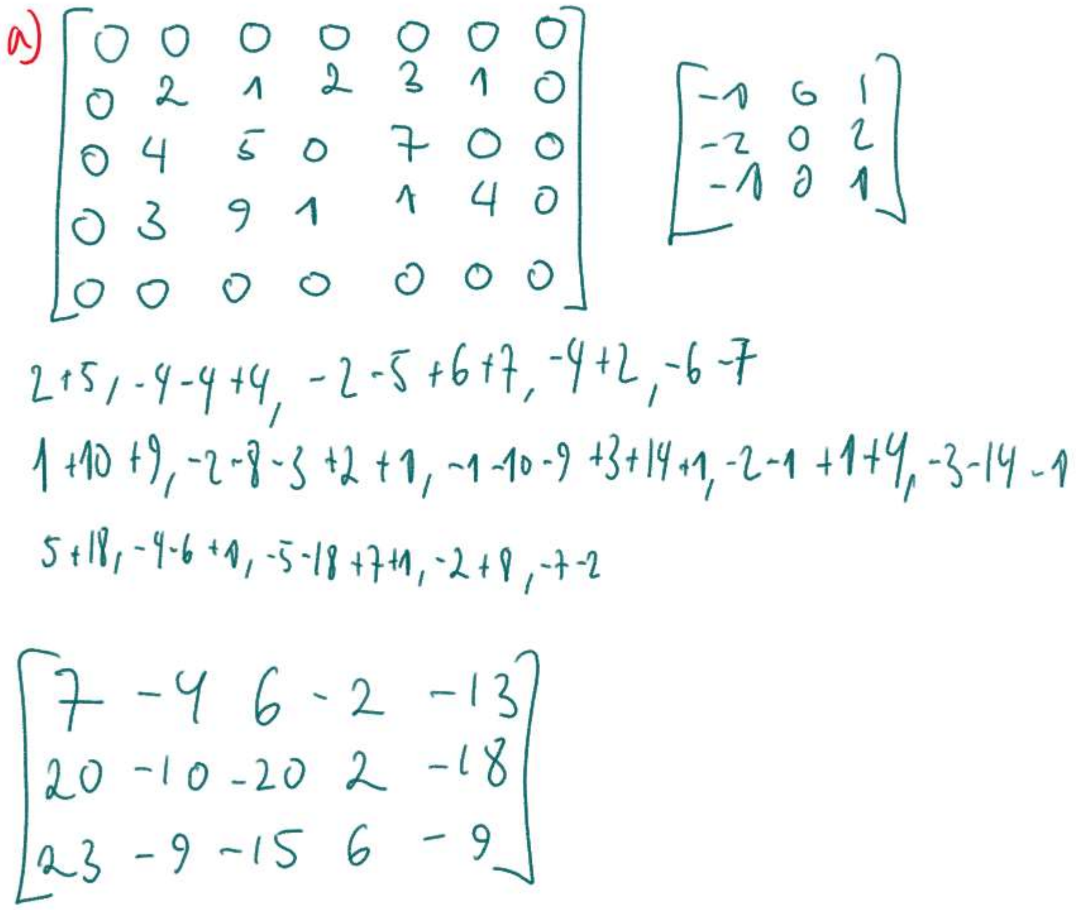
Assignment 3 Group 131 Report

Task 1

task 1a)



a)

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 1 & 2 & 3 & 1 & 0 \\ 0 & 4 & 5 & 0 & 7 & 0 & 0 \\ 0 & 3 & 9 & 1 & 1 & 4 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad \begin{bmatrix} -1 & 6 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

