Assignment Report - Group 14

March 4, 2021

1 Assignment 3 Report - Group 14

Kristoffer Norli & Bendik Stokke

2 Task 1

2.1 task 1a)

For task 1a) we chose to handle the boundary conditions by adding a one pixel wide padding of zeros around the image.

1

a)

$$\begin{array}{ccccccc} 0&0&0&0&0&0&0\\ 0&1&0&2&3&1&0\\ 0&3&2&0&7&0&0\\ 0&0&6&1&1&4&0\\ 0&0&0&0&0&0&0 \end{array} \quad \boxed{\begin{array}{ccc} -1&0&1\\ -2&0&2\\ -1&0&1 \end{array}} \Rightarrow \boxed{\begin{array}{ccccc} 1&-1&11&-2&-13\\ 10&-4&8&2&-11\\ 14&-1&-5&6&-9 \end{array}}$$

Image with padding Kernel Output

$P_W = P_H = S_H = S_W = 1$

b) iii) Max Pooling layer reduces the sensitivity to translational variations in input.

c) We have $S_W = S_H = 1 \quad C_1 = 6 = C_2$

$F_W = F_H = 5$

Want $W_2 = W_1 \Rightarrow \quad P_W = P_H = \underline{\underline{2}}$

d) $P_W = 0 \quad S_W = 1$

$$W_2 = \frac{W_1 - F_W + 2P_W}{S_W} + 1 = \frac{512 - F_W}{1} + 1 = 504$$

$\Rightarrow \underline{F_W = 9} \Rightarrow (\text{height}) \times (\text{width}) = \underline{\underline{9 \times 9}}$

e) Using the same formula again.

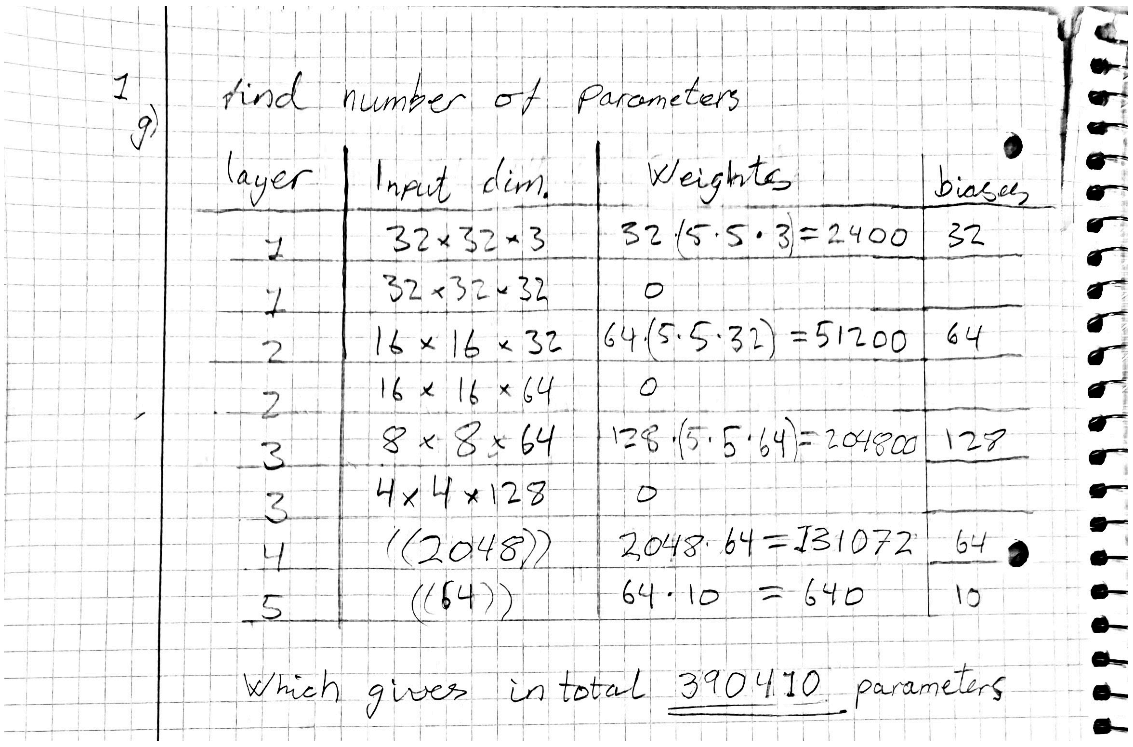
$$W = \frac{W - F_W}{S_W} + 1 = \frac{504 - 2}{2} + 1 = \underline{252}$$

