Assignment 3 - Group 170

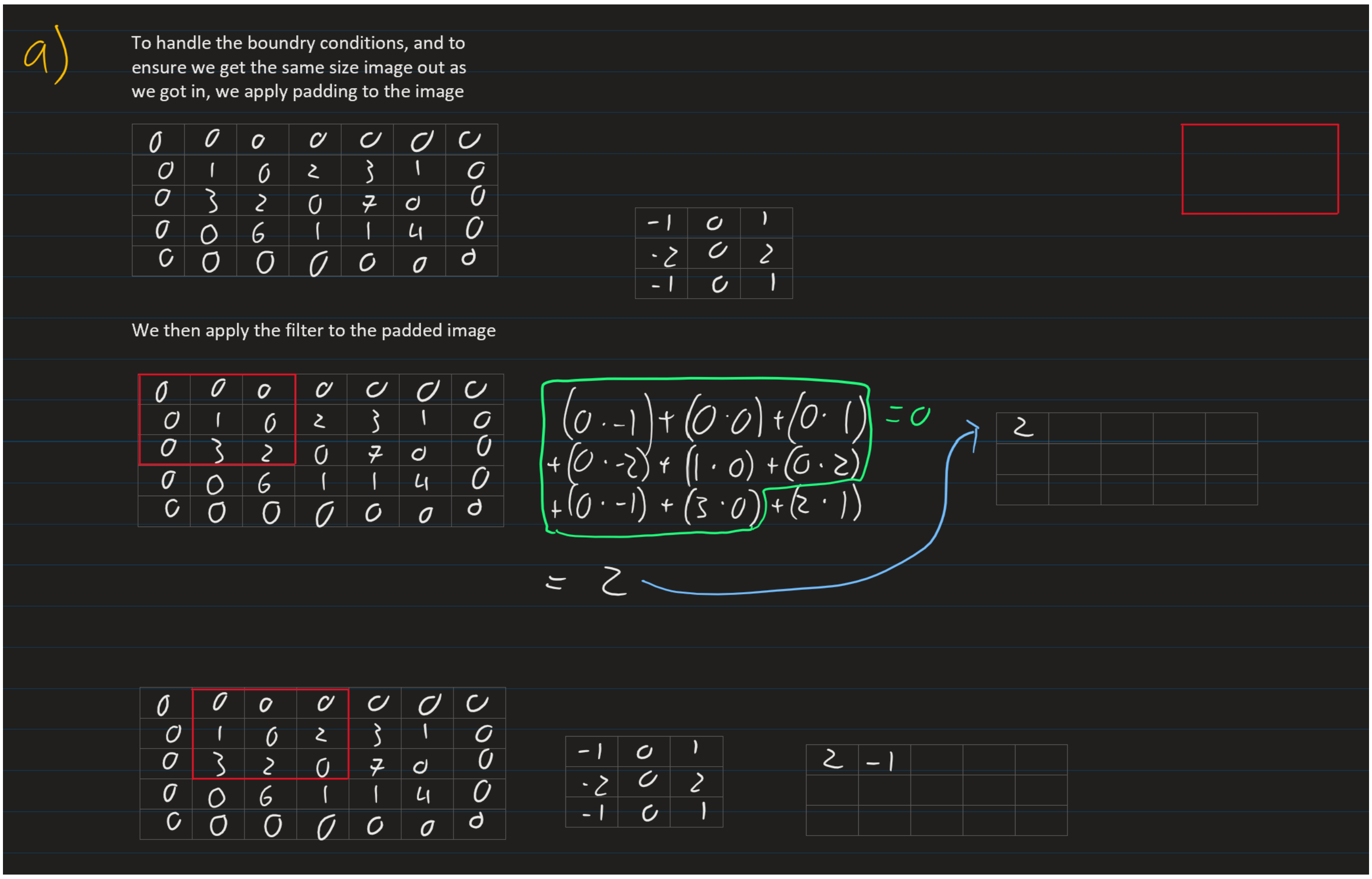
This is an outline for your report to ease the amount of work required to create your report. Jupyter notebook supports markdown, and I recommend you to check out this cheat sheet (https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet). If you are not familiar with markdown.

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)



a)

To handle the boundry conditions, and to ensure we get the same size image out as we got in, we apply padding to the image

0	0	0	0	0	0	0
0	1	0	2	3	1	0
0	3	2	0	7	0	0
0	0	6	1	1	4	0
0	0	0	0	0	0	0

-1	0	1
-2	0	2
-1	0	1

We then apply the filter to the padded image

0	0	0	0	0	0	0
0	1	0	2	3	1	0
0	3	2	0	7	0	0
0	0	6	1	1	4	0
0	0	0	0	0	0	0

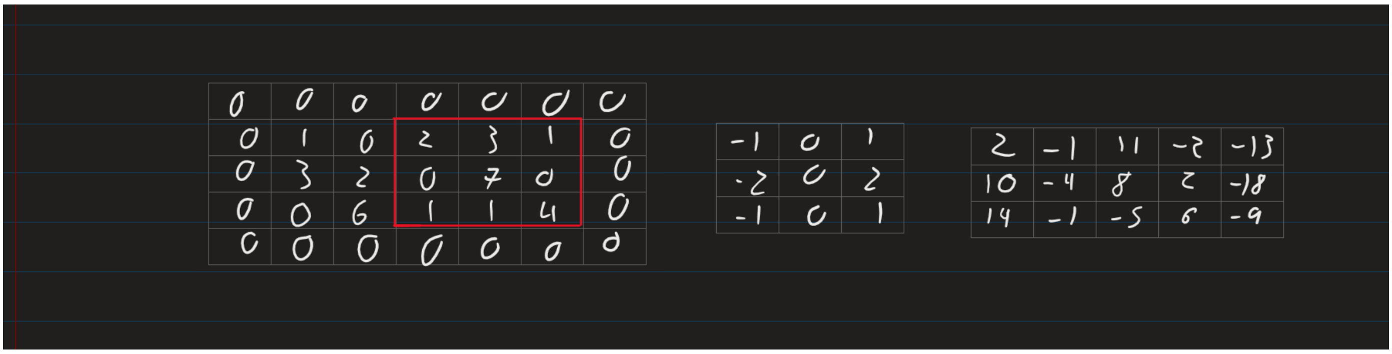
$$(0\cdot -1)+(0\cdot 0)+(0\cdot 1)+(0\cdot -2)+(1\cdot 0)+(0\cdot 2)+(0\cdot -1)+(3\cdot 0) = 0$$
$$+(2\cdot 1)$$
$$= 2$$

2				

0	0	0	0	0	0	0
0	1	0	2	3	1	0
0	3	2	0	7	0	0
0	0	6	1	1	4	0
0	0	0	0	0	0	0

-1	0	1
-2	0	2
-1	0	1

2	-1			



0	0	0	0	0	0	0
0	1	0	2	3	1	0
0	3	2	0	7	0	0
0	0	6	1	1	4	0
0	0	0	0	0	0	0

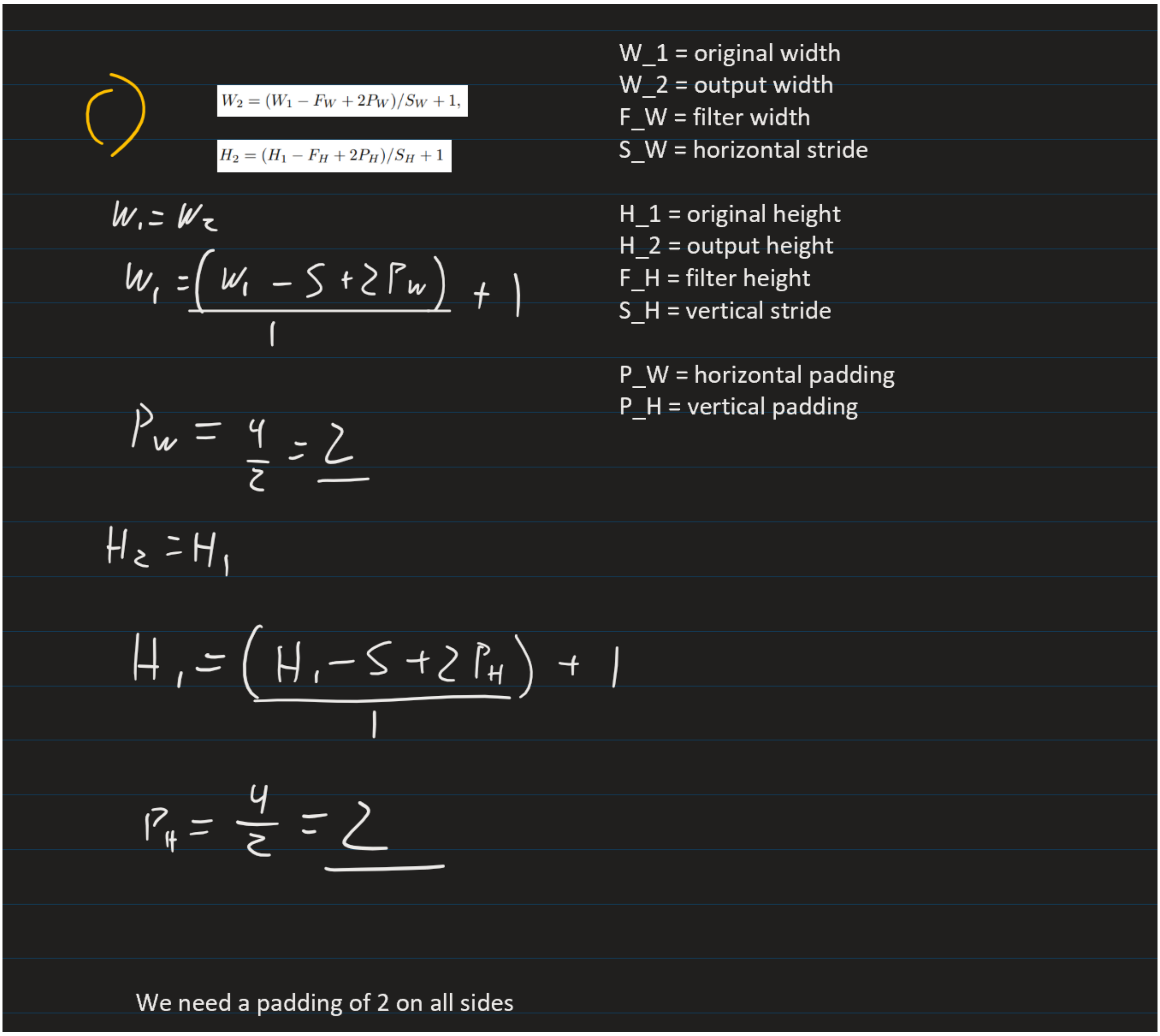
-1	0	1
-2	0	2
-1	0	1

2	-1	11	-2	-13
10	-4	8	2	-18
14	-1	-5	6	-9

task 1b)

the convolutional layer

task 1c)



c)

$W_2 = (W_1 - F_W + 2P_W)/S_W + 1,$

$H_2 = (H_1 - F_H + 2P_H)/S_H + 1$

W_1 = original width
W_2 = output width
F_W = filter width
S_W = horizontal stride

H_1 = original height
H_2 = output height
F_H = filter height
S_H = vertical stride

P_W = horizontal padding
P_H = vertical padding

$W_1 = W_2$

$$W_1 = \frac{(W_1 - 5 + 2P_W)}{1} + 1$$

$$P_W = \frac{4}{2} = \underline{2}$$

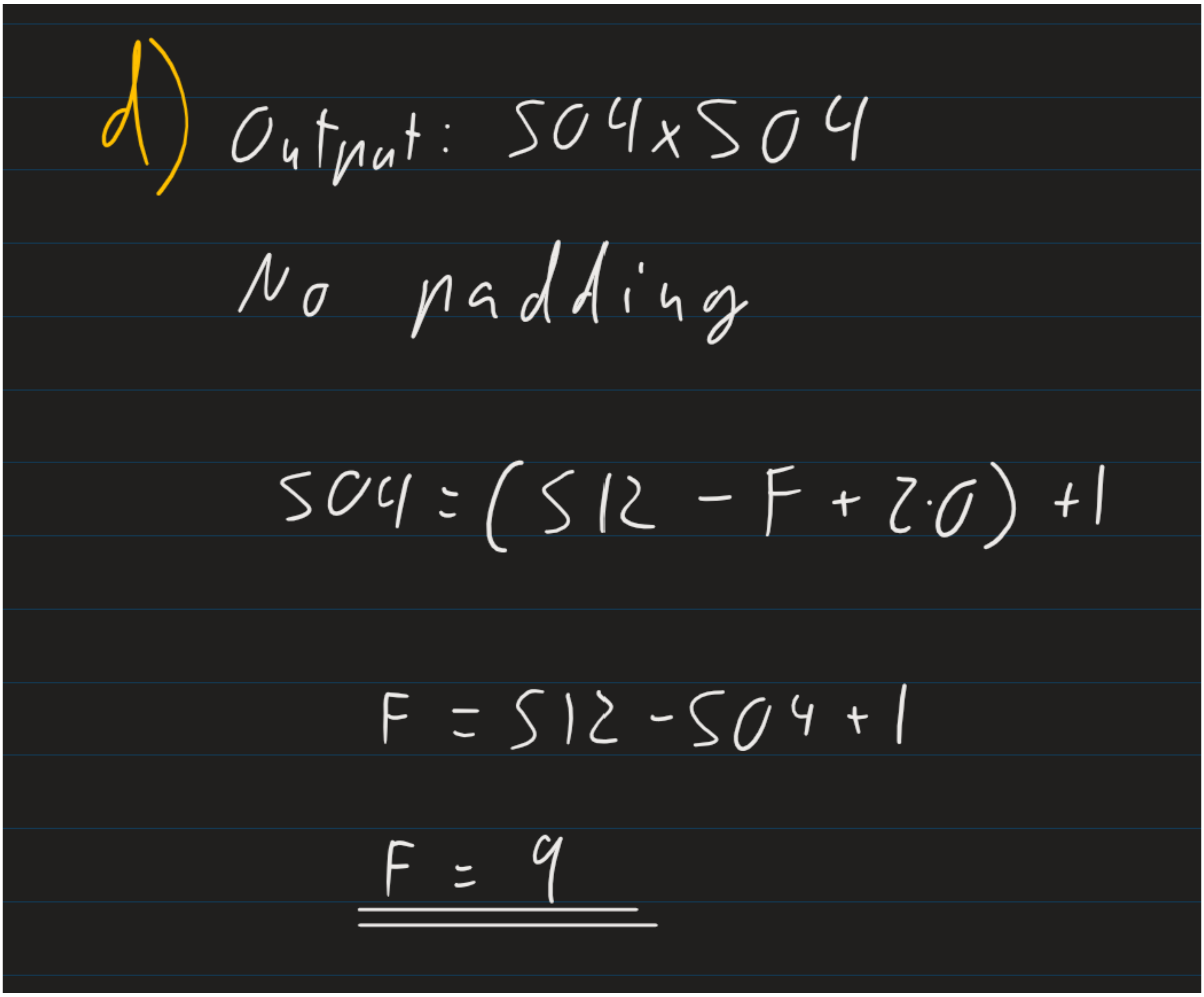
$H_2 = H_1$

$$H_1 = \frac{(H_1 - 5 + 2P_H)}{1} + 1$$

$$P_H = \frac{4}{2} = \underline{2}$$

We need a padding of 2 on all sides

task 1d)



d) Output: 504x504

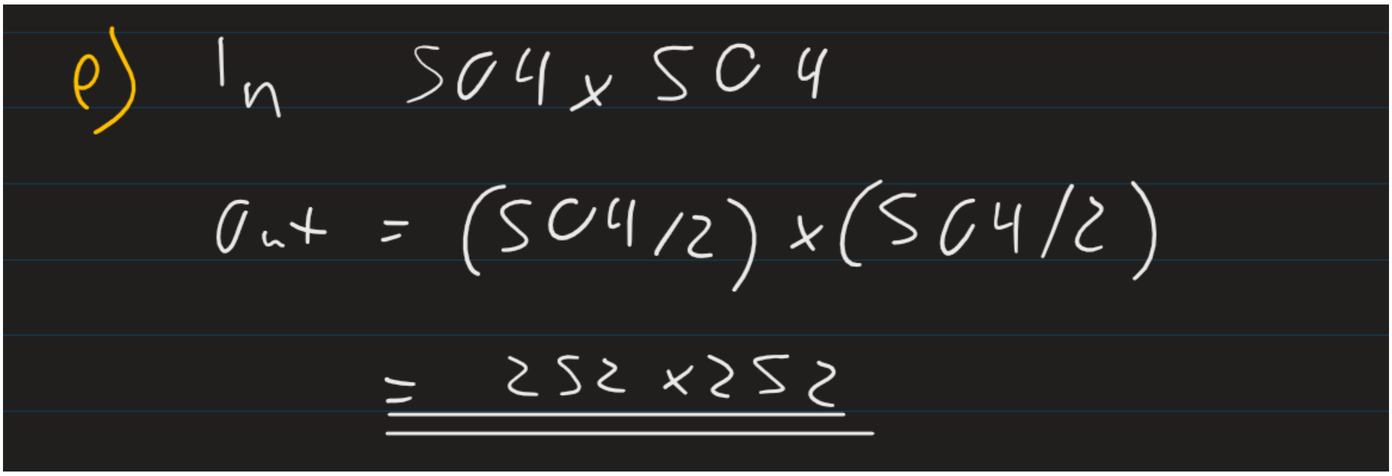
No padding

$$504 = (512 - F + 2 \cdot 0) + 1$$

$$F = 512 - 504 + 1$$

$$\underline{\underline{F = 9}}$$

task 1e)

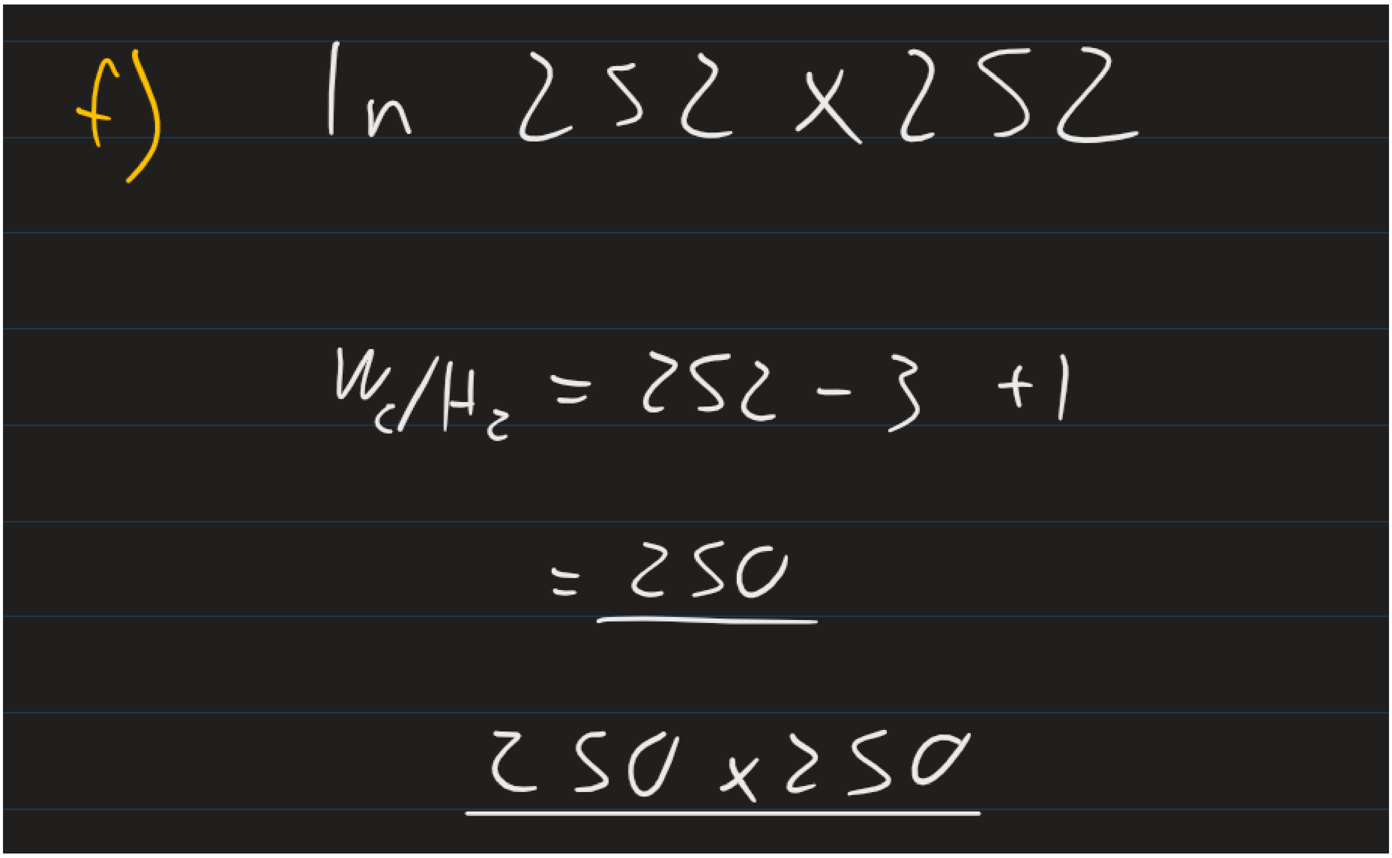


e) In 504x504

$$Out = (504/2) \times (504/2)$$

$$\underline{\underline{= 252 \times 252}}$$

task 1f)



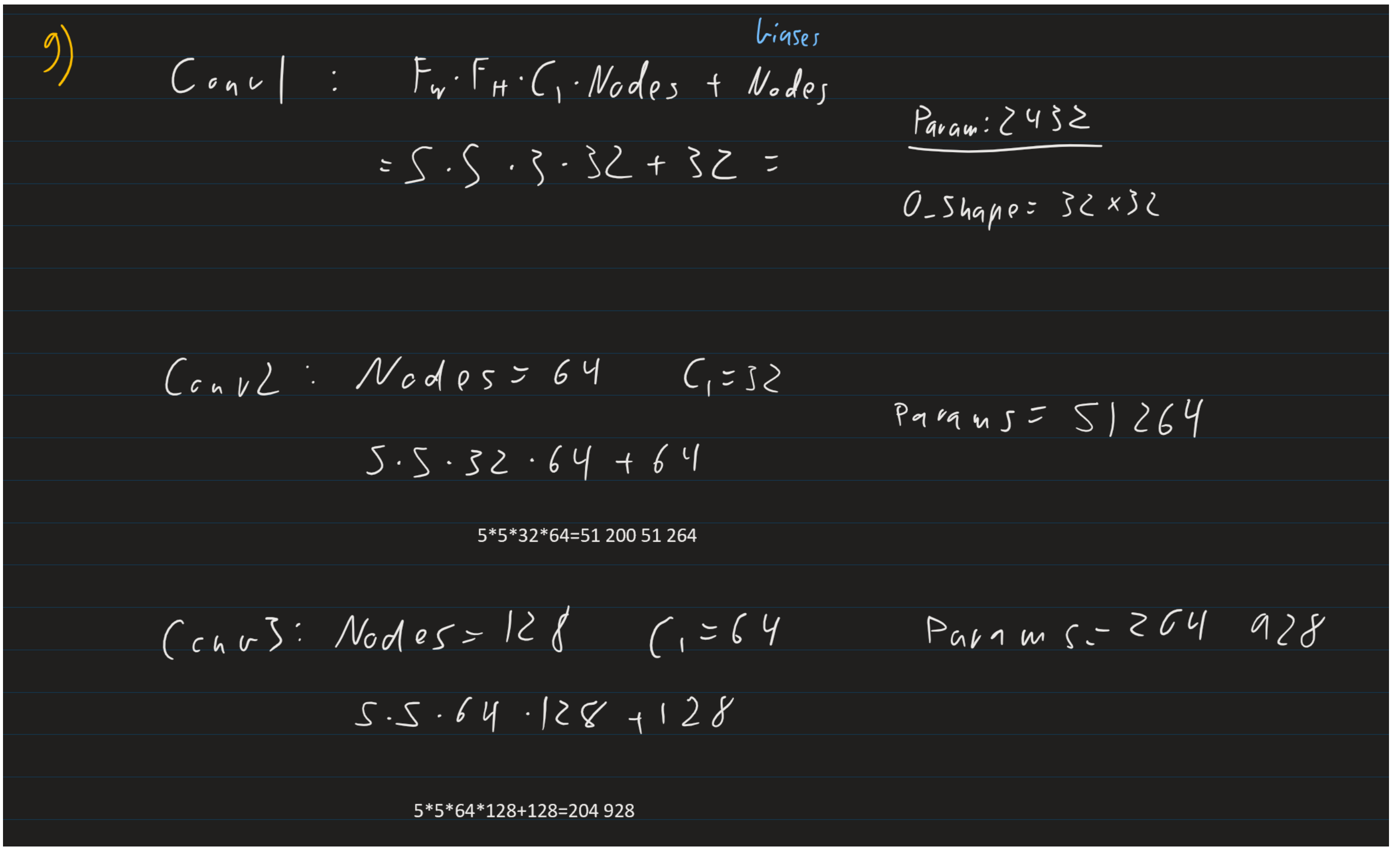
f) In 252 x 252

$W_c/H_c = 252 - 3 + 1$

$= \underline{250}$

$\underline{250 \times 250}$

task 1g)



g)

$$\text{Conv1} : F_W \cdot F_H \cdot C_1 \cdot Nodes + \overset{\text{biases}}{Nodes}$$

$$= 5 \cdot 5 \cdot 3 \cdot 32 + 32 =$$

$\underline{\text{Param}: 2432}$

O_Shape = 32x32

Conv2 : Nodes = 64 $C_1 = 32$

$5 \cdot 5 \cdot 32 \cdot 64 + 64$

Params = 51264

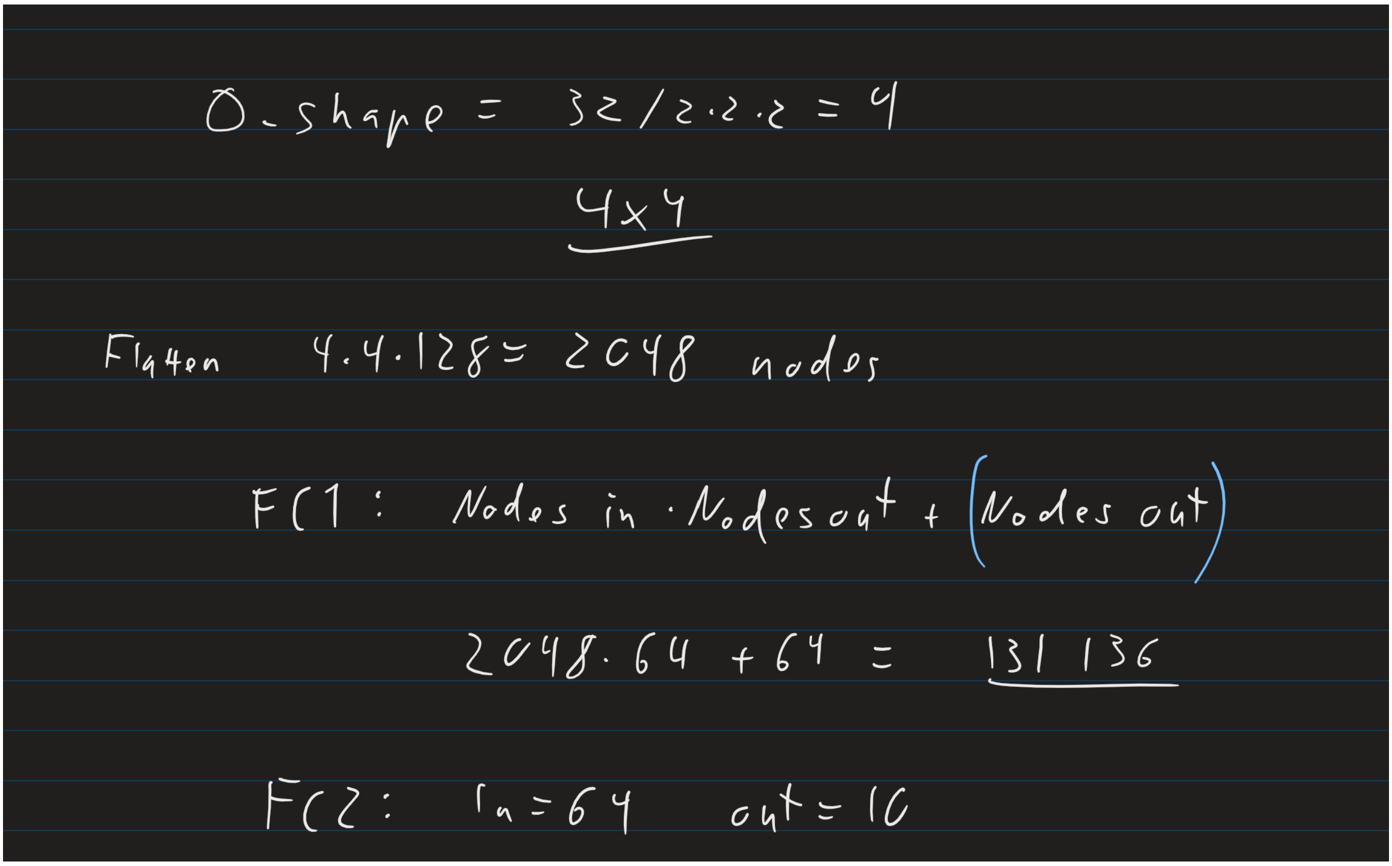
5*5*32*64=51 200 51 264

Conv3: Nodes = 128 $C_1 = 64$

Params = 204 928

$5 \cdot 5 \cdot 64 \cdot 128 + 128$

5*5*64*128+128=204 928



O-shape = 32 / 2·2·2 = 4

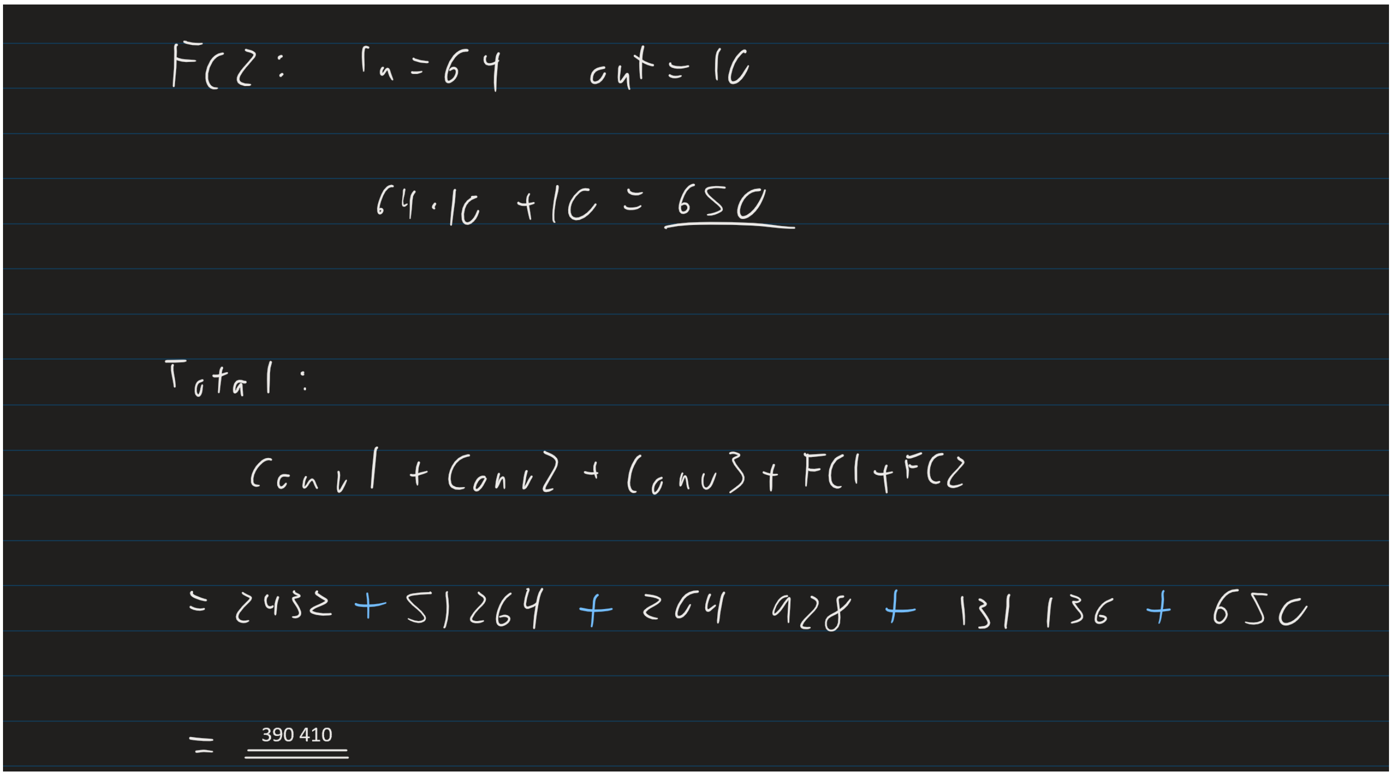
$\underline{4 \times 4}$

Flatten 4·4·128 = 2048 nodes

FC1 : Nodes in · Nodes out + (Nodes out)

2048·64 + 64 = $\underline{131\ 136}$

FC2 : in = 64 out = 10



FC2 : in = 64 out = 10

64·10 + 10 = $\underline{650}$

Total :

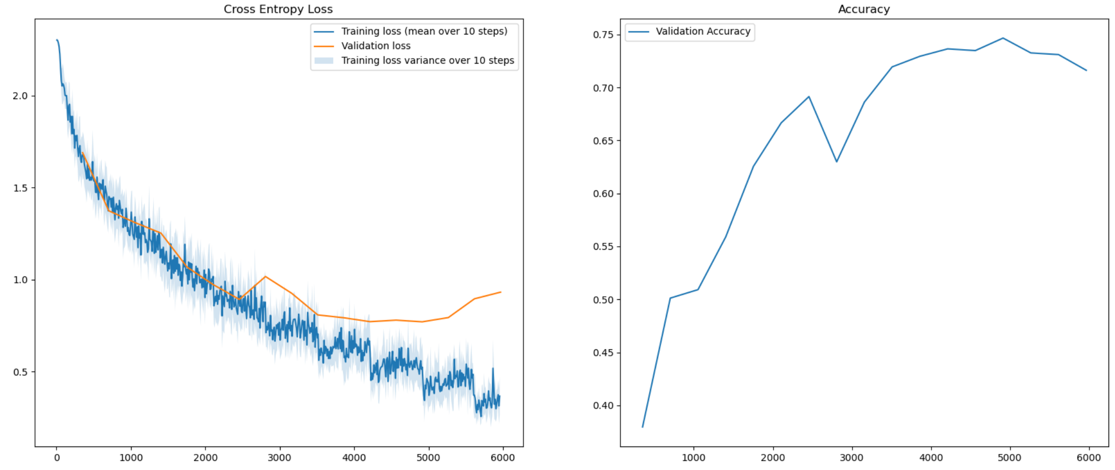
Conv1 + Conv2 + Conv3 + FC1 + FC2

= 2432 + 51264 + 264 928 + 131 136 + 650

= $\underline{\underline{390\ 410}}$

Task 2

Task 2a)



Task 2b)

train accuracy 0.8824, val accuracy 0.7162, test accuracy 0.7179

Task 3

Task 3a)

For the first conv net I used this architecture without changing anything but the filter size and padding.

Parameter	Value
optimizer	SGD
Learning rate	5e-2
batch size	64
Filter size	3
padding	1

==========

Layer	LayerType	Number of Hidden Units/filters	Activation func
1	conv2d	64	ReLu
2	MaxPool2d	-	-
4	conv2d	128	ReLu
4	MaxPool2d	-	-
6	conv2d	256	ReLu
6	MaxPool2d	-	-
	Flatten	-	
7	Fully-connected	64	ReLU
8	Fully-connected	10	Softmax

For the second conv net I used this architecture with batch normalization after the convolutional layers. I also turned off bias for the convolutional layers, as I've read it is not supposed to be used with batch normalization.

Parameter	Value
optimizer	SGD
Learning rate	5e-2
batch size	64
Filter size	3
padding	1

=====

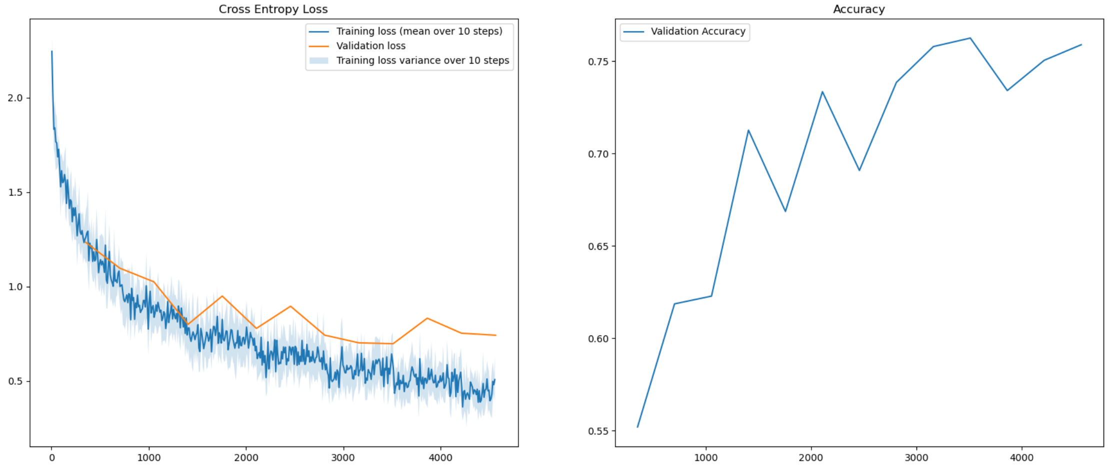
Layer	LayerType	Number of Hidden Units/filters	Activation func
1	conv2d	16	ReLu
2	conv2d	16	ReLu
2	MaxPool2d	-	-
3	conv2d	32	ReLu
4	conv2d	32	ReLu
4	MaxPool2d	-	-
5	conv2d	64	ReLu
6	conv2d	64	ReLu
6	MaxPool2d	-	-
	Flatten	-	
7	Fully-connected	64	ReLU
8	Fully-connected	10	Softmax

Accuracies

Model 1		Model 2	
final train loss	0.75	final train loss	0.79
train accuracy	0.902	train accuracy	0.845
validation accuracy	0.766	validation accuracy	0.768
test accuracy	0.751	test accuracy	0.769

Task 3b)

I chose highest test accuracy as the measurement of what was the best model



Task 3c)

Changing the filter size to 3 with 1 padding helped a lot. I think this is because you recogize finer detailes with a smaller filter. It's more thurough if you will.

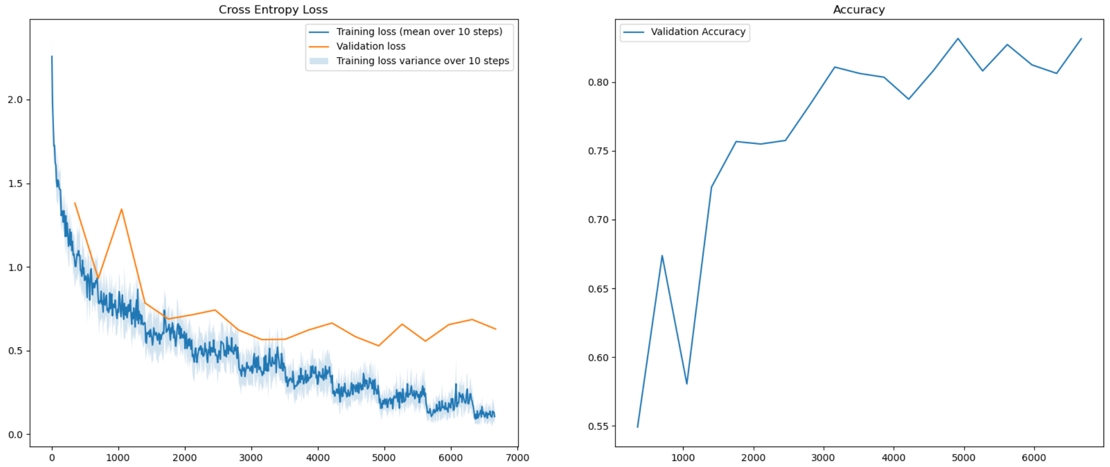
Adding batch normalization sped up the training.

Changing the architecture and number of conv layers helped quite a bit as well. This is likely since it makes the model more complex.

Additionally, changing the number of channels helped.

Task 3d)

Task 3e)



This time I changed the channels back to [32, 32, 64, 64, 128, 128] for the conv layers

Improved model	
Optimizer	Adam
Learning rate	0.001
batch size	64

Improved model	
Filter size	3
padding	1
weight_decay	1e-5
early stop count	6

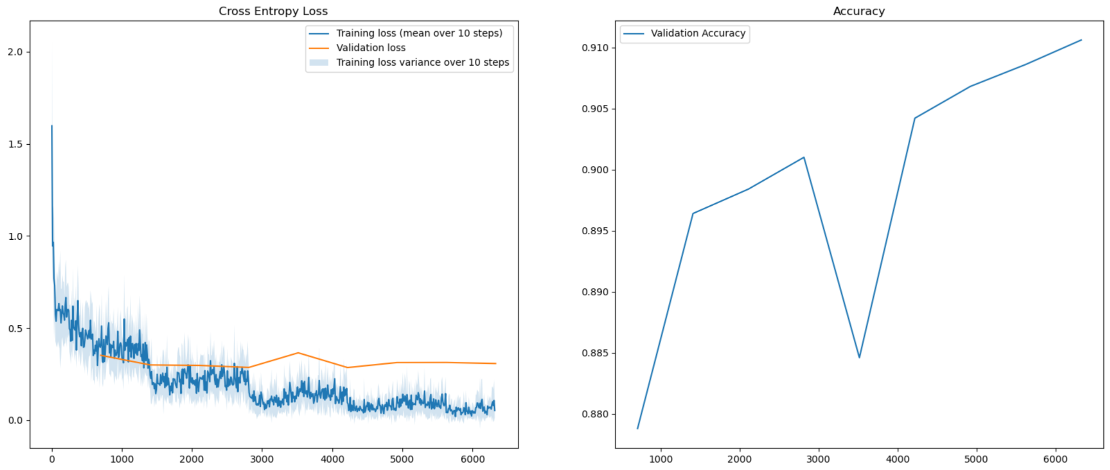
Improved Model	
final train loss	0.63
train accuracy	0.969
validation accuracy	0.832
test accuracy	0.827

Task 3f)

Yes, we clearly see signs of overfitting. The model has an accuracy of almost 97% for the training dataset, and only around 83% for the validation and test sets.

Task 4

Task 4a)



Accuracies

final loss 0.31 train 0.9897 val 0.9106 test 0.8991

Task 4b)

FILL IN ANSWER

Task 4c)

FILL IN ANSWER