Assignment 4

TDT4265 Computer Vision and Deep Learning

Lars Martin Moe

Spring, 2020

Task 1: Object Detection Metrics (0.5 point)

Task 1a)

Intersection over union is a measure of the overlap between 2 boundaries. These boundaries can for example be our predicted bounding boxes and the ground truth bounding boxes. We can then calculate the overlap between these two bounding boxes as a measure of the accuracy of an object detector on a particular dataset. The intersection over union is the area of overlap divided by the area of the union, as shown in fig. 1. This value will be between 0 and 1, where a value greater than 0.5 is often regarded as a match.

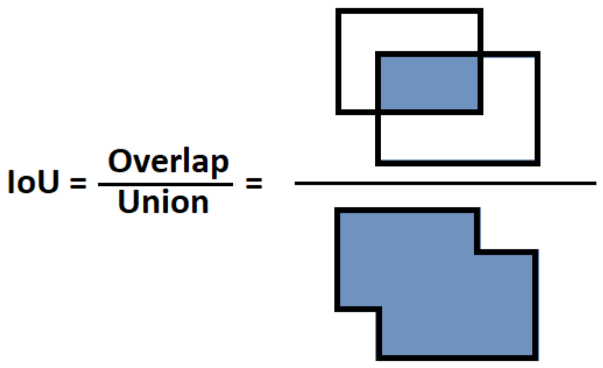

Figure 1: Illustration of intersection over union.

Task 1b)

The equations for precision and recall are shown in eq. (1) and eq. (2). Precision is the percentage of predictions that are correct, and recall is a measure of how well we find all the positives.

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

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative} \tag{2}$$

A true positive is an outcome where the model has correctly predicted the positive class, and a false positive is an outcome where it has incorrectly predicted the positive class. An example of true positive would be the correct prediction of a hand being in a picture, when the picture truly contains a hand. A false positive would be a prediction of a hand being in a picture where there truly are no hands in it.

Task 1c)

The Average Precision (AP) is given by:

$$AP = \int_0^1 p(r)dr \tag{3}$$

We make the calculated AP value less susceptible to small variations in the ranking by replacing the precision value for recall $\tilde{r}$ with the maximum precision for recalls to the right of $\tilde{r}$, causing the curve to decrease monotonically. We then have:

$$AP_{interpolated} = \int_0^1 p_{interpolated}(r)dr \tag{4}$$

where

$$p_{interpolated}(r) = \max_{\tilde{r} \geq r} p(\tilde{r}) \tag{5}$$

For this task we can start by finding an average for the 11-point (0.0, 0.1, ..., 1.0) interpolated AP with:

$$AP = \frac{1}{11} \sum_{r \in \{0.0, \ldots, 1.0\}} APr \tag{6}$$

Precision and recall curve for class 1:
Precision$_1$ = [1.0, 1.0, 1.0, 0.5, 0.20]
Recall$_1$ = [0.05, 0.1, 0.4, 0.7, 1.0]

Precision and recall curve for class 2:
Precision$_2$ = [1.0, 0.80, 0.60, 0.5, 0.20]
Recall$_2$ = [0.3, 0.4, 0.5, 0.7, 1.0]

Interpolated values are shown in fig. 2 and fig. 3.

$$AP_1 = \frac{5 \cdot 1.0 + 3 \cdot 0.5 + 3 \cdot 0.2}{11} = 0.645 \tag{7}$$

$$AP_2 = \frac{4 \cdot 1.0 + 1 \cdot 0.8 + 1 \cdot 0.6 + 2 \cdot 0.5 + 3 \cdot 0.2}{11} = 0.636 \tag{8}$$

The mean average precision (mAP) can be found by taking the average of the mean we found for the two classes:

$$mAP = \frac{AP_1 + AP_2}{2} = \frac{0.645 + 0.636}{2} = 0.6405 \tag{9}$$

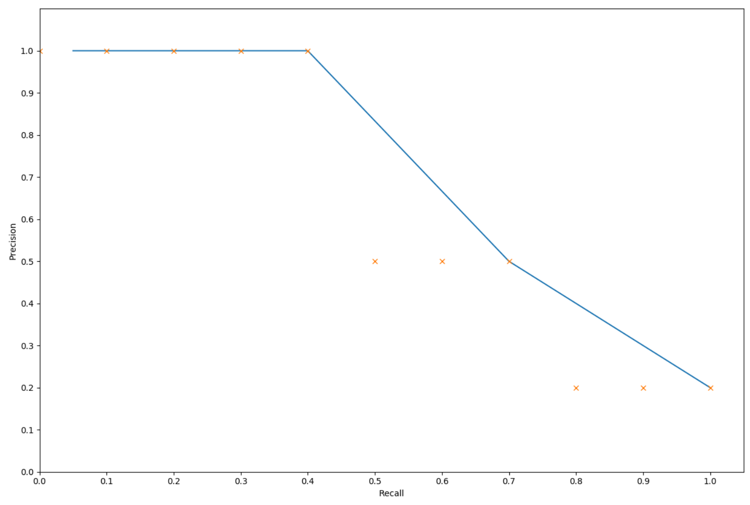

Figure 2: Precision plotted against recall for class 1 with interpolated values as crosses.