$\Rightarrow (\text{height}) \times (\text{width}) = \underline{\underline{252 \times 252}}$

f)

$$W_2 = \frac{W_1 - F_W + 2P_W}{S_W} + 1 = \frac{252 - 3 + 2 \cdot 0}{1} + 1 = \underline{250}$$

$\Rightarrow (\text{height}) \times (\text{width}) = \underline{\underline{250 \times 250}}$



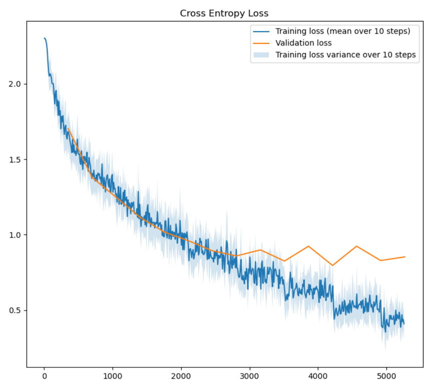
1 g) find number of Parameters

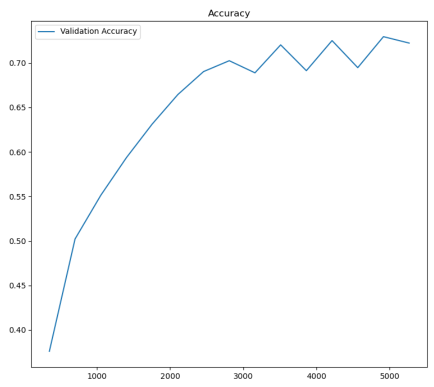
layer	Input dim.	Weights	biases
1	32×32×3	32·(5·5·3)=2400	32
1	32×32×32	0	
2	16×16×32	64·(5·5·32)=51200	64
2	16×16×64	0	
3	8×8×64	128·(5·5·64)=204800	128
3	4×4×128	0	
4	((2048))	2048·64=131072	64
5	((64))	64·10 = 640	10

Which gives in total 390410 parameters

3 Task 2

3.1 2 a)





