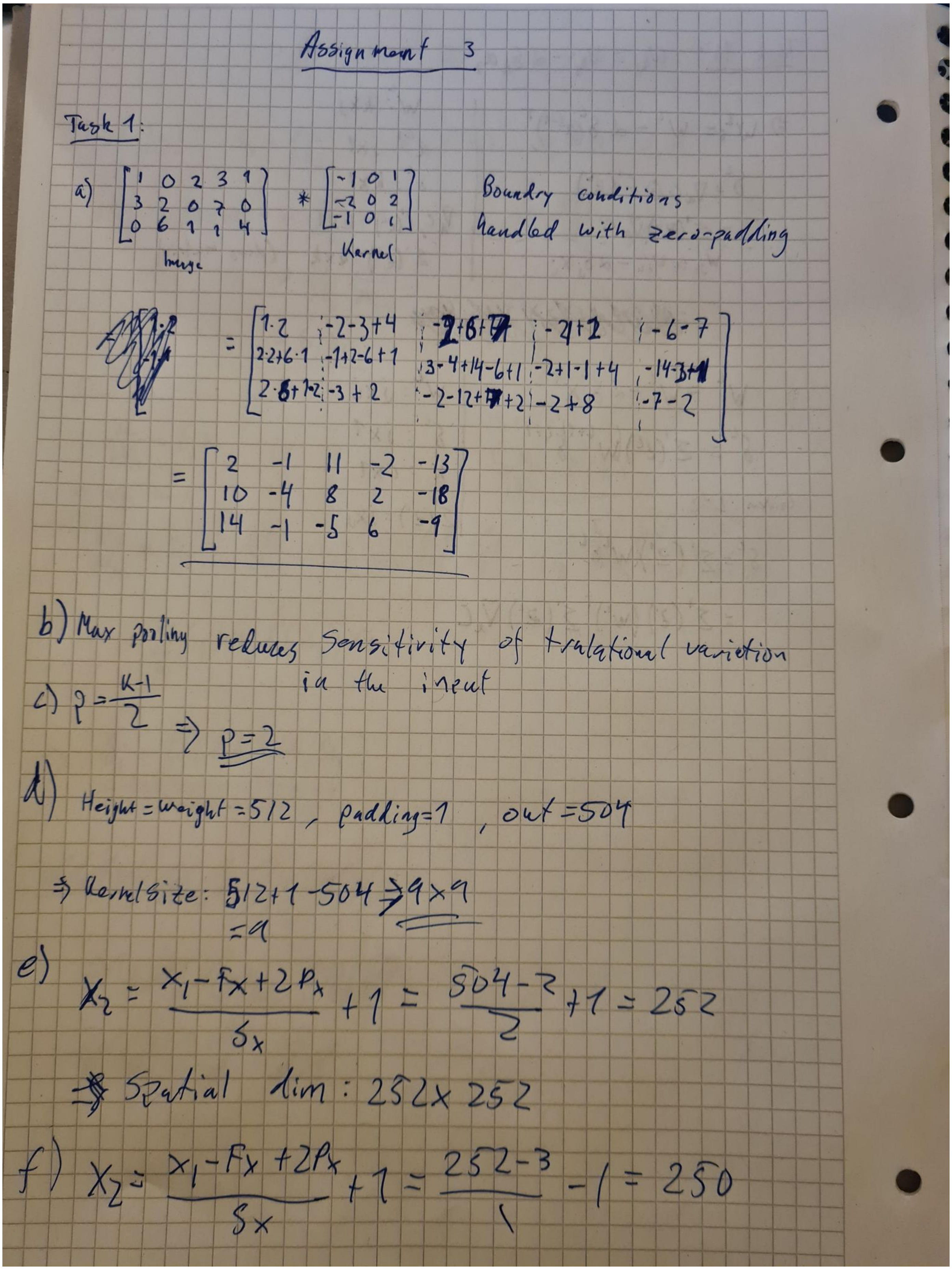
Assignment 3 Report

Task 1 ¶



Assignment 3

Task 1:

a) $\begin{bmatrix} 1 & 0 & 2 & 3 & 1 \\ 3 & 2 & 0 & 7 & 0 \\ 0 & 6 & 1 & 1 & 4 \end{bmatrix} * \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$

