Assignment 1 Report

This is an outline for your report to ease the amount of work required to create your report. Jupyter notebook supports markdown, and I recommend you to check out this cheat sheet (https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet). If you are not familiar with markdown.

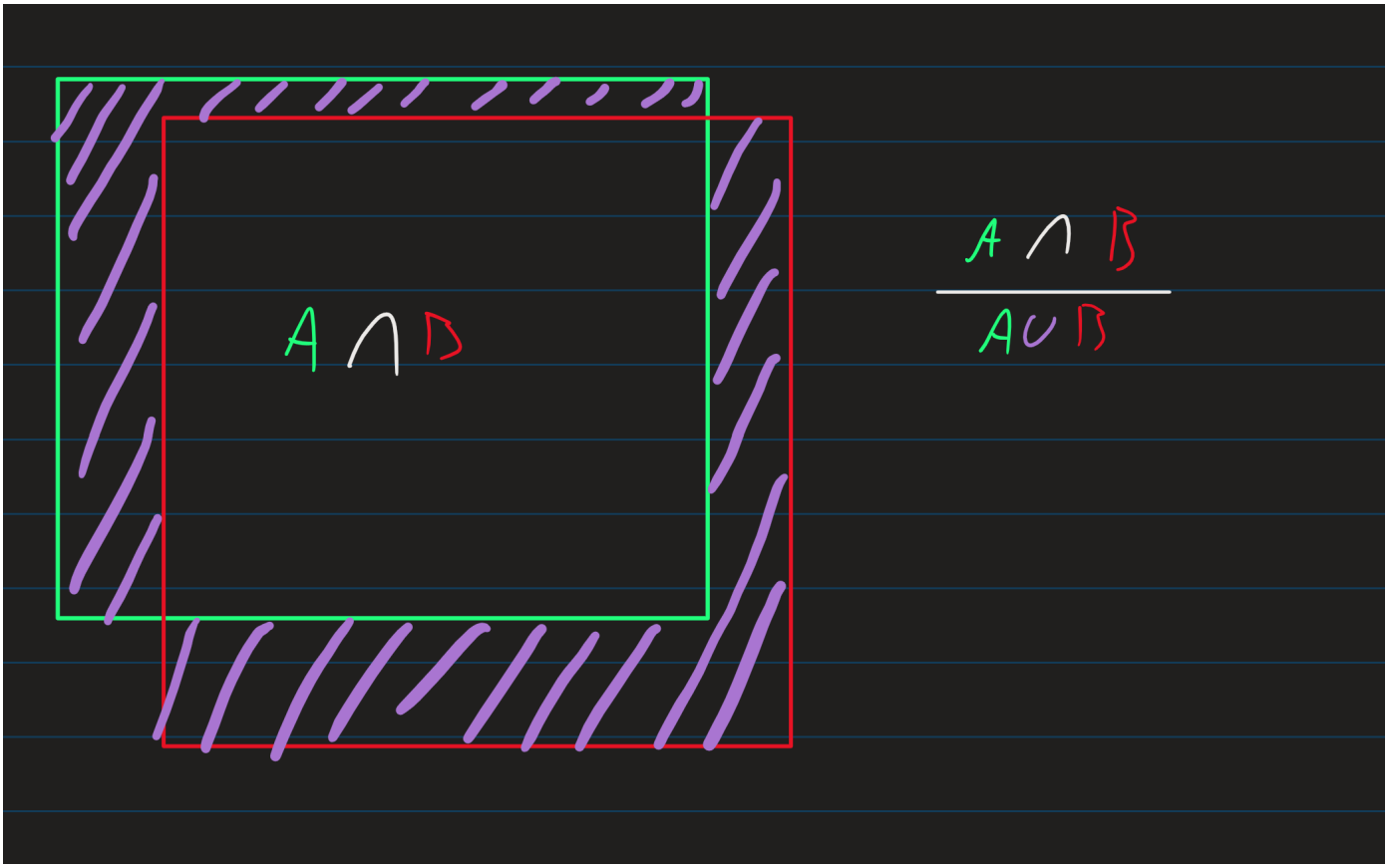
Before delivery, **remember to convert this file to PDF**. You can do it in two ways:

1. Print the webpage (ctrl+P or cmd+P)
2. Export with latex. This is somewhat more difficult, but you'll get somehwat of a "prettier" PDF. Go to File -> Download as -> PDF via LaTeX. You might have to install nbconvert and pandoc through conda; `conda install nbconvert pandoc` .

Task 1

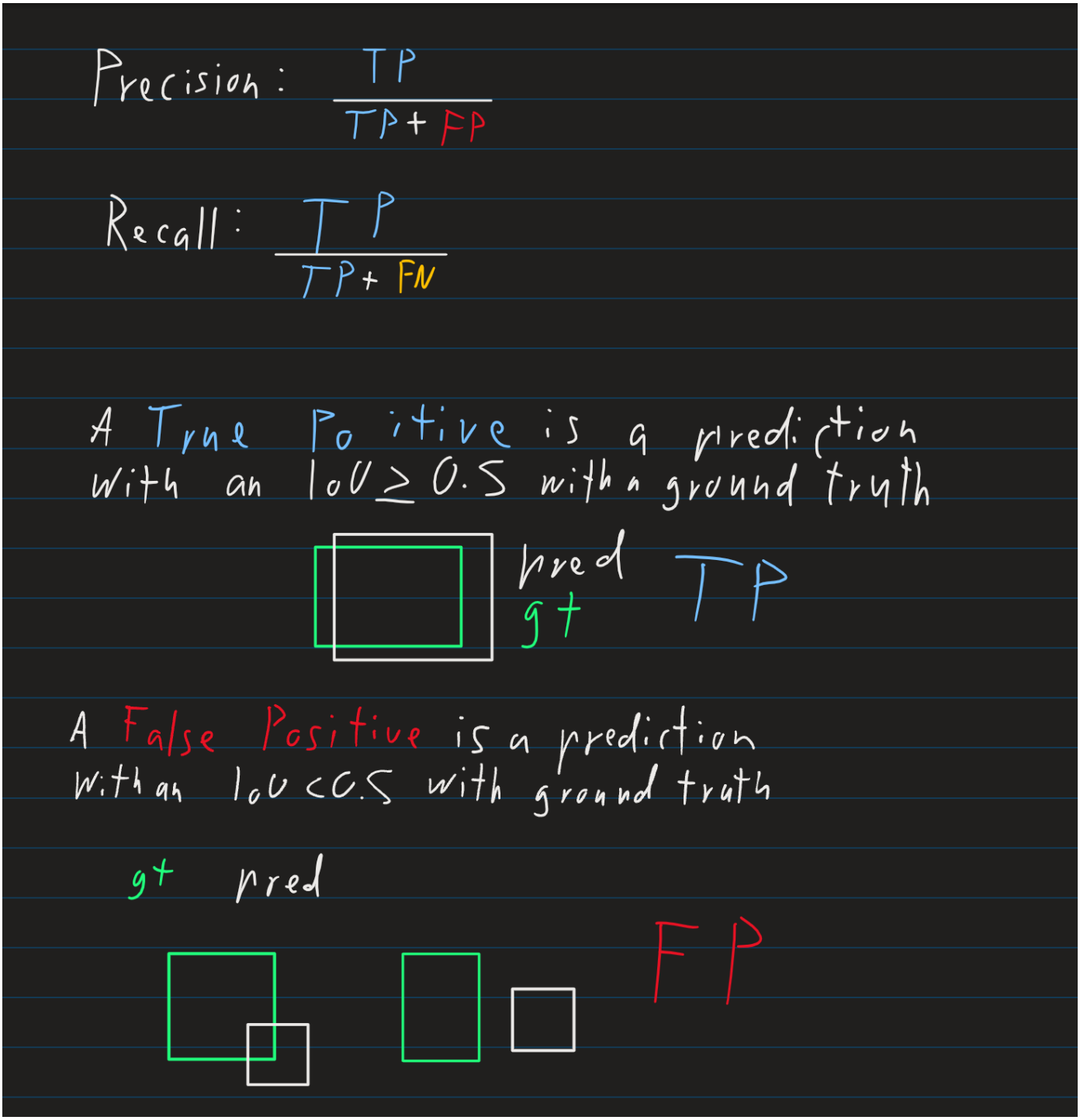
task 1a)

IoU or intersection of Union is a measurement of accuracy for object detection models. It takes the overlap between the ground truth bounding box, and the predicted bounding box, and divides it on the area of the union.



task 1b)

A True Positive is the part of a prediction which intersects with the ground truth. A false Positive is the part of a prediction which does not intersect with the ground truth.

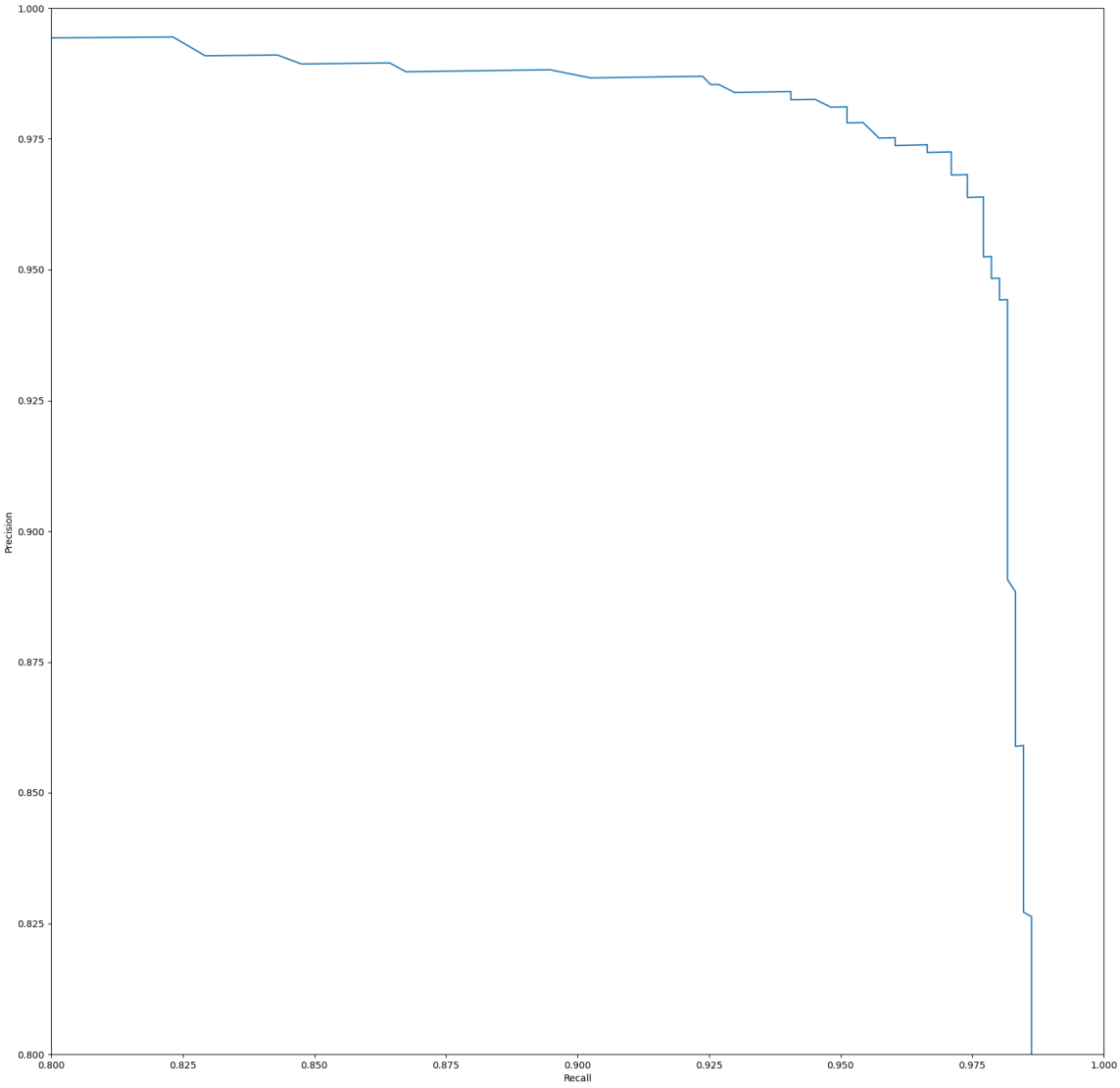


task 1c)

mAP: 0.64

Task 2

Task 2f)



Task 3

Task 3a)

The operation of filtering out overlapping bounding boxes during inference is called **non-maximum suppression**.

Task 3b)

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

This is **false**. The deeper we go into the network, the lower the resolution becomes. The higher resolution feature maps are responsible for detecting smaller objects, meaning the more "shallow" or higher layers preform this task.

Task 3c)

We use different bounding box aspect ratios at the same spatial location because it enhances the early stage of training. If we have diverse box shapes, it is easier to detect multiple objects with varying shapes. This makes sense as a lot of objects in real life do not have random aspect ratios (Humans for instance are typically way longer than they are wide), and therefore applying different default boxes will increase the chance that one of them become a positive match. This again is important for training as *SSD only uses positive matches in calculating the localization cost* as stated in the medium article by Jonathan Hui.

Task 3d)

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

The difference is that the SSD model adds extra feature layers to the end of the base network. *which predict the offsets to default boxes of different scales and aspect ratios and their associated confidences*.

Essensially, YOLO operates on a single scale feature map, while SSD operates on several which allows it to detect objects in a wide range of scales. The SSD also uses convolutional filters to determine the shape offsets relative to the default boxes coordinates, where as YOLO uses a dense layer for this task.

Task 3e)

