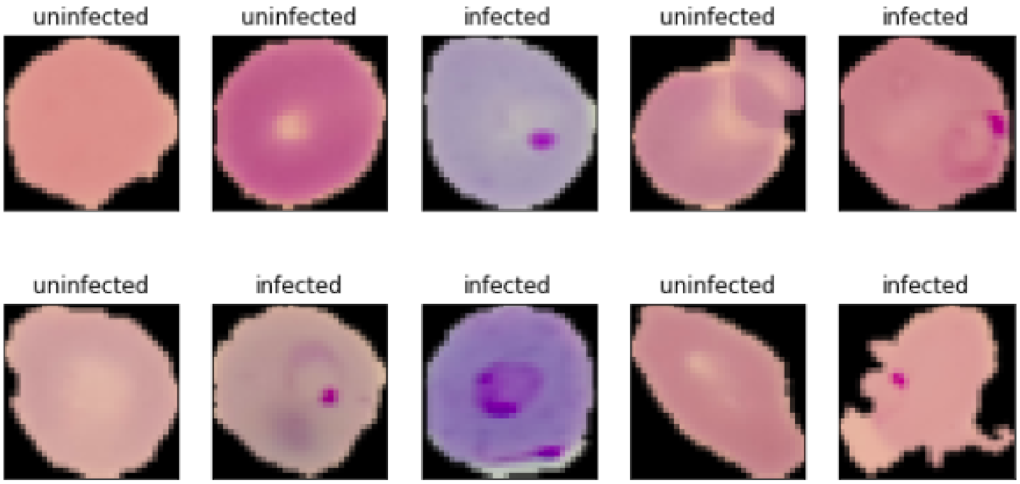
CSE 641 | Deep Learning

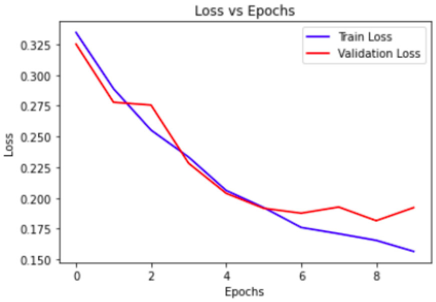
Assignment 2

Question 1

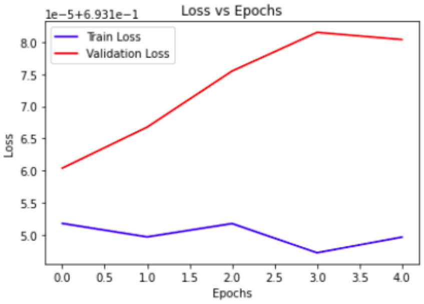
- The images while reading were resized to 32*32, converted to tensors and normalized by the factors of [0.5, 0.5, 0.5], [0.5, 0.5, 0.5]
- DataLoader class was used to create the train, test and validation datasets.
- The ratio of train:test:validation datasets was 80:10:10
- To display the images, a function was created which took in the image in np.array format.



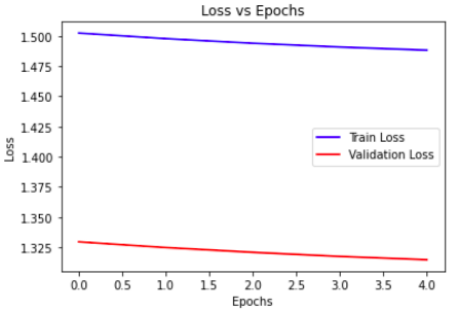
- A function to display the loss vs epochs plot for both train data and validation data was also used.
- CNN network with the given specifications was implemented.
  - For each convolution layer, a padding of 1 was used.
  - The channels were increased from 3 to 16 in 1st block, 16 to 32 in second block and finally from 32 to 64 in the last block.
  - For the pooling layers, kernel size of 3*3 was used with stride of 2
- BCE Loss and adam optimizer(lr = 0.0003) was used to train the model.
- The given CNN network gave an accuracy of 93.51% on the test set in 10 epochs.



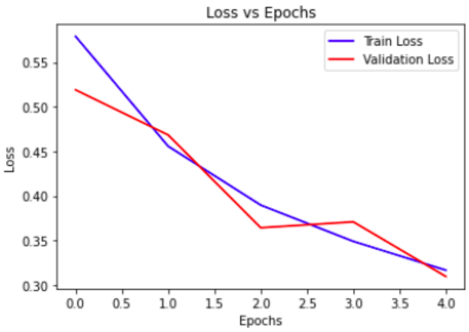
- For weight initialization, the models were run for 5 epochs.
- Zero Weight:
  - Both weights and bias were assigned 0 weights.
  - The model gave an accuracy of 49.82% and didn't learn anything.



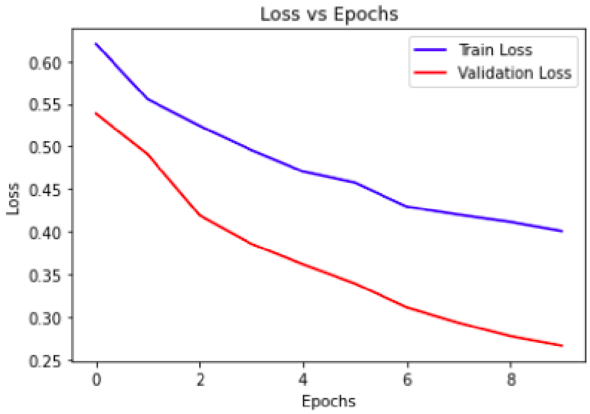
- Random Weight:
  - Both weights and bias were assigned random normal weights.
  - The model again gave an accuracy of 49.82% but from the graphs it was evident that the model is slowly but gradually starting to learn. Had the number of epochs were increased, the results would have been better.



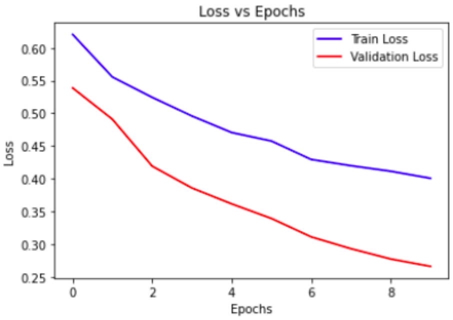
- He Initialization:
  - Weights were assigned distribution according to kaiming_normal or the normally distributed he initialization.
  - The model gave an accuracy of 86.90% in just 5 epochs and it still had not converged properly.



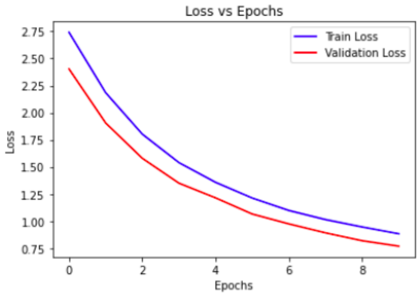
- It is evident from the results that He initialization is the best amongst the three. This is because in zero weight initialization, the model is learning next to nothing as the weight update equation is reduced to 0 eventually. In the random distribution, due to the random nature of weights, the model will learn, but that would be very slow. In the He initialization, the weights were decided on the basis of a prediefined mathematical formula which takes into account various parameters like the sizes of layers among other things. Using this, the jump in accuracy was evident.
- For dropouts, the probability of dropping a neuron was kept at 0.4
- Between convolution layers:
  - Dropouts were used after blockA and blockB
  - After training for 10 epochs, an accuracy of 90.82% was reported



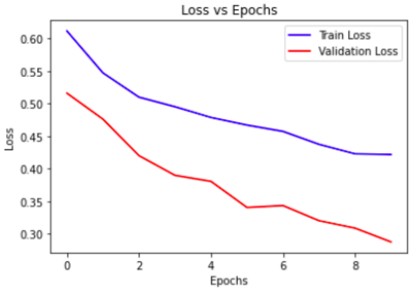
- After fc layer
  - Dropout layer was used after the final fc layer.
  - After training for 10 epochs, an accuracy of 89.73% was reported