Image Kernel

Boundry conditions handled with zero-padding

$$= \begin{bmatrix} 1\cdot 2 & -2-3+4 & -2+6+7 & -4+2 & -6-7 \\ 2\cdot 2+6\cdot 1 & -1+2-6+1 & 3-4+14-6+1 & -2+1-1+4 & -14-3+1 \\ 2\cdot 6+1\cdot 2 & -3+2 & -2-12+7+2 & -2+8 & -7-2 \end{bmatrix}$$

$$= \begin{bmatrix} 2 & -1 & 11 & -2 & -13 \\ 10 & -4 & 8 & 2 & -18 \\ 14 & -1 & -5 & 6 & -9 \end{bmatrix}$$

b) Max pooling reduces sensitivity of tralational variation in the input

c) $p = \frac{k-1}{2} \Rightarrow \underline{p=2}$

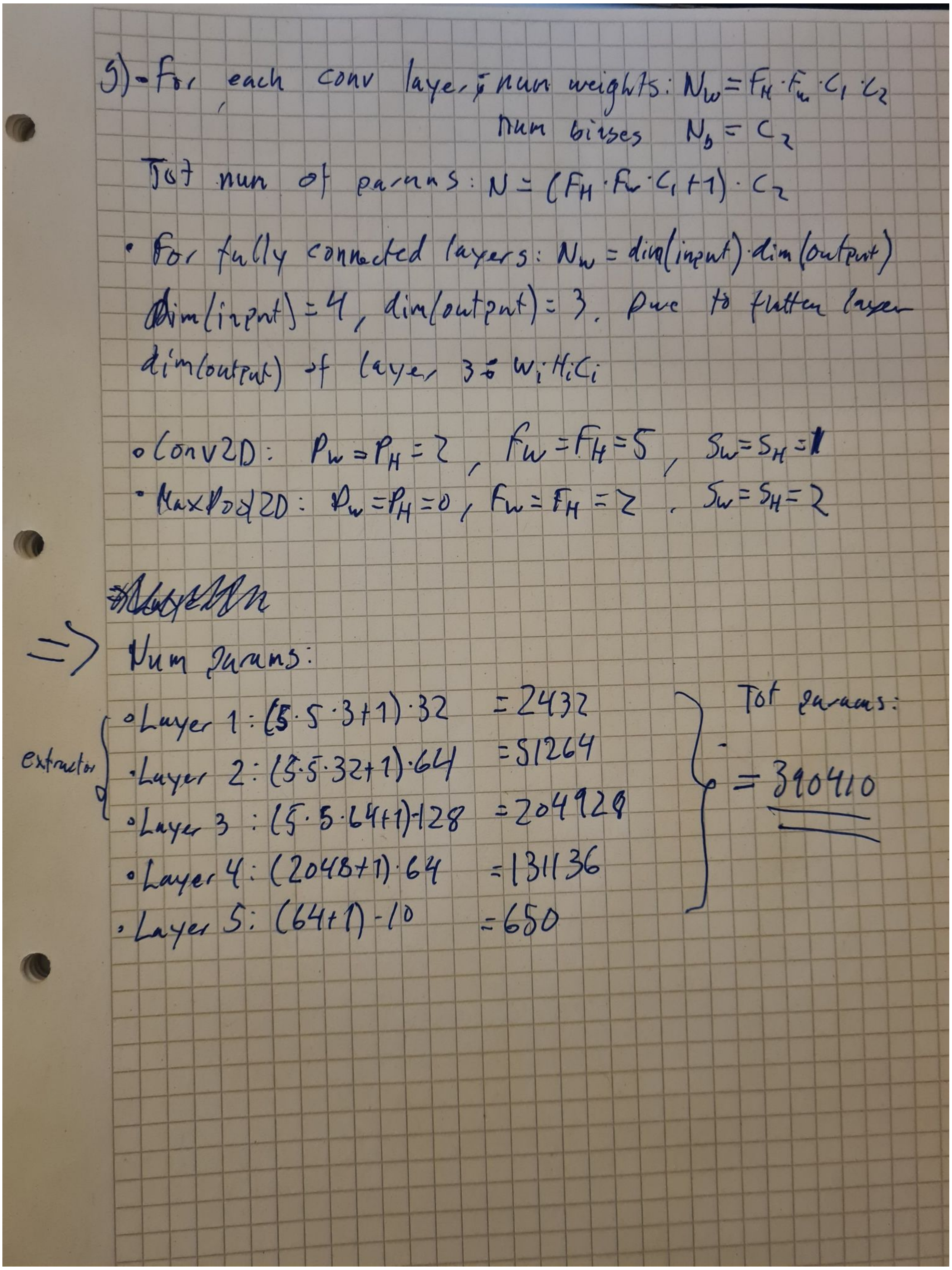
d) Height = weight = 512, padding = 1, out = 504