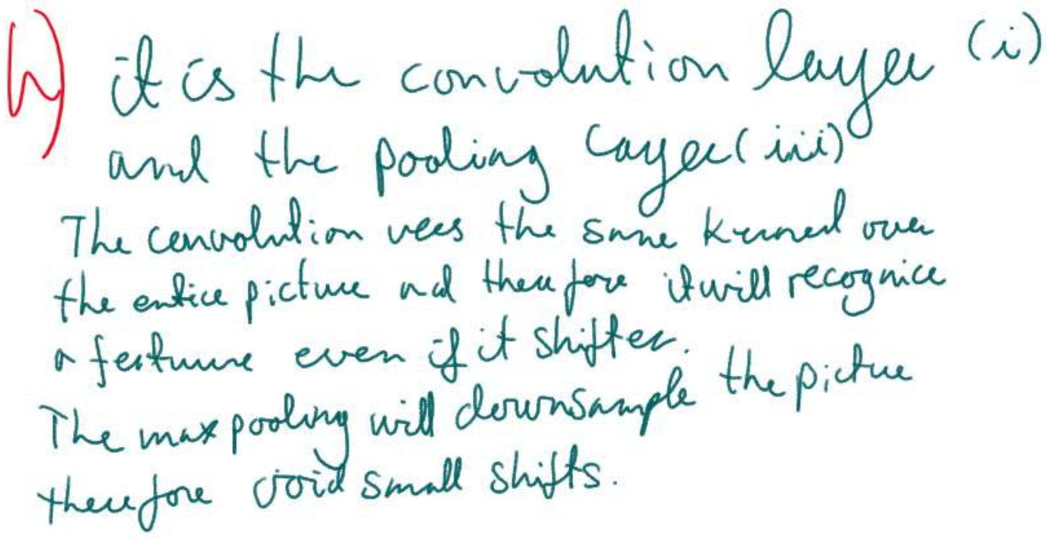
$2+5, -4-4+4, -2-5+6+7, -4+2, -6-7$

$1+10+9, -2-8-3+2+1, -1-10-9+3+14+1, -2-1+1+4, -3-14-1$

$5+18, -4-6+1, -5-18+7+1, -2+8, -7-2$

$$\begin{bmatrix} 7 & -4 & 6 & -2 & -13 \\ 20 & -10 & -20 & 2 & -18 \\ 23 & -9 & -15 & 6 & -9 \end{bmatrix}$$

task 1b)

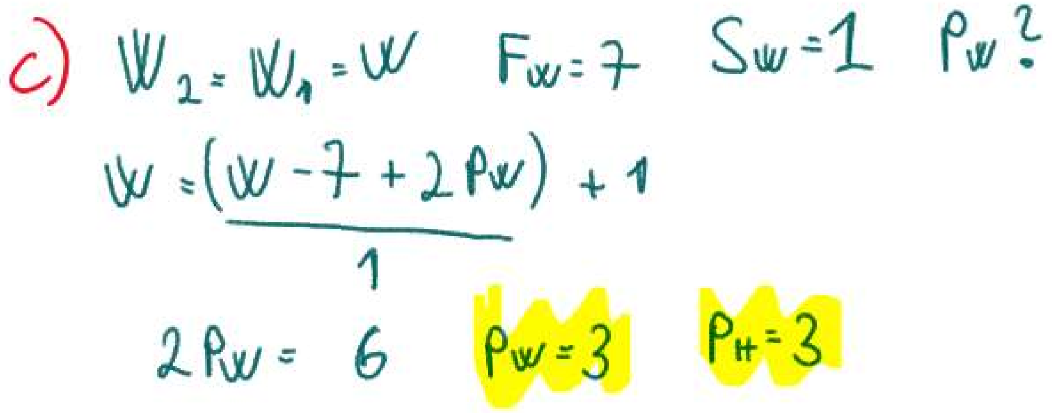


b) it is the convolution layer (i) and the pooling Cayer (iii)

The convolution uses the same kernel over the entire picture and therefore it will recognice a feature even if it shifted.