38x38x6 = 8664 anchor boxes for the feature map

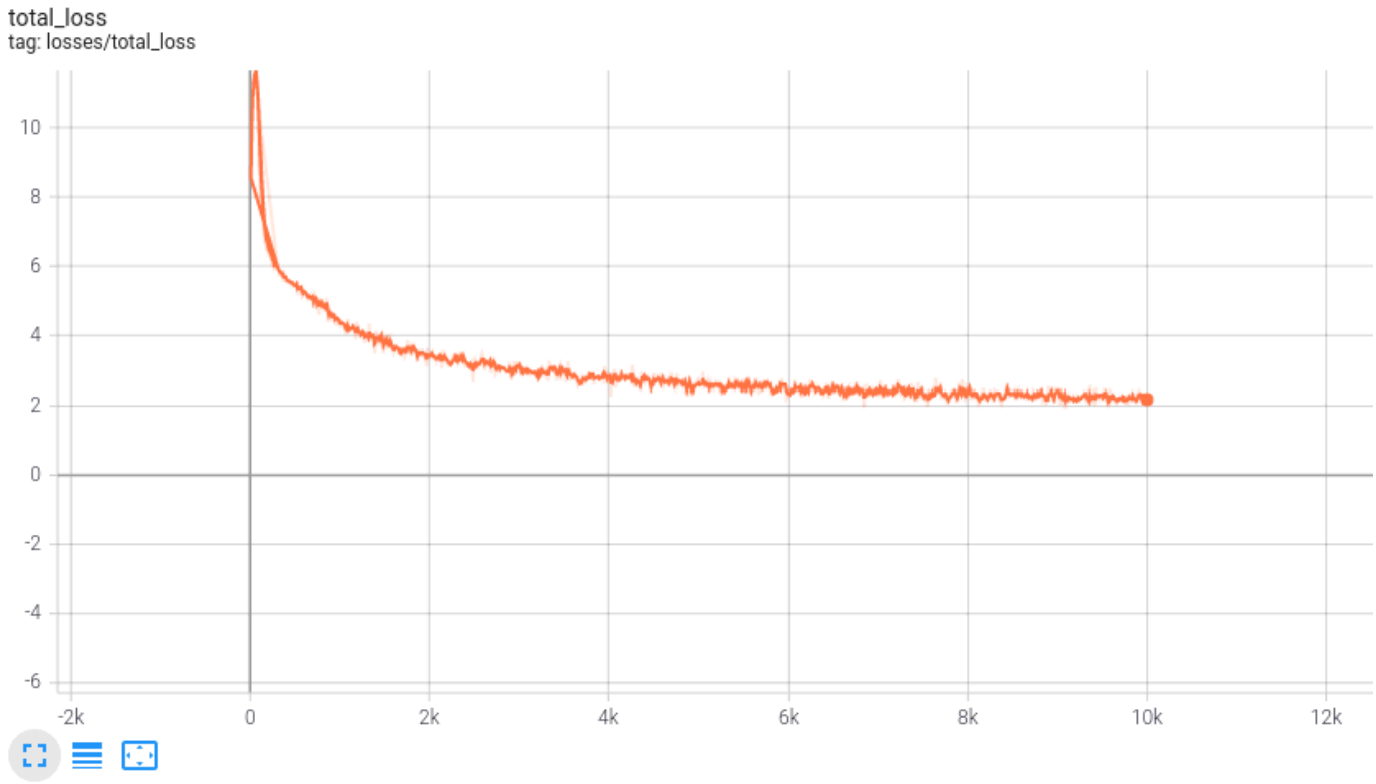
Task 3f)

38 x 38 x 6 = 8664 19 x 19 x 6 = 2166 10 x 10 x 6 = 600 5 x 5 x 6 = 150 3 x 3 x 6 = 54 1 x 1 x 6 = 6

= 11640 total anchor boxes

Task 4

Task 4b)



I trained for 10k iterations

mAP after 6k iterations: 0.7576 Final mAP: 0.8008

Task 4c)

I got to 0.8708 mAP by adding two extra conv layers with more filters in the backbone, and adding batchnormalization before all conv layers. Additionally I set learning rate to 1e-3

mAP: 0.8710

number	AP
0	: 0.8892
1	: 0.8618
2	: 0.8540
3	: 0.8552
4	: 0.8798
5	: 0.8590
6	: 0.8874
7	: 0.8654
8	: 0.8924
9	: 0.8656

Backbone

Layer	LayerType	Number of Hidden Units/filters	stride
1	conv2d	32	1
-	MaxPool2d	-	2

Layer	LayerType	Number of Hidden Units/filters	stride
-	ReLU	-	-
2	conv2d	64	1
-	MaxPool2d	-	2
-	ReLU	-	-
3	conv2d	128	1
-	ReLU	-	-
4	conv2d	128	1
-	ReLU	-	-
5	conv2d	64	1
-	ReLU	-	-
6	conv2d	64	2

please note that I use a nn.BatchNorm2d layer after every conv layer

Task 4d)

To get the mAP above 90% i changed the optimizer to Adam, set the learning rate to 5e-4, and added a couple of more filters and layers. already by 6.5k iterations the mAP was at 0.9015. the mAP was at its highest after 8.5k iterations at 0.9029. Also, I'm still using batchnormalization before every conv

mAP: 0.9029

number	AP
0	: 0.9032
1	: 0.8985
2	: 0.9053
3	: 0.9061
4	: 0.9046
5	: 0.8971
6	: 0.9037
7	: 0.9030
8	: 0.9069
9	: 0.9006

I also made significant changes to the architecture of the network

Backbone

Layer	LayerType	Number of Hidden Units/filters	stride
1	conv2d	32	1
-	MaxPool2d	-	2
-	ReLU	-	-