$\Rightarrow$ Kernel size: $512+1-504 = 9 \Rightarrow \underline{9\times 9}$

e) $$X_2 = \frac{X_1 - F_x + 2P_x}{S_x} + 1 = \frac{504-2}{2} + 1 = 252$$

$\Rightarrow$ Spatial dim: 252×252

f) $$X_2 = \frac{X_1 - F_x + 2P_x}{S_x} + 1 = \frac{252-3}{1} - 1 = 250$$



5) - For each conv layer, num weights: $N_w = F_H \cdot F_w \cdot C_1 \cdot C_2$

num bisses $N_b = C_2$

Tot num of params: $N = (F_H \cdot F_w \cdot C_1 + 1) \cdot C_2$

• For fully connected layers: $N_w = \dim(\text{input}) \cdot \dim(\text{output})$

dim(input) = 4, dim(output) = 3. Due to flatten layer

dim(output) of layer 3 = $W_i \cdot H_i \cdot C_i$

∘ Conv2D: $P_W = P_H = 2$, $F_W = F_H = 5$, $S_W = S_H = 1$

• MaxPool2D: $P_w = P_H = 0$, $F_w = F_H = 2$, $S_w = S_H = 2$

~~[illegible]~~

$\Rightarrow$ Num params:

extractor

∘ Layer 1: $(5 \cdot 5 \cdot 3 + 1) \cdot 32 = 2432$

• Layer 2: $(5 \cdot 5 \cdot 32 + 1) \cdot 64 = 51264$

∘ Layer 3: $(5 \cdot 5 \cdot 64 + 1) \cdot 128 = 204928$

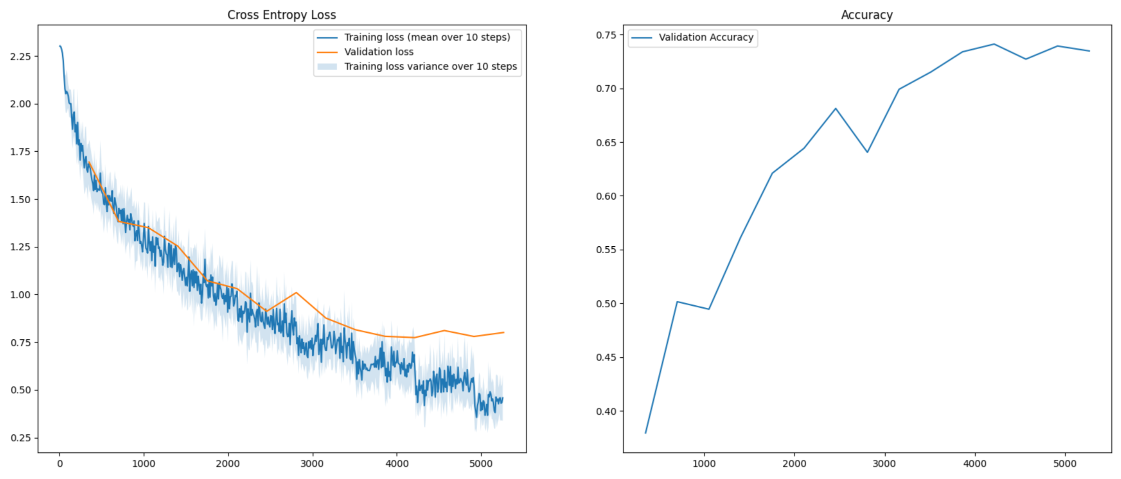
• Layer 4: $(2048 + 1) \cdot 64 = 131136$

• Layer 5: $(64 + 1) \cdot 10 = 650$

Tot params: $= \underline{\underline{390410}}$

Task 2

Task 2a)



Task 2b)