The max pooling will downsample the picture therefore void small shifts.

task 1c)

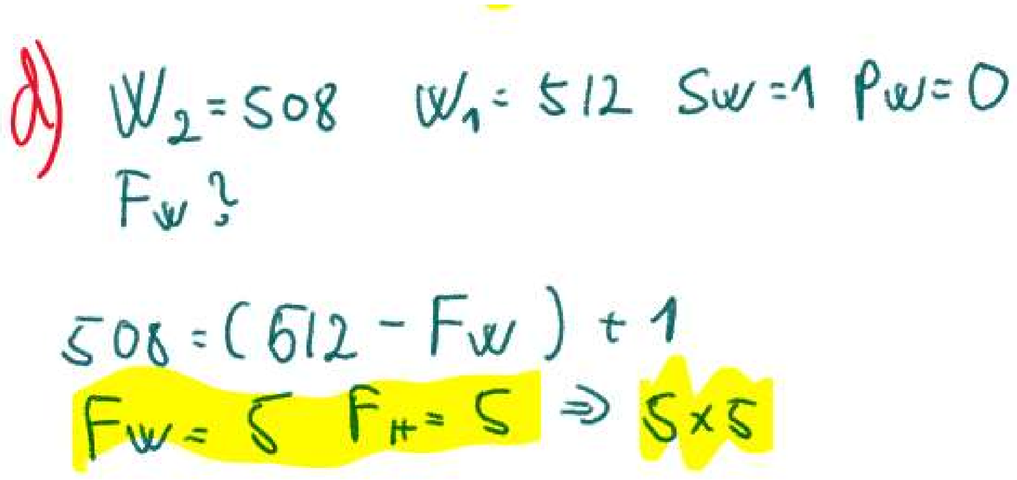


c) $W_2 = W_1 = W$ $F_W = 7$ $S_W = 1$ P_W?

$$W = \frac{(W - 7 + 2P_W)}{1} + 1$$

$2P_W = 6$ $P_W = 3$ $P_H = 3$

task 1d)



d) $W_2 = 508$ $W_1 = 512$ $S_W = 1$ $P_W = 0$

F_W?

$508 = (512 - F_W) + 1$

$F_W = 5$ $F_H = 5 \Rightarrow 5 \times 5$

task 1e)

e) $\frac{508}{2} = 254$ 254×254

task 1f)

f) $W_2 = \frac{(254-3)}{1} + 1 = 252$

252×252

task 1g)