- For Regularization, again the model was trained for 10 epochs for each regularization type.
- L1 Regularization:
  - lambda value for l1 regularization was fixed at 1e-3
  - The model gave an accuracy of 85.78%



- L2 Regularization:
  - Weight decay parameter for the optimizer was set at 1e-5
  - The model gave an accuracy of 89.40%

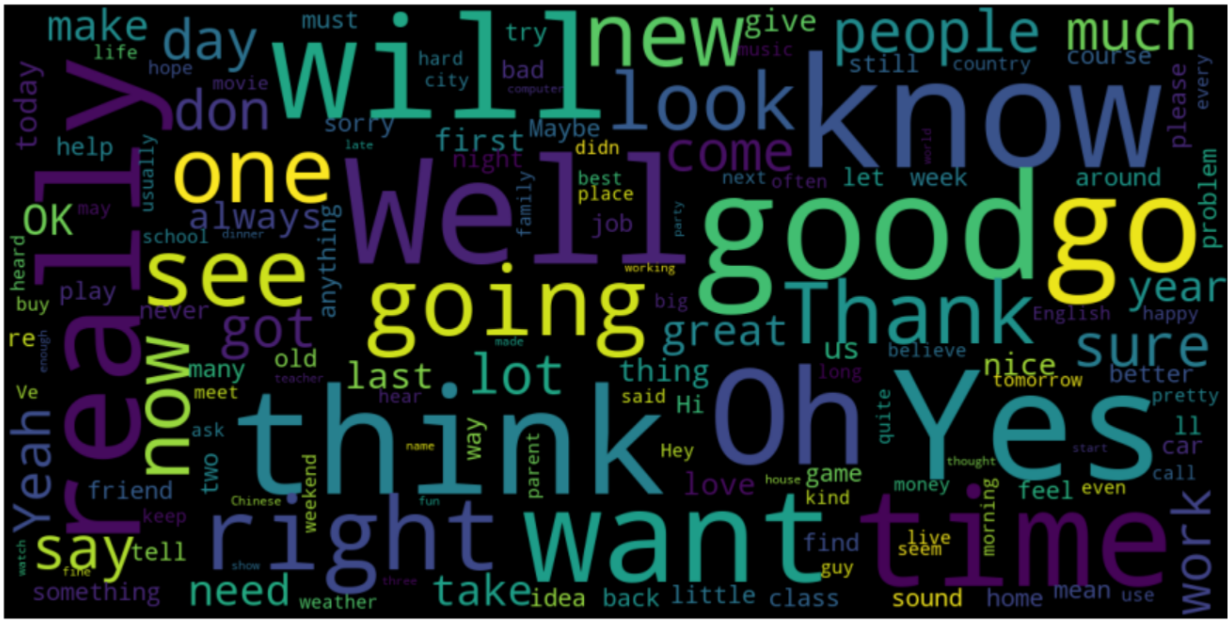


- Please Note that for this question, random seed was not fixed, as a result, there are minor deviations in the results. The models when loaded would give the exact same outputs but if the models are trained again, there can be significant changes in the results obtained.

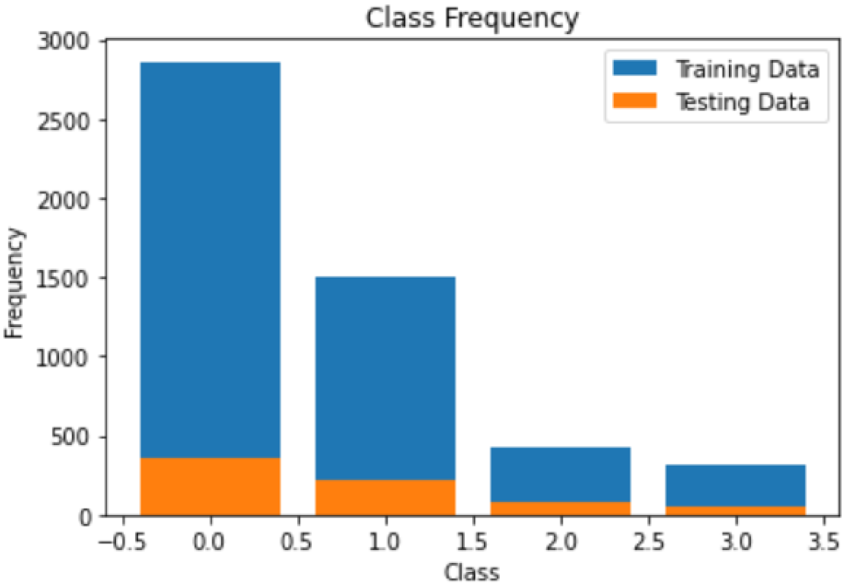
PART II: Long Short-Term Memory (LSTM)

Dialog Corpus Visualization

- Size defines the frequency of words present in dataset. Larger the size of the words present here more the frequency.
- Here we can see words like Well, will, good and time are most frequently used words.

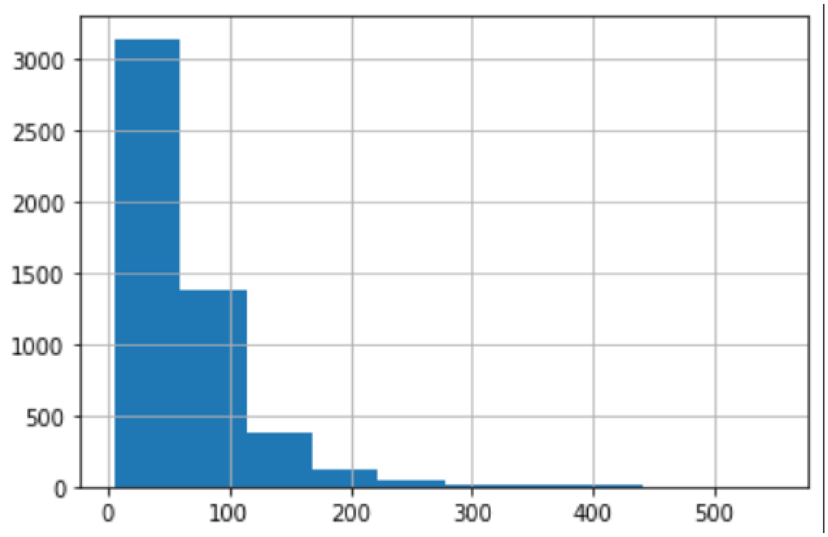


Class distribution: Frequency of each class in training and testing data



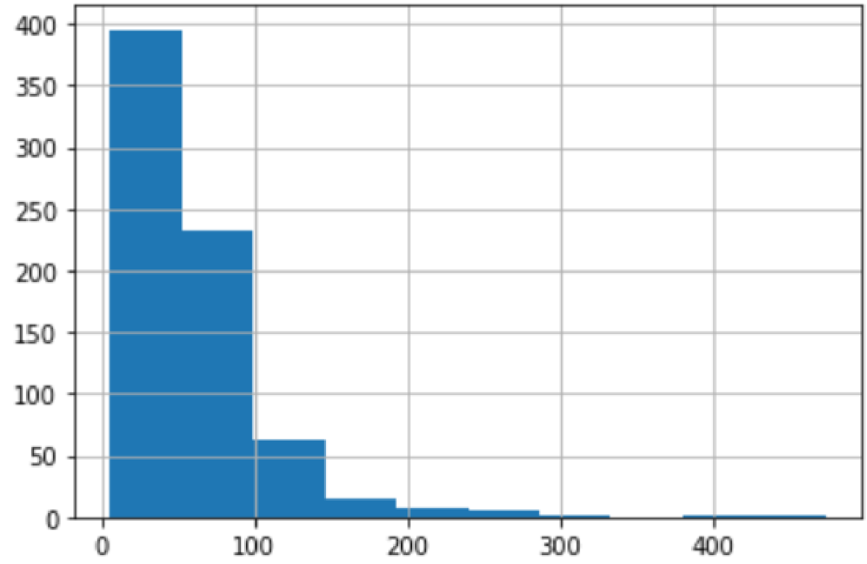
From the class frequency distribution plot we can see that frequency of Class 0 i.e inform class is maximum. Data is biased towards inform class.