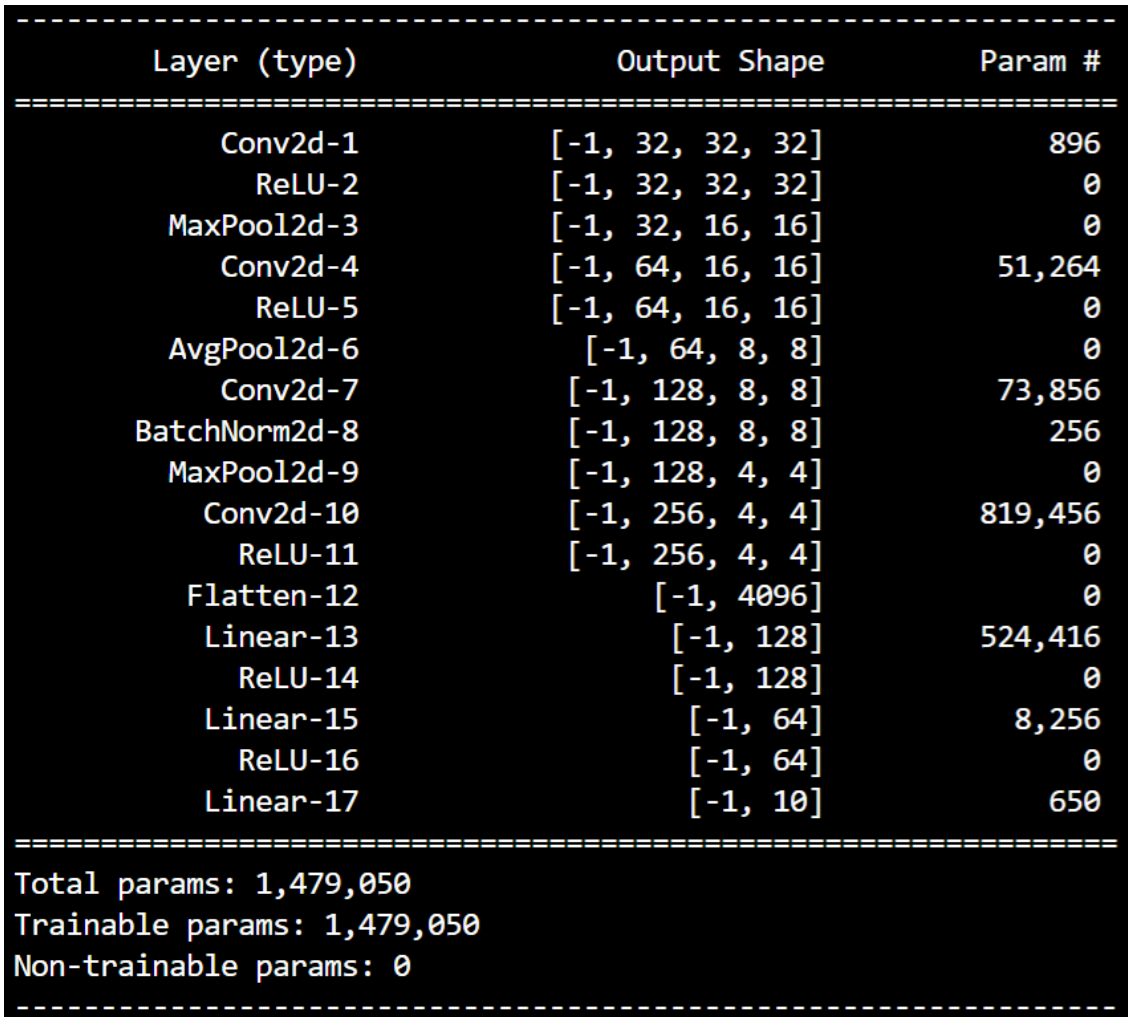
3.2 2 b)

Final validation accuracy 72,21%
Final train accuracy 87,03%
Final test accuracy 73,11%

4 Task 3

4.1 3 a)

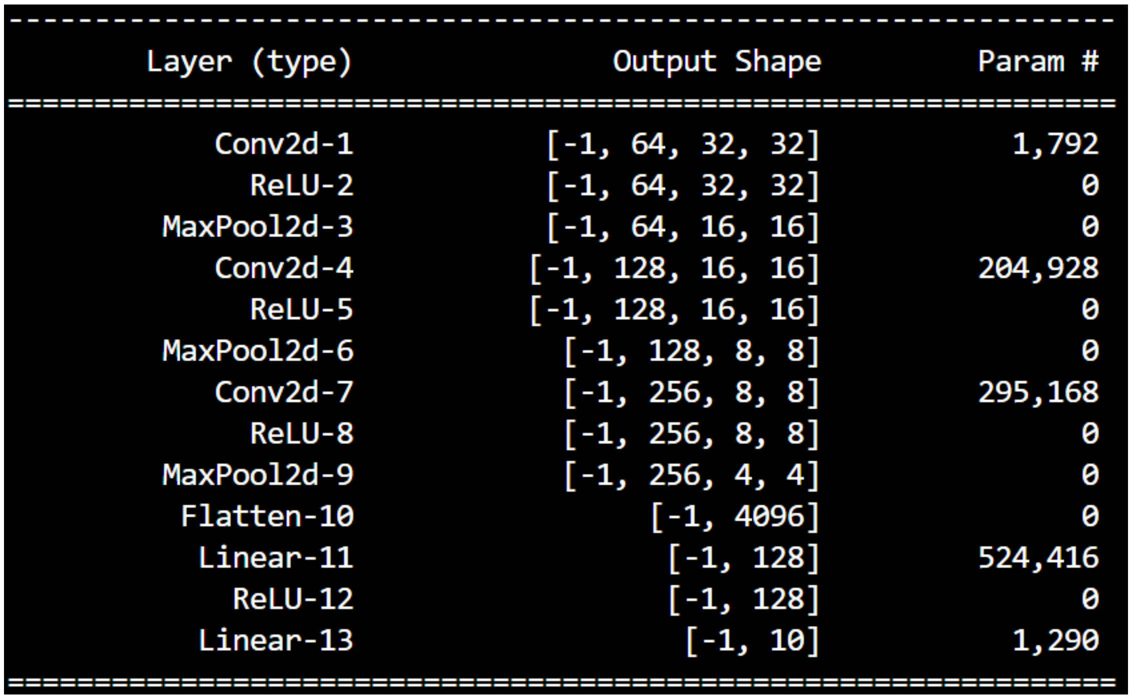
Model 1



```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1           [-1, 32, 32, 32]             896
              ReLU-2           [-1, 32, 32, 32]               0
         MaxPool2d-3           [-1, 32, 16, 16]               0
            Conv2d-4           [-1, 64, 16, 16]          51,264
              ReLU-5           [-1, 64, 16, 16]               0
         AvgPool2d-6             [-1, 64, 8, 8]               0
            Conv2d-7            [-1, 128, 8, 8]          73,856
       BatchNorm2d-8            [-1, 128, 8, 8]             256
         MaxPool2d-9            [-1, 128, 4, 4]               0
           Conv2d-10            [-1, 256, 4, 4]         819,456
             ReLU-11            [-1, 256, 4, 4]               0
          Flatten-12                 [-1, 4096]               0
           Linear-13                  [-1, 128]         524,416
             ReLU-14                  [-1, 128]               0
           Linear-15                   [-1, 64]           8,256
             ReLU-16                   [-1, 64]               0
           Linear-17                   [-1, 10]             650
================================================================
Total params: 1,479,050
Trainable params: 1,479,050
Non-trainable params: 0
----------------------------------------------------------------
```