g) ex: $F_W \times F_H \times C_{in} \times C_{out} + C_{out}$

CNN

$$5 \times 5 \times 3 \times 32 + 32 = 2432$$

$$+ \; 5 \times 5 \times 32 \times 64 + 64 = 51264$$

$$5 \times 5 \times 64 \times 128 + 128 = 204928$$

$$= \boxed{258624} \text{ CNN Part.}$$

fully Connected

$$32 \rightarrow 16 \rightarrow 8 \rightarrow 4$$

$$4 \times 4 \times 128 = 2048$$

2048 64 10

$$2048 \cdot 64 + 64 \cdot 10 + 64 + 10$$

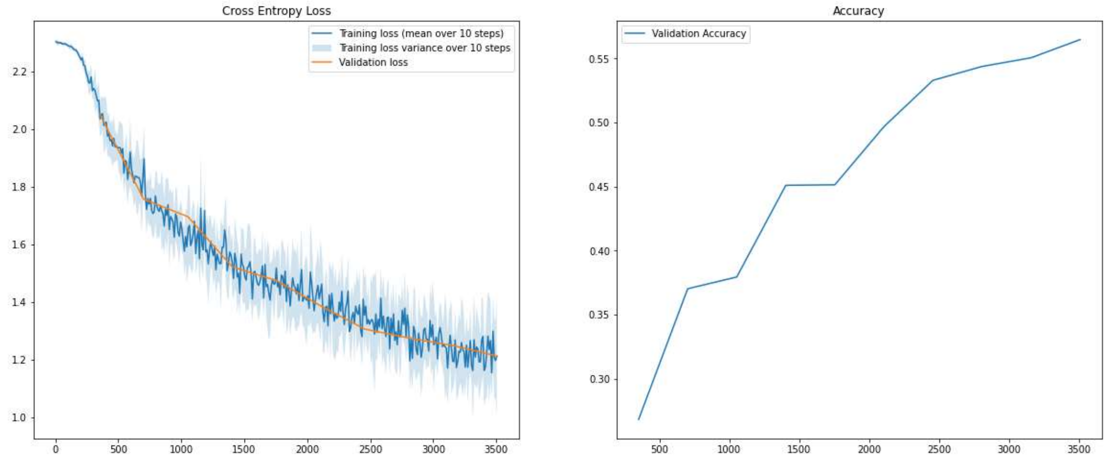
$$= \boxed{131786}$$

$$\text{Total} = 258624 + 131786 = \boxed{390410}$$

Task 2

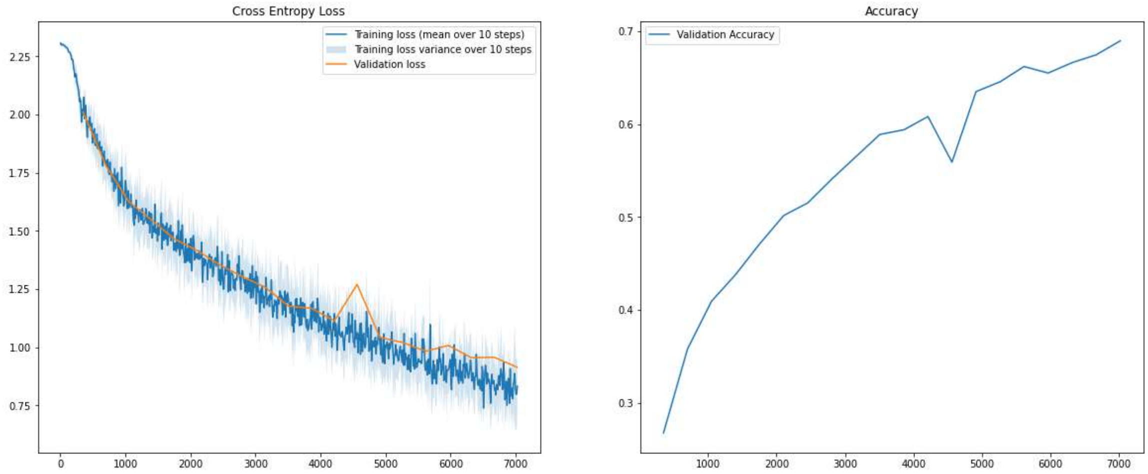
Task 2a)

Over 4 Epochs



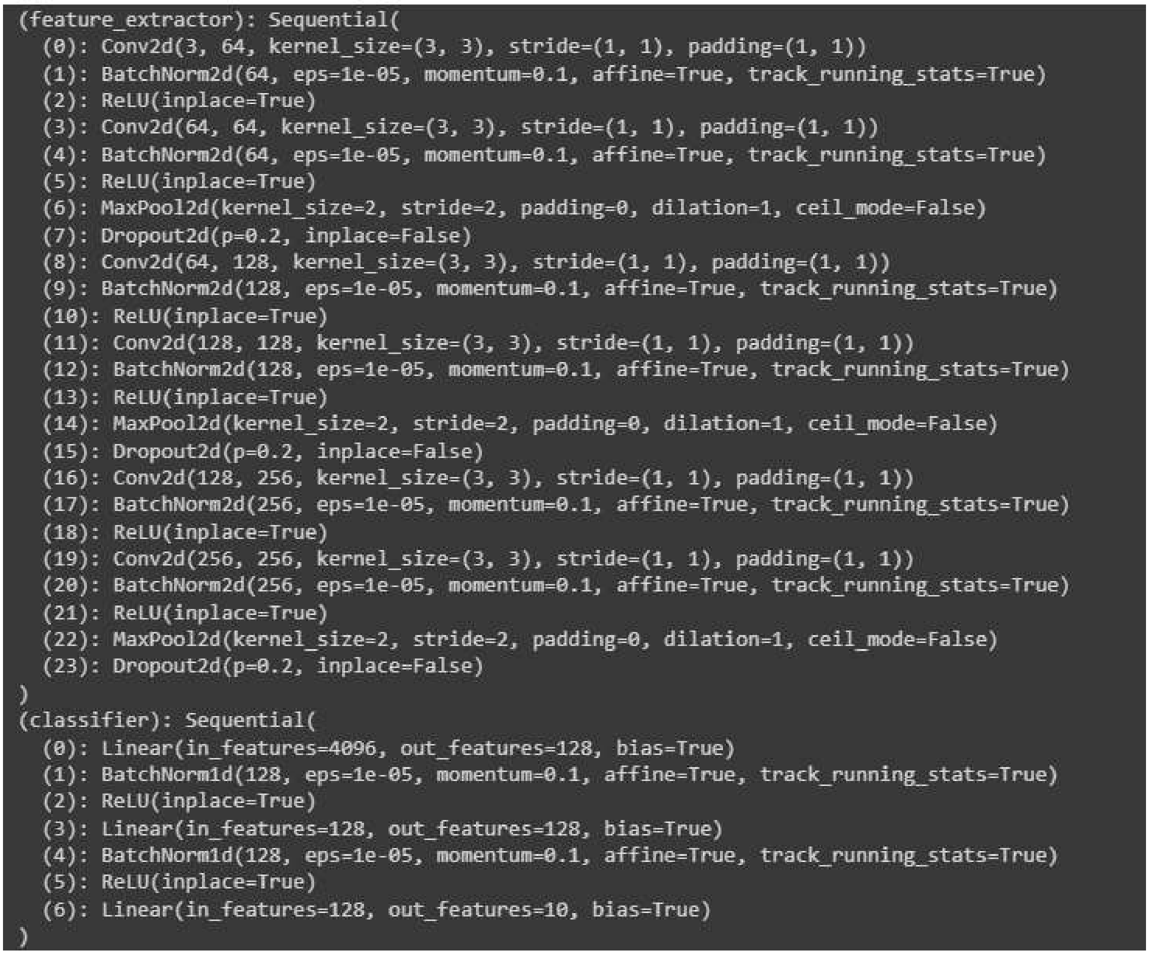
Task 2b)

Over 10 Epochs



Task 3

Task 3a)



```
(feature_extractor): Sequential(
  (0): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (2): ReLU(inplace=True)
  (3): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (4): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (5): ReLU(inplace=True)
  (6): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (7): Dropout2d(p=0.2, inplace=False)
  (8): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (9): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (10): ReLU(inplace=True)
  (11): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (12): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (13): ReLU(inplace=True)
  (14): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (15): Dropout2d(p=0.2, inplace=False)
  (16): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (17): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (18): ReLU(inplace=True)
  (19): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (20): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (21): ReLU(inplace=True)
  (22): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (23): Dropout2d(p=0.2, inplace=False)
)
(classifier): Sequential(
  (0): Linear(in_features=4096, out_features=128, bias=True)
  (1): BatchNorm1d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (2): ReLU(inplace=True)
  (3): Linear(in_features=128, out_features=128, bias=True)
  (4): BatchNorm1d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (5): ReLU(inplace=True)
  (6): Linear(in_features=128, out_features=10, bias=True)
)
```

Optimzer: Adam

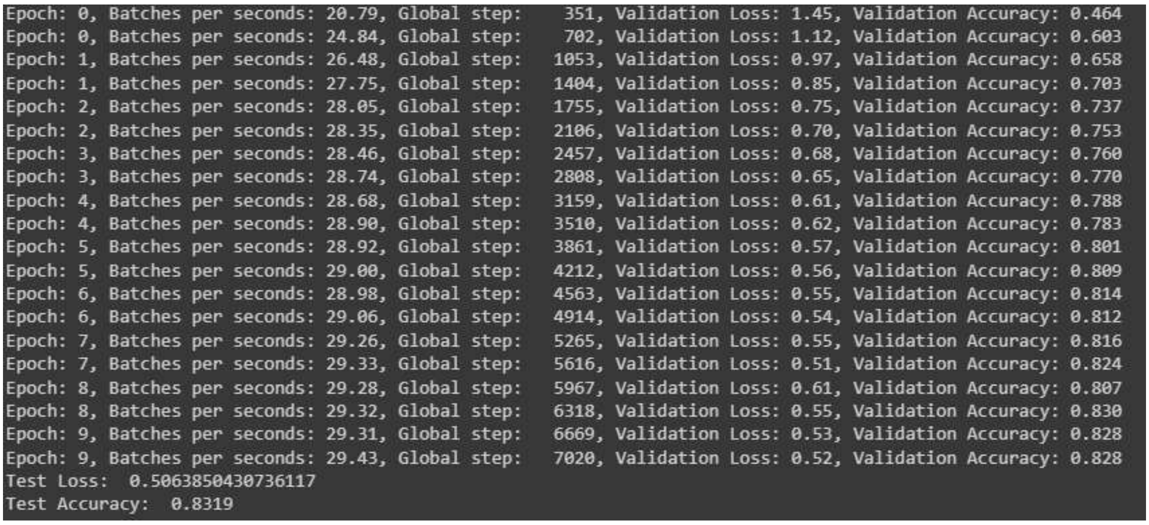
Learning rate: 0.02

Max Epoch: 10

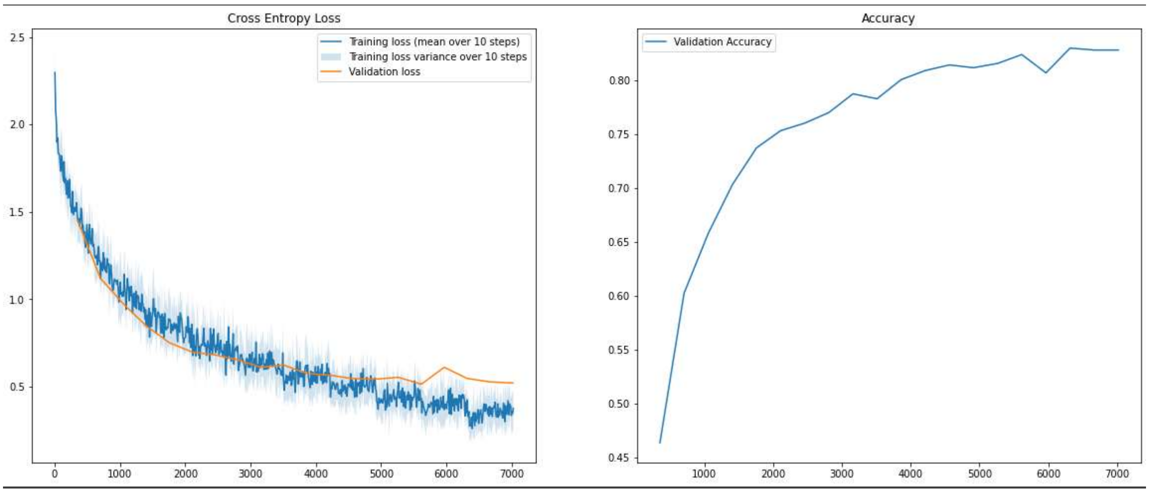
Batch size: 64

Wheight init: standard

Task 3b)



```
Epoch: 0, Batches per seconds: 20.79, Global step:    351, Validation Loss: 1.45, Validation Accuracy: 0.464
Epoch: 0, Batches per seconds: 24.84, Global step:    702, Validation Loss: 1.12, Validation Accuracy: 0.603
Epoch: 1, Batches per seconds: 26.48, Global step:   1053, Validation Loss: 0.97, Validation Accuracy: 0.658
Epoch: 1, Batches per seconds: 27.75, Global step:   1404, Validation Loss: 0.85, Validation Accuracy: 0.703
Epoch: 2, Batches per seconds: 28.05, Global step:   1755, Validation Loss: 0.75, Validation Accuracy: 0.737
Epoch: 2, Batches per seconds: 28.35, Global step:   2106, Validation Loss: 0.70, Validation Accuracy: 0.753
