TDT4265 Assignment 4

Anders Thallaug Fagerli

Task 1

a)

Intersection over Union (IoU) in the context of object detection is the fraction of area of intersection over area of union between two bounding boxes. Figure 1 displays a typical scenario, where the IoU is found by dividing the sum of pixels that contain the intersection by the sum of pixels that contain the union.

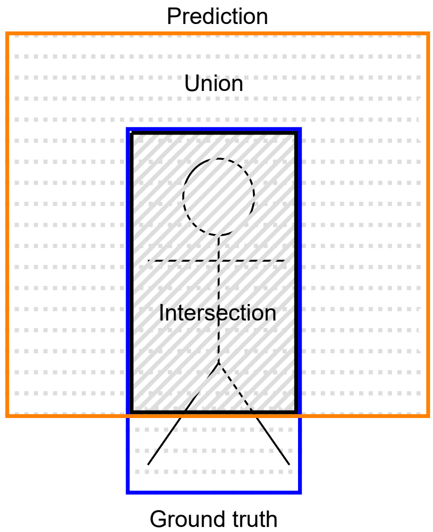


Figure 1: *Intersection (dashed) and union (dots) for two bounding boxes when detecting a person from an image.*

b)

The equations for precision and recall are

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}},$$
$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}},$$

where TP are true positives, FP are false positives and FN are false negatives. A true positive is a correct prediction of a class, while a false positive is an incorrect prediction of a class. In Figure 1, we have a true positive if we predict there is a person in the image, and a false positive if we predict there is a person in an image when there is not.

c)

The average precision (AP) for each class is

$$\text{Class 1:}\quad \text{AP}_1 = \frac{5 \cdot 1 + 3 \cdot 0.5 + 3 \cdot 0.2}{11} = 0.645,$$
$$\text{Class 2:}\quad \text{AP}_2 = \frac{4 \cdot 1 + 1 \cdot 0.8 + 1 \cdot 0.6 + 2 \cdot 0.5 + 3 \cdot 0.2}{11} = 0.636.$$

The mean average precision (mAP) is then

$$\text{mAP} = \frac{1}{K}\sum_{i=1}^{K} \text{AP}_i = \frac{0.645 + 0.636}{2} = 0.6405.$$

Task 2

f)