Batch size: 50, Learning rate: 0.02
Optimizer: Average SGD, weight decay = 0.001, Weight init: Xavier uniform

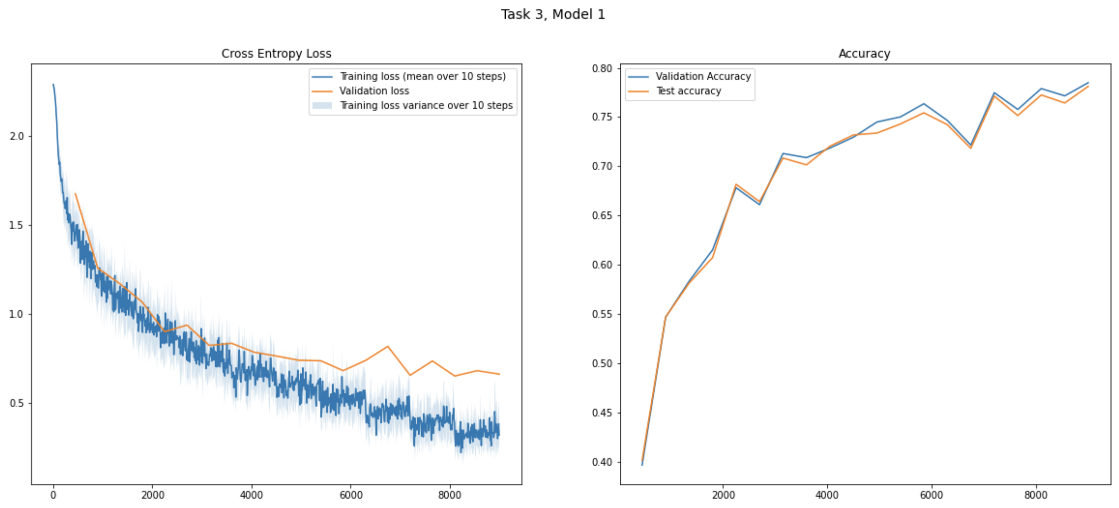
Model 2

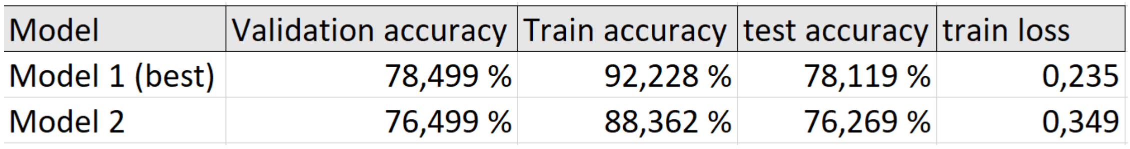


```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1           [-1, 64, 32, 32]           1,792
              ReLU-2           [-1, 64, 32, 32]               0
         MaxPool2d-3           [-1, 64, 16, 16]               0
            Conv2d-4          [-1, 128, 16, 16]         204,928
              ReLU-5          [-1, 128, 16, 16]               0
         MaxPool2d-6            [-1, 128, 8, 8]               0
            Conv2d-7            [-1, 256, 8, 8]         295,168
              ReLU-8            [-1, 256, 8, 8]               0
         MaxPool2d-9            [-1, 256, 4, 4]               0
          Flatten-10                 [-1, 4096]               0
           Linear-11                  [-1, 128]         524,416
             ReLU-12                  [-1, 128]               0
           Linear-13                   [-1, 10]           1,290
================================================================
```

Batch size: 64, Learning rate: 0.05
Optimizer: Average SGD, weight decay = 0.001

4.2 3 b)





Model	Validation accuracy	Train accuracy	test accuracy	train loss
Model 1 (best)	78,499 %	92,228 %	78,119 %	0,235
Model 2	76,499 %	88,362 %	76,269 %	0,349

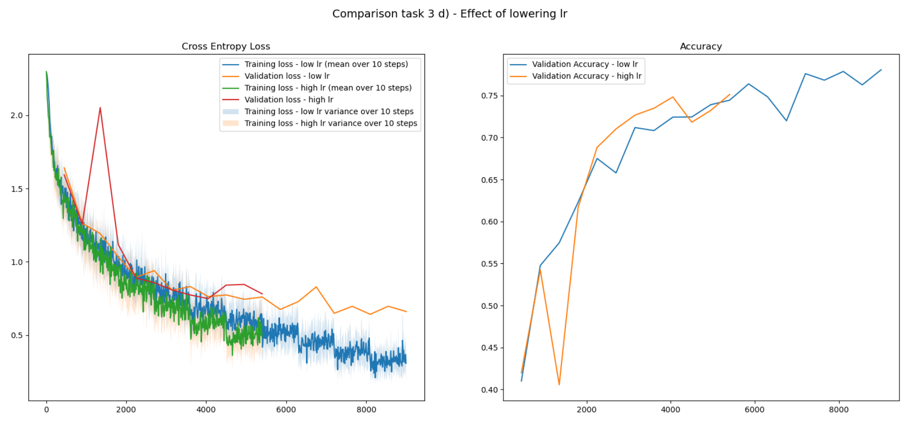
4.3 3 c)

We saw a big impact from adjust the learning rate. A too high learning rate cause very fast training but bad accuracy and high loss. It seemed also like the network overfitted more with a high learning rate. Lowering the learning rate made the network perform better in terms of both accuracy and loss, but it then used more epochs to converge.

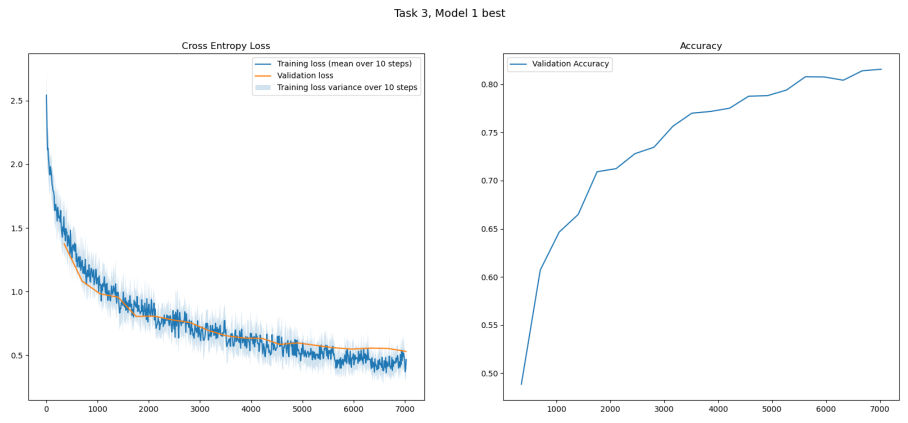
We tried many things which did not work that well. We tried using adaptive average pooling, using batch normalization on all layers, data augmentation with random brightness, and strided convolutions instead of pooling layers. We also tried using the Adam optimizer, with a weight decay of 0, but then the accuracy dropped drasticly.