Epoch: 3, Batches per seconds: 28.46, Global step:   2457, Validation Loss: 0.68, Validation Accuracy: 0.760
Epoch: 3, Batches per seconds: 28.74, Global step:   2808, Validation Loss: 0.65, Validation Accuracy: 0.770
Epoch: 4, Batches per seconds: 28.68, Global step:   3159, Validation Loss: 0.61, Validation Accuracy: 0.788
Epoch: 4, Batches per seconds: 28.90, Global step:   3510, Validation Loss: 0.62, Validation Accuracy: 0.783
Epoch: 5, Batches per seconds: 28.92, Global step:   3861, Validation Loss: 0.57, Validation Accuracy: 0.801
Epoch: 5, Batches per seconds: 29.00, Global step:   4212, Validation Loss: 0.56, Validation Accuracy: 0.809
Epoch: 6, Batches per seconds: 28.98, Global step:   4563, Validation Loss: 0.55, Validation Accuracy: 0.814
Epoch: 6, Batches per seconds: 29.06, Global step:   4914, Validation Loss: 0.54, Validation Accuracy: 0.812
Epoch: 7, Batches per seconds: 29.26, Global step:   5265, Validation Loss: 0.55, Validation Accuracy: 0.816
Epoch: 7, Batches per seconds: 29.33, Global step:   5616, Validation Loss: 0.51, Validation Accuracy: 0.824
Epoch: 8, Batches per seconds: 29.28, Global step:   5967, Validation Loss: 0.61, Validation Accuracy: 0.807
Epoch: 8, Batches per seconds: 29.32, Global step:   6318, Validation Loss: 0.55, Validation Accuracy: 0.830
Epoch: 9, Batches per seconds: 29.31, Global step:   6669, Validation Loss: 0.53, Validation Accuracy: 0.828
Epoch: 9, Batches per seconds: 29.43, Global step:   7020, Validation Loss: 0.52, Validation Accuracy: 0.828
Test Loss:  0.5063850430736117
Test Accuracy:  0.8319
```



Task 3c)

Expanding the network after the VGGnet helped a lot. more convolutions with a smaller kernel and zero padding to keep the size. The deeper network gave better accuracy per epoch.

Switching to Adam on the deeper network gave a lot better learning rate.

Datch normalization also gave better learning time.

Did not see improvement with ther activation functions.

Image normalization was kept, other Data augmentation slowed down the learning rate but could give better generaization in the long run.

Added a hidden layer and more nodes in both layers in the fully connected network, gave some better accuracy in trade of training time.

Used maxpooling 2x2 step 2, improves performance of the network

Task 3d)

The best improvement was seen with the switch to Adam (could not get double graph to work)

Task 3e)

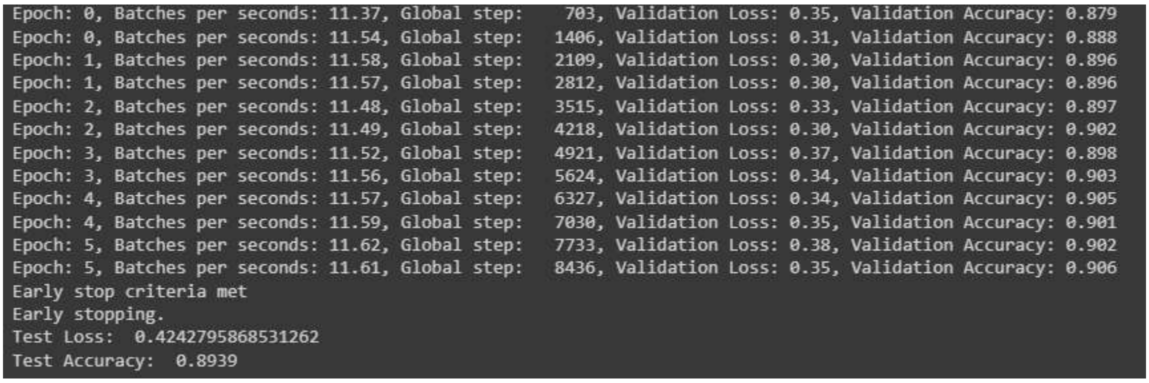
See image in 3b)

Task 3f)

