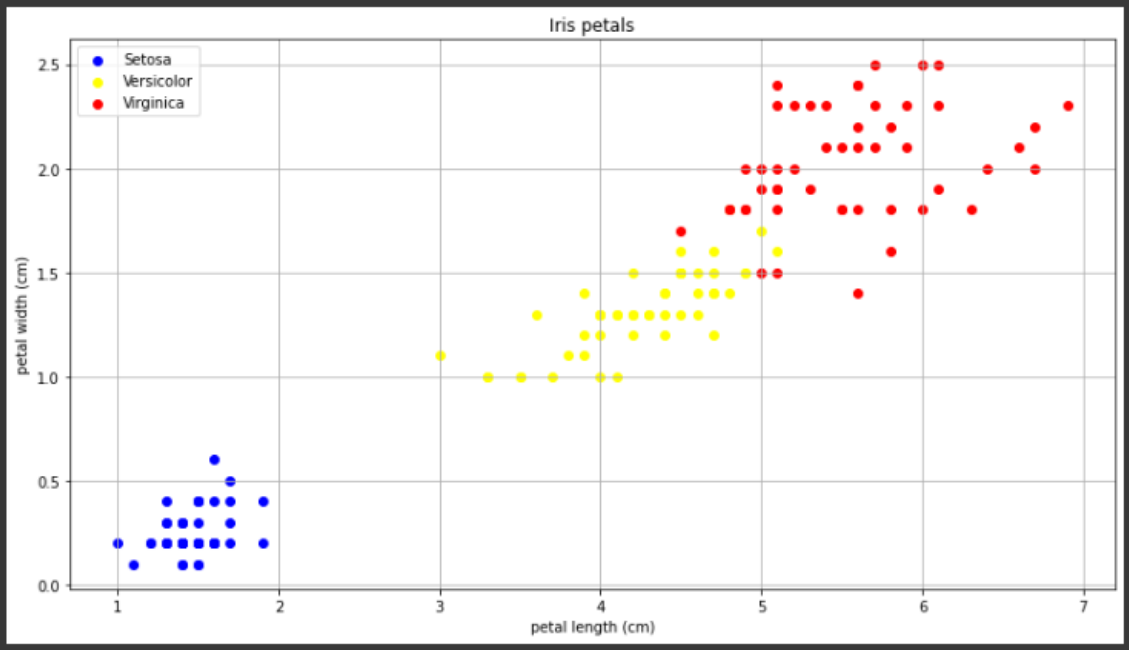
# Iris data

| | sepal length (cm) | sepal width (cm) | petal length (cm) | petal width (cm) | target |
|---|---|---|---|---|---|
| count | 150.000000 | 150.000000 | 150.000000 | 150.000000 | 150.000000 |
| mean | 5.843333 | 3.057333 | 3.758000 | 1.199333 | 1.000000 |
| std | 0.828066 | 0.435866 | 1.765298 | 0.762238 | 0.819232 |
| min | 4.300000 | 2.000000 | 1.000000 | 0.100000 | 0.000000 |
| 25% | 5.100000 | 2.800000 | 1.600000 | 0.300000 | 0.000000 |
| 50% | 5.800000 | 3.000000 | 4.350000 | 1.300000 | 1.000000 |
| 75% | 6.400000 | 3.300000 | 5.100000 | 1.800000 | 2.000000 |
| max | 7.900000 | 4.400000 | 6.900000 | 2.500000 | 2.000000 |




remuving unuseful columns

spliting to train data 80% and test data 20% with random selection

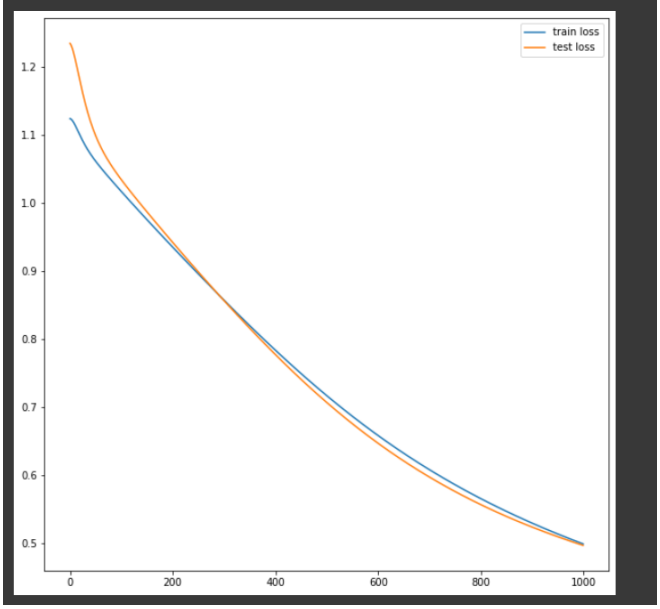
scaling data with 0 mean remaning the standar diviation

