# Assignment final report : Neural networks with numpy

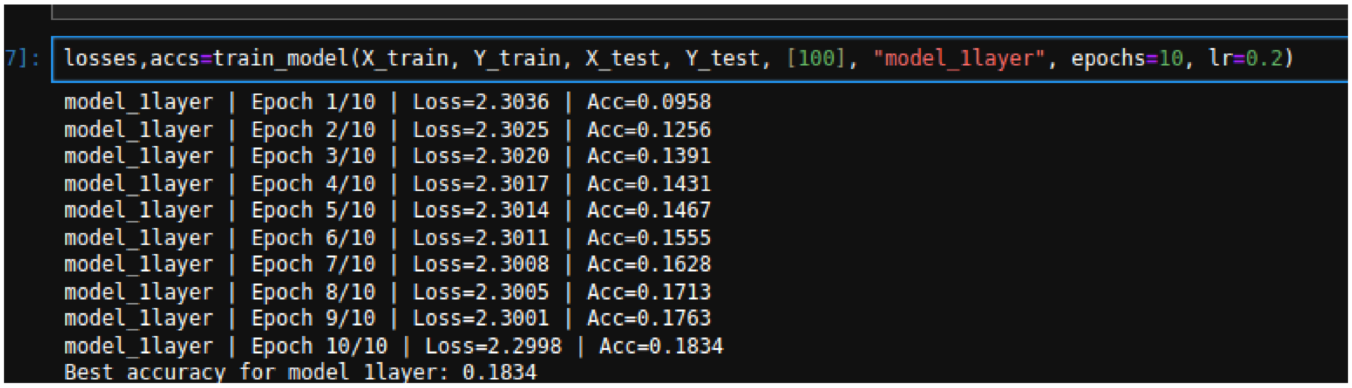
Hrittik Bhattacharya
hrittikbhattacharya_25rco01@dtu.ac.in

Sanjay Singh Gurjar
sanjaysinghgurjar_25rco02@dtu.ac.in

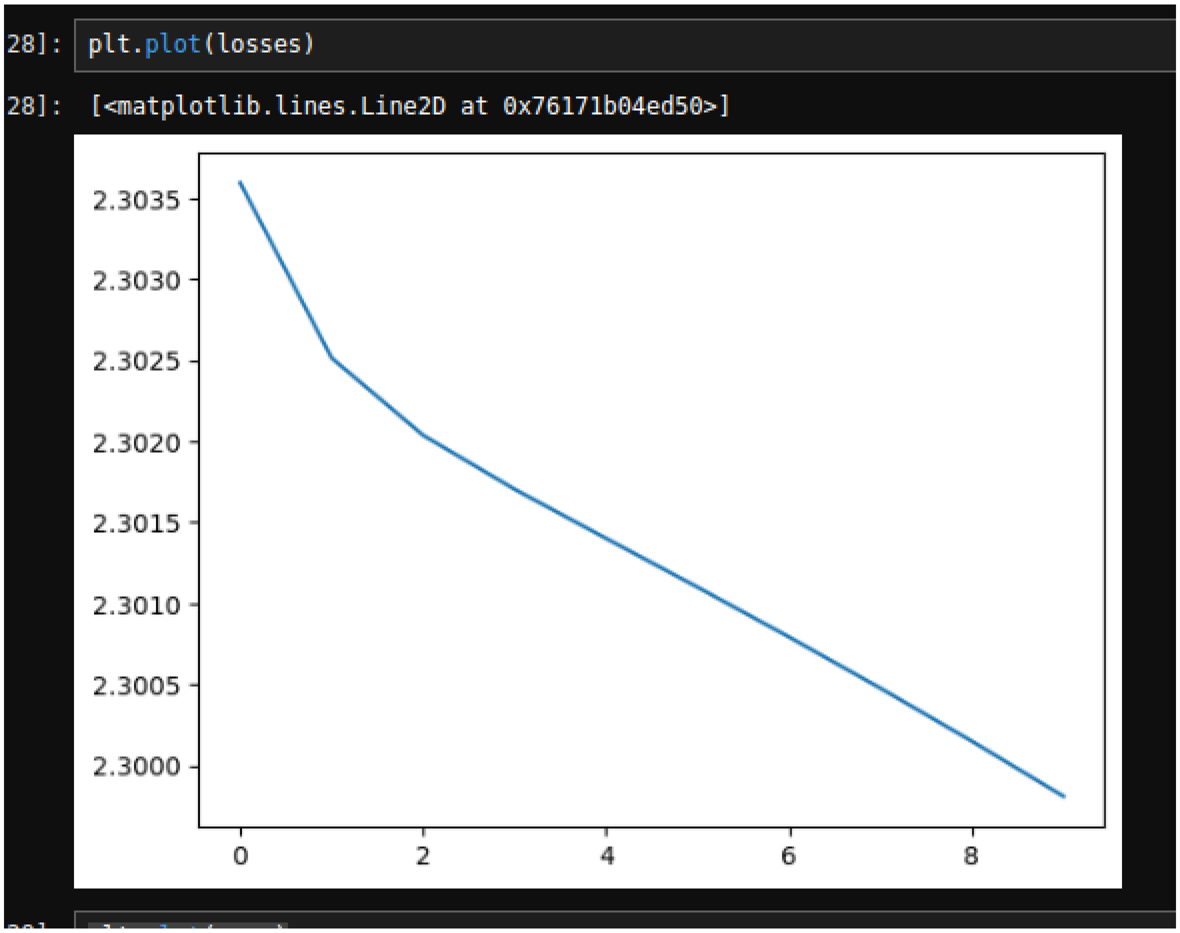
Code link:
ANN_A1_Group01

## Solution

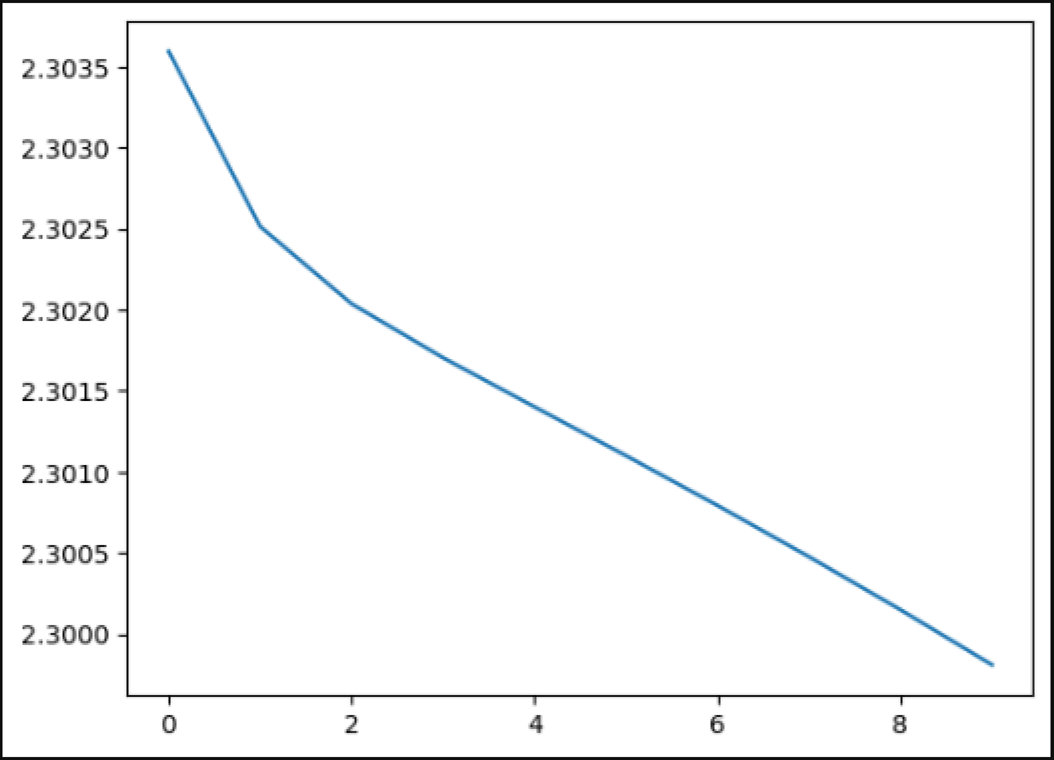
## P1:



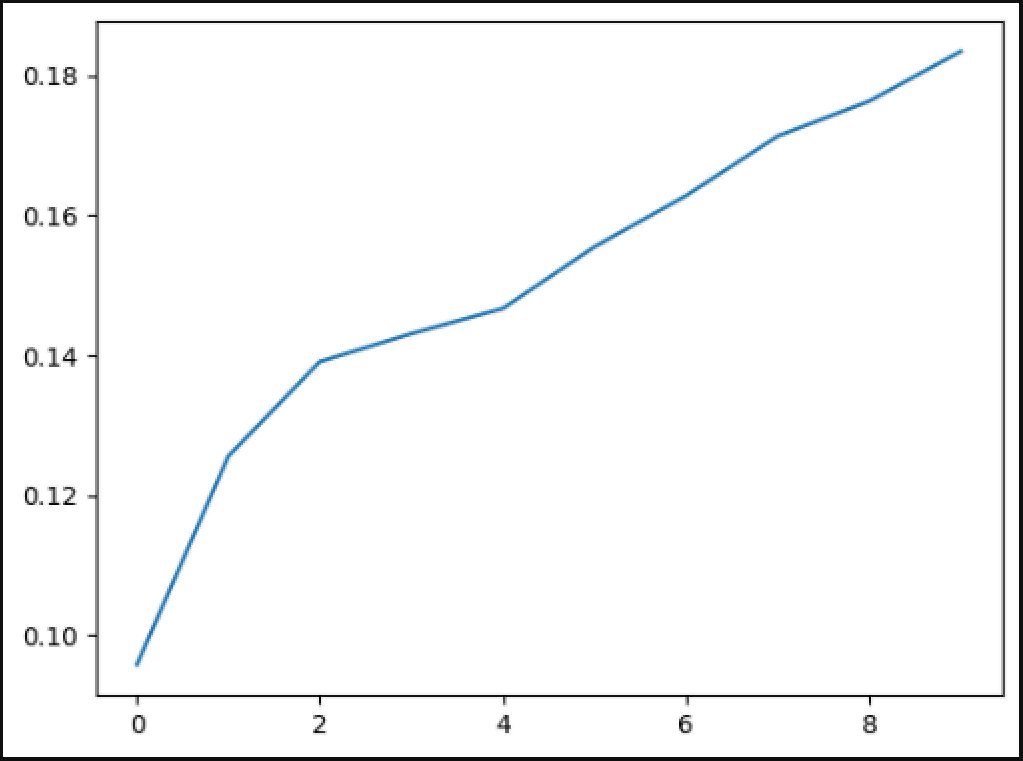
```
7]: losses,accs=train_model(X_train, Y_train, X_test, Y_test, [100], "model_1layer", epochs=10, lr=0.2)

    model_1layer | Epoch 1/10 | Loss=2.3036 | Acc=0.0958
    model_1layer | Epoch 2/10 | Loss=2.3025 | Acc=0.1256
    model_1layer | Epoch 3/10 | Loss=2.3020 | Acc=0.1391
    model_1layer | Epoch 4/10 | Loss=2.3017 | Acc=0.1431
    model_1layer | Epoch 5/10 | Loss=2.3014 | Acc=0.1467
    model_1layer | Epoch 6/10 | Loss=2.3011 | Acc=0.1555
    model_1layer | Epoch 7/10 | Loss=2.3008 | Acc=0.1628
    model_1layer | Epoch 8/10 | Loss=2.3005 | Acc=0.1713
    model_1layer | Epoch 9/10 | Loss=2.3001 | Acc=0.1763
    model_1layer | Epoch 10/10 | Loss=2.2998 | Acc=0.1834
    Best accuracy for model_1layer: 0.1834
```

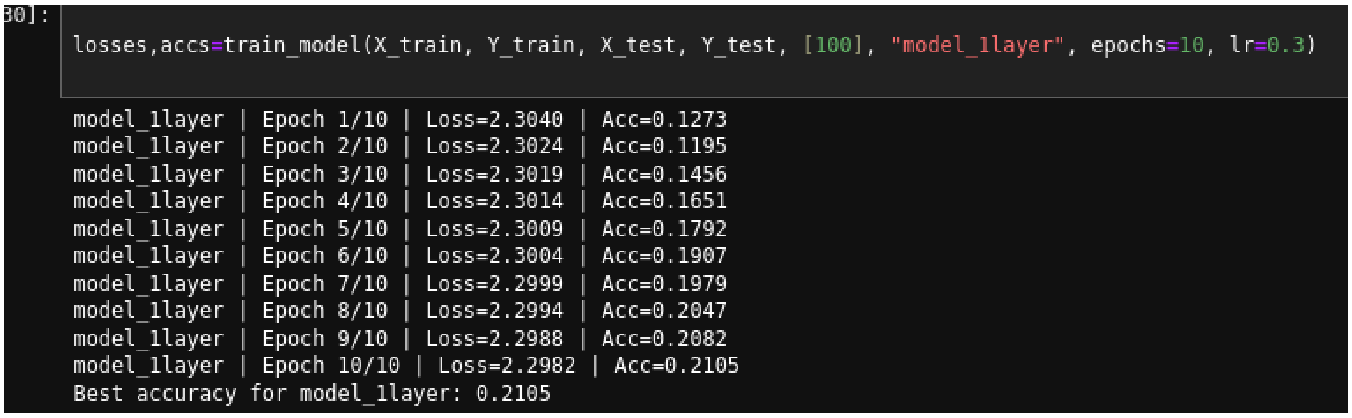


```
28]: plt.plot(losses)

28]: [<matplotlib.lines.Line2D at 0x76171b04ed50>]
```

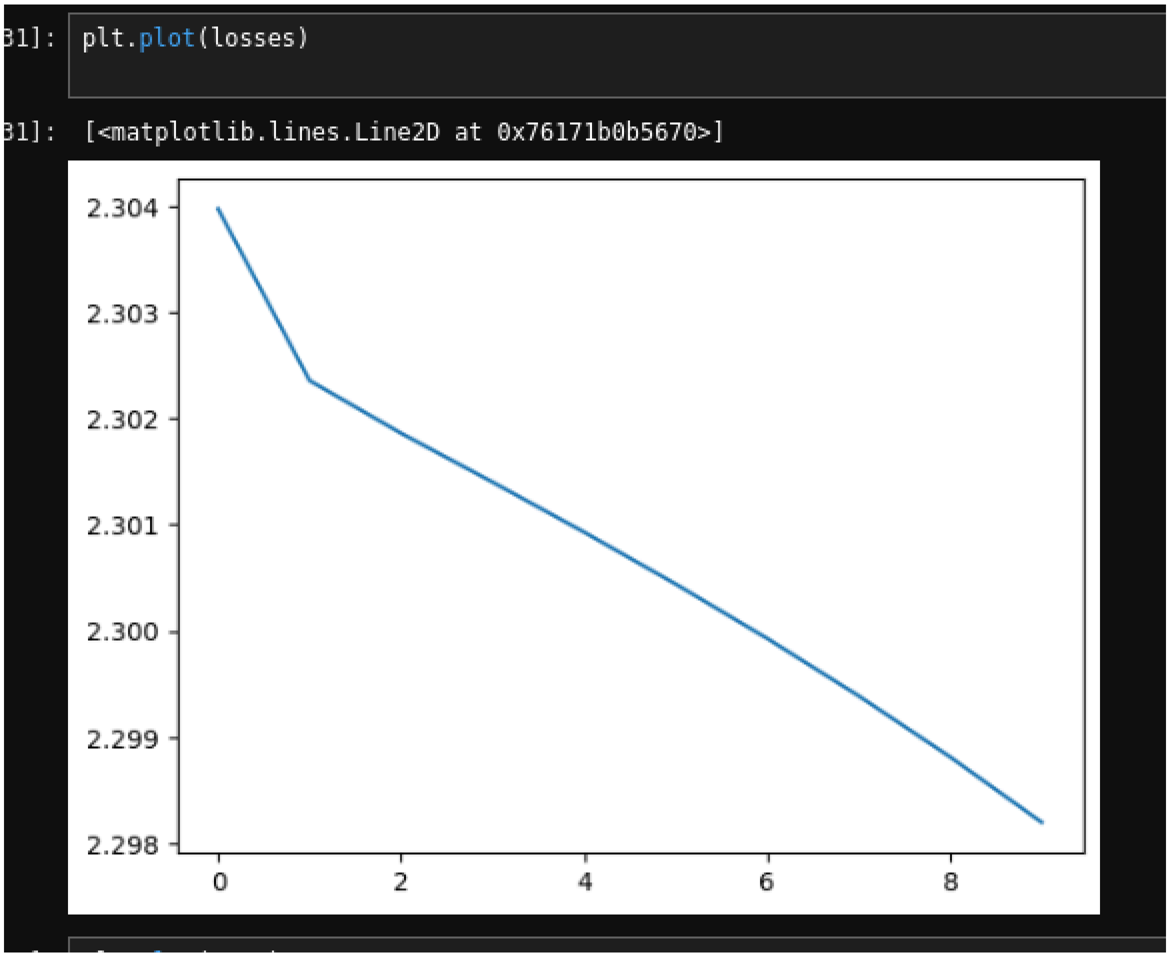


```
29]:  plt.plot(accs)

29]:  [<matplotlib.lines.Line2D at 0x76171b0e01d0>]
```

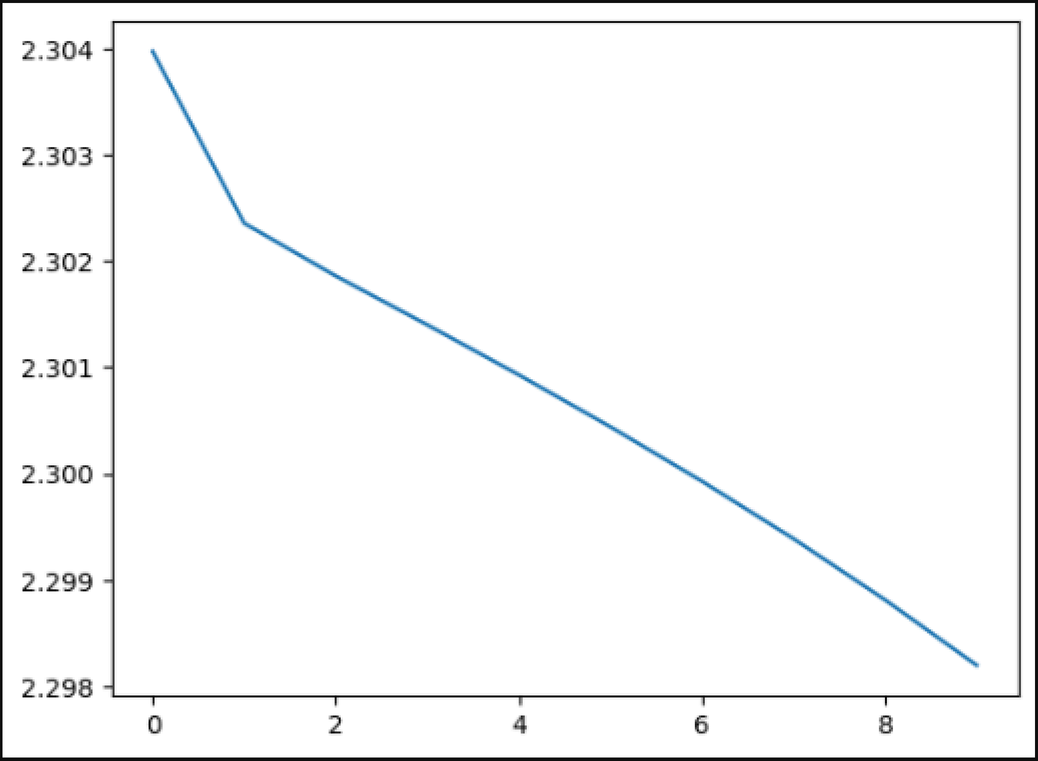


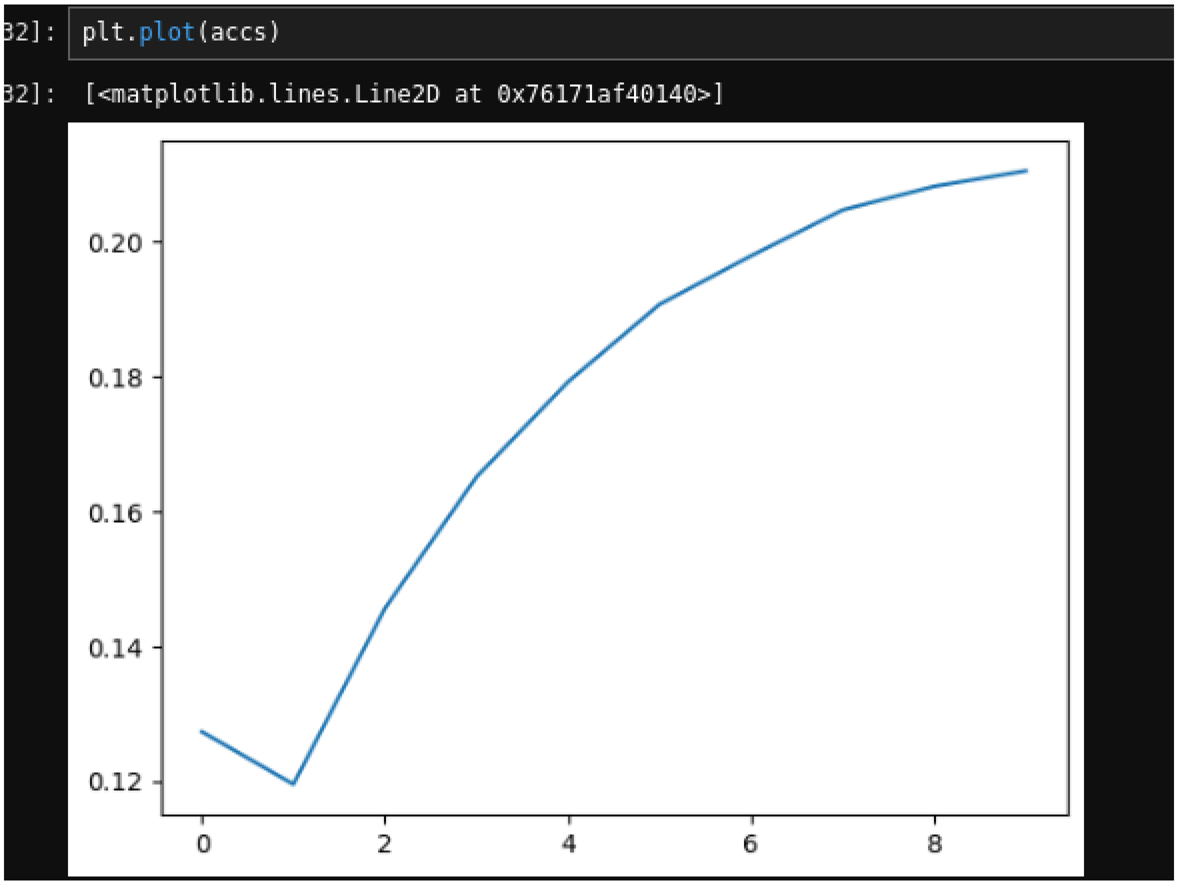


```
30]:  losses,accs=train_model(X_train, Y_train, X_test, Y_test, [100], "model_1layer", epochs=10, lr=0.3)

      model_1layer | Epoch 1/10 | Loss=2.3040 | Acc=0.1273
      model_1layer | Epoch 2/10 | Loss=2.3024 | Acc=0.1195
      model_1layer | Epoch 3/10 | Loss=2.3019 | Acc=0.1456
      model_1layer | Epoch 4/10 | Loss=2.3014 | Acc=0.1651
      model_1layer | Epoch 5/10 | Loss=2.3009 | Acc=0.1792
      model_1layer | Epoch 6/10 | Loss=2.3004 | Acc=0.1907
      model_1layer | Epoch 7/10 | Loss=2.2999 | Acc=0.1979
      model_1layer | Epoch 8/10 | Loss=2.2994 | Acc=0.2047
      model_1layer | Epoch 9/10 | Loss=2.2988 | Acc=0.2082
      model_1layer | Epoch 10/10 | Loss=2.2982 | Acc=0.2105
      Best accuracy for model_1layer: 0.2105
```

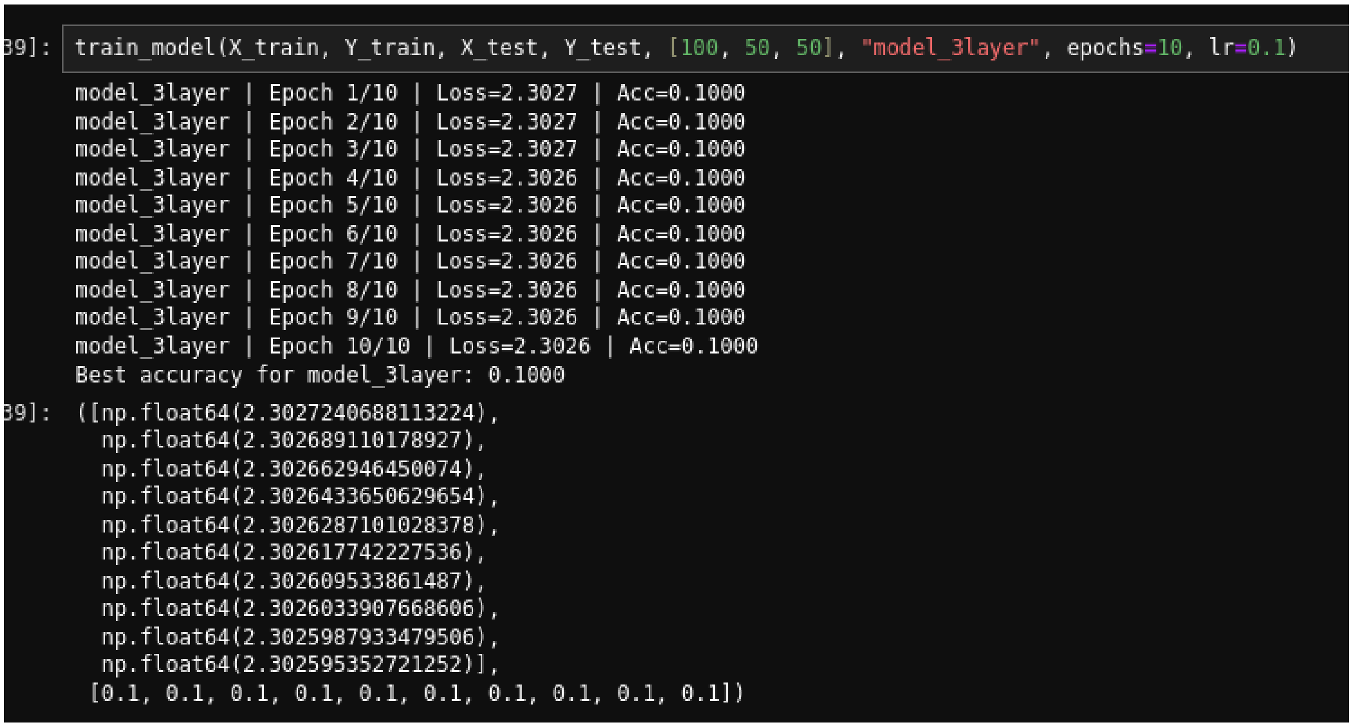


```
31]: plt.plot(losses)

31]: [<matplotlib.lines.Line2D at 0x76171b0b5670>]
```



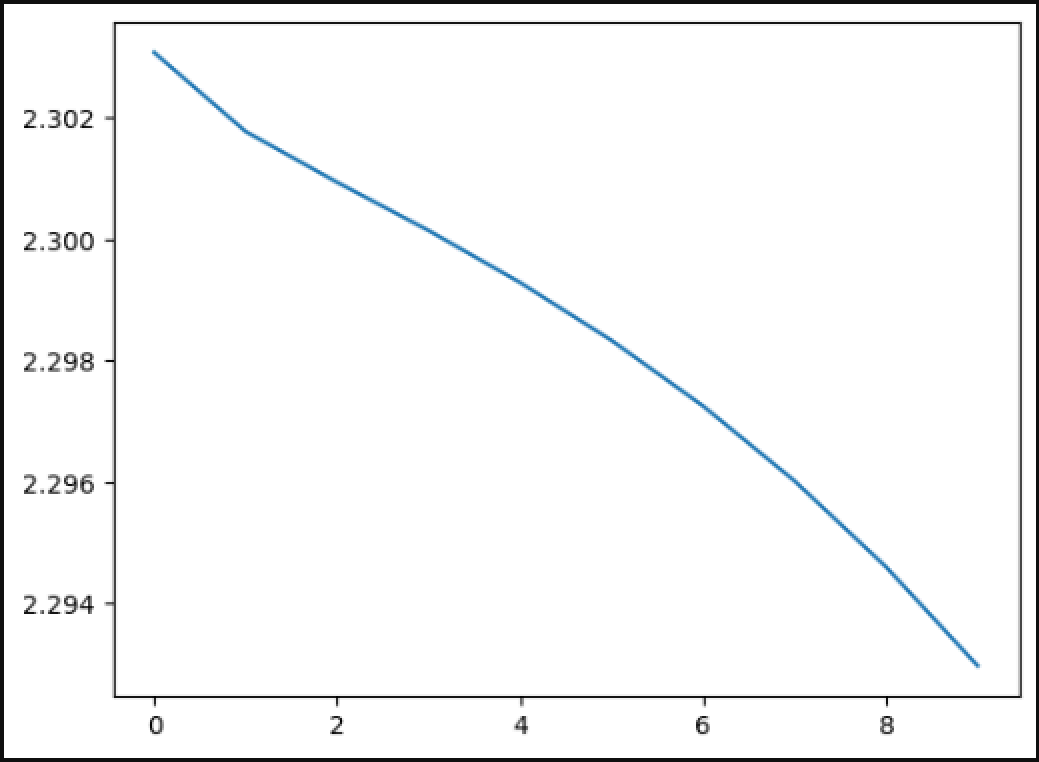


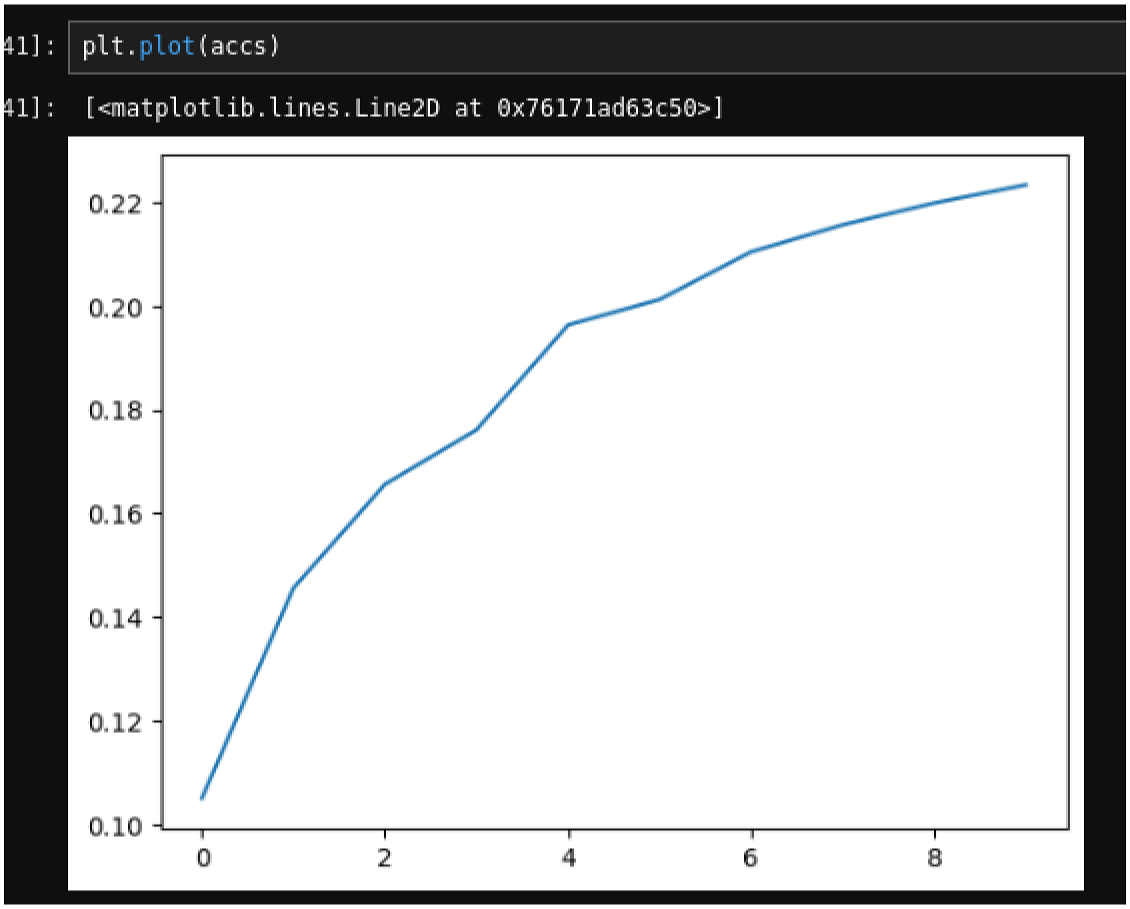
```
32]:  plt.plot(accs)

32]:  [<matplotlib.lines.Line2D at 0x76171af40140>]
```



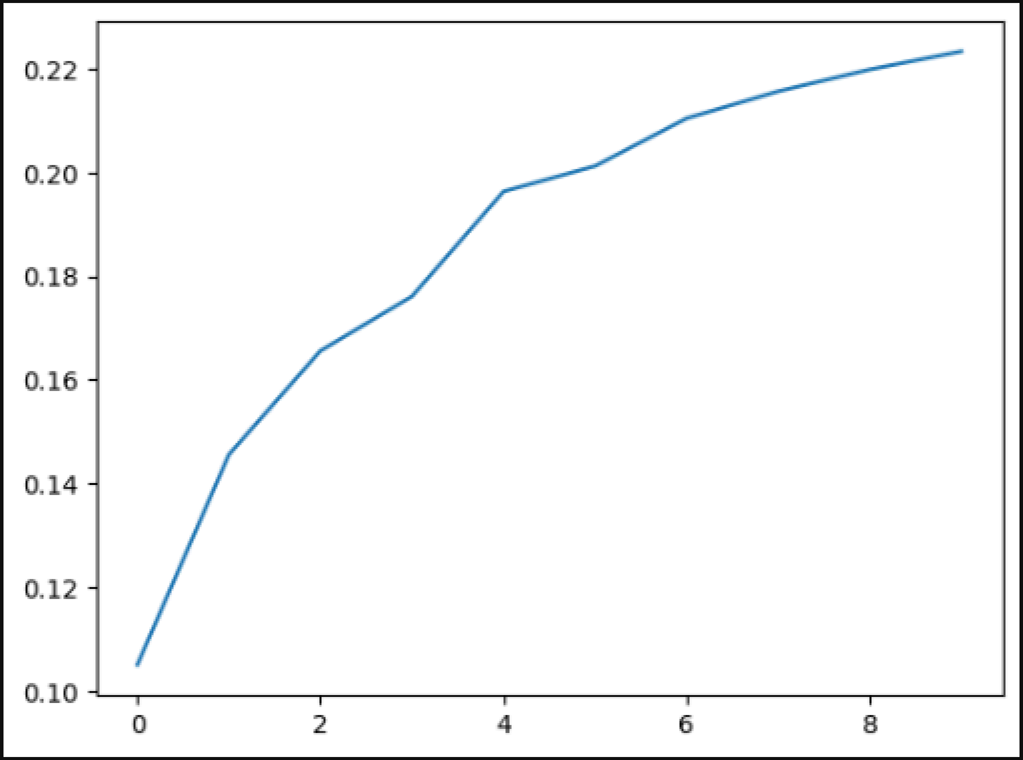
```
39]: train_model(X_train, Y_train, X_test, Y_test, [100, 50, 50], "model_3layer", epochs=10, lr=0.1)

     model_3layer | Epoch 1/10 | Loss=2.3027 | Acc=0.1000
     model_3layer | Epoch 2/10 | Loss=2.3027 | Acc=0.1000
     model_3layer | Epoch 3/10 | Loss=2.3027 | Acc=0.1000
     model_3layer | Epoch 4/10 | Loss=2.3026 | Acc=0.1000
     model_3layer | Epoch 5/10 | Loss=2.3026 | Acc=0.1000
     model_3layer | Epoch 6/10 | Loss=2.3026 | Acc=0.1000
     model_3layer | Epoch 7/10 | Loss=2.3026 | Acc=0.1000
     model_3layer | Epoch 8/10 | Loss=2.3026 | Acc=0.1000
     model_3layer | Epoch 9/10 | Loss=2.3026 | Acc=0.1000
     model_3layer | Epoch 10/10 | Loss=2.3026 | Acc=0.1000
     Best accuracy for model_3layer: 0.1000

39]: ([np.float64(2.3027240688113224),
       np.float64(2.302689110178927),
       np.float64(2.302662946450074),
       np.float64(2.3026433650629654),
       np.float64(2.3026287101028378),
       np.float64(2.302617742227536),
       np.float64(2.302609533861487),
       np.float64(2.3026033907668606),
       np.float64(2.3025987933479506),
       np.float64(2.302595352721252)],
      [0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1])
```

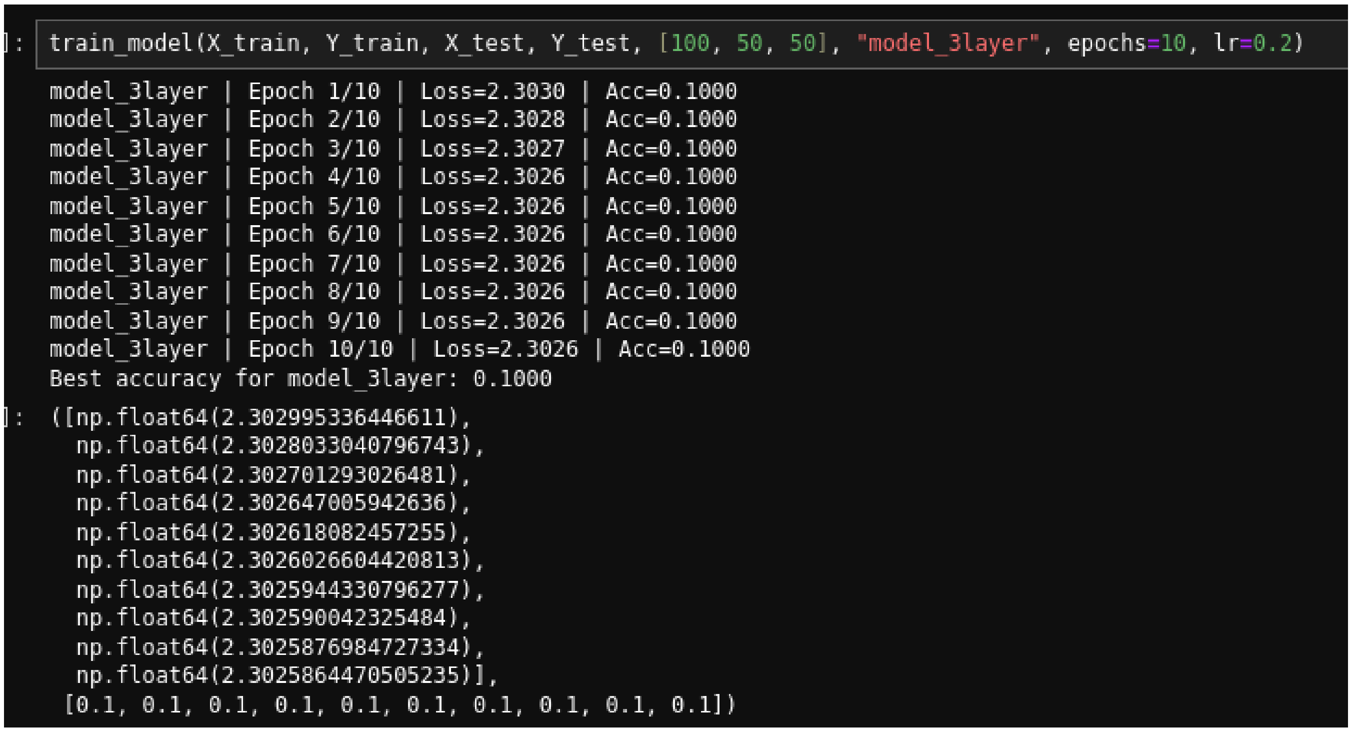
```
      [0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1])

40]: plt.plot(losses)

40]: [<matplotlib.lines.Line2D at 0x76171ae8bdd0>]
```



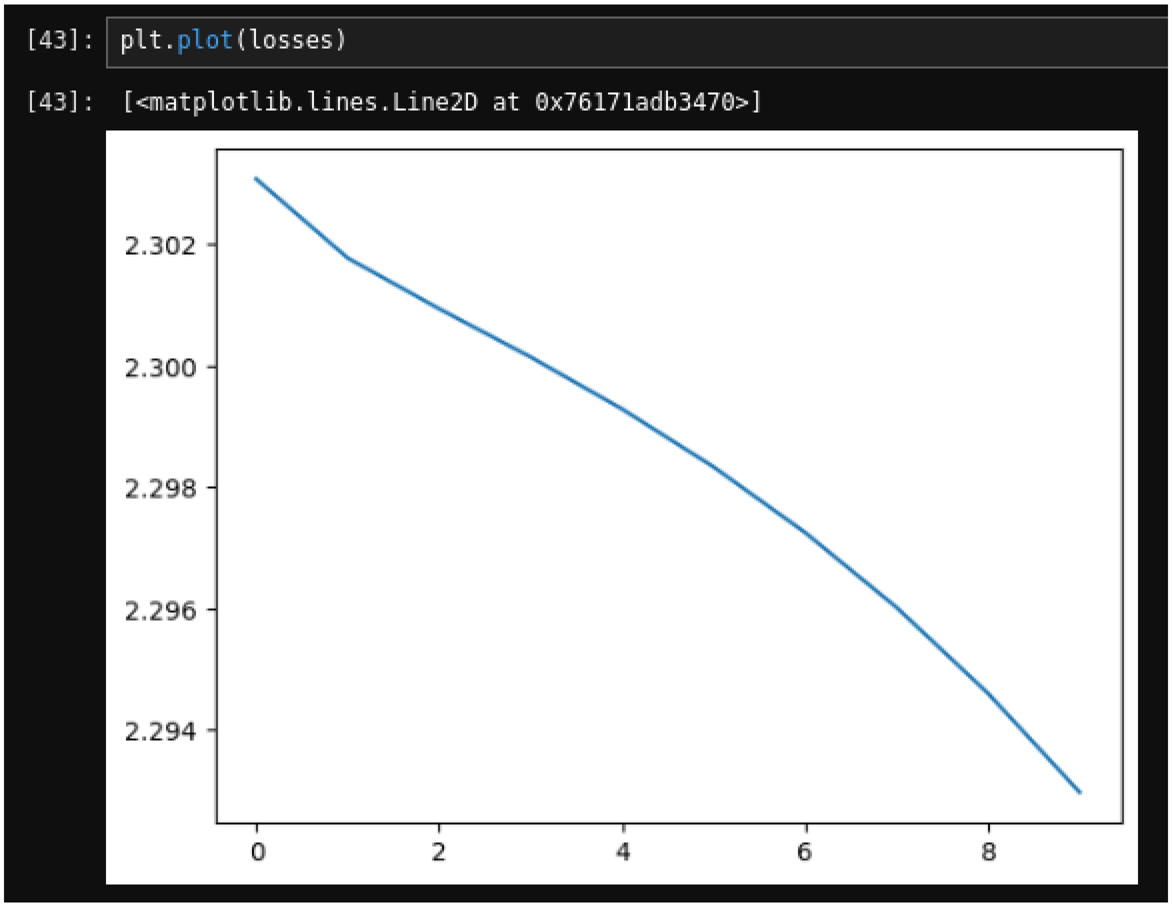


```
41]: plt.plot(accs)

41]: [<matplotlib.lines.Line2D at 0x76171ad63c50>]
```

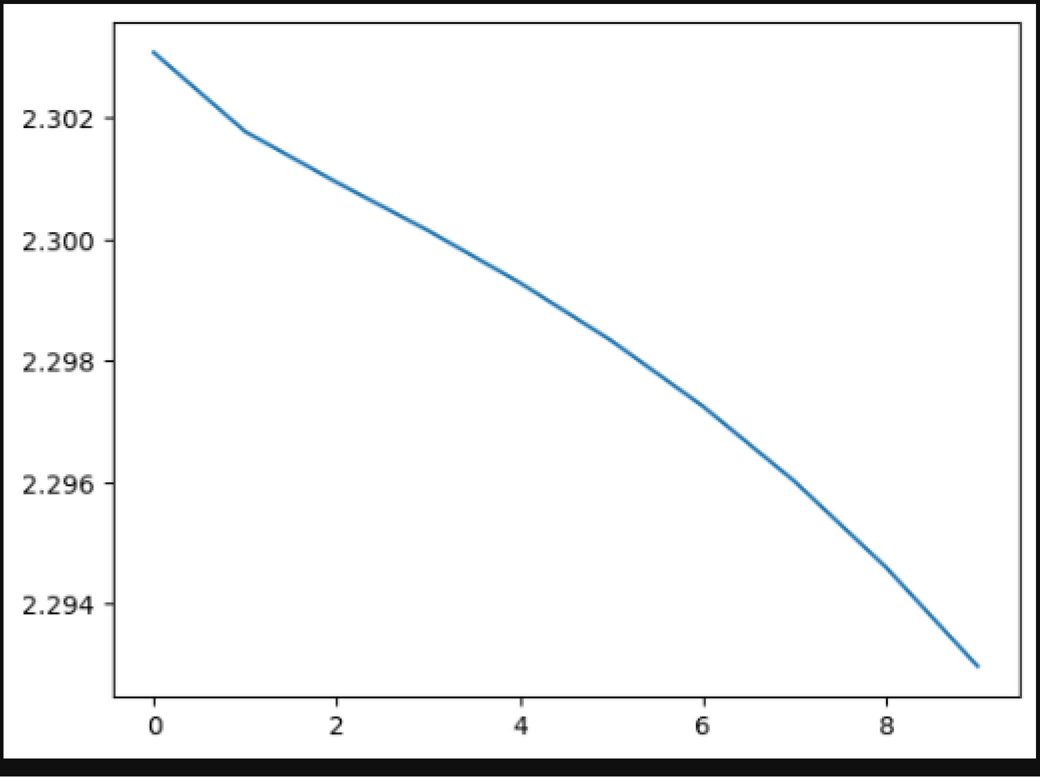


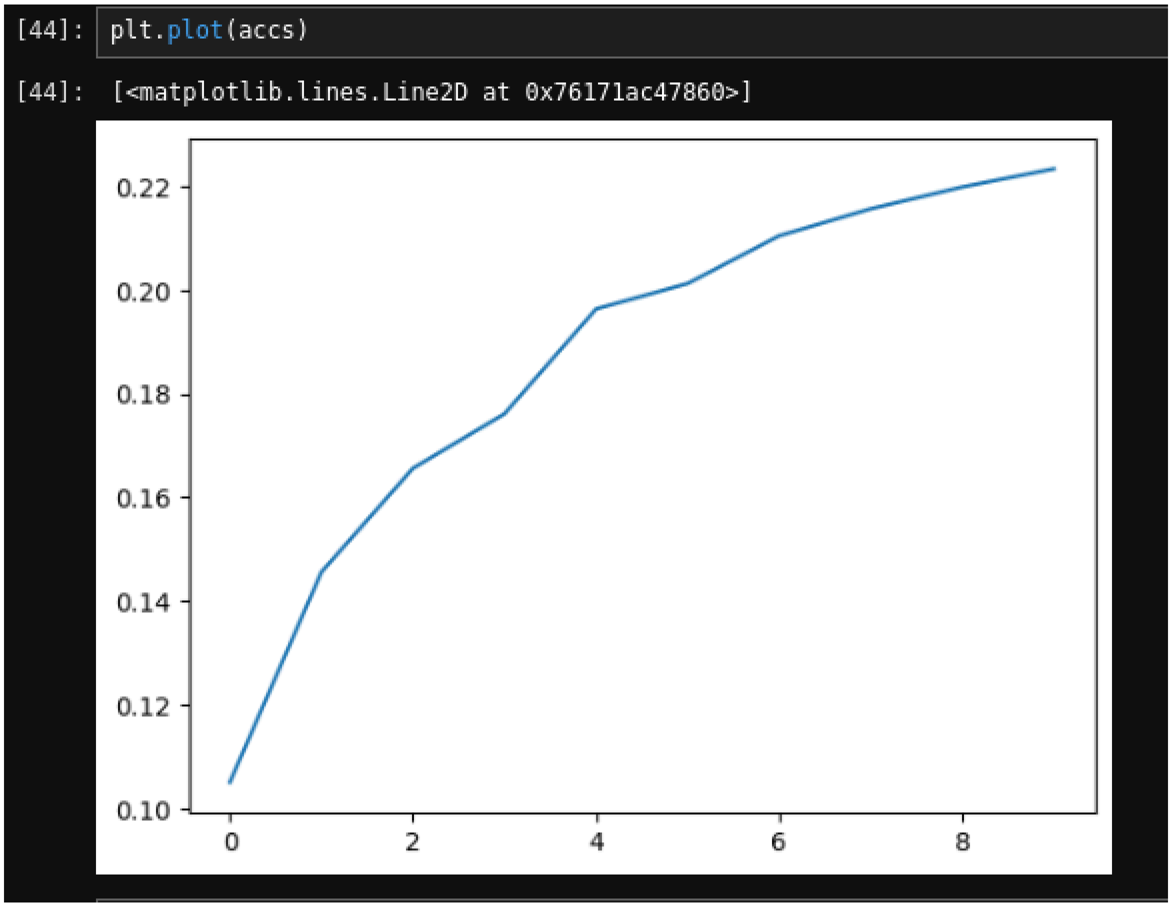


```
]: train_model(X_train, Y_train, X_test, Y_test, [100, 50, 50], "model_3layer", epochs=10, lr=0.2)

   model_3layer | Epoch 1/10 | Loss=2.3030 | Acc=0.1000
   model_3layer | Epoch 2/10 | Loss=2.3028 | Acc=0.1000
   model_3layer | Epoch 3/10 | Loss=2.3027 | Acc=0.1000
   model_3layer | Epoch 4/10 | Loss=2.3026 | Acc=0.1000
   model_3layer | Epoch 5/10 | Loss=2.3026 | Acc=0.1000
   model_3layer | Epoch 6/10 | Loss=2.3026 | Acc=0.1000
   model_3layer | Epoch 7/10 | Loss=2.3026 | Acc=0.1000
   model_3layer | Epoch 8/10 | Loss=2.3026 | Acc=0.1000
   model_3layer | Epoch 9/10 | Loss=2.3026 | Acc=0.1000
   model_3layer | Epoch 10/10 | Loss=2.3026 | Acc=0.1000
   Best accuracy for model_3layer: 0.1000

]: ([np.float64(2.302995336446611),
     np.float64(2.3028033040796743),
     np.float64(2.302701293026481),
     np.float64(2.302647005942636),
     np.float64(2.302618082457255),
     np.float64(2.3026026604420813),
     np.float64(2.3025944330796277),
     np.float64(2.302590042325484),
     np.float64(2.3025876984727334),
     np.float64(2.3025864470505235)],
    [0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1])
```



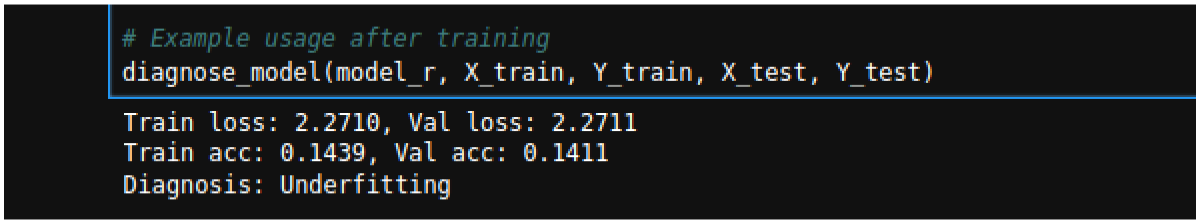
```
[43]: plt.plot(losses)

[43]: [<matplotlib.lines.Line2D at 0x76171adb3470>]
```





```
[44]: plt.plot(accs)

[44]: [<matplotlib.lines.Line2D at 0x76171ac47860>]
```

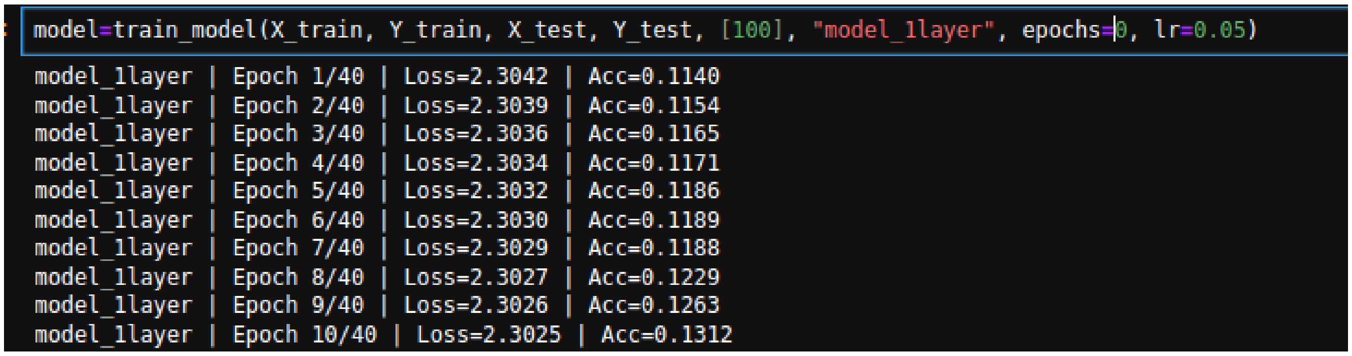
## Q2: overfitting or underfitting:



```
# Example usage after training
diagnose_model(model_r, X_train, Y_train, X_test, Y_test)

Train loss: 2.2710, Val loss: 2.2711
Train acc: 0.1439, Val acc: 0.1411
Diagnosis: Underfitting
```

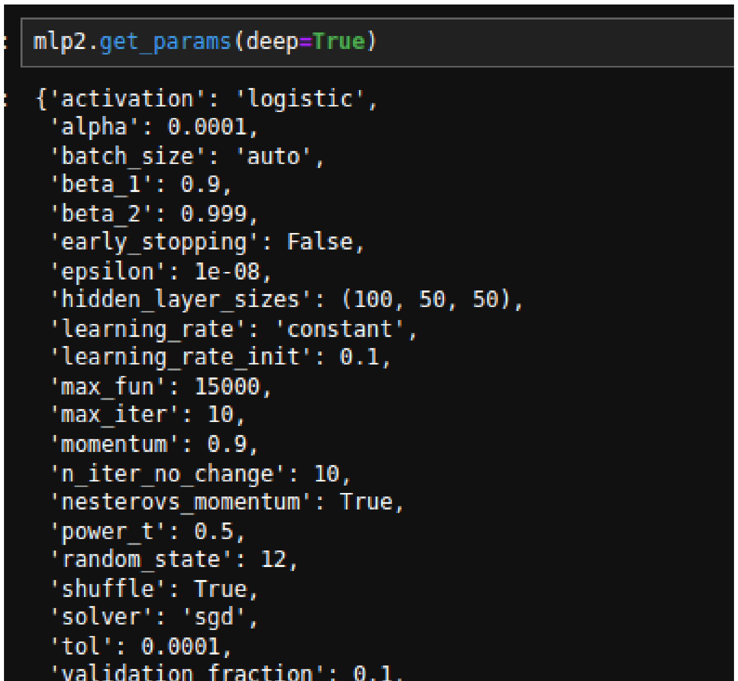
Our model underfits.

## Q3: nn with relu()



```
model=train_model(X_train, Y_train, X_test, Y_test, [100], "model_1layer", epochs=0, lr=0.05)

model_1layer | Epoch 1/40 | Loss=2.3042 | Acc=0.1140
model_1layer | Epoch 2/40 | Loss=2.3039 | Acc=0.1154
model_1layer | Epoch 3/40 | Loss=2.3036 | Acc=0.1165
model_1layer | Epoch 4/40 | Loss=2.3034 | Acc=0.1171
model_1layer | Epoch 5/40 | Loss=2.3032 | Acc=0.1186
model_1layer | Epoch 6/40 | Loss=2.3030 | Acc=0.1189
model_1layer | Epoch 7/40 | Loss=2.3029 | Acc=0.1188
model_1layer | Epoch 8/40 | Loss=2.3027 | Acc=0.1229
model_1layer | Epoch 9/40 | Loss=2.3026 | Acc=0.1263
model_1layer | Epoch 10/40 | Loss=2.3025 | Acc=0.1312
```

## Q4: Implement NN with sklearn



```
mlp2.get_params(deep=True)

{'activation': 'logistic',
 'alpha': 0.0001,
 'batch_size': 'auto',
 'beta_1': 0.9,
 'beta_2': 0.999,
 'early_stopping': False,
 'epsilon': 1e-08,
 'hidden_layer_sizes': (100, 50, 50),
 'learning_rate': 'constant',
 'learning_rate_init': 0.1,
 'max_fun': 15000,
 'max_iter': 10,
 'momentum': 0.9,
 'n_iter_no_change': 10,
 'nesterovs_momentum': True,
 'power_t': 0.5,
 'random_state': 12,
 'shuffle': True,
 'solver': 'sgd',
 'tol': 0.0001,
 'validation_fraction': 0.1,
```

Accuracy of my model is very low compared to the MLP Classifier and the reason lies in the image above as mlp is optimized for performance.

Our model accuracy: 18 %

MLP accuracy: 44%