Final train accuracy = 0.8788895606994629

Final validation accuracy = 0.7346000075340271

Final test accuracy = 0.7339999675750732

Epochs needed: 7, Global steps: 5265

Task 3

Task 3a)

```
Both networks:
Optimizer: SGD
Learning rate: 5e-2
Batch size: 64
Weight initialization: kaiming unfirom
Data augmentation added: transforms.RandomHorizontalFlip(p=0.5)

Network 1:
Net1(
  (feature_extractor): Sequential(
    (0): Conv2d(3, 32, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
    (1): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU()
    (3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (4): Conv2d(32, 64, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
    (5): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (6): ReLU()
    (7): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (8): Conv2d(64, 128, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
    (9): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (10): ReLU()
    (11): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (classifier): Sequential(
    (0): Linear(in_features=2048, out_features=64, bias=True)
    (1): BatchNorm1d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU()
    (3): Linear(in_features=64, out_features=10, bias=True)
    (4): BatchNorm1d(10, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  )
)

Network 2:
Net2(
  (feature_extractor): Sequential(
    (0): Conv2d(3, 32, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
    (1): ReLU()
    (2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (3): Dropout(p=0.05, inplace=False)
    (4): Conv2d(32, 64, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
    (5): ReLU()
    (6): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (7): Dropout(p=0.05, inplace=False)
    (8): Conv2d(64, 128, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
    (9): ReLU()
    (10): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (11): Dropout(p=0.05, inplace=False)
  )
  (classifier): Sequential(
    (0): Linear(in_features=2048, out_features=64, bias=True)
    (1): ReLU()
    (2): Linear(in_features=64, out_features=10, bias=True)
  )
)
```

Task 3b)

Network 1: Final train accuracy = 0.8329036235809326

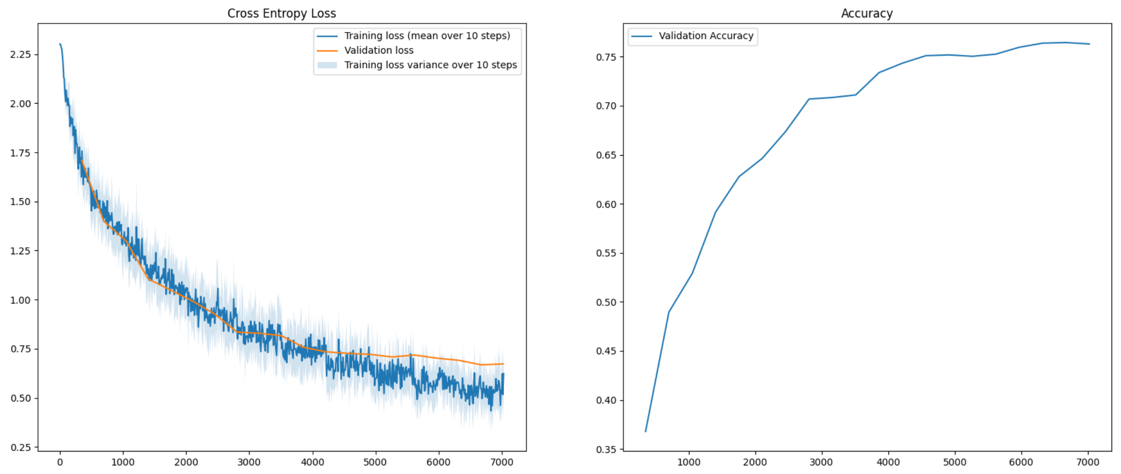
Final validation accuracy = 0.7591999769210815

Final test accuracy = 0.7532999515533447

Network 2: Final train accuracy = 0.8435944318771362

Final validation accuracy = 0.7753999829292297

Final test accuracy = 0.7613999843597412

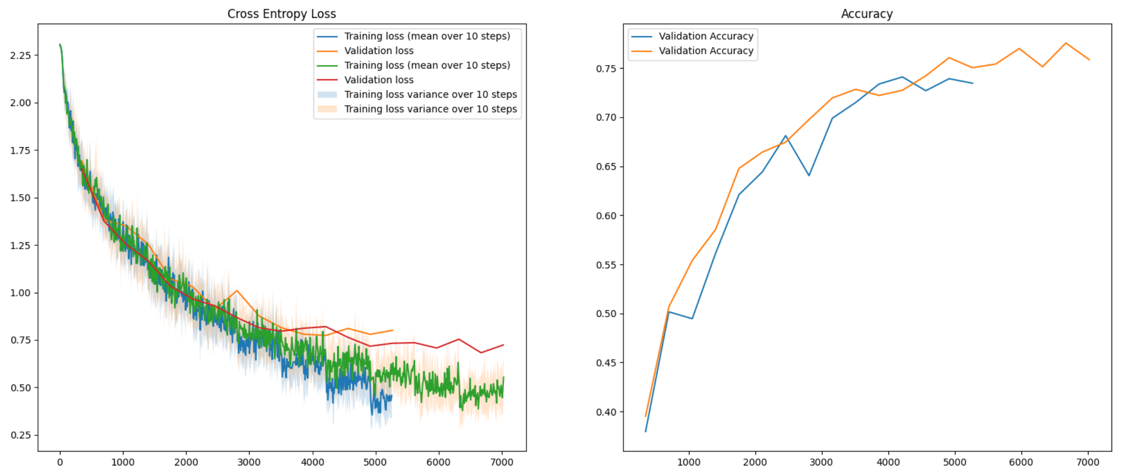


Task 3c)

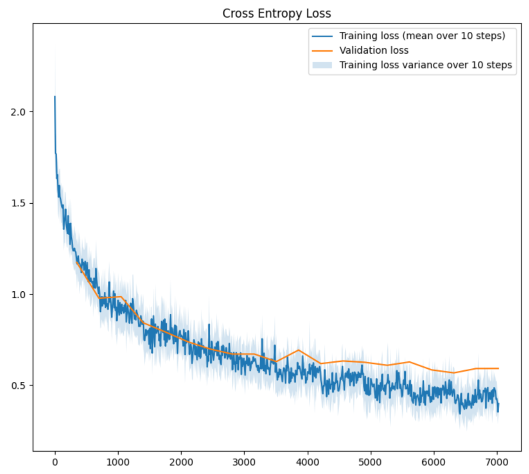
Introducing data augmentation improved the accuracy the most, since this gives the traning set a bigger variation which leads to better generalization. New network arcitecture: (conv-relu-conv-relu-pool)x3. This did not improve accuracy and as well increased computation time. Did not work since it overfits. ELU was tried instead of ReLU this worked better on the training data, but worsened the accuracy for validation and test, which is likely to be overfitting. It is hard to say why this is the case. To prevent overfitting regularization was also introduced in the form of Dropout, this slightly improved the results, but not by much. Maybe because we sat the probability of dropping a layer too low. Introducing batch normalization also improved the net slightly by reducing the training time. The weight initalization was not changed as it is uniformly distributed by default.

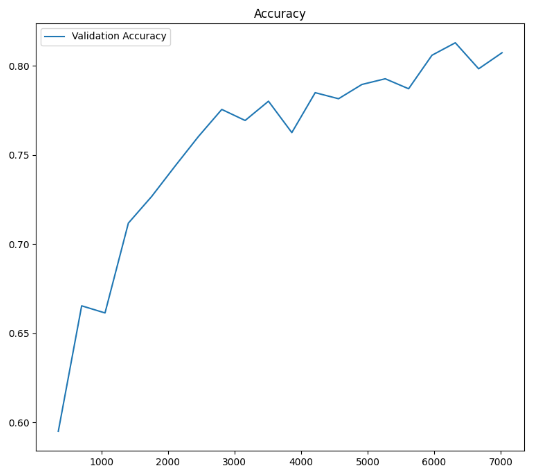
Task 3d)

As seen under including data augmentation also increases the traing steps



Task 3e)





Final test accuracy = 0.8025999665260315

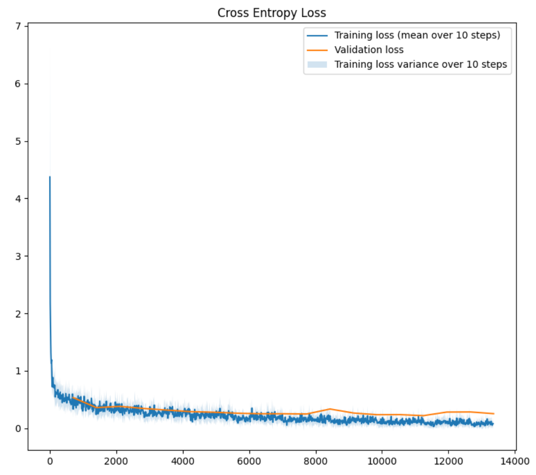
Task 3f)

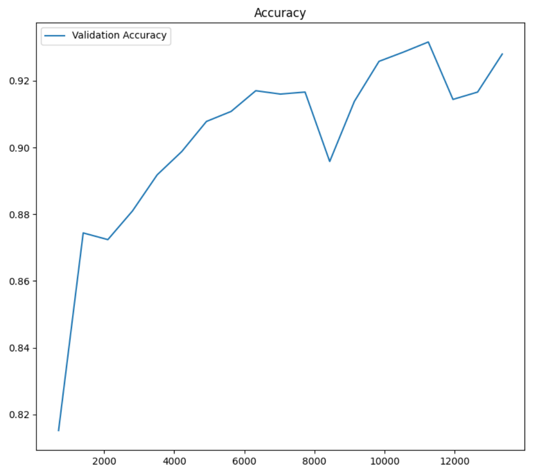
See some small sign of overfitting the best model, where the validation loss slightly diverges from the training loss.