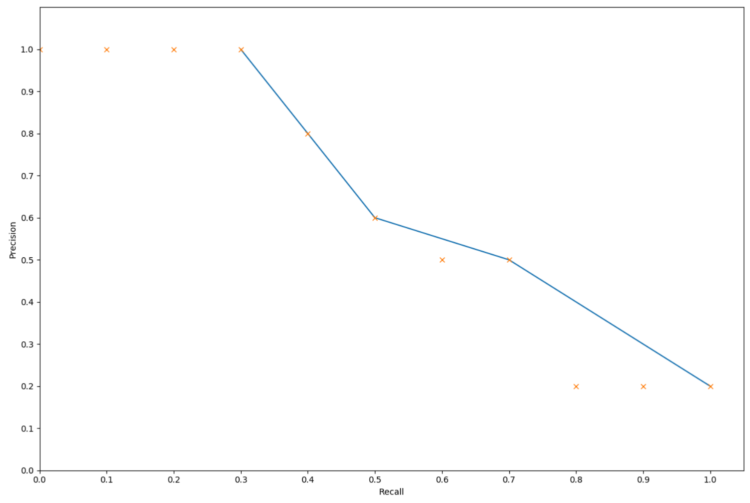

Figure 3: Precision plotted against recall for class 2 with interpolated values as crosses.

Implementing Mean Average Precision (2 points)

All tasks were implemented in task2.py. The tests passed and calculate_mean_average_precision returned 0.9066 for an IoU threshold of 0.5.

Task 2f)

The final precision-recall curve for this task is shown in fig. 4:

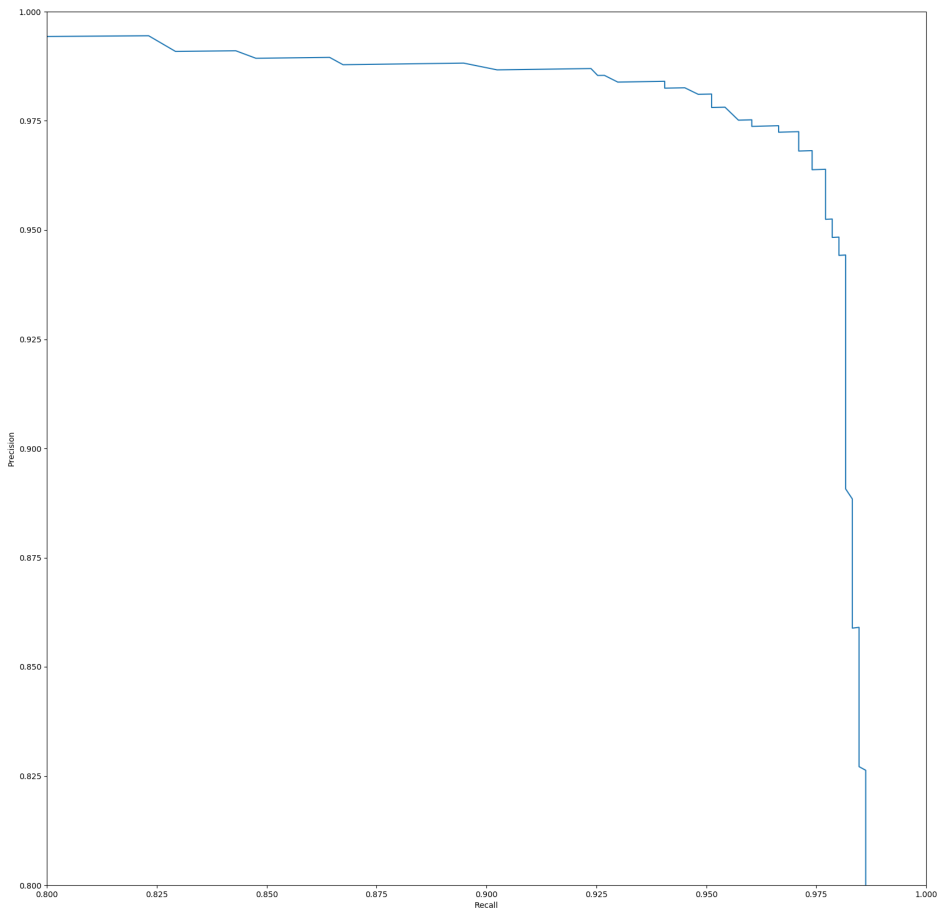

Figure 4: Final precision-recall curve

Task 3: Theory (0.5 points)

Task 3a

The final step that filters out overlapping boxes is called non-maximum suppression.

Task 3b

Statement:

"Predictions from the deeper layers in SSD are responsible to detect small objects"

Deeper layers in the SSD architecture have lower resolutions because CNNs reduce spatial dimensions. This means that information will be lost, and it harder to make out small objects. The lower resoultion layers are used to detect objects at larger scales, and the higher ones are used for smaller objects.

Statement verdict: **FALSE**

Task 3c

SSD uses different bounding box aspect ratios at the same spatial location because it allows us to partition what shape of the ground truth that a prediction is responsible for, encouraging it to predict shapes closer to the corresponding default box. This simplifies the learning problem (predictions become more diverse and more stable in training) and allows the network to predict high scores for multiple overlapping default boxes, instead of requiring it to only pick the one with maximum overlap.