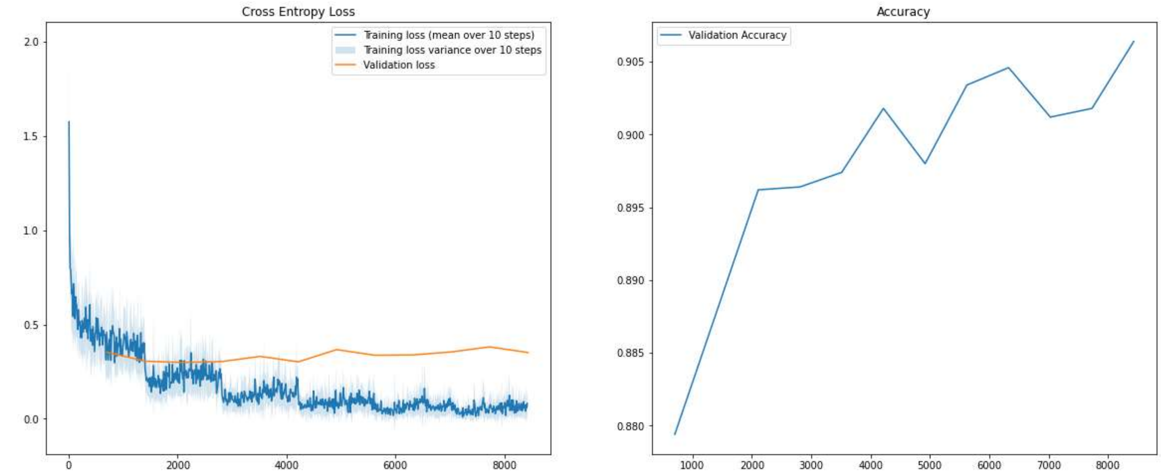
Test accuracy and loss was better that validation. See some overfitting at the end of training, could be better with more data augmentarion to make the training set more diverse

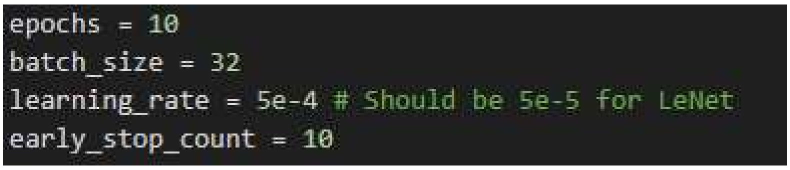
Task 4

Task 4a)



```
Epoch: 0, Batches per seconds: 11.37, Global step:    703, Validation Loss: 0.35, Validation Accuracy: 0.879
Epoch: 0, Batches per seconds: 11.54, Global step:   1406, Validation Loss: 0.31, Validation Accuracy: 0.888
Epoch: 1, Batches per seconds: 11.58, Global step:   2109, Validation Loss: 0.30, Validation Accuracy: 0.896
Epoch: 1, Batches per seconds: 11.57, Global step:   2812, Validation Loss: 0.30, Validation Accuracy: 0.896
Epoch: 2, Batches per seconds: 11.48, Global step:   3515, Validation Loss: 0.33, Validation Accuracy: 0.897
Epoch: 2, Batches per seconds: 11.49, Global step:   4218, Validation Loss: 0.30, Validation Accuracy: 0.902
Epoch: 3, Batches per seconds: 11.52, Global step:   4921, Validation Loss: 0.37, Validation Accuracy: 0.898
Epoch: 3, Batches per seconds: 11.56, Global step:   5624, Validation Loss: 0.34, Validation Accuracy: 0.903
Epoch: 4, Batches per seconds: 11.57, Global step:   6327, Validation Loss: 0.34, Validation Accuracy: 0.905
Epoch: 4, Batches per seconds: 11.59, Global step:   7030, Validation Loss: 0.35, Validation Accuracy: 0.901
Epoch: 5, Batches per seconds: 11.62, Global step:   7733, Validation Loss: 0.38, Validation Accuracy: 0.902
Epoch: 5, Batches per seconds: 11.61, Global step:   8436, Validation Loss: 0.35, Validation Accuracy: 0.906
Early stop criteria met
Early stopping.
Test Loss:  0.4242795868531262
Test Accuracy:  0.8939
```





```
epochs = 10
batch_size = 32
learning_rate = 5e-4 # Should be 5e-5 for LeNet
early_stop_count = 10
```