# Στατιστικα

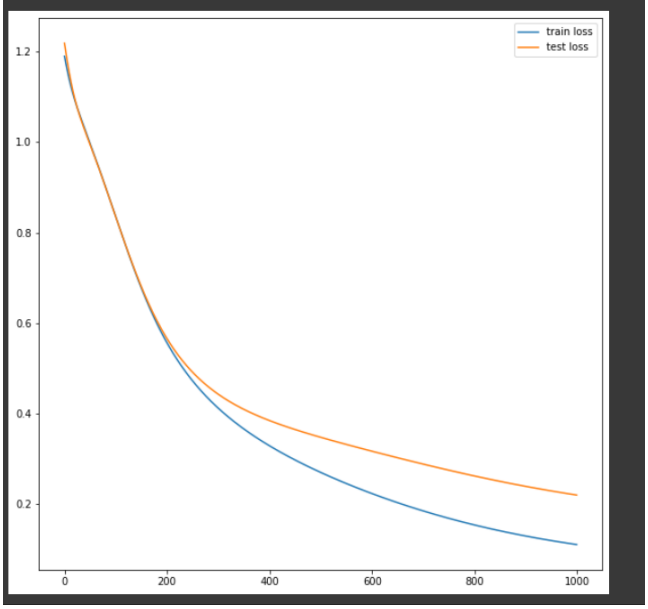
## NN 4:30:3 with Sigmoid activation

```
NN_Sigmoid(
  (linear_stack): Sequential(
    (0): Linear(in_features=4, out_features=30, bias=True)
    (1): Sigmoid()
    (2): Linear(in_features=30, out_features=3, bias=True)
  )
)
Number of learnable parameters' sets:  4
torch.Size([30, 4])
torch.Size([30])
torch.Size([3, 30])
torch.Size([3])
```

```
Epoch 1000/1000, Train Loss: 0.4989, Test Loss: 0.4967
Accuracy: 83.3%

Weights data have been saved in best_weights_SGD.pth
```




```
Epoch 1000/1000, Train Loss: 0.1113, Test Loss: 0.2206
Accuracy: 86.7%

Weights data have been saved in best_weights_Adam.pth
```

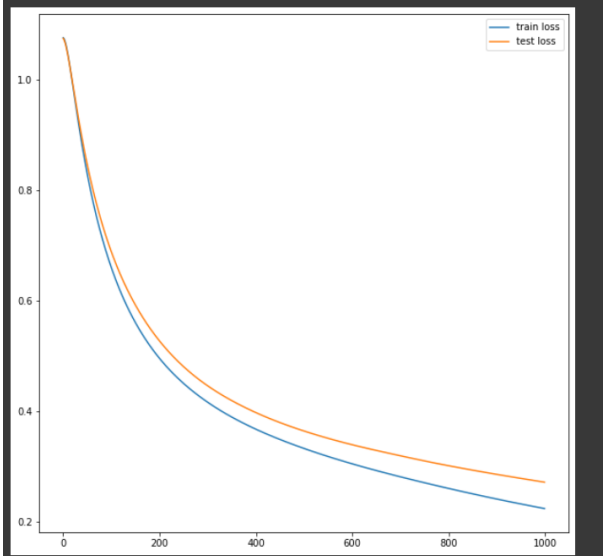



```
Precision, Recall, Confusion matrix, in testing

              precision    recall  f1-score   support

           0      1.000     1.000     1.000        11
           1      0.667     1.000     0.800         8
           2      1.000     0.636     0.778        11

    accuracy                          0.867        30
   macro avg      0.889     0.879     0.859        30
weighted avg      0.911     0.867     0.865        30

[[11  0  0]
 [ 0  8  0]
 [ 0  4  7]]
```

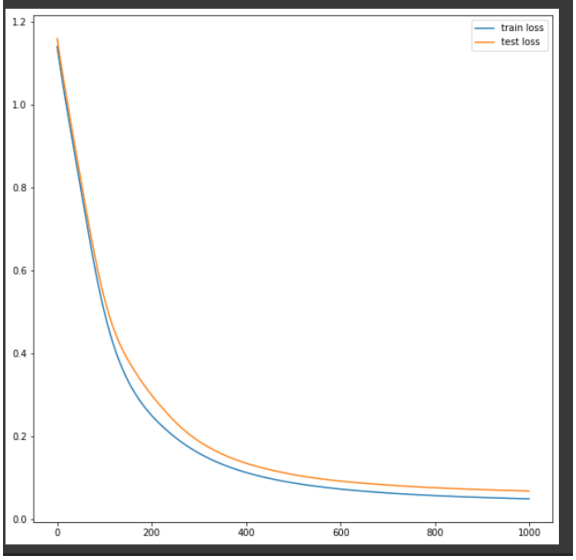
## NN with 4:30:3 with Relu activation

```
NN_ReLU(
  (flatten): Flatten(start_dim=1, end_dim=-1)
  (linear_stack): Sequential(
    (0): Linear(in_features=4, out_features=30, bias=True)
    (1): ReLU()
    (2): Linear(in_features=30, out_features=3, bias=True)
  )
)
Number of learnable parameters' sets:  4
torch.Size([30, 4])
torch.Size([30])
torch.Size([3, 30])
torch.Size([3])
```

```
Epoch 1000/1000, Train Loss: 0.2235, Test Loss: 0.2712
Accuracy: 93.3%

Weights data have been saved in best_weights_SGD.pth
```




```
Epoch 1000/1000, Train Loss: 0.0486, Test Loss: 0.0674
Accuracy: 96.7%

Weights data have been saved in best_weights_Adam.pth
```



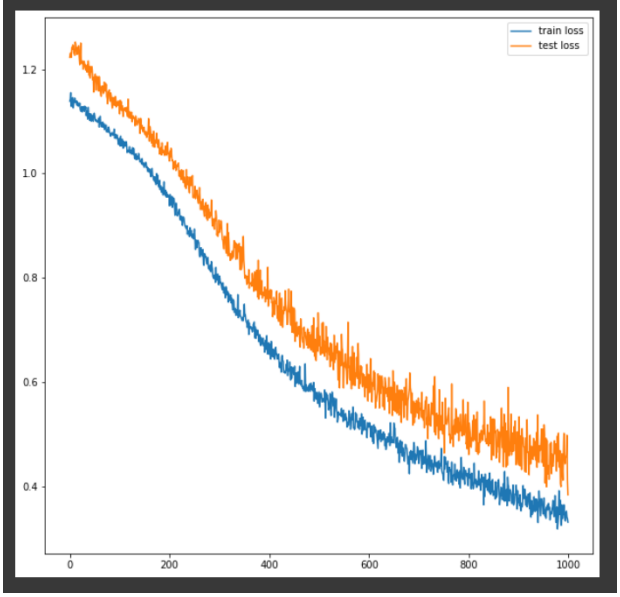

Confusion map and statistics for best outcome

```
Precision, Recall, Confusion matrix, in testing

              precision    recall  f1-score   support

           0      1.000     1.000     1.000         5
           1      0.929     1.000     0.963        13
           2      1.000     0.917     0.957        12

    accuracy                          0.967        30
   macro avg      0.976     0.972     0.973        30
weighted avg      0.969     0.967     0.967        30

[[ 5  0  0]
 [ 0 13  0]
 [ 0  1 11]]
```

## My NN 4:20:10:3 with Relu

```
My_NN_Relu(
  (linear_stack): Sequential(
    (0): Linear(in_features=4, out_features=20, bias=True)
    (1): ReLU()
    (2): Linear(in_features=20, out_features=10, bias=True)
    (3): ReLU()
    (4): Linear(in_features=10, out_features=3, bias=True)
  )
  (dropout): Dropout(p=0.1, inplace=False)
)
Number of learnable parameters' sets:  6
torch.Size([20, 4])
torch.Size([20])
torch.Size([10, 20])
torch.Size([10])
torch.Size([3, 10])
torch.Size([3])
```

```
Weights data have been saved in best_weights_SGD.pth
```

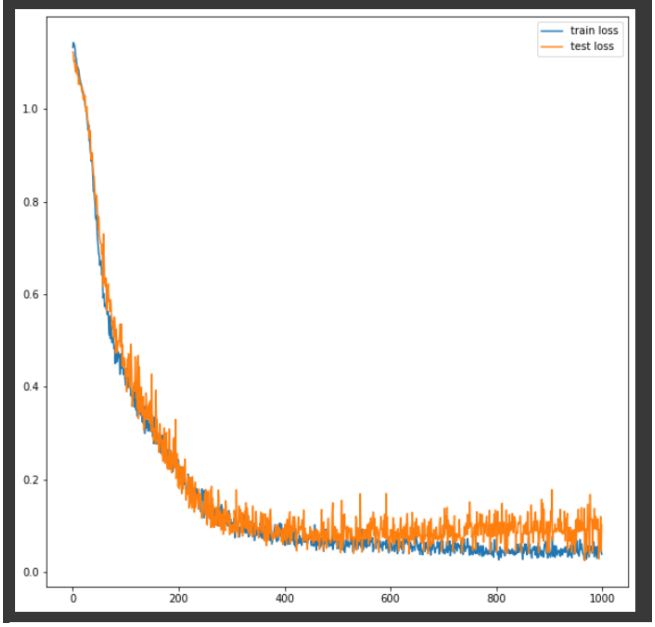



```
Precision, Recall, Confusion matrix, in training

              precision    recall  f1-score   support

           0      1.000     1.000     1.000        43
           1      0.846     0.805     0.825        41
           2      0.789     0.833     0.811        36

    accuracy                          0.883       120
   macro avg      0.879     0.879     0.879       120
weighted avg      0.884     0.883     0.883       120

[[43  0  0]
 [ 0 33  8]
 [ 0  6 30]]
```

```
Precision, Recall, Confusion matrix, in testing

              precision    recall  f1-score   support

           0      1.000     1.000     1.000         7
           1      0.800     0.889     0.842         9
           2      0.923     0.857     0.889        14

    accuracy                          0.900        30
   macro avg      0.908     0.915     0.910        30
weighted avg      0.904     0.900     0.901        30

[[ 7  0  0]
 [ 0  8  1]
 [ 0  2 12]]
```

## Add Learning rate Scheduler

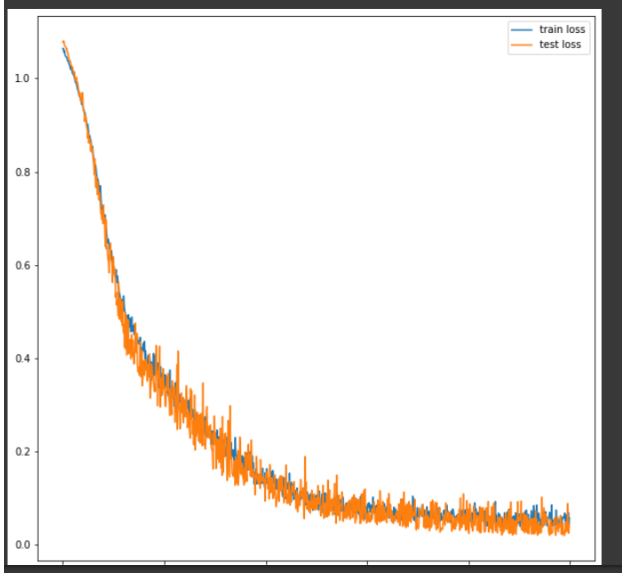
```
Weights data have been saved in best_weights_SGD.pth
```




```
Precision, Recall, Confusion matrix, in training

              precision    recall  f1-score   support

           0      0.933     1.000     0.966        42
           1      1.000     0.949     0.974        39
           2      1.000     0.974     0.987        39

    accuracy                          0.975       120
   macro avg      0.978     0.974     0.975       120
weighted avg      0.977     0.975     0.975       120

[[42  0  0]
 [ 2 37  0]
 [ 1  0 38]]
```

```
Precision, Recall, Confusion matrix, in testing

              precision    recall  f1-score   support

           0      1.000     1.000     1.000         8
           1      0.846     1.000     0.917        11
           2      1.000     0.818     0.900        11

    accuracy                          0.933        30
   macro avg      0.949     0.939     0.939        30
weighted avg      0.944     0.933     0.933        30

[[ 8  0  0]
 [ 0 11  0]
 [ 0  2  9]]
```

```
Epoch 1000/1000, Train Loss: 0.0579, Test Loss: 0.0444
Accuracy: 96.7%

Weights data have been saved in best_weights_Adam.pth
```

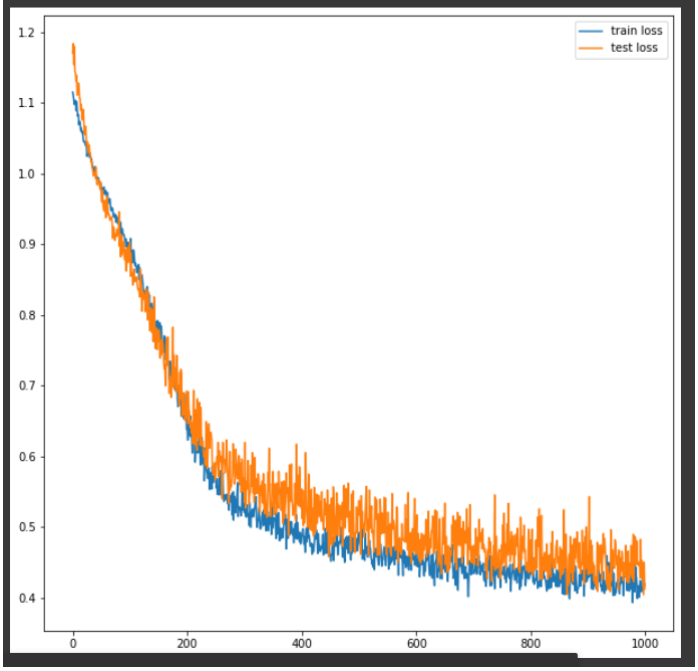



```
Precision, Recall, Confusion matrix, in training

              precision    recall  f1-score   support

           0      0.976     1.000     0.988        41
           1      0.977     0.977     0.977        43
           2      0.971     0.944     0.958        36

    accuracy                          0.975       120
   macro avg      0.975     0.974     0.974       120
weighted avg      0.975     0.975     0.975       120

[[41  0  0]
 [ 0 42  1]
 [ 1  1 34]]
```

```
Precision, Recall, Confusion matrix, in testing

              precision    recall  f1-score   support

           0      1.000     1.000     1.000         9
           1      1.000     1.000     1.000         7
           2      1.000     1.000     1.000        14

    accuracy                          1.000        30
   macro avg      1.000     1.000     1.000        30
weighted avg      1.000     1.000     1.000        30

[[ 9  0  0]
 [ 0  7  0]
 [ 0  0 14]]
```

## Adam with wreigth decay

```
saved   Epoch 1000/1000, Train Loss: 0.4178, Test Loss: 0.4128
Accuracy: 96.7%

Weights data have been saved in best_weights_Adam_2.pth
```




```
Precision, Recall, Confusion matrix, in training

              precision    recall  f1-score   support

           0      0.956     1.000     0.977        43
           1      0.909     0.750     0.822        40
           2      0.786     0.892     0.835        37

    accuracy                          0.883       120
   macro avg      0.883     0.881     0.878       120
weighted avg      0.888     0.883     0.882       120

[[43  0  0]
 [ 1 30  9]
 [ 1  3 33]]
```

```
Precision, Recall, Confusion matrix, in testing

              precision    recall  f1-score   support

           0      1.000     1.000     1.000         7
           1      1.000     0.800     0.889        10
           2      0.867     1.000     0.929        13

    accuracy                          0.933        30
   macro avg      0.956     0.933     0.939        30
weighted avg      0.942     0.933     0.932        30

[[ 7  0  0]
 [ 0  8  2]
 [ 0  0 13]]
```

Scaler :

We indrodused a standar deviation scaler standardize features of the data set by scaling to unit variance and removing the mean using column summary statistics on the samples in the training set .This methode helps with data faraway from center that otherwise would contribute very small amount to the change.

Sigmoid vs Relu :

Because sigmoid compres back propagations adjustments needs more repetions-epoch to reach a optimum solution. If we let epoche be bigger the network woud give a good solution because the data are very small and the network has one hidden layer . Although the data and the networks layout thre is a difference with Relu activation function producing betters results in average .

Std “with momentum :

By introducing a momentun in optimizer we overcome the small “hills” by not following the gradience all the time like a heavy ball

Sceduler :

Scheduler slowly decrease learning rate when the system does not move a lot. With that addition we can “reach deeper” in lower minimums sometimes .if the learning rate is too hight tha step accordingly would be to long and we would not reach the tru minimus.

Std vs Adam :

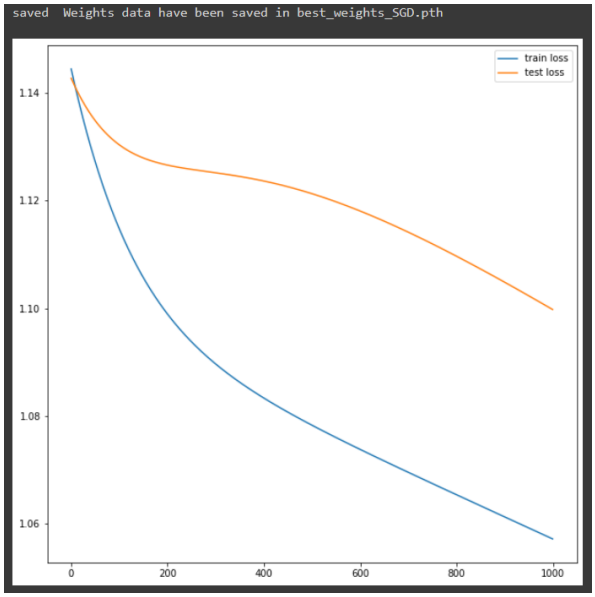
Adam method inroduse two optimisations :

- Adaptive Gradient Algorithm (AdaGrad) that maintains a per-parameter learning rate that improves performance on problems with sparse gradients (e.g. natural language and computer vision problems).
- Root Mean Square Propagation (RMSProp) that also maintains per-parameter learning rates that are adapted based on the average of recent magnitudes of the gradients for the weight (e.g. how quickly it is changing). This means the algorithm does well on online and non-stationary problems (e.g. noisy).

Source: Gentle Introduction to the Adam Optimization Algorithm for Deep Learning - MachineLearningMastery.com

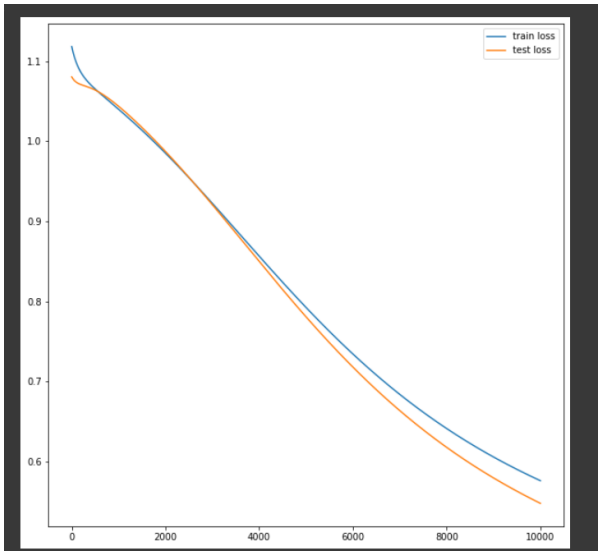
Without all the optimizations on NN Sigmoid 30 :

```
saved  Weights data have been saved in best_weights_SGD.pth
```




```
Precision, Recall, Confusion matrix, in training

              precision    recall  f1-score   support

           0      0.000     0.000     0.000        35
           1      0.000     0.000     0.000        42
           2      0.361     1.000     0.531        43

    accuracy                          0.358       120
   macro avg      0.120     0.333     0.177       120
weighted avg      0.129     0.358     0.190       120

[[ 0  1 34]
 [ 0  0 42]
 [ 0  0 43]]
```

After epoch 10000




```
Precision, Recall, Confusion matrix, in training

              precision    recall  f1-score   support

           0      1.000     1.000     1.000        38
           1      1.000     0.475     0.644        40
           2      0.667     1.000     0.800        42

    accuracy                          0.825       120
   macro avg      0.889     0.825     0.815       120
weighted avg      0.883     0.825     0.811       120

[[38  0  0]
 [ 0 19 21]
 [ 0  0 42]]
```

```
Precision, Recall, Confusion matrix, in testing

              precision    recall  f1-score   support

           0      1.000     1.000     1.000        12
           1      1.000     0.600     0.750        10
           2      0.667     1.000     0.800         8

    accuracy                          0.867        30
   macro avg      0.889     0.867     0.850        30
weighted avg      0.911     0.867     0.863        30

[[12  0  0]
 [ 0  6  4]
 [ 0  0  8]]
```

With 10x times the epoch the network cant even reach the worst case of the nectorks above