We also tried using a lot of dropout with different probability, and from a test with dropout on nearly every convolution iteration we observed a significant drop in test accuracy. This may be due to the fact that it essentially removes too many nodes from the network so that it in the end won't have "learned" as much as it should. It is also worth mentioning that with a lot of dropout we saw a lower effect of overfitting. This was to be expected, as the main task for dropout is to reduce overfitting.

4.4 3 d)



4.5 3 e)



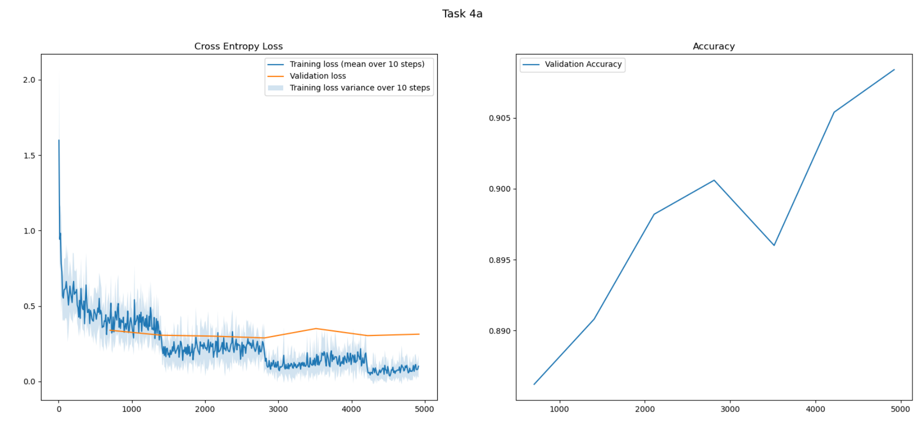
Final validation accuracy 81,559%
Final train accuracy 90,460%
Final test accuracy 81,299%
Final train loss 0,5346

4.6 3 f)

We see that the train accuracy is a lot higher than the test accuracy, 90.4% versus 81.3%. That is a sign of some overfitting as the network performs better on the training dataset than the “unseen” test dataset.

5 Task 4

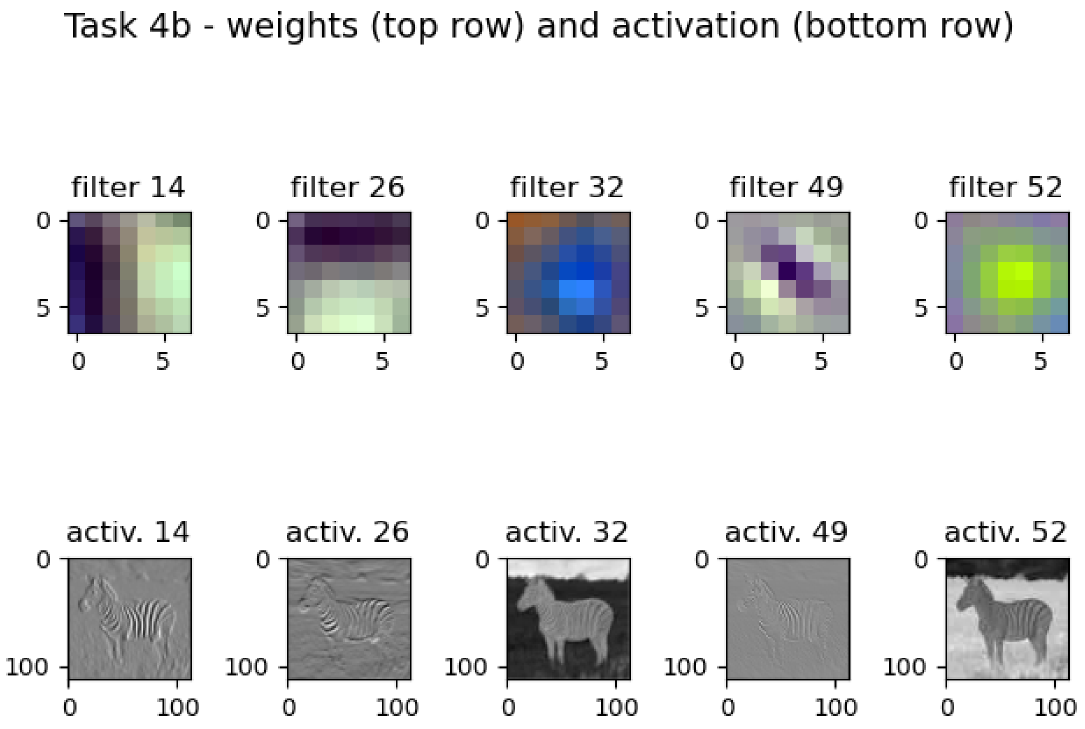
5.1 4 a)



