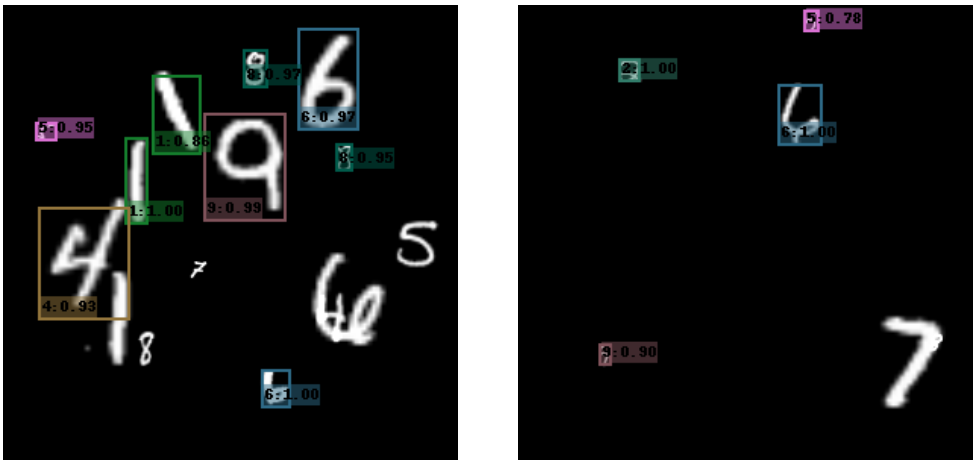


Figure 12: Two of the test images side by side

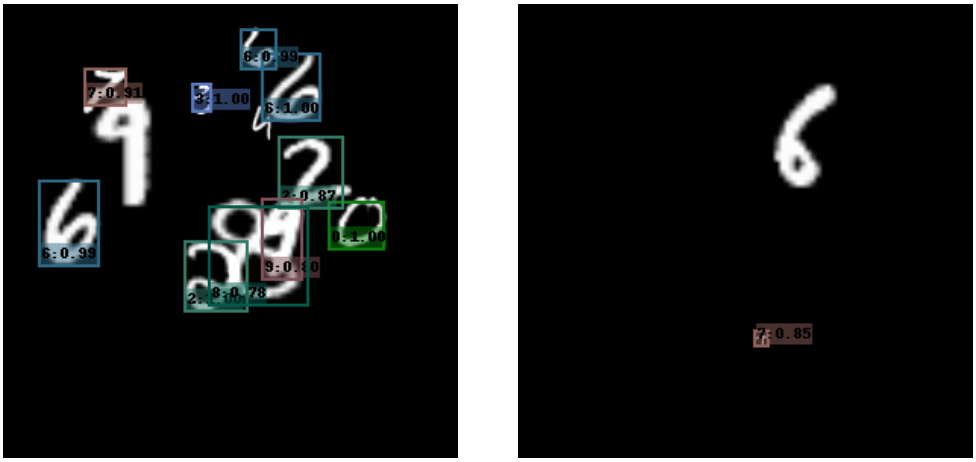


Figure 13: Two of the test images side by side

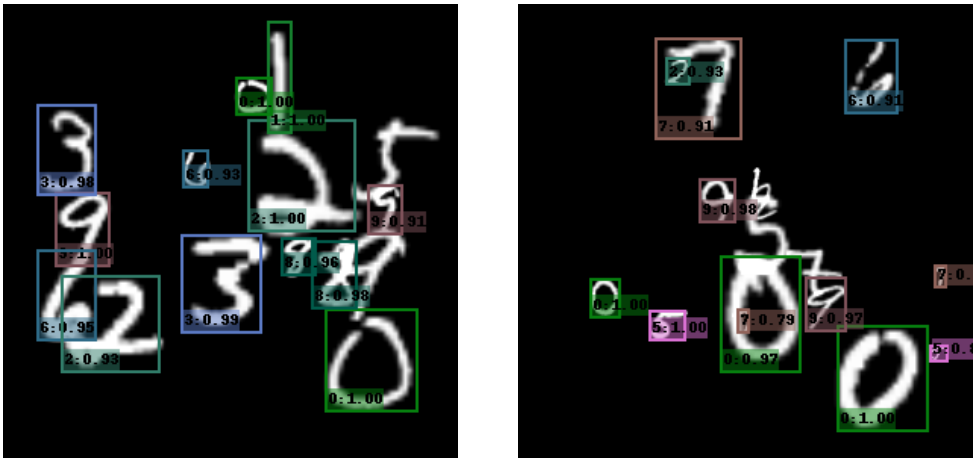


Figure 14: Two of the test images side by side

Figure 15: Two of the test images side by side

Figure 16: The final test image

4.5 Task f

For the last task a the pretrained network VGG16 was used as a backbone for the detector. The detector was trained for 5000 gradient descent iterations on the VOC dataset. The mAp and total loss is shown in Figure 17 and Figure 18.

The trained detector was tested on five images from the PASCAL VOC dataset and the resulting predicted boxes and their category is shown in Figure 19, Figure 20 and Figure 21. The score threshold used for this test was 0.2.