Task 4

Task 4a)

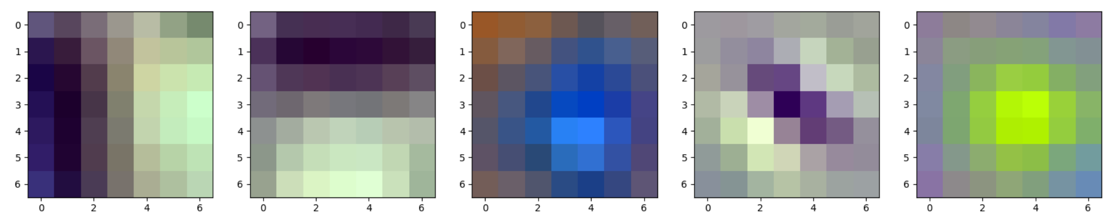
ResNet: Optimizer: Adam Learning rate: 5e-4 Batch size: 32 Data augmentation added: transforms.RandomHorizontalFlip(p=0.5), transforms.Resize(224) .

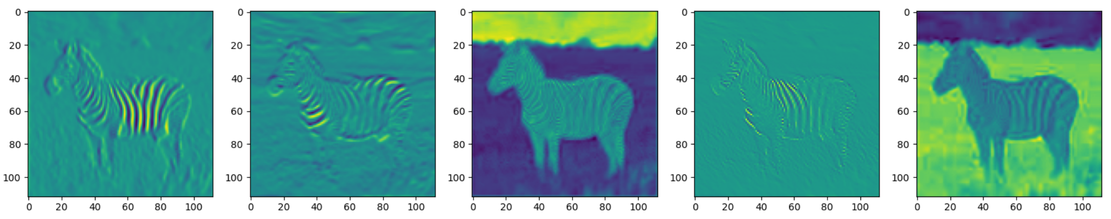




Final test accuracy = 0.9279999732971191

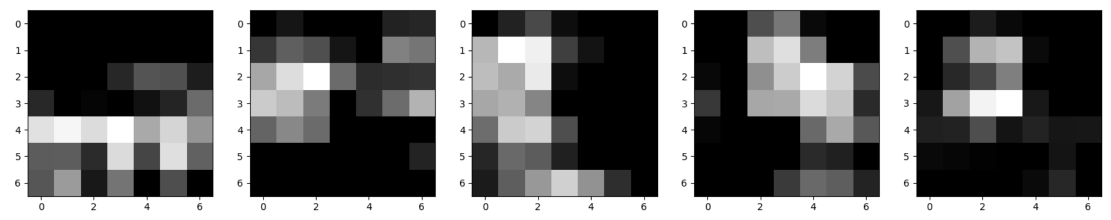
Task 4b)

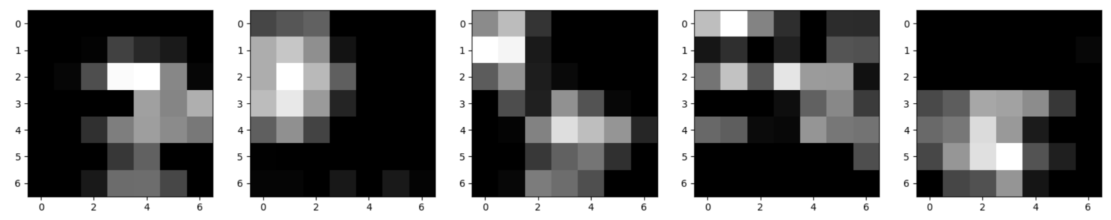




We see that some of the filters are are easy to see classify a zebra and others not as much. We also see that some of the filters focus on the details of the zebra, while others on the shape and background. In the first two pictures we can see e.g. vertical and horizontal edges.

Task 4c)





We see that with these filters it is impossible to see any resemblance to the zebra image, and that there is not much information to extract with the human eye.