We used the Adam optimizer Hyperparameters: batch_size = 32
learning_rate = 0.0005

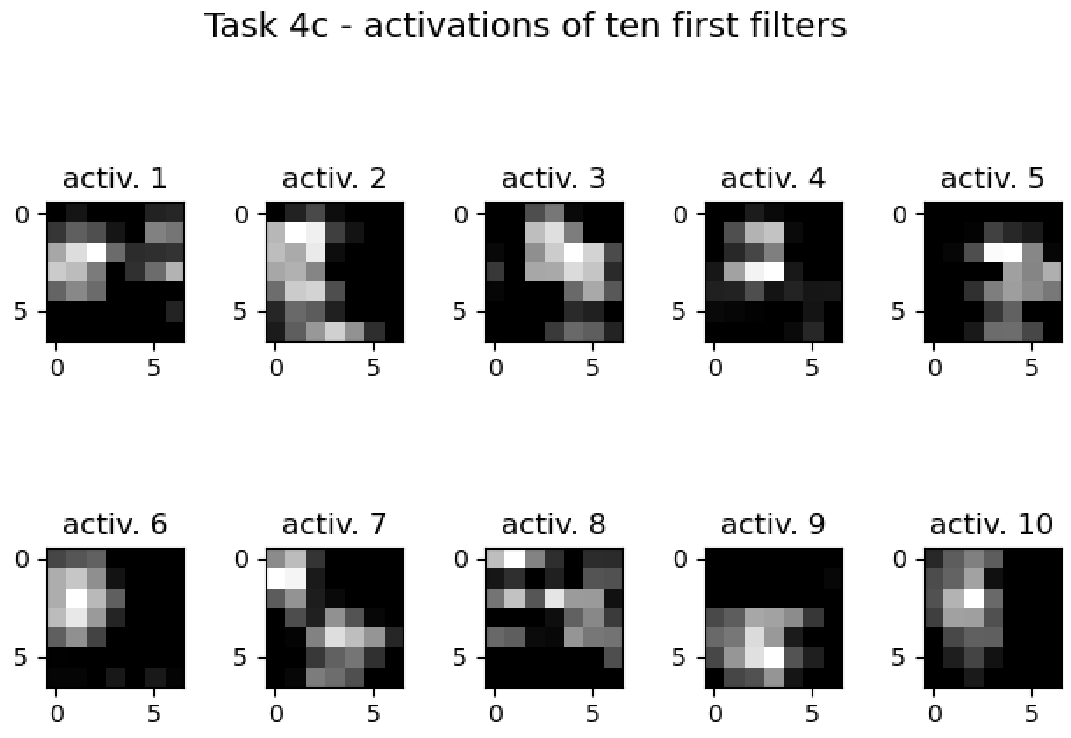
Final test accuracy: 89,96%

5.2 4 b)



We see that filter 14 looks for vertical lines, filter 26 for horizontal lines and filter 49 looks for diagonal lines in the image. Filter 32 and 52 looks like it tries to get the contour of the zebra by separating it from the background.

5.3 4 c)



At the last layer, the activations are very small only 7x7 pixels. So we can see that the features the filters extract very specific things in the image, that seems very random.