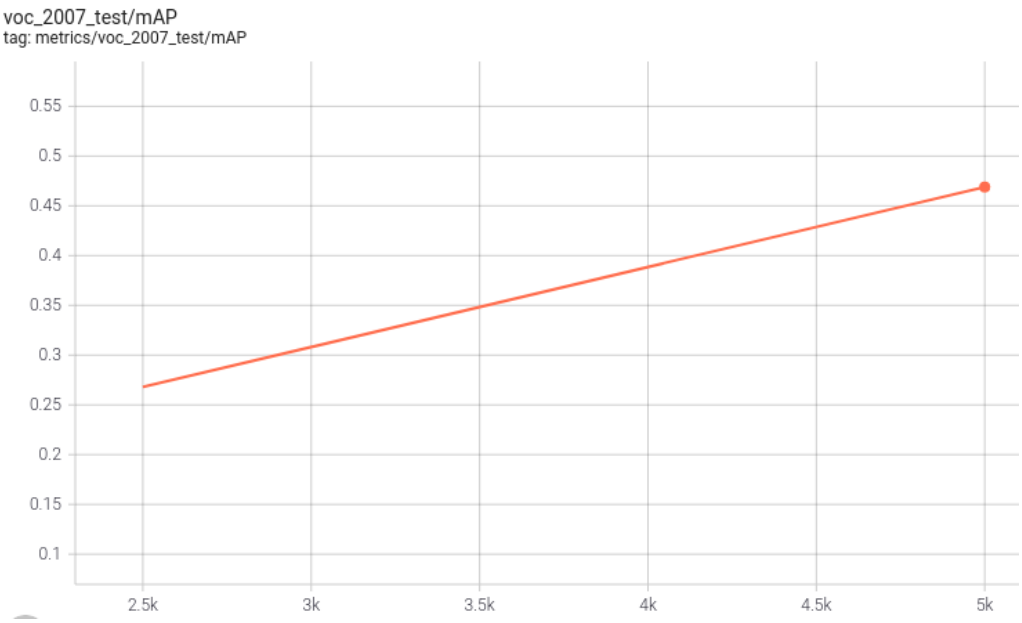


Figure 17: Caption

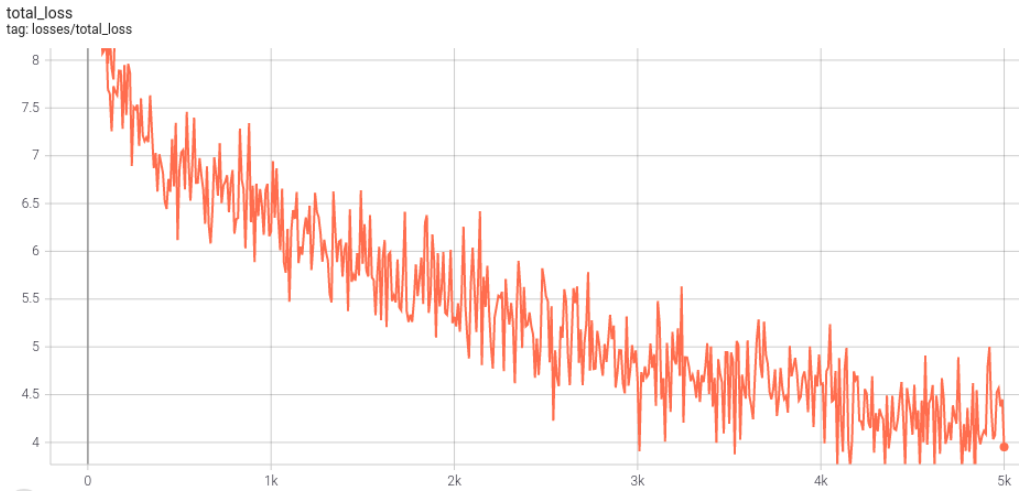
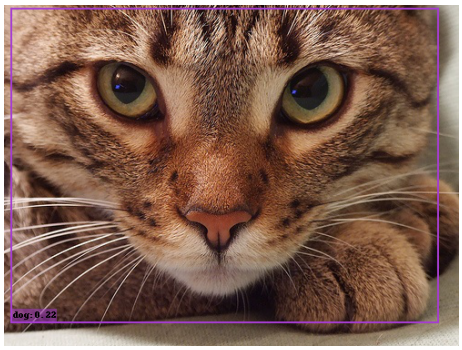


Figure 18: Caption

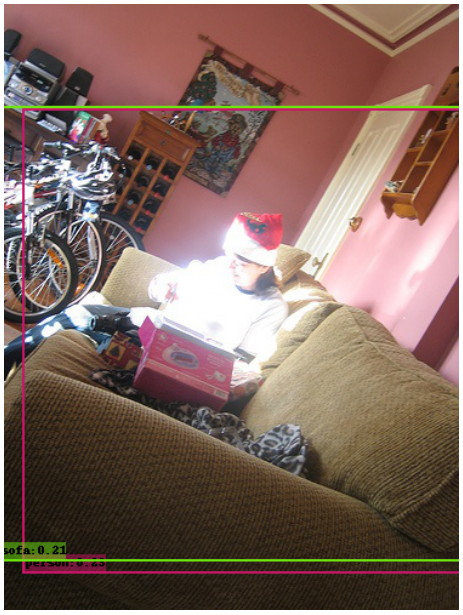
(a) TV-monitor labeled as person, and person labeled correctly



(b) Cat labeled as a dog.

Figure 19: Two of the test images side by side

(a) Image of a dinner party with a lot of accurate prediction boxes



(b) Person on sofa with correct labels, but very large prediction box for persoon 2

Figure 20: Two of the test images side by side

Figure 21: Dinner table with only a bottle labeled and bounded by a prediction box.