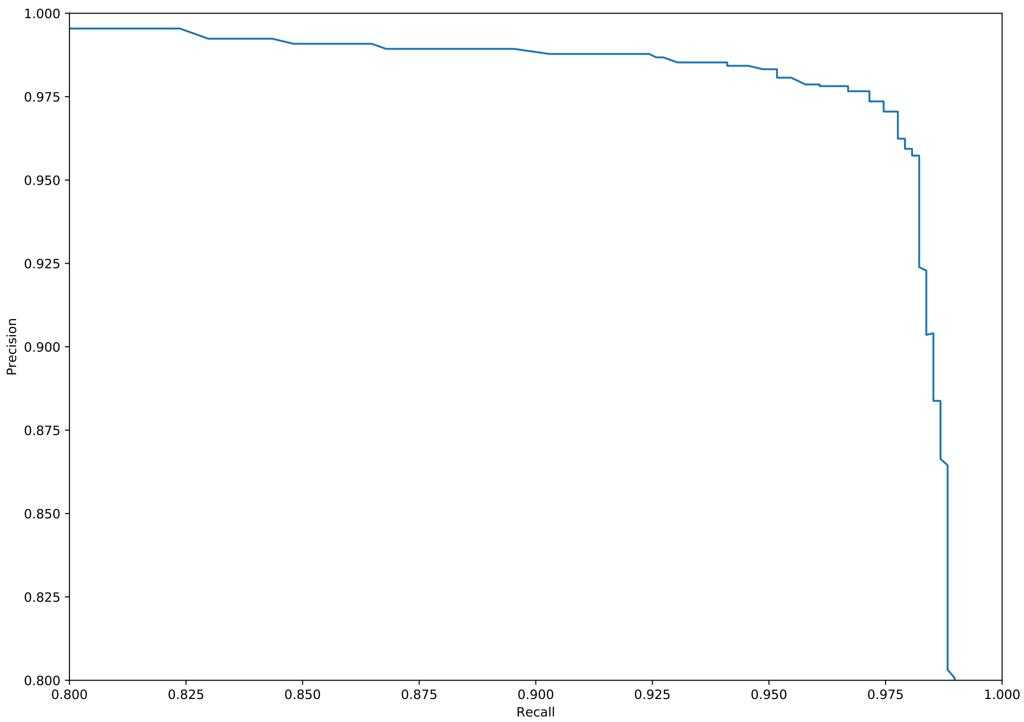


Figure 2: *Precision-recall curve.*

Task 3

a)

The filtering operation is called **non-maximum suppression** (nms).

b)

False, predictions from deeper layers detect larger objects.

c)

Different aspect ratios are used to detect objects of varying shape, as the different classes often have different shapes, giving faster and more stable convergence towards the final prediction boxes.

d)

The main difference between SSD and YOLO is that SSD uses multi-scale feature maps for detection, while YOLO uses a single scale feature map.

e)

We place 6 anchors for each pixel in the 38×38 feature map, giving $38 \cdot 38 \cdot 6 = 8664$ anchor boxes in total for this feature map.

f)

For the entire network we have $6 \cdot (38 \cdot 38 + 19 \cdot 19 + 10 \cdot 10 + 5 \cdot 5 + 3 \cdot 3 + 1 \cdot 1) = 11640$ anchor boxes.

Task 4

b)

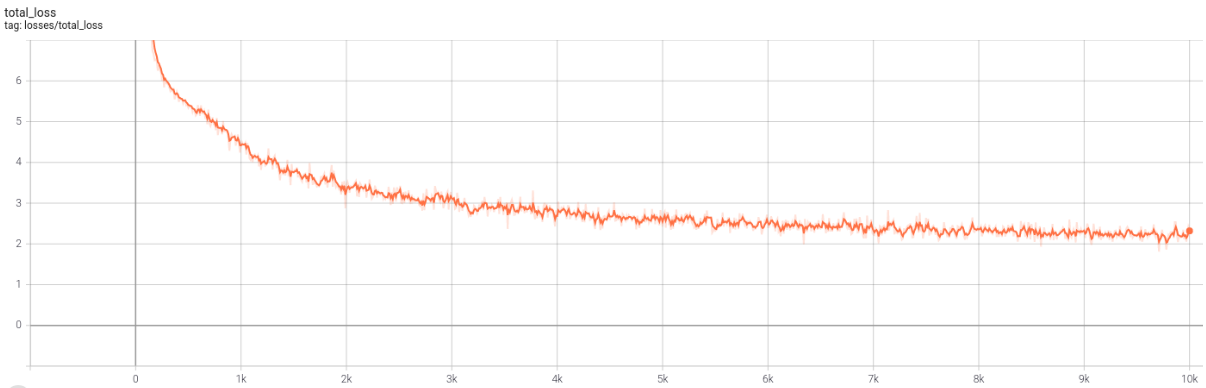


Figure 3: *Total loss over 10k iterations.*

After 10k gradient descent iterations the mean average precision was 0.8004.

c) and d)

The improved model is found in `ssd/modeling/backbone/improved.py` with config `mnist_improved.yaml`.

From Assignment 3 I saw that batch normalization contributed greatly to the overall performance, which was therefore added after each convolutional layer. I also switched to the Adam optimizer, and set the learning rate to $5 \cdot 10^{-4}$. One more convolutional layer (with batch normalization and ReLU) was also added to each feature map. This gave a mAP of around 85% within 10k iterations. After running `demo.py` I saw that the network struggled with smaller scales, so I added a 76×76 feature map for detecting numbers with smaller scale. This finally gave a mAP of 0.9047 after 14k iterations, barely achieving the goal. Output from `test.py` with the network is shown in Figure 4. The mAP over all iterations is shown in Figure 5, where we see that the actual mAP barely reaches 90% at iteration 14k. The total loss is shown in Figure 6.

```
2021-03-19 12:12:22,234 SSD.inference INFO: Loading checkpoint from outputs/improved/model_014000.pth
2021-03-19 12:12:22,367 SSD.inference INFO: Evaluating mnist_detection_val dataset(1000 images):
100%|
2021-03-19 12:12:28,874 SSD.inference INFO: mAP: 0.9047
0               : 0.9355
1               : 0.8479
2               : 0.9196
3               : 0.9081
4               : 0.9081
5               : 0.9076
6               : 0.9075
7               : 0.9003
8               : 0.9085
9               : 0.9042
```