Task 3d

The main difference is that the SSD approach uses multi-scale feature maps for detection, while YOLOv1/v2 operates on a single scale feature map. fig. 5 shows the added feature layers at the end of the base network for the SSD architecture. These layers predict the offsets to default boxes of different scales and aspect ratios and their associated confidences

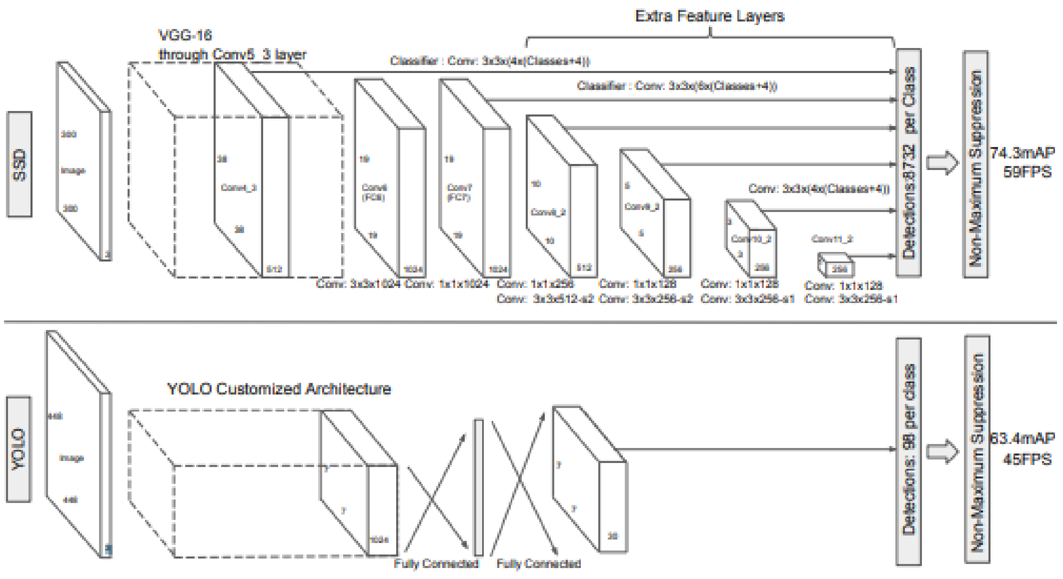

Figure 5: SDD and YOLO architecture from the original paper.

Task 4: Single Shot Detector (4 points)

Task 4b

Final mean average precision (mAP): 0.7874.

A plot of the total loss after 6000 iterations is shown in fig. 6:

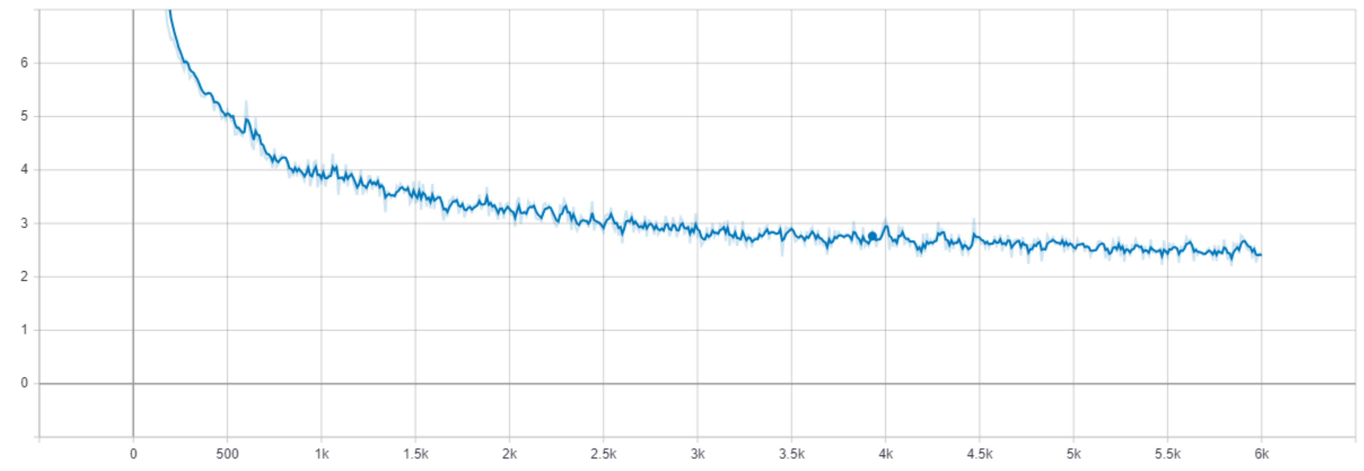

Figure 6: Total_loss plot from tensorboard.

Task 4c

Spatial batch normalization was added after each convolution. The architecture was updated with additional convolutional layers according to table 1. The SGD optimizer was switched out with the Adagrad optimizer with a learning rate of 0.02. The network has a total of 5.93M parameters.

The model reached a mAP of 0.8510 after 9500 iterations, as shown in fig. 7. Total loss is shown in fig. 8.

Table 1: New CNN backbone.

Is Output	Layer Type	Number of Hidden Units/Filters	Stride
	Conv2D	32	1
	BatchNorm2D	-	-
	MaxPool2D	-	2
	ReLU	-	-
	Conv2D	64	1
	BatchNorm2D	-	-
	MaxPool2D	-	2
	ReLU	-	-
	Conv2D	64	1
	BatchNorm2D	-	-
	ReLU	-	-
Yes, 38x38	Conv2D	output_channels[0]	2
	BatchNorm2D	-	-
	ReLU	-	-
	Conv2D	128	1
	BatchNorm2D	-	-
	ReLU	-	-
	Conv2D	256	1
	BatchNorm2D	-	-
	ReLU	-	-
yes, 19x19	Conv2D	output_channels[1]	2
	BatchNorm2D	-	-
	ReLU	-	-
	Conv2D	256	1
	BatchNorm2D	-	-
	ReLU	-	-
	Conv2D	512	1
	BatchNorm2D	-	-
	ReLU	-	-
Yes 9x9	Conv2D	output_channels[2]	2
	BatchNorm2D	-	-
	ReLU	-	-
	Conv2D	128	1
	BatchNorm2D	-	-
	ReLU	-	-
	Conv2D	256	1
	BatchNorm2D	-	-
	ReLU	-	-
Yes 5x5	Conv2D	output_channels[3]	2
	BatchNorm2D	-	-

Table 1: New CNN backbone.

Is Output	Layer Type	Number of Hidden Units/Filters	Stride
	ReLU	-	-
	Conv2D	128	1
	BatchNorm2D	-	-
	ReLU	-	-
	Conv2D	256	1
	BatchNorm2D	-	-
	ReLU	-	-
Yes, 3x3	Conv2D	output_channels[4]	2
	BatchNorm2D	-	-
	ReLU	-	-
	Conv2D	128	1
	BatchNorm2D	-	-
	ReLU	-	-
	Conv2D	256	1
	BatchNorm2D	-	-
	ReLU	-	-
Yes, 1x1	Conv2D	output_channels[5]	2

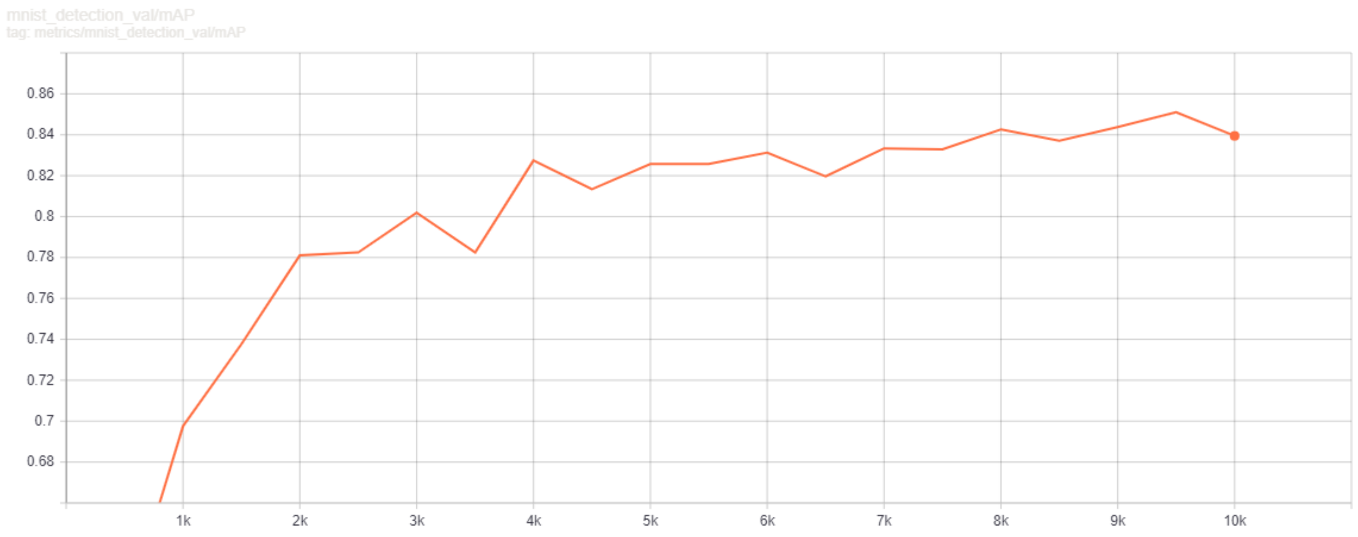

Figure 7: mAP for the model in table 1.

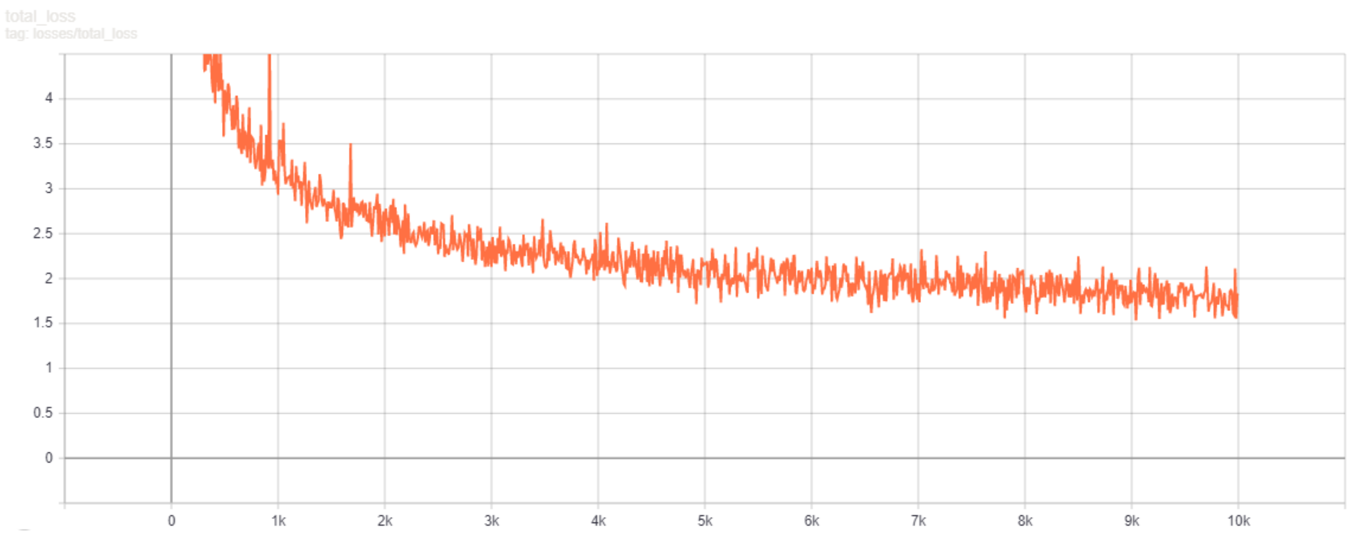

Figure 8: Total_loss plot for the model in table 1.

Task 4d

The model was changed again by adding a couple of convolutional layers, and adding an extra feature output layer with a higher resolution (75x75) at the start of the network, in order to better detect small-scaled numbers. The added layer and the changes made to the first layer is shown in 2. The rest of the layers are the same as in table 1.

The model reached a mAP of 0.9007 after 7000 iterations. After further training it was able to reach a mAP of 0.9110 at iteration nr 14000. The tmAP and total loss is shown in fig. 9 and fig. 18.

The backbone model has 5.67M parameters. The convolutions and first max pooling operation in the first layer could have been replaced with a 7x7 convolution with stride=2 in order to make the operations computationally cheaper. I would like to try this out later as the model currently requires a bit of time to train. The model currently uses no data augmentation techniques. For further work I would like to try to give the network more small-scale examples to train on by using augmentation techniques that zoom out on some of the inputs, and see how it performs. The current network seems to have to most problems with correctly classifying the 1-digit. I could try to improve this by altering some of the default boxes.

Table 2: New CNN backbone for task 4d.

Is Output	Layer Type	Number of Hidden Units/Filters	Stride
	Conv2D	32	1
	BatchNorm2D	-	-
	ReLU	-	-
	Conv2D	64	1
	BatchNorm2D	-	-
	MaxPool2D	-	2
	ReLU	-	-
	Conv2D	64	1
	BatchNorm2D	-	-
	MaxPool2D	-	2
	ReLU	-	-
Yes, 75x75	Conv2D	output_channels[0]	1
	BatchNorm2D	-	-
	ReLU	-	-
	Conv2D	64	1
	BatchNorm2D	-	-
	ReLU	-	-
	Conv2D	128	1
	BatchNorm2D	-	-
	ReLU	-	-
Yes 38x38	Conv2D	output_channels[1]	2

Task 4e

The results from the demo is shown in fig. 9 - fig 17. The model correctly classified all numbers. There is a number in figure 11 which I am not sure 100

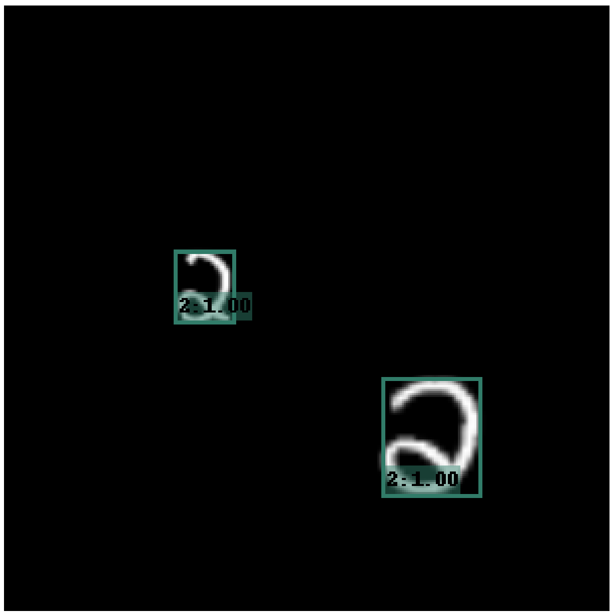

Figure 9: mAP for the model in table 2.

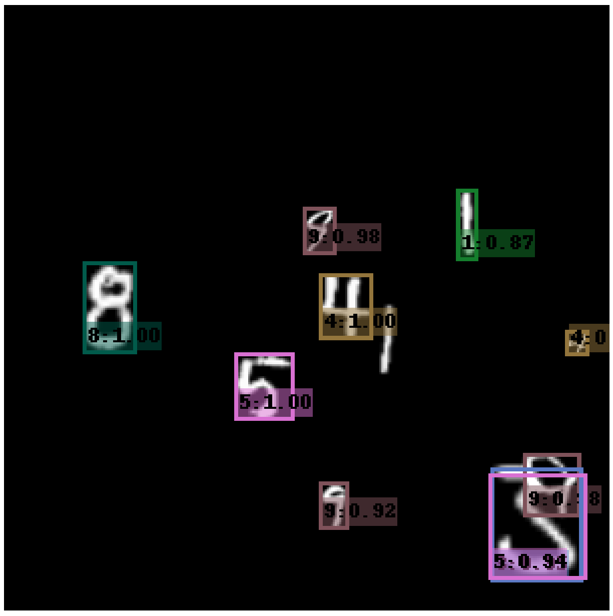

Figure 10: Total_loss plot for the model in table 2.

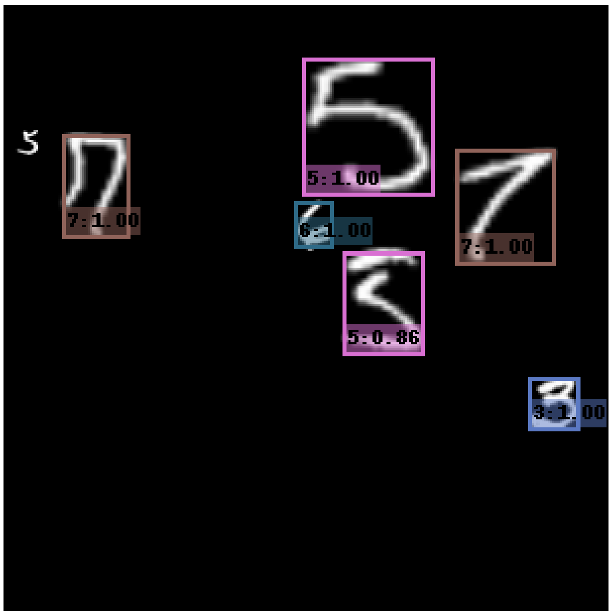

Figure 11:

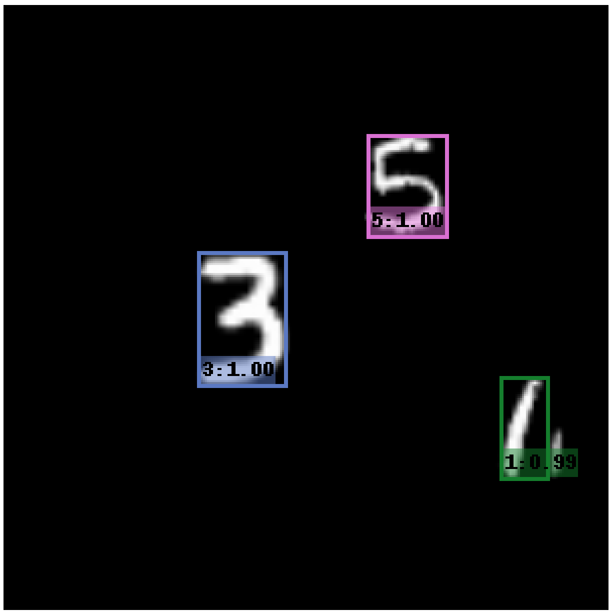

Figure 12:

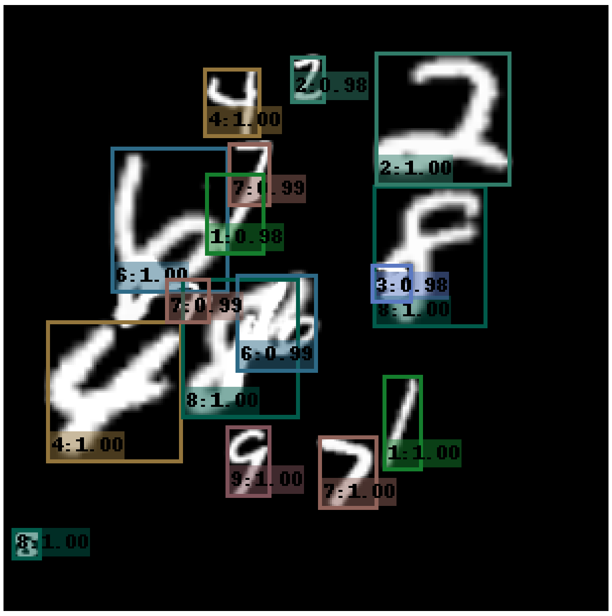

Figure 13:

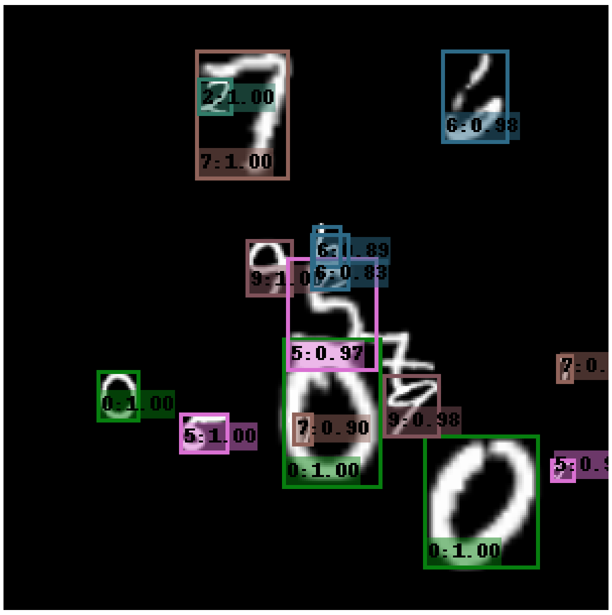

Figure 14:

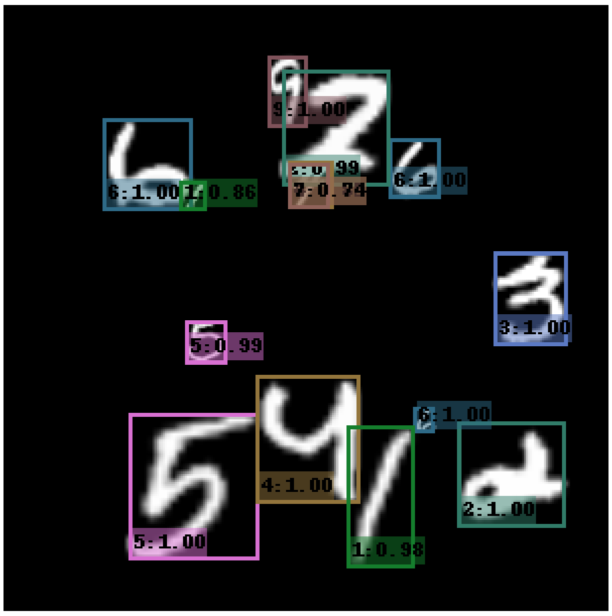

Figure 15:

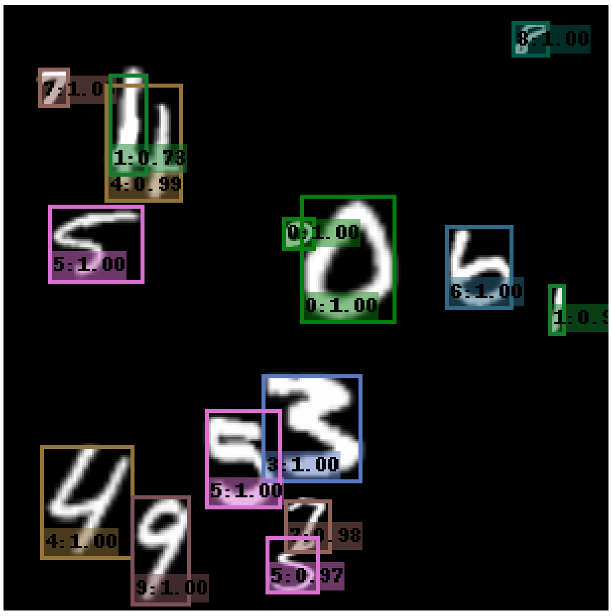

Figure 16:

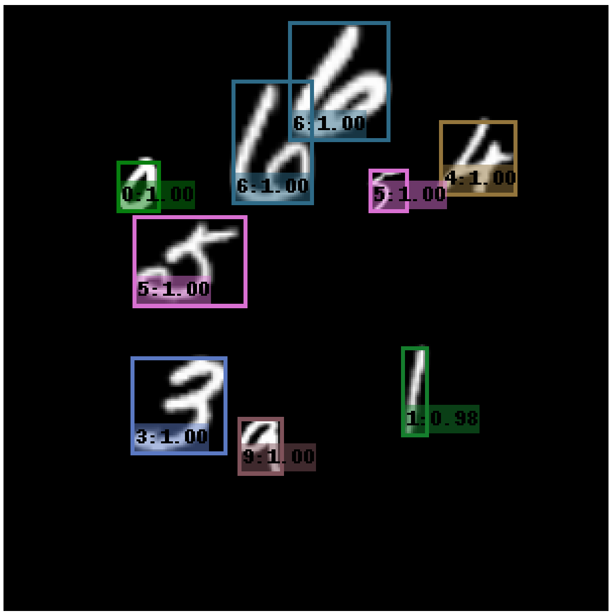

Figure 17:

Task 4f

For this task, all changes made to the files were reverted (no more Adagrad optimizer, original nr. of feature maps etc). The attached files are for the previous tasks.

The model was trained for 5000 gradient descent iterations on the VOC dataset. The total loss for the validation set is shown in **??**. The final mAP was 0.5622.

The model was unable to identify several objects when running the demo. The resulting images can be seen in fig. 19 to fig 23.

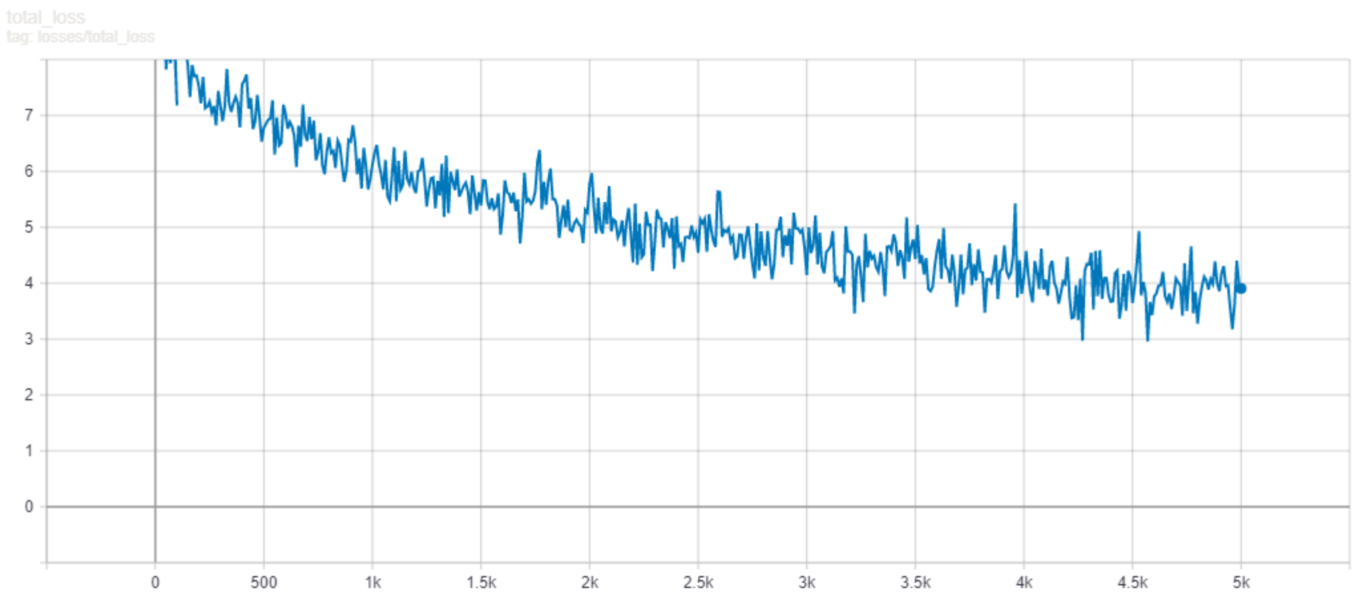

Figure 18: Total loss for the VGG16 model on the VOC validation set

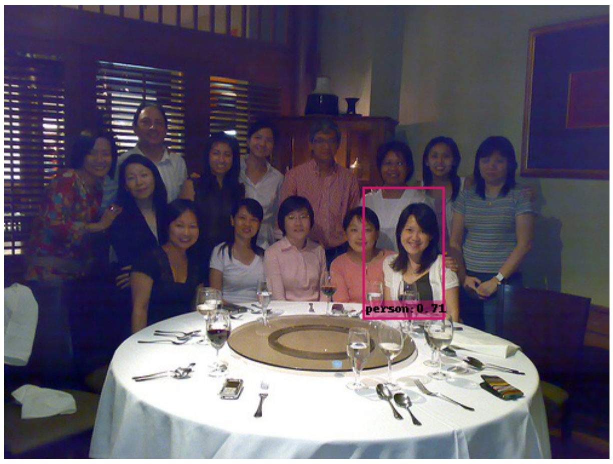

Figure 19:

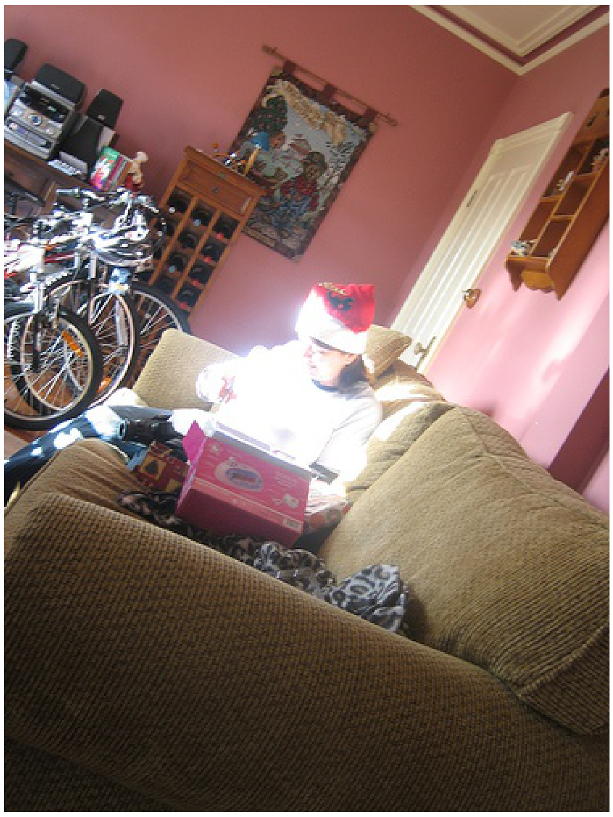

Figure 20:

Figure 21:

Figure 22:

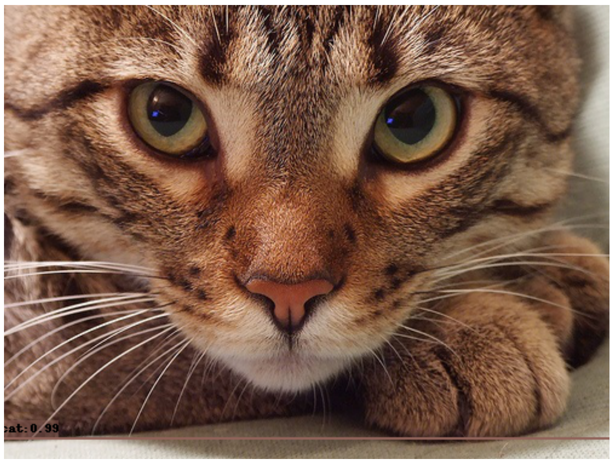

Figure 23: