CSE641 - Deep Learning

Assignment 2

- Abhimanyu Gupta (2019226)
- Meenal Gurbaxani (2019434)

Contribution

Abhimanyu Gupta - Part 1

Meenal Gurbaxani - Part 2

Part 1

Methodology:

➢ **Preprocessing**

- The input images were present in two folders namely ‘Parasitized’ and ‘Uninfected’ and so were read one by one and put in a common list, which was ultimately shuffled.
- The images were of varying sizes so each images irrespective of size was scaled to 32*32 to introduce uniformity.
- Also, the pixel values were in the range of 0 to 255, hence it was also normalised by reducing the range between 0 to 1.
- For better accessibility of images later and avoid repeating preprocessing again, the images after being converted to numpy array was saved in the folder using numpy.save()
- Sklearn’s train_test_split() was used to divide the data into train, val and test set.
- Finally, 5 values from each Parasitized and Uninfected were displayed through matplotlib for getting an idea of the data set.

➢ **Learning**

- A Model class extending torch.nn.Module was used to implement the required model.
- The connections of various layers was defined in the Model class forward function.
- Further, to have ease in accessing the dataset CellImageDataset class was implemented extending the torch.utils.data.Dataset after being wrapped around by torch.utils.data.DataLoader.
- Weight Initialisation of model was done through the torch.nn.Module.apply method to which functions taking a model argument was passed.

- For Regularisation, the loss was calculated through the user defined function ‘regularize’.
- The model’s criterion was set to be torch.nn.BCELoss as we have only 2 labels.
- To set up the above mentioned model, dataloader, criterion, etc make_model function was created to provide a high level interface.
- At the end, to train the model ‘train’ function was created which included the complete training cycle.
- On top of this a number of helper functions like evaluate, score, etc was created to aid the implementation wherever required.
- Further, class MyModel was implemented to pack everything into a single entity with easy accessibility.
- Through the training cycles, training and validation losses and accuracies were calculated for each epoch to have better insights into the training phase.
- In total ‘Epochs v/s Samples Seen’, ‘Epochs v/s Loss’ and ‘Epochs v/s Accuracy’ plots were generated.
- From the loss plots of training and validation data, it was evident that the learning keeps on getting better till 12 ± 2 epochs, after which their was signs of overfitting.

➢ **Saving Results**

- All the generated plots were saved with adequate name for future referencing.
- To store the learnt parameters pickle was used.
- The complete instance of MyModel was saved with a name havingl the information about model configuration.

Observation:

Running the model on several configurations, the following things were observed:

➔ It was observed that in absence of dropout and regularization, he initialization showed maximum performance, which can attributed to fact how carefully he initialization has been crafted. Further, bad performance of zero initialization is because all the weights are the same, so all the neurons try to do the same thing and hence are not able to learn anything meaