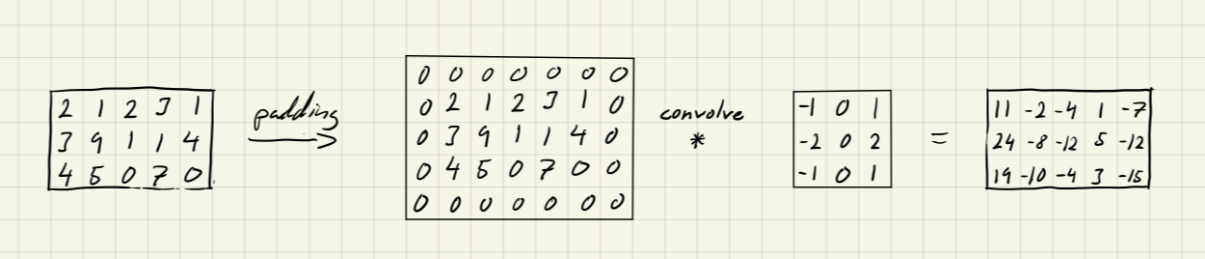
Assignment 3

Task 1

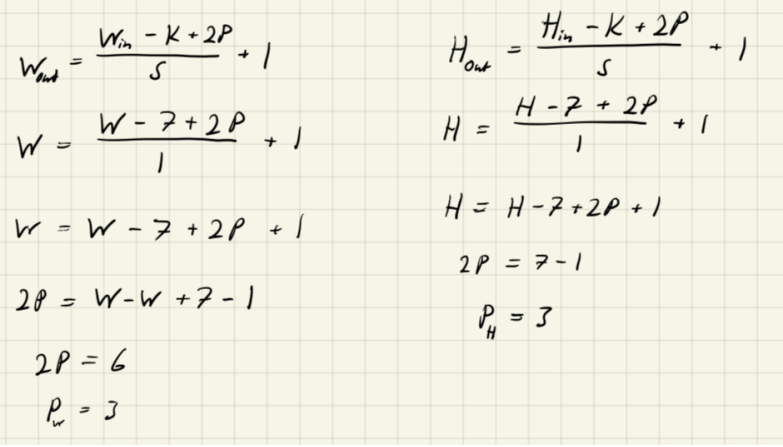
a)



b)

Max pooling makes the CNN invariant to small translational shifts because the highest pixel value in the patch will be pooled either way, no matter if it is at pixel position i,j or i+1, j+1 (as long as i,j and i+1, j+1 are in the same patch).

c)



$$W_{out} = \frac{W_{in} - K + 2P}{S} + 1 \qquad H_{out} = \frac{H_{in} - K + 2P}{S} + 1$$

$$W = \frac{W - 7 + 2P}{1} + 1 \qquad H = \frac{H - 7 + 2P}{1} + 1$$

$$W = W - 7 + 2P + 1 \qquad H = H - 7 + 2P + 1$$

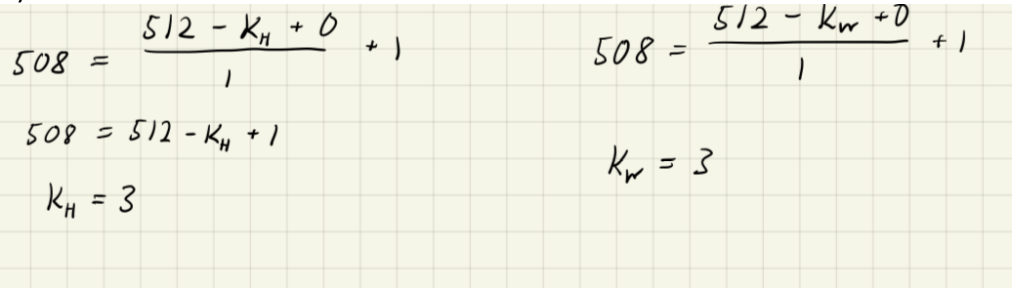
$$2P = W - W + 7 - 1 \qquad 2P = 7 - 1$$

$$2P = 6 \qquad P_H = 3$$

$$P_W = 3$$

Zero padding needs to be 3 in both the height and the width.

d)



$$508 = \frac{512 - K_H + 0}{1} + 1 \qquad 508 = \frac{512 - K_W + 0}{1} + 1$$

$$508 = 512 - K_H + 1 \qquad K_W = 3$$

$$K_H = 3$$

The Kernel size is (3,3)

e)
With a 2x2 pooling layer the output shape will be (508/2, 508/2) = (254, 254)

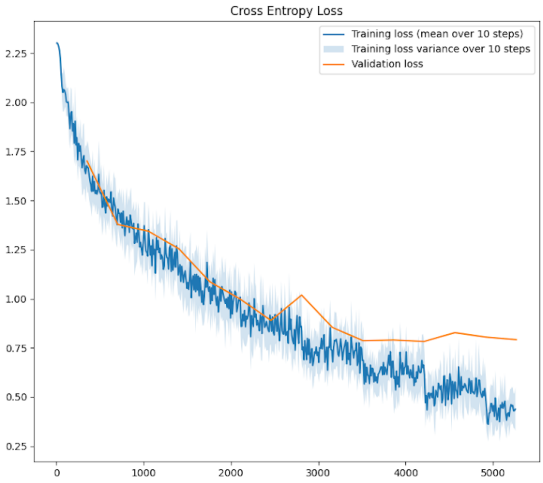
f)

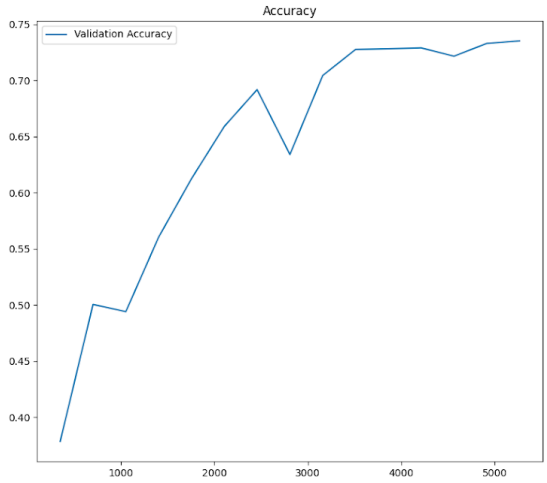
$$H_{out} = \frac{254 - 3 + 0}{1} + 1 = 252 \qquad W_{out} = \frac{254 - 3 + 0}{1} + 1 = 252$$

The output is of shape (252, 252)

g)
#parameters = (5*5*3*32 + 32) + (5*5*32*64 + 64) + (5*5*64*128 + 128) + (4*4*128*64 + 64) + (64*10 + 10) = 390410

Task 2
a)





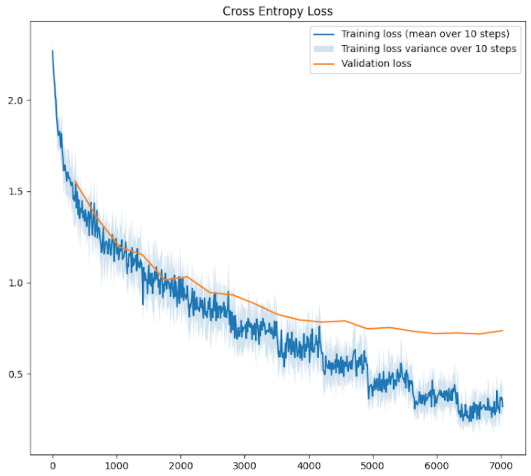
Final validation accuracy = 0.735

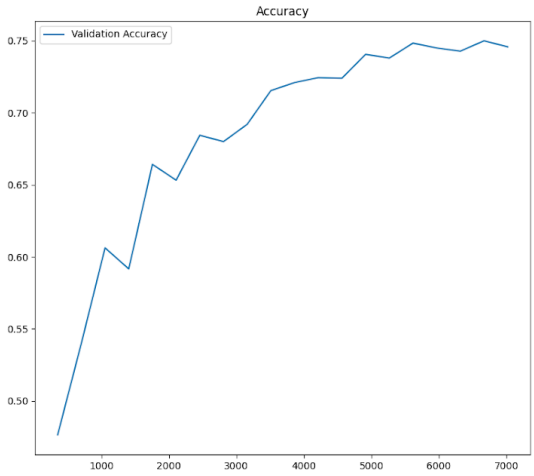
Task 3
a)

Layer	Layer type	#Hidden units	Activation function
1	Conv2D	32	ReLU
1	MaxPool2D		
2	Conv2D	64	ReLU
2	MaxPool2D		
3	Flatten		

3	FC	10368	ReLU
3	Dropout		
4	FC	10	None

b)
Validation accuracy = 0.75
Test accuracy = 0.725





c)
Improvements:

- Switching kernel_size to 3x3.
- Adding more parameters in the conv filters. The network was likely underfitting and couldn't properly model the training data distribution (too much bias)
- Adding more neurons in dense layers. Not surprising that increasing the capacity helped the model learn quicker. Basically it had more neurons to learn the feature space with. However, increasing the model parameters too much made it overfit.

Not useful:

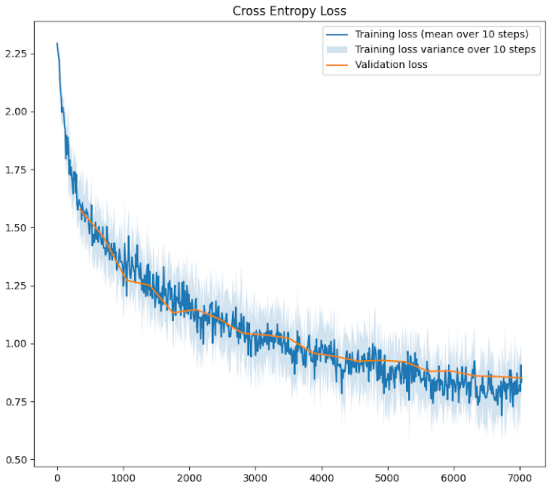
- Adam: The training is very unstable with Adam, especially at higher learning rates. This is not surprising as it uses momentum. With lower learning rates the training process becomes slightly smoother but it does not reach a high accuracy (the training is too slow)
- Data augmentation did not work. The model may not have had enough capacity to adapt to the variations in the data. RandomVerticalFlip, RandomRotation and ColorJitter were used in the transforms.

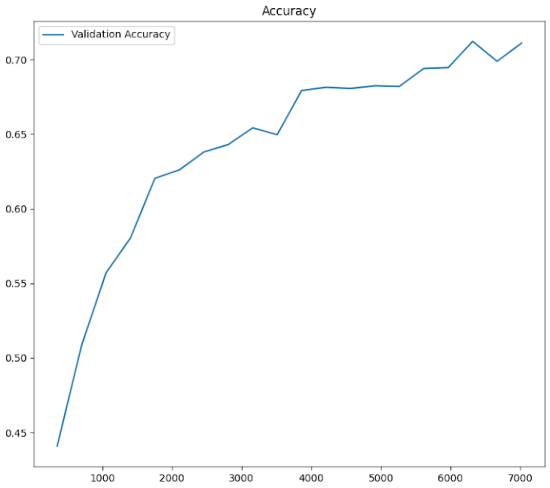
Not worked well:

- Data augmentation: probably not enough epochs

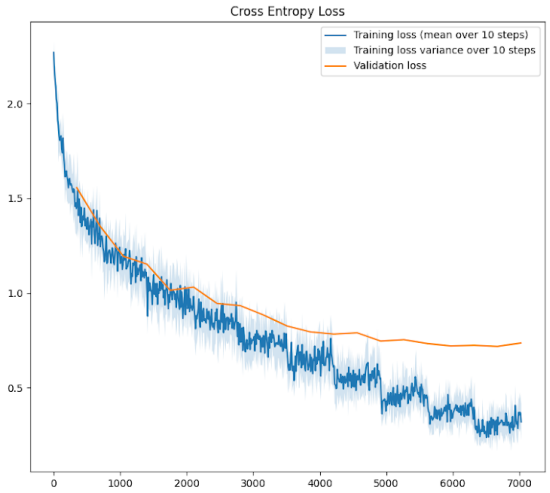
d)
By far best improvements from increasing model parameters and adding extra layers

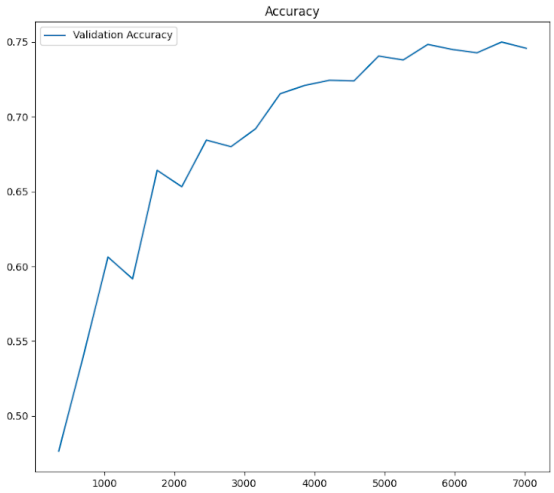
Before increase:





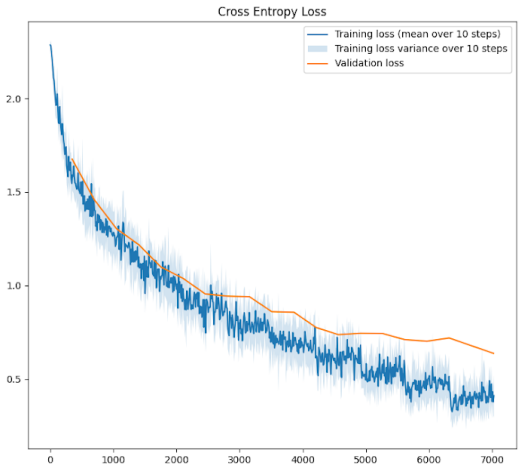
After increase of parameters:

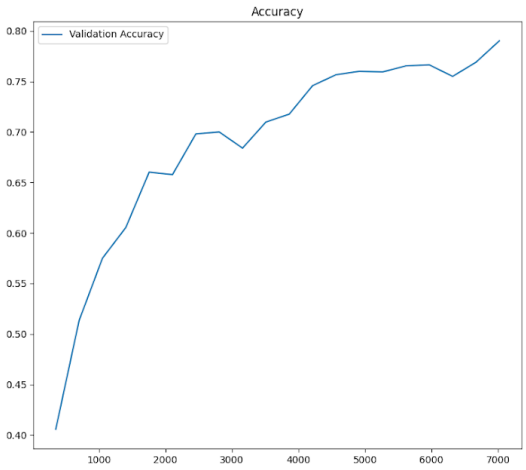




The accuracy increases, but we can see that it starts to overfit on the data as the capacity increases as well.

e)

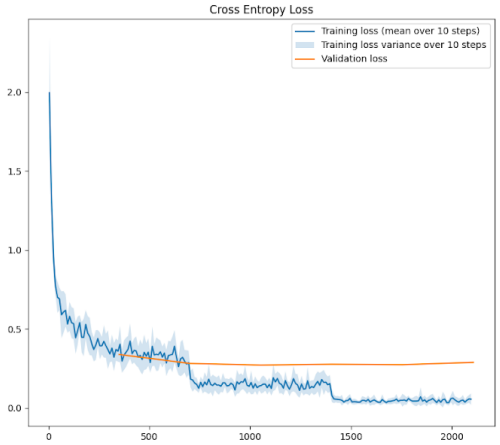


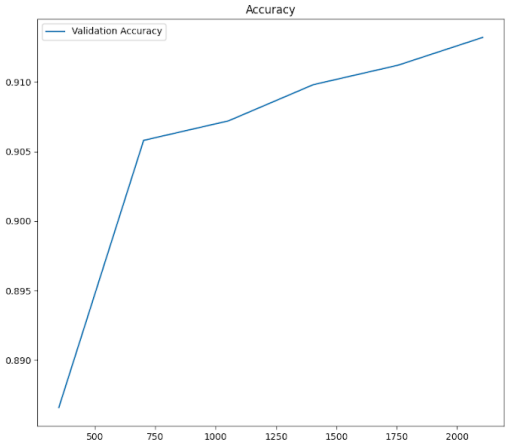


Validation accuracy = 0.79
Test accuracy = 0.75

f)
We can see that the validation loss stops decreasing proportionally to the training loss towards the last epochs. This is a sign of the model overfitting because it means that it is starting to perform better on the training data (which it is trained on) than on unseen data.

Task 4





Final validation accuracy: 0.913
Final test accuracy: 0.89