Layer	LayerType	Number of Hidden Units/filters	stride
2	conv2d	64	1
-	MaxPool2d	-	2
-	ReLU	-	-
3	conv2d	128	1
-	ReLU	-	-
4	conv2d	256	1
-	ReLU	-	-
5	conv2d	256	1
-	ReLU	-	-
6	conv2d	256	1
-	ReLU	-	-
7 (output)	conv2d	256	2

Extra filters

Layer	LayerType	Number of Hidden Units/filters	stride
-	ReLuU	-	-
8	conv2d	256	1
-	ReLU	-	-
9 (output)	conv2d	512	2

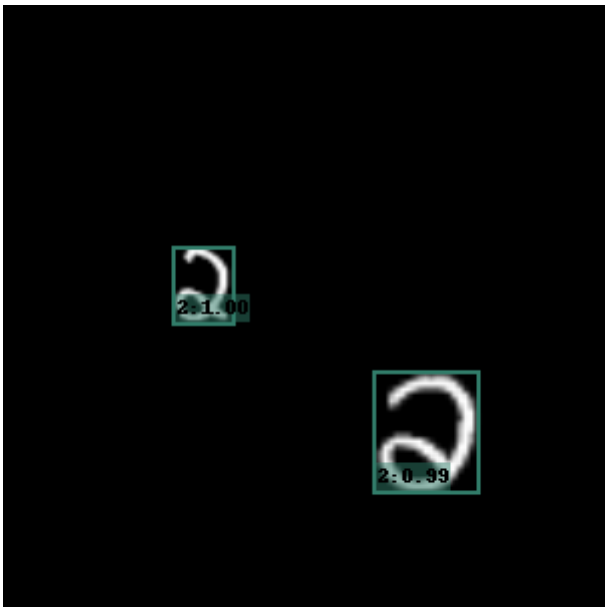
Layer	LayerType	Number of Hidden Units/filters	stride
-	ReLuU	-	-
10	conv2d	512	1
-	ReLU	-	-
11 (output)	conv2d	256	2

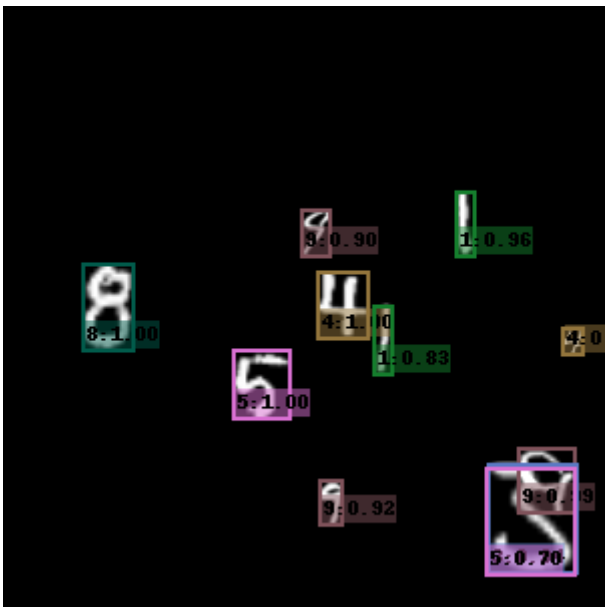
Layer	LayerType	Number of Hidden Units/filters	stride
-	ReLuU	-	-
12	conv2d	256	1
-	ReLU	-	-
13 (output)	conv2d	256	2

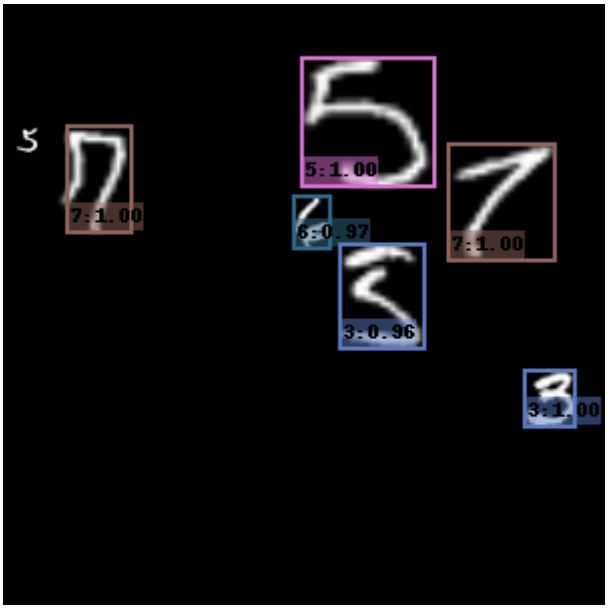
Layer	LayerType	Number of Hidden Units/filters	stride
-	ReLuU	-	-
14	conv2d	256	1
-	ReLU	-	-
15 (output)	conv2d	128	2

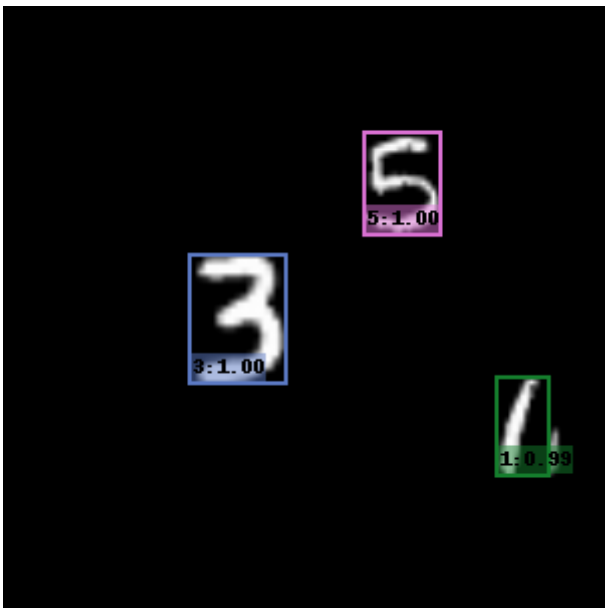
Layer	LayerType	Number of Hidden Units/filters	stride
-	ReLuU	-	-
16	conv2d	128	1
-	ReLU	-	-
17 (output)	conv2d	128	1

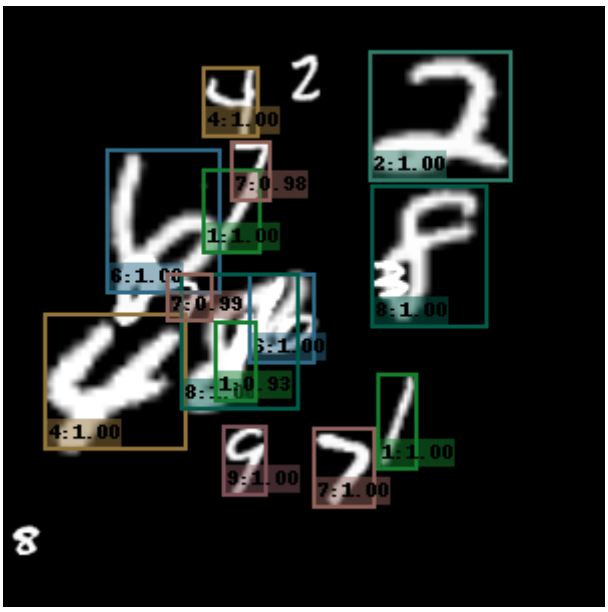
Task 4e)

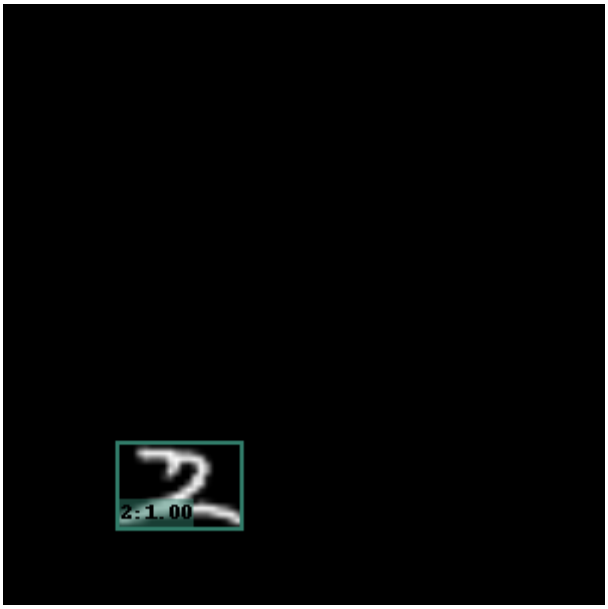


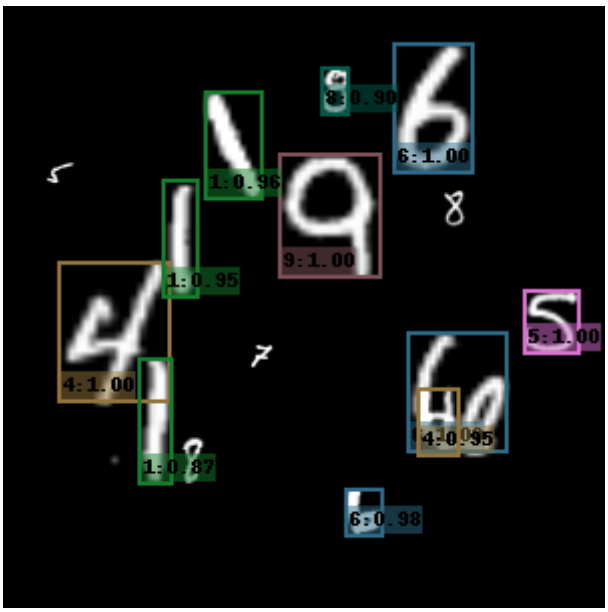


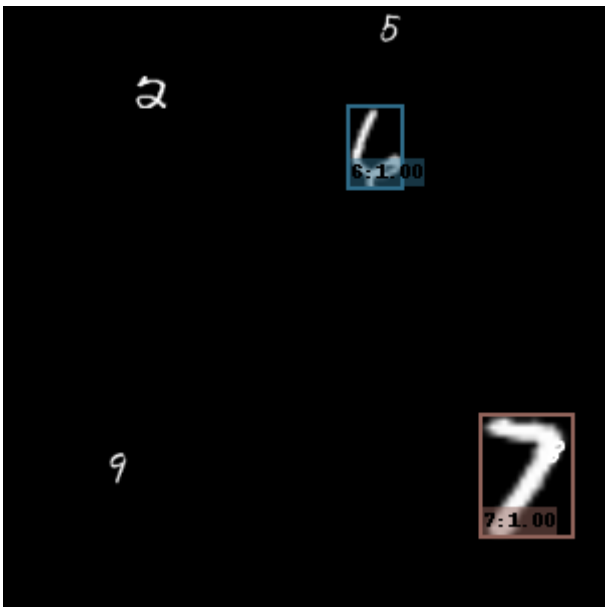

5
7:1.00
5:1.00
6:0.97
7:1.00
3:0.96
3:1.00

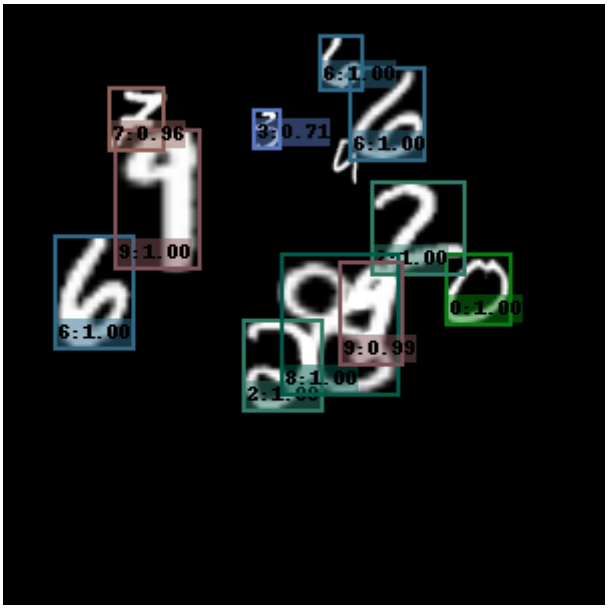

5:1.00
3:1.00
1:0.93

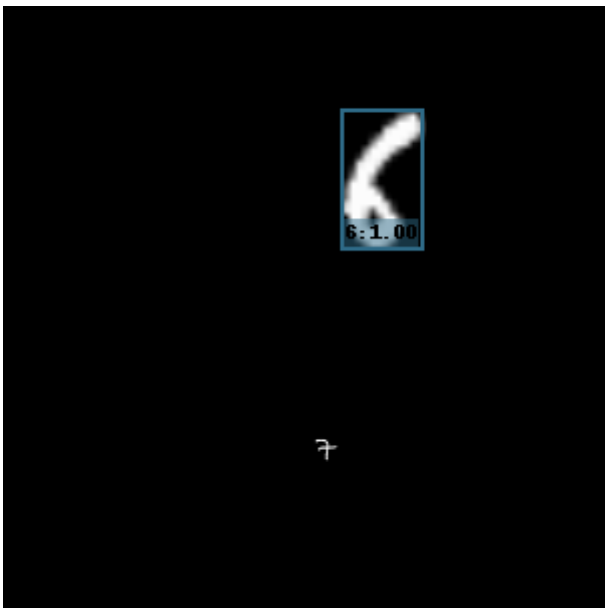

2
4:1.00
2:1.00
7:0.98
1:1.00
6:1.00
7:0.99
3:1.00
8:1.00
4:1.00
9:1.00
7:1.00
1:1.00
8

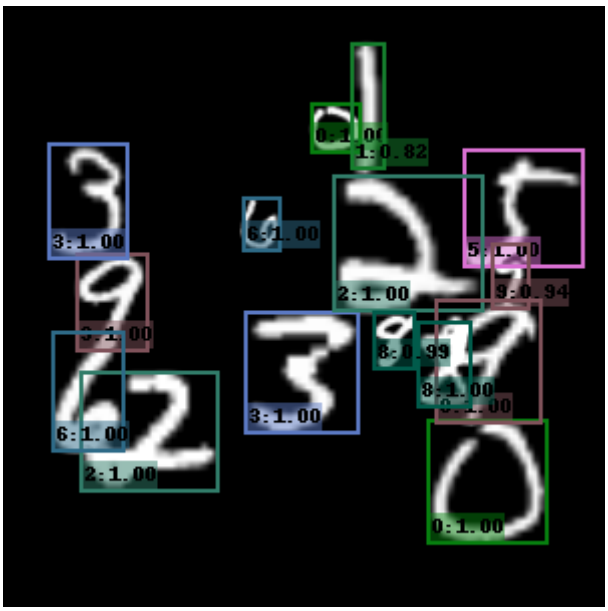

2: 1.00

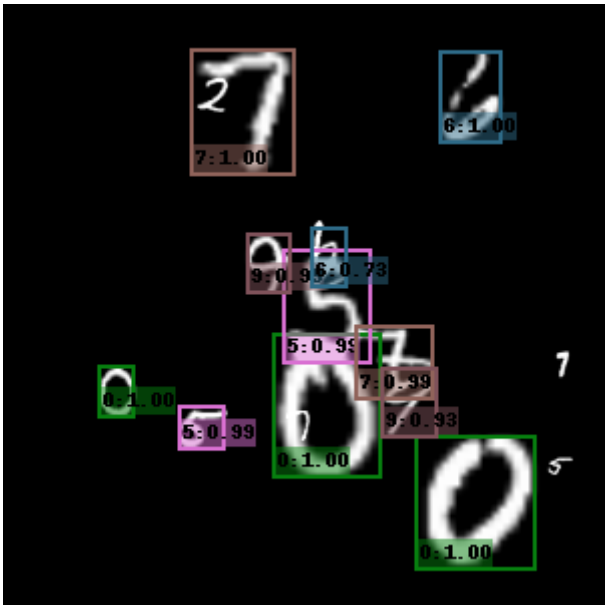

8: 0.90
6: 1.00
1: 0.96
9: 1.00
1: 0.95
4: 1.00
5: 1.00
4: 0.95
1: 0.87
6: 0.98


6: 1.00
7: 1.00

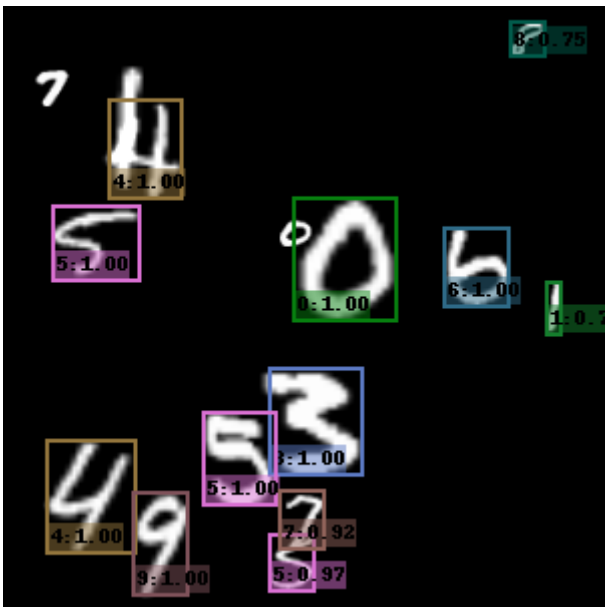

6:1.00
7:0.96
6:1.00
9:1.00
6:1.00
9:0.99
8:1.00
0:1.00

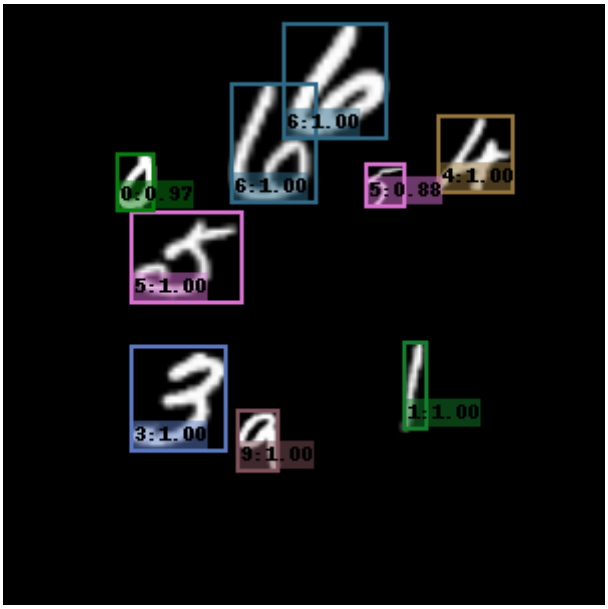

6:1.00


1:0.82
3:1.00
2:1.00
9:0.94
8:0.99
8:1.00
6:1.00
3:1.00
2:1.00
0:1.00


7:1.00
6:1.00
9:0.99
6:0.73
5:0.99
7:0.99
9:0.93
0:1.00
5:0.99
0:1.00
0:1.00


3:1.00
2:1.00
6:1.00
4:0.86
6:1.00
3:1.00
6:0.80
4:1.00
2:0.99
5:1.00
1:0.98


8:0.75
4:1.00
5:1.00
0:1.00
6:1.00
3:1.00
5:1.00
4:1.00
7:0.92
9:1.00
5:0.97



We see that my model struggled quite a lot with the really small numbers. It also makes a few mistakes, and sometimes can't detect numbers that are overlapping

Task 4f)