Figure 4: *Output of* `test.py` *at checkpoint from iteration 14k.*

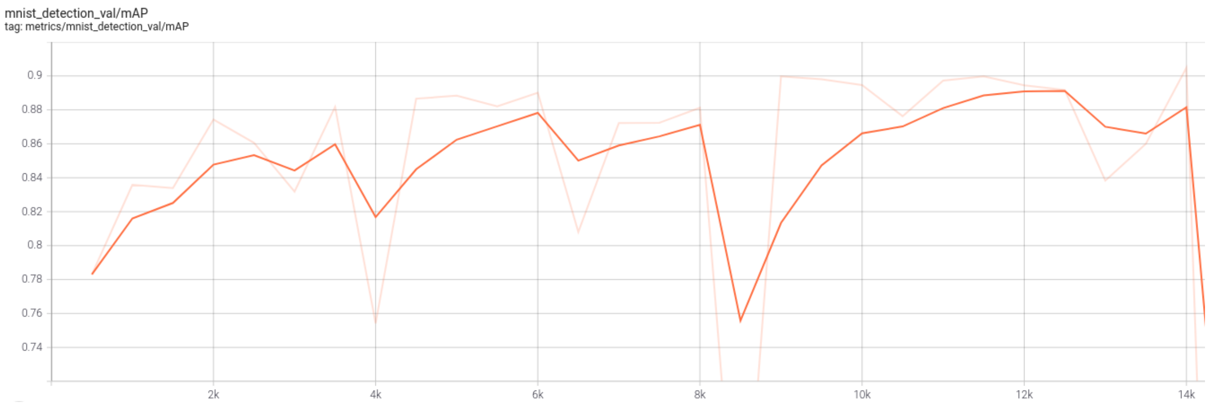


Figure 5: *mAP over all iterations, with smoothed value as strong line and actual value as weak line.*

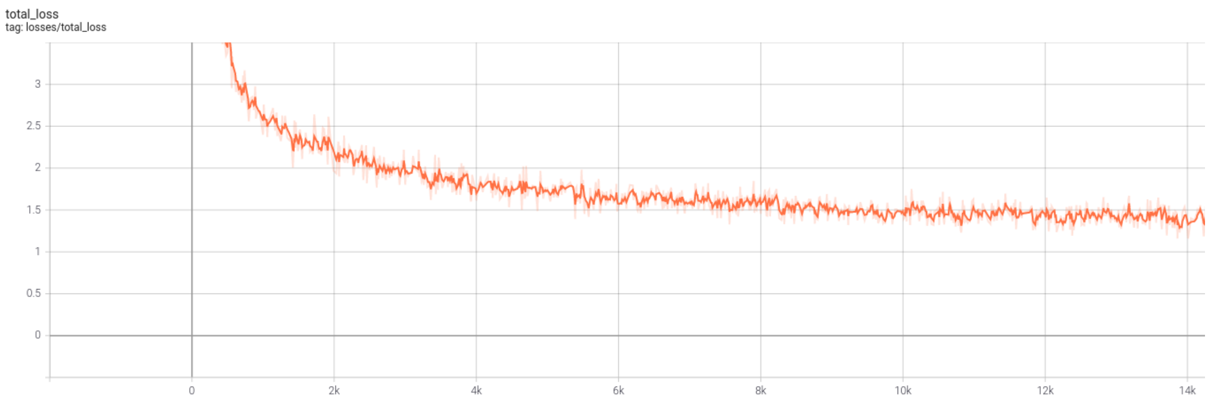


Figure 6: *Total loss over all iterations..*

e)