Distribution of utterances length in training set



Most of the utterances lie in the range of 0 to 100.

Distribution of utterances length in testing set



LSTM and linear layers to predict act of utterance at time T considering previous X utterances' context:

Preprocessing steps:

- Convert all words into lowercase
- Replace all contractions such as don't to do not
- Tokenization using nltk

Sentence Embedding:

- Using Sbert transformer: Generate sentence embeddings for each utterance using sbert transformer and give it to LSTM.
- Using Glove: First apply word embeddings using glove and then average pooling to get sentence level embeddings

Data preparation for sending into LSTM according to context:

- Input data (both train and test) are sent in batches of X+1 if previous X utterances have to be considered. In a sliding window way: If X=3 we will send data as (0,1,2,3) then (1,2,3,4) which will be continued in a similar fashion. We are sending data in this way because LSTM considers each row of the data as a sample to be given to it's one cell.
- Output data is sent in following way: for each batch final sample's label is sent to LSTM for calculation loss and sending it for back propagation.

LSTM:

- Use of pytorch inbuilt LSTM function along with Linear function to get linear layer.

Save Model: Using pickle library

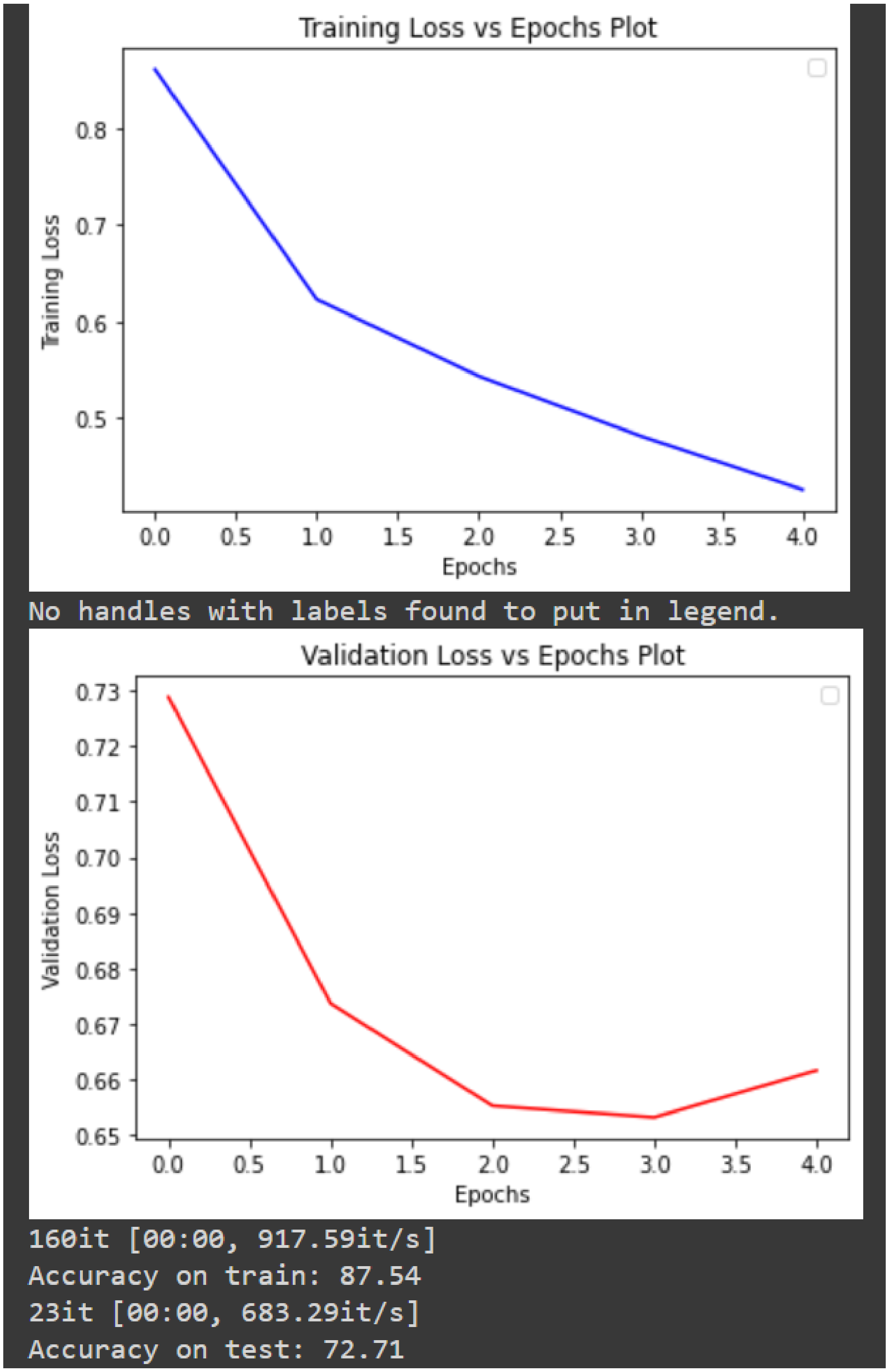
Architecture proposed:

LSTM:

Input_size given to LSTM = Embedding dimension = No of features of embedding vector = 768
Hidden_size = 8 (Simplified model otherwise we saw overfitting)
Num_layers = 1
Batch_first = True
Dropout = 0.5 (To avoid overfitting)
No of epochs = 5
Learning rate = 0.01
Optimizer = Adam
Loss function = Cross Entropy Loss

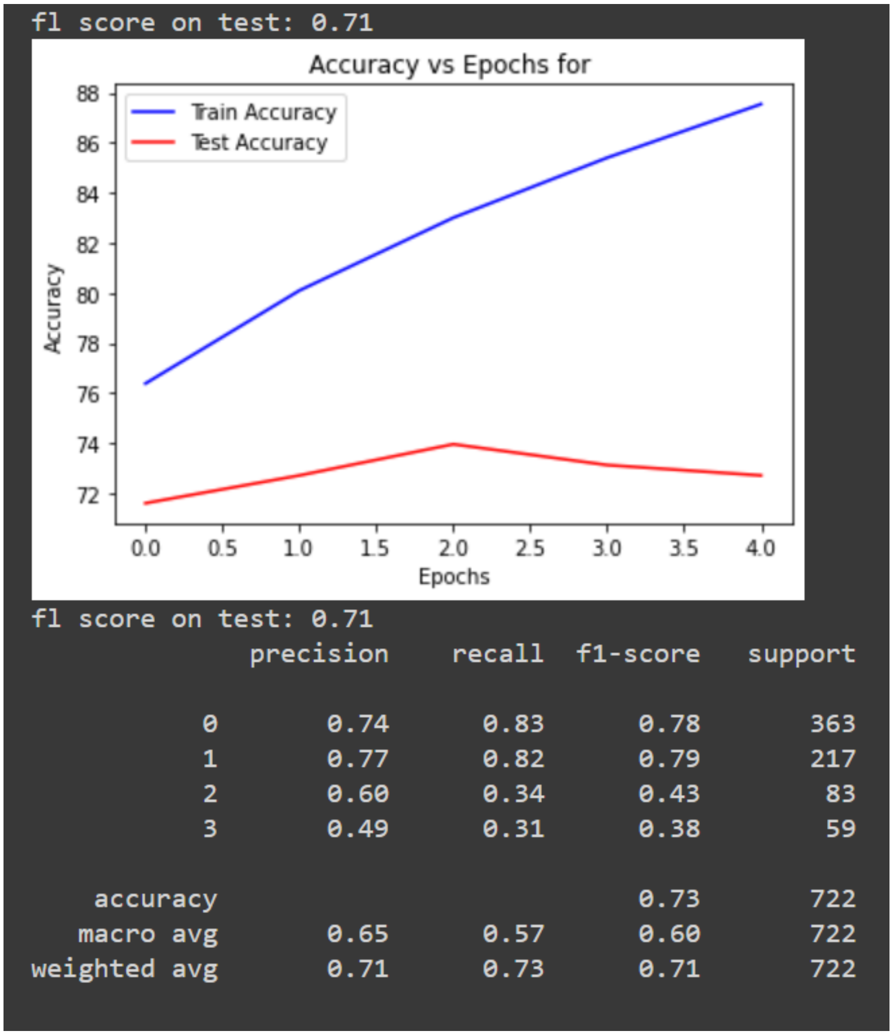
Training Loss vs Epochs, Validation Loss vs Epochs and Accuracy plots for each X = {0,1,2,3,4}

X = 0



```
No handles with labels found to put in legend.
```

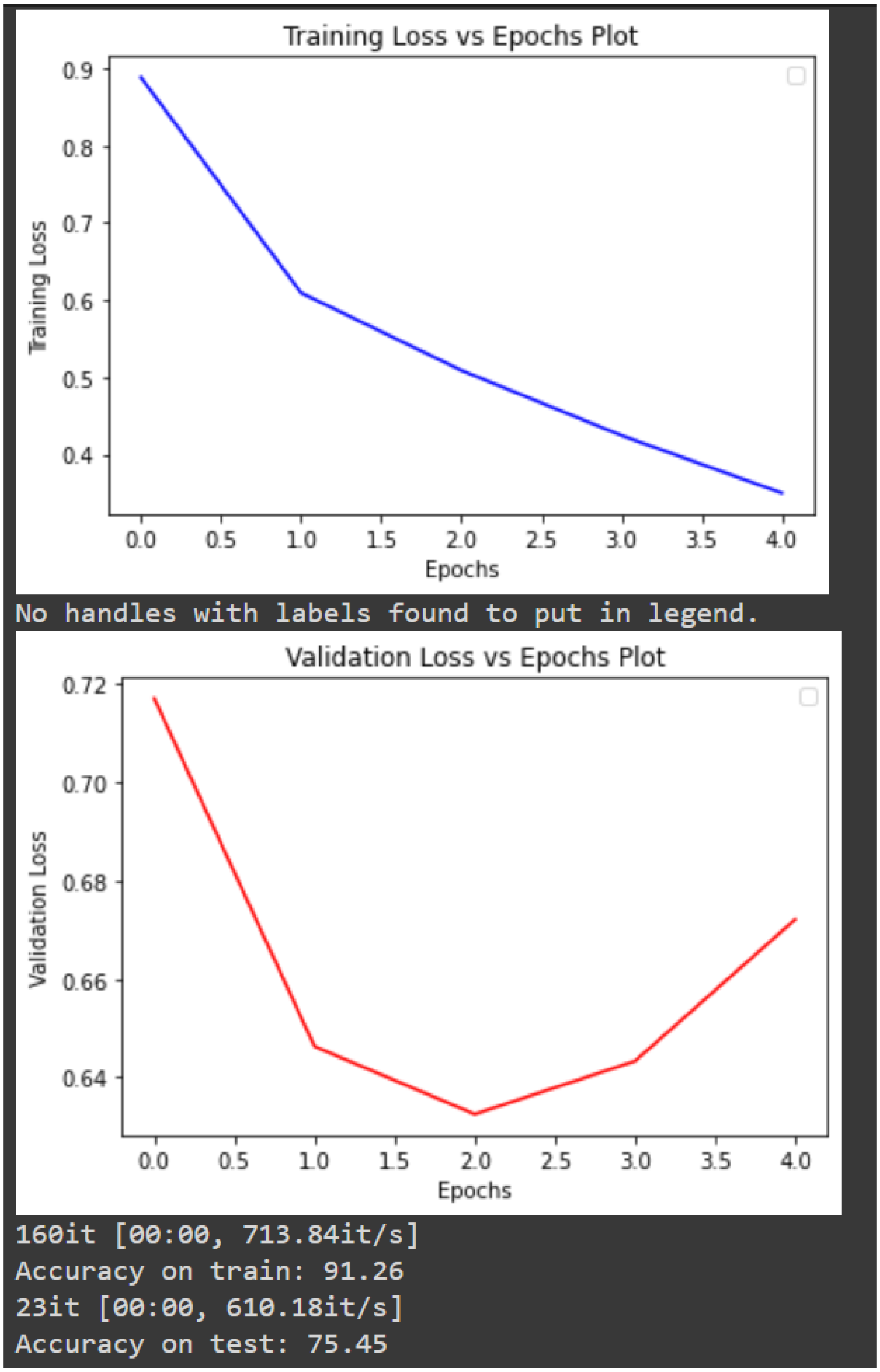
```
160it [00:00, 917.59it/s]
Accuracy on train: 87.54
23it [00:00, 683.29it/s]
Accuracy on test: 72.71
```



```
f1 score on test: 0.71
```

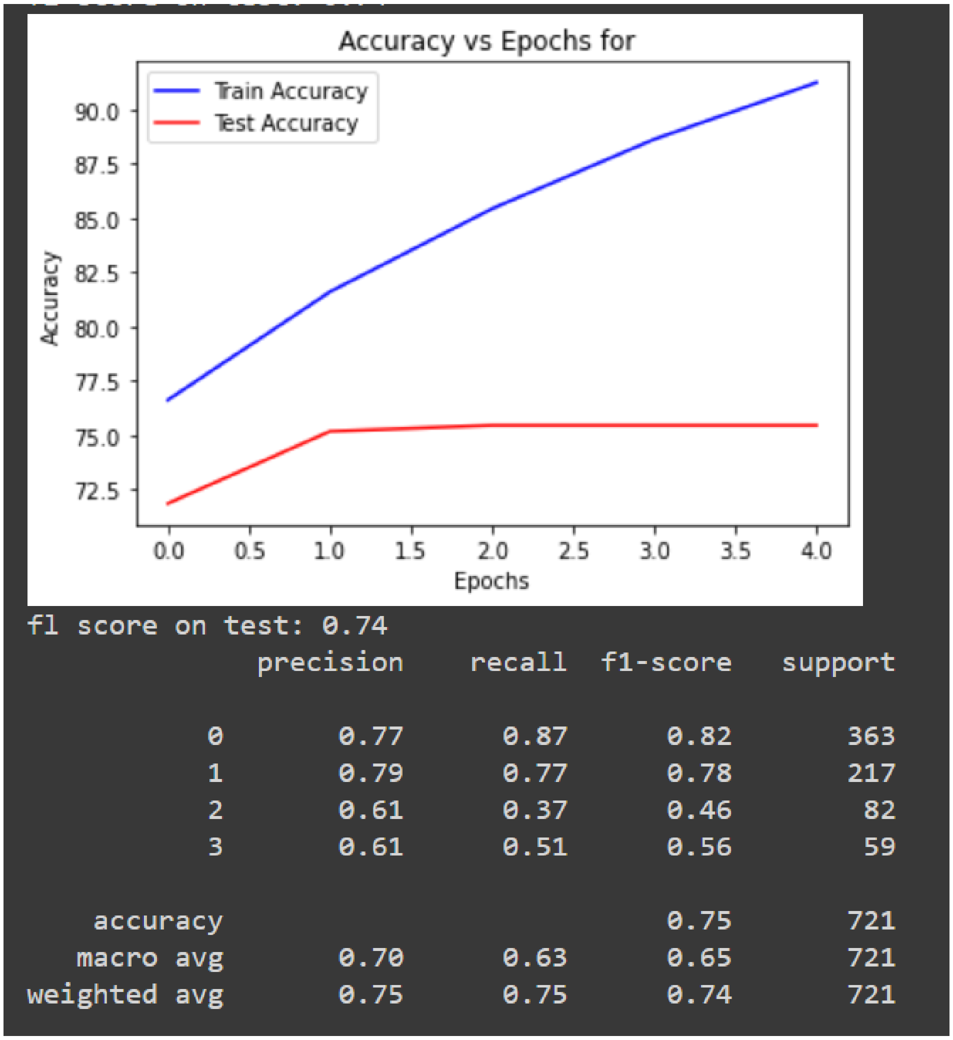
```
f1 score on test: 0.71
              precision    recall  f1-score   support

           0       0.74      0.83      0.78       363
           1       0.77      0.82      0.79       217
           2       0.60      0.34      0.43        83
           3       0.49      0.31      0.38        59

    accuracy                           0.73       722
   macro avg       0.65      0.57      0.60       722
weighted avg       0.71      0.73      0.71       722
```

X = 1



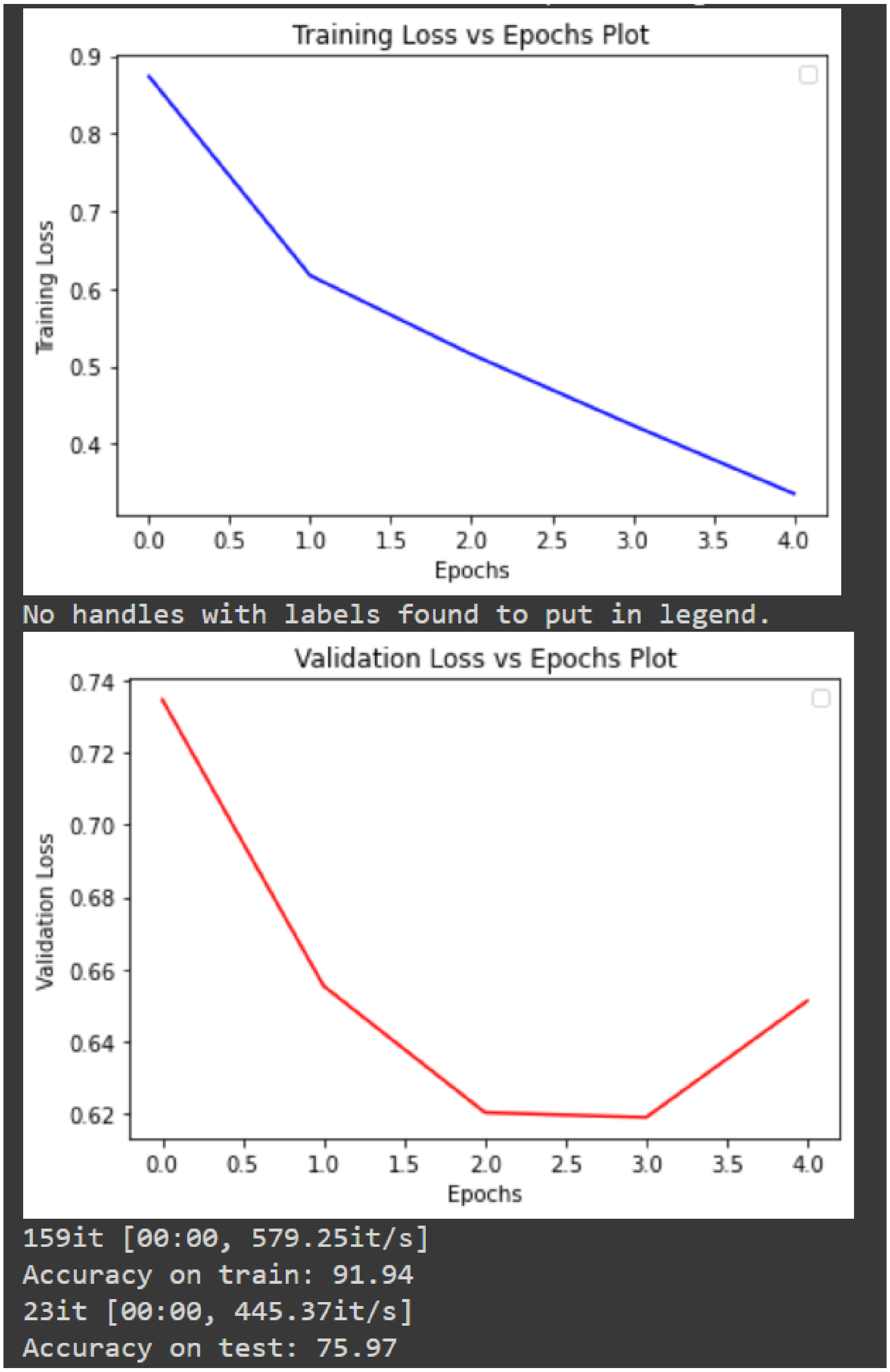
```
No handles with labels found to put in legend.
```

```
160it [00:00, 713.84it/s]
Accuracy on train: 91.26
23it [00:00, 610.18it/s]
Accuracy on test: 75.45
```



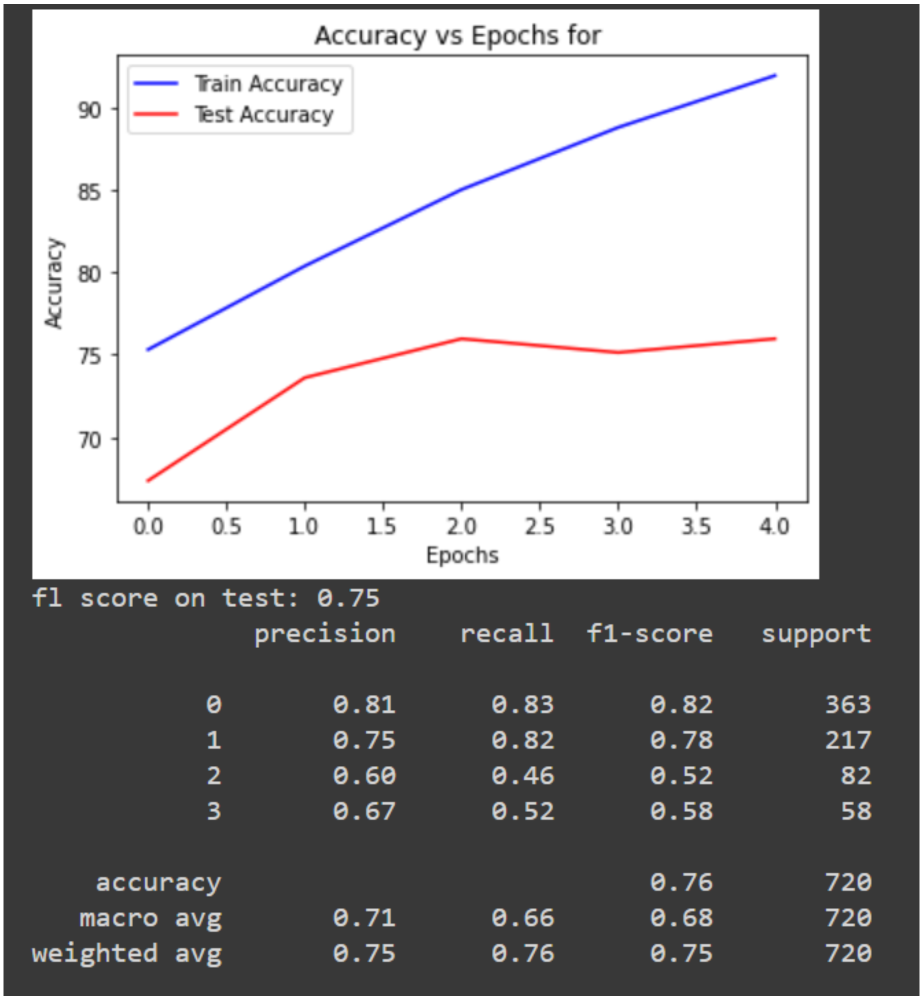
```
f1 score on test: 0.74
              precision    recall  f1-score   support

           0       0.77      0.87      0.82       363
           1       0.79      0.77      0.78       217
           2       0.61      0.37      0.46        82
           3       0.61      0.51      0.56        59

    accuracy                           0.75       721
   macro avg       0.70      0.63      0.65       721
weighted avg       0.75      0.75      0.74       721
```

X = 2



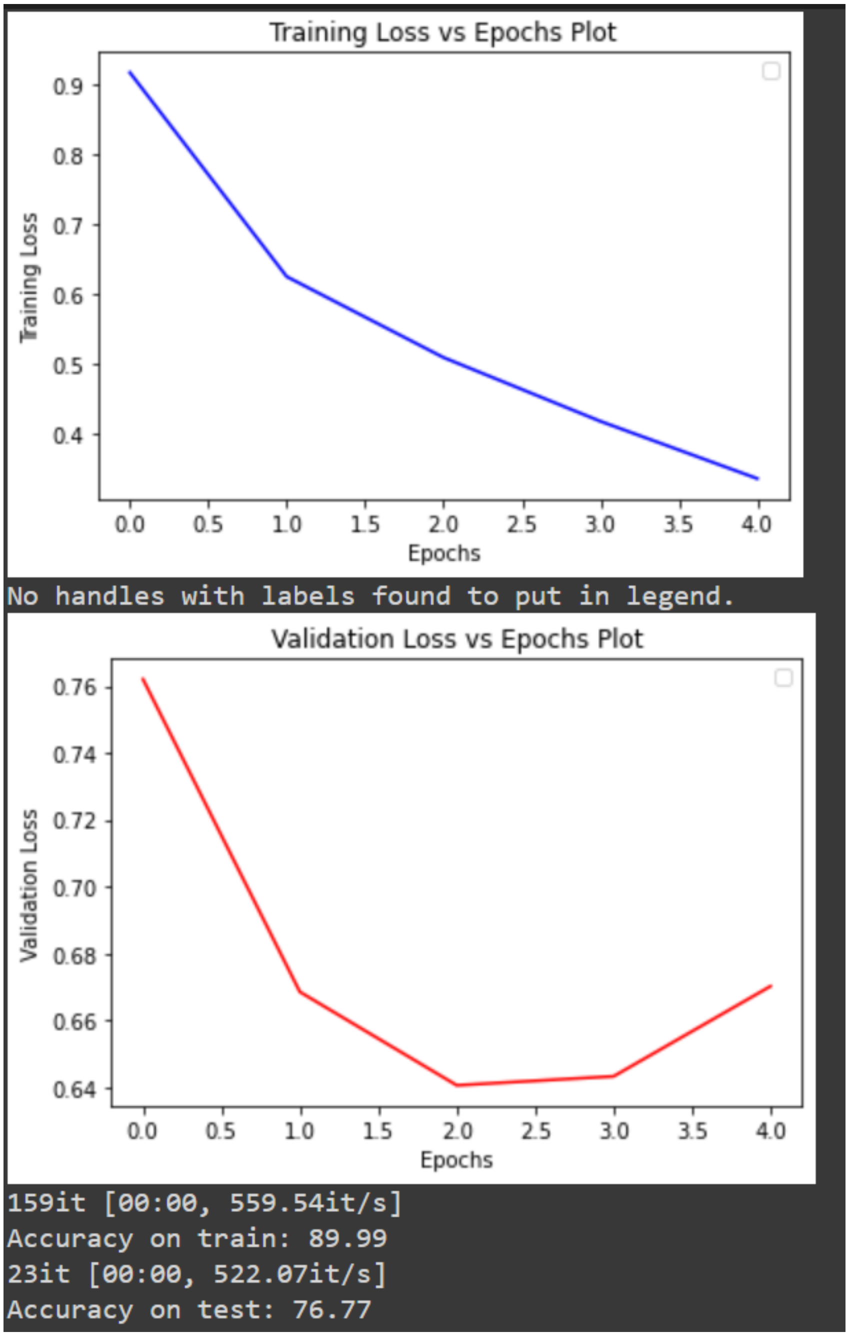
```
No handles with labels found to put in legend.
```

```
159it [00:00, 579.25it/s]
Accuracy on train: 91.94
23it [00:00, 445.37it/s]
Accuracy on test: 75.97
```



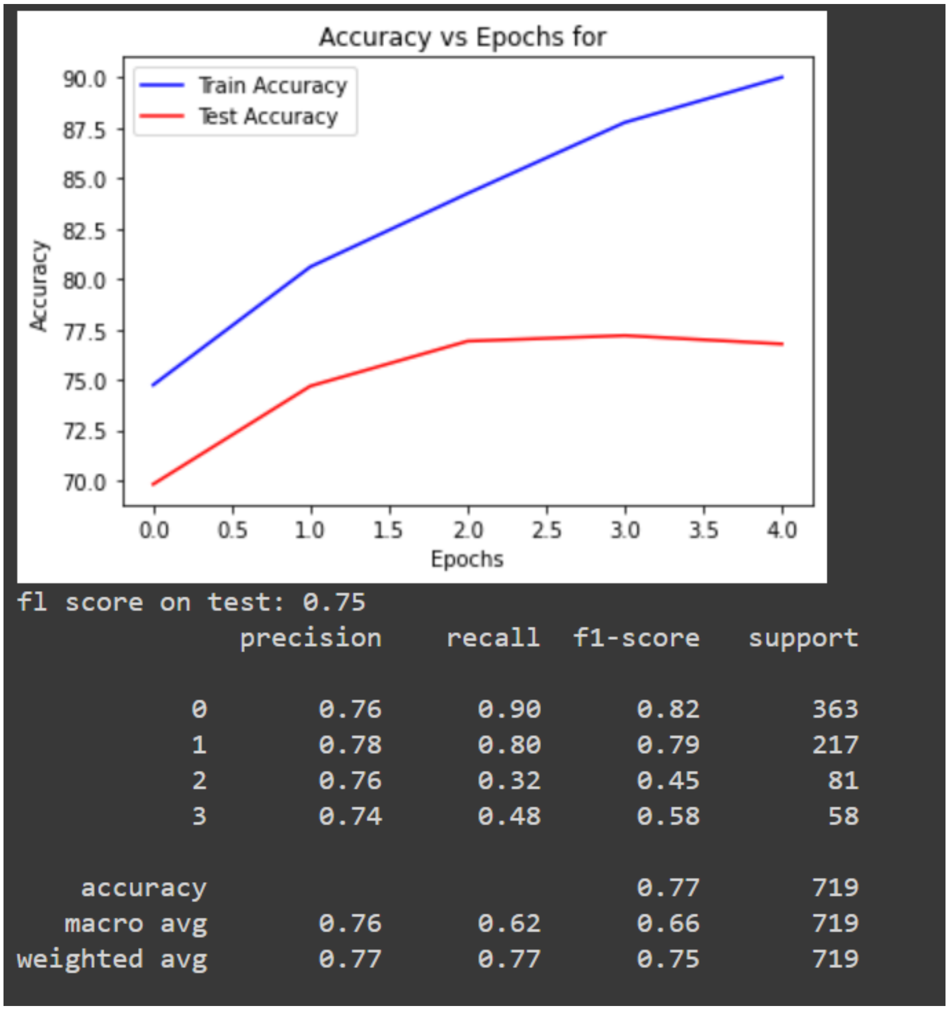
```
f1 score on test: 0.75
              precision    recall  f1-score   support

           0       0.81      0.83      0.82       363
           1       0.75      0.82      0.78       217
           2       0.60      0.46      0.52        82
           3       0.67      0.52      0.58        58

    accuracy                           0.76       720
   macro avg       0.71      0.66      0.68       720
weighted avg       0.75      0.76      0.75       720
```

X = 3



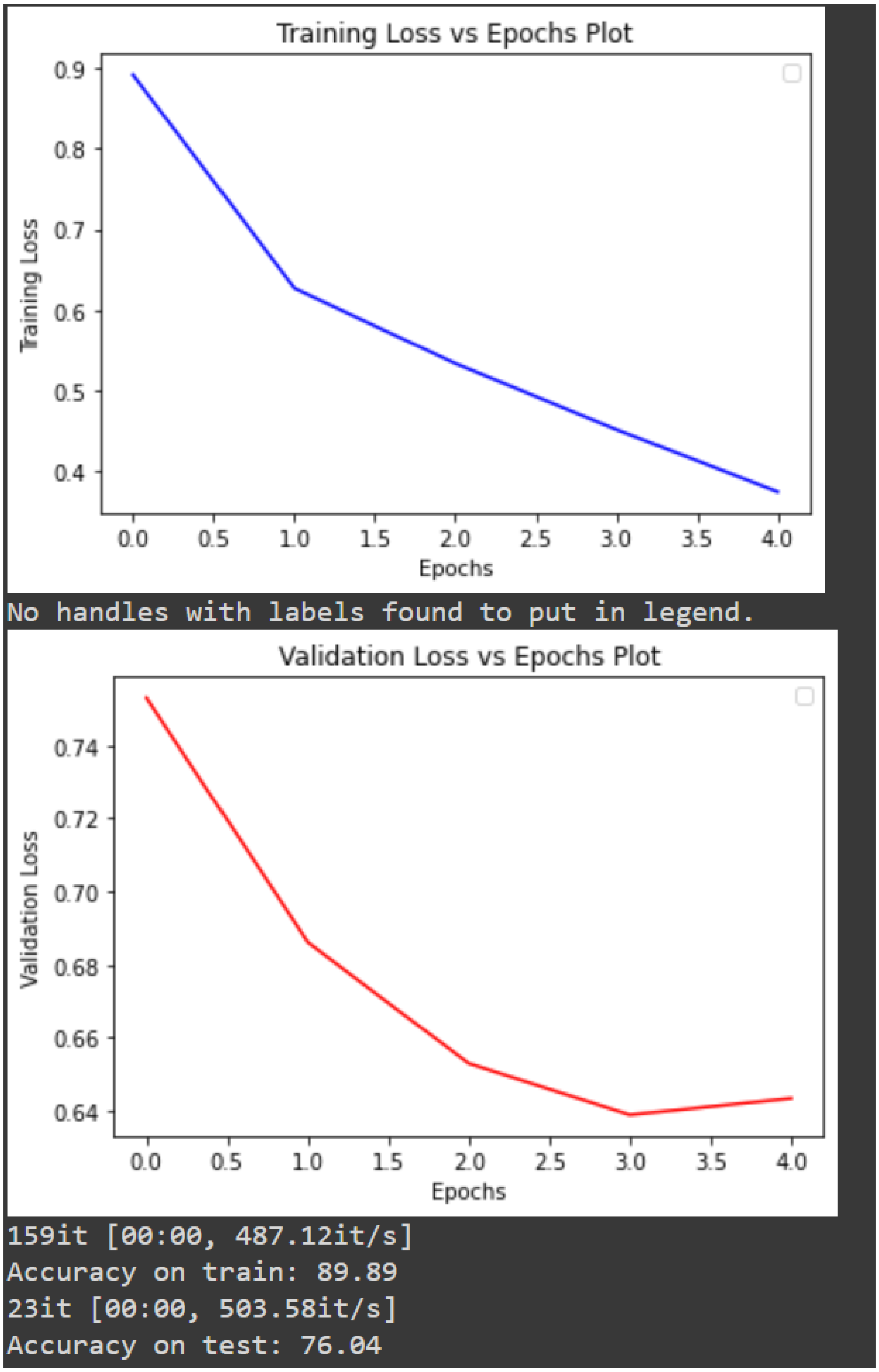
```
No handles with labels found to put in legend.
```

```
159it [00:00, 559.54it/s]
Accuracy on train: 89.99
23it [00:00, 522.07it/s]
Accuracy on test: 76.77
```



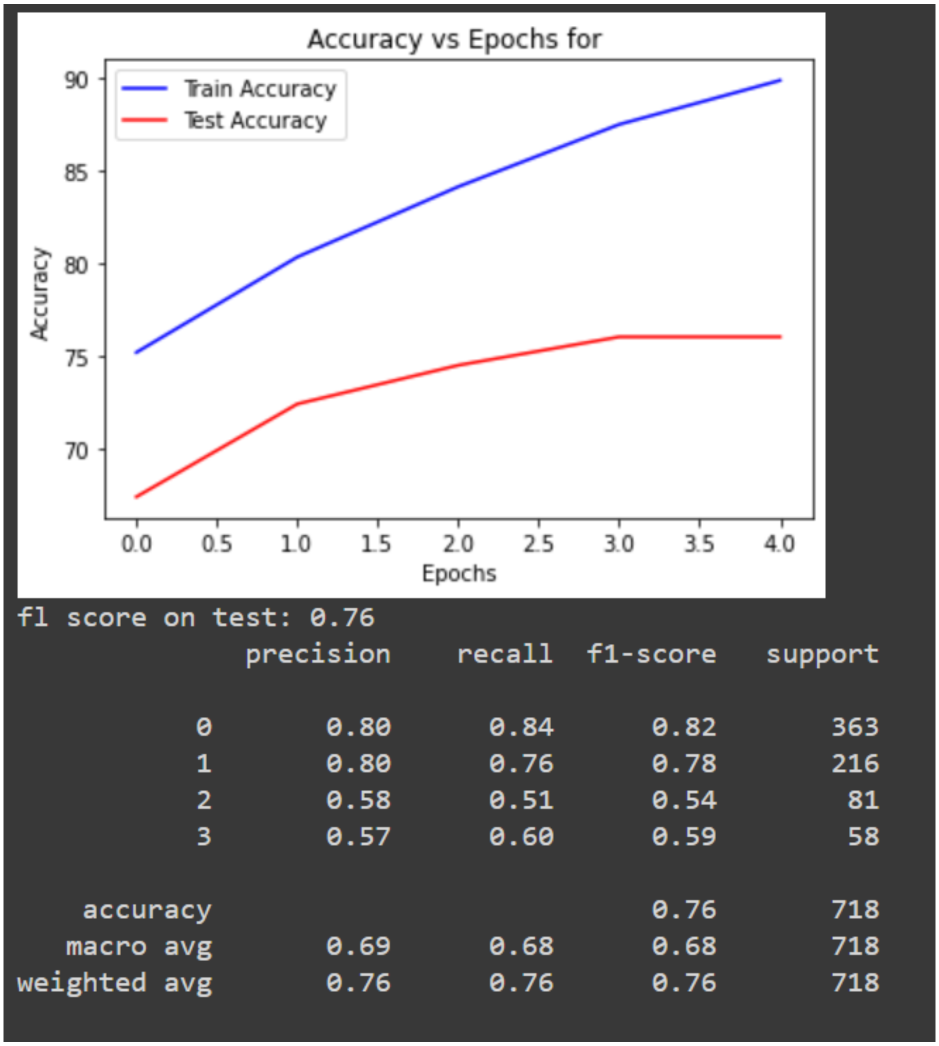
```
f1 score on test: 0.75
              precision    recall  f1-score   support

           0       0.76      0.90      0.82       363
           1       0.78      0.80      0.79       217
           2       0.76      0.32      0.45        81
           3       0.74      0.48      0.58        58

    accuracy                           0.77       719
   macro avg       0.76      0.62      0.66       719
weighted avg       0.77      0.77      0.75       719
```

X = 4



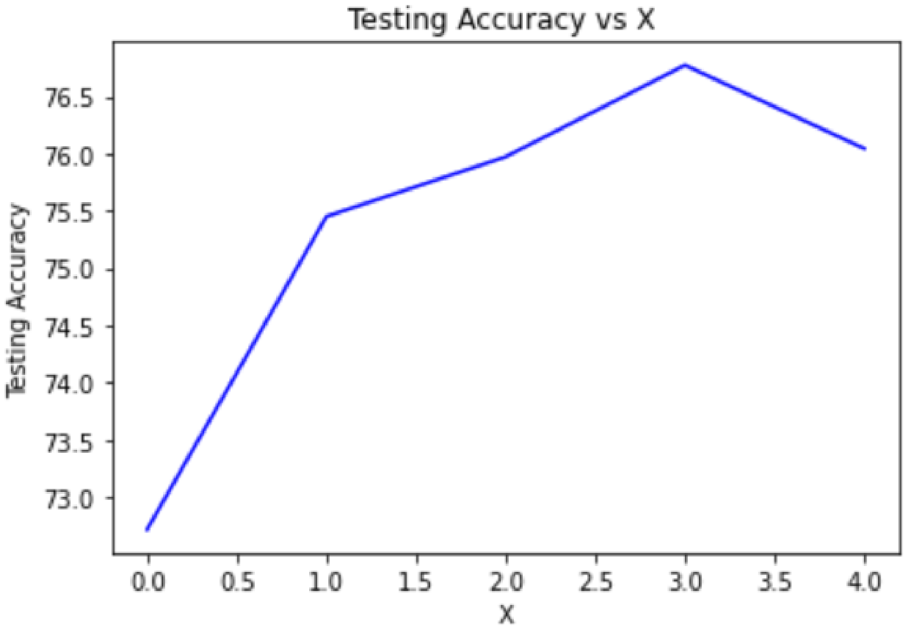
```
No handles with labels found to put in legend.
```

```
159it [00:00, 487.12it/s]
Accuracy on train: 89.89
23it [00:00, 503.58it/s]
Accuracy on test: 76.04
```

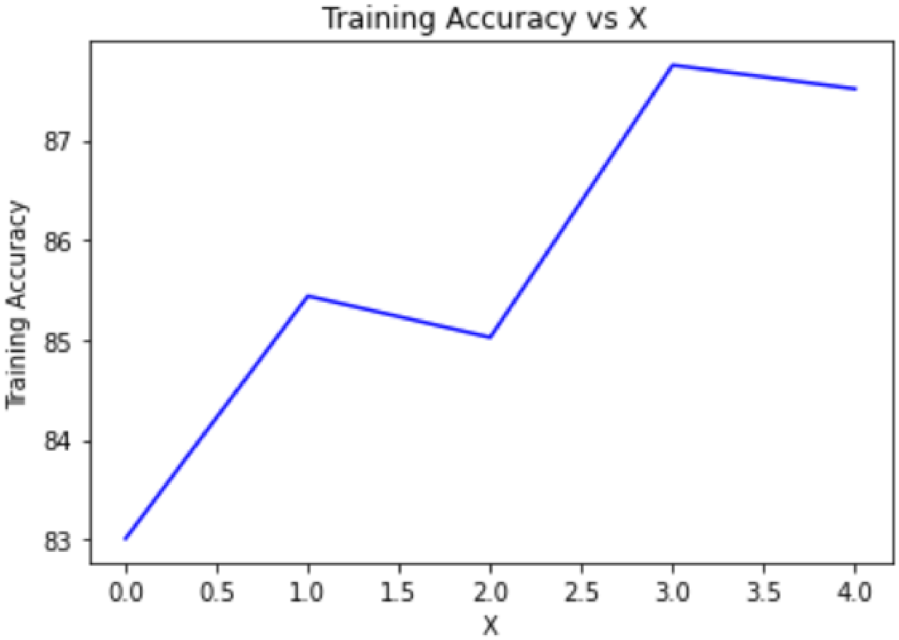


```
f1 score on test: 0.76
              precision    recall  f1-score   support

           0       0.80      0.84      0.82       363
           1       0.80      0.76      0.78       216
           2       0.58      0.51      0.54        81
           3       0.57      0.60      0.59        58

    accuracy                           0.76       718
   macro avg       0.69      0.68      0.68       718
weighted avg       0.76      0.76      0.76       718
```

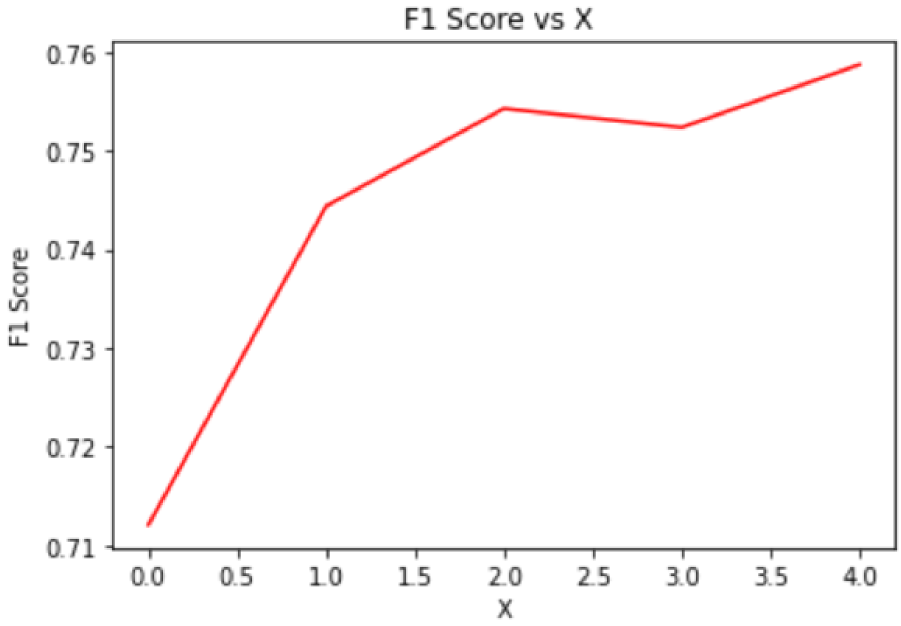
Test Accuracy vs X



Training Accuracy vs X



F1 Score vs X



4. Yes, the performance of the model increase with increase in X as shown in above plots of accuracy and f1 score vs X. As the number of utterances as context increases LSTM gets more previous information to decipher the class of the given sample. LSTM is a memory based model in which information from previous samples are passed on from first cell to last cell but it has the capacity to maintain a cell state which stores only relevant information, thus avoiding vanishing gradient problem. So as we increase the value of X, model gets more relevant information to train faster and more accurately. It can be seen that with X = 0 and 1 it is not able to converge completely in 5 epochs while incase of X = 3 and 4 it is able to learn faster as it has more context and thus converges in just 5 epochs.

How to run the file:

- Need to update the directory for reading dataset
- Need to update the directory wherever train embeddings are present.