The results from `demo.py` with the model from d) is shown in Figure 7, Figure 8 and Figure 9. We see that the model struggles with numbers that have smaller scale, and mostly with the digit "1" or numbers that resemble "1". The 76×76 feature map did thus not solve the problem completely, and in afterthought a better solution may be to modify the bounding box parameters instead, to give smaller default bounding boxes.

f)

Despite fixing the errors in the original config, I was not able to achieve a mAP higher than 0.2409, as seen from Figure 11. The predictions did therefore not give any good results, and I had to lower the threshold for accepting a bounding box to 0.4 to show any results at all. Total loss is shown in Figure 10.

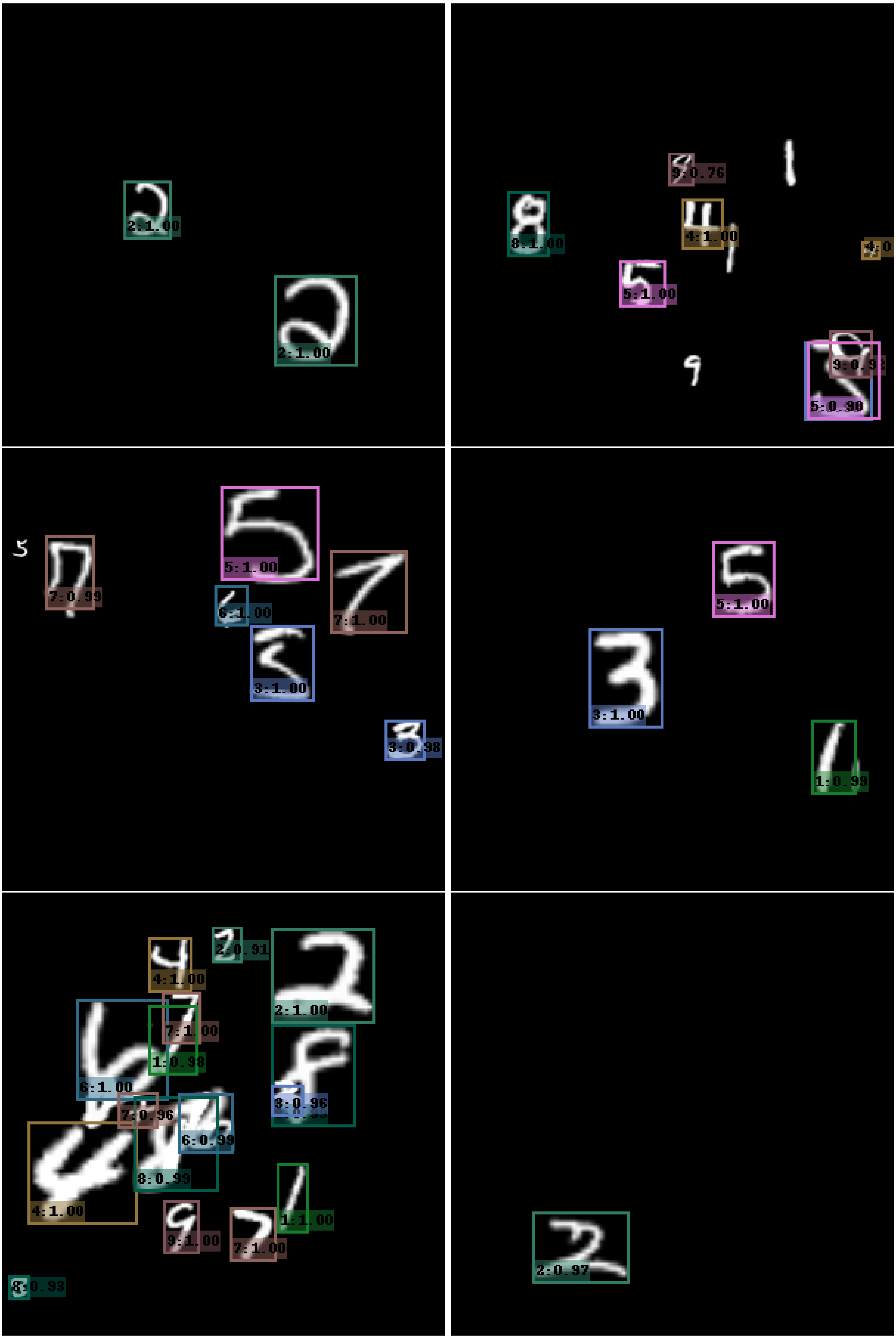


Figure 7: *Images 0-5.*

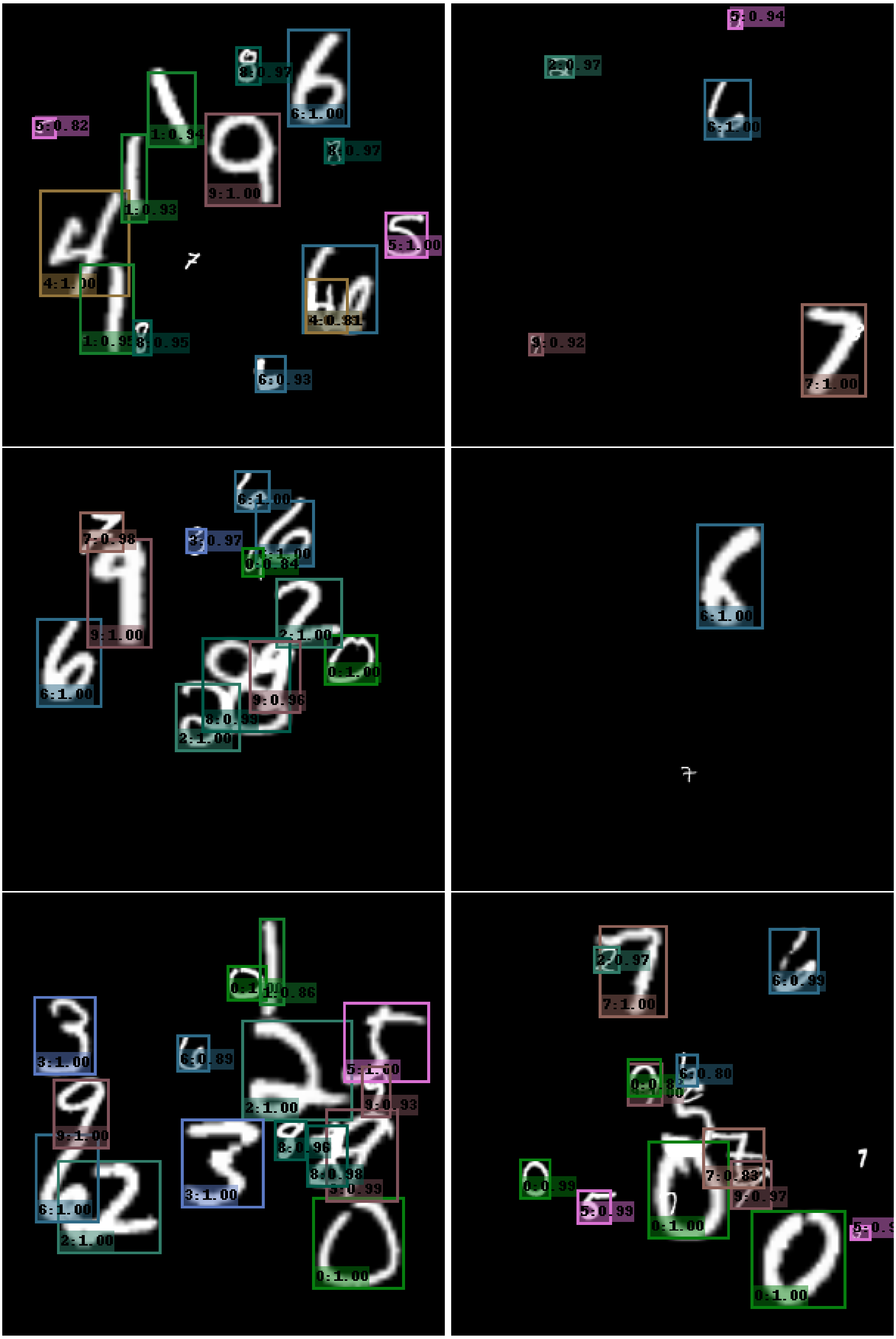


Figure 8: Images 6-11.

Figure 9: *Images 12-14.*

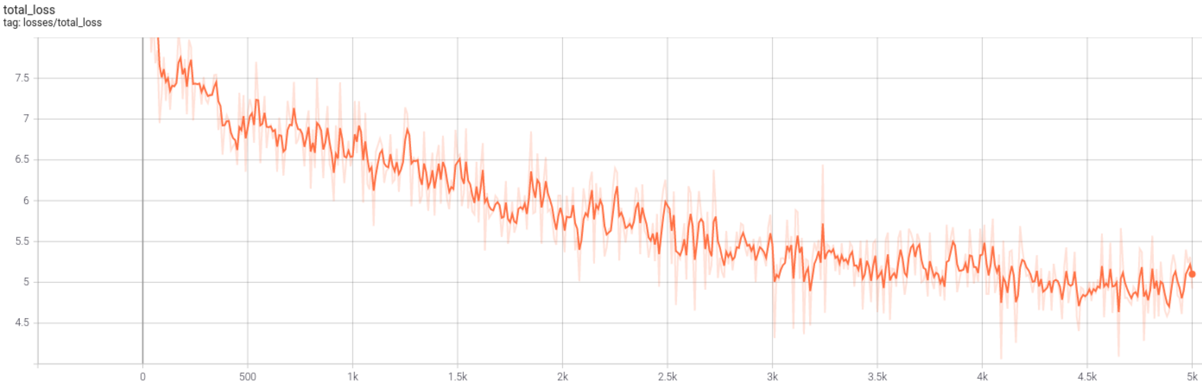
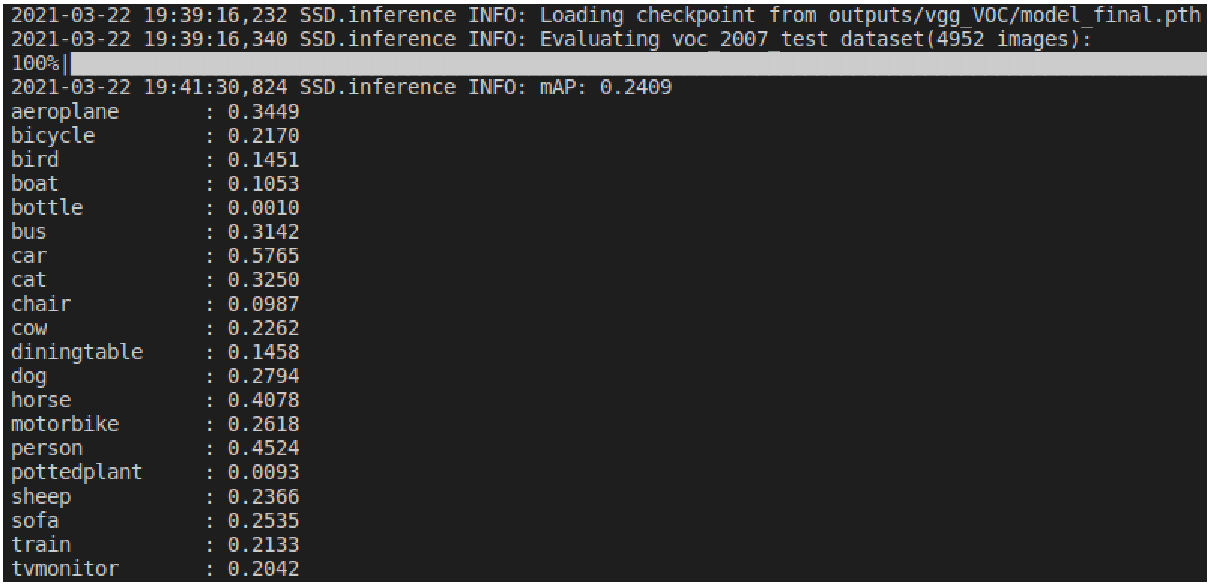
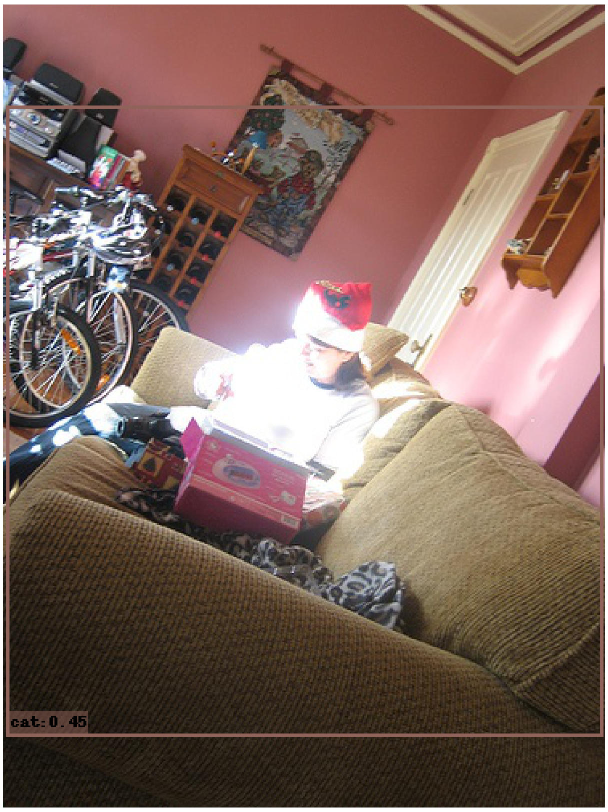


Figure 10: *Total loss.*



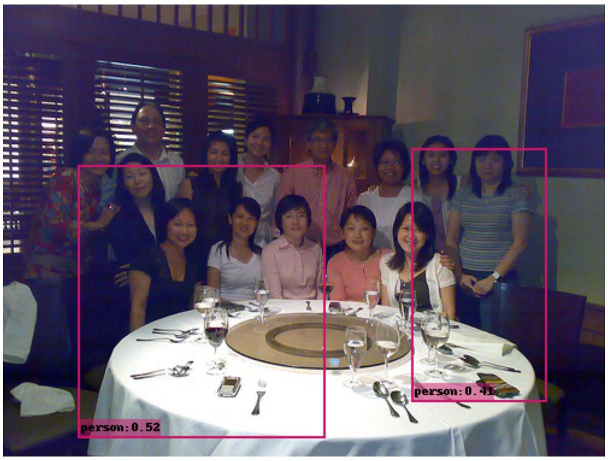
```
2021-03-22 19:39:16,232 SSD.inference INFO: Loading checkpoint from outputs/vgg_VOC/model_final.pth
2021-03-22 19:39:16,340 SSD.inference INFO: Evaluating voc_2007_test dataset(4952 images):
100%|
2021-03-22 19:41:30,824 SSD.inference INFO: mAP: 0.2409
aeroplane       : 0.3449
bicycle         : 0.2170
bird            : 0.1451
boat            : 0.1053
bottle          : 0.0010
bus             : 0.3142
car             : 0.5765
cat             : 0.3250
chair           : 0.0987
cow             : 0.2262
diningtable     : 0.1458
dog             : 0.2794
horse           : 0.4078
motorbike       : 0.2618
person          : 0.4524
pottedplant     : 0.0093
sheep           : 0.2366
sofa            : 0.2535
train           : 0.2133
tvmonitor       : 0.2042
```

Figure 11: *Output of* `test.py`*.*


cat: 0.45


person: 0.51
AGR
23


person: 0.43
person: 0.66


person: 0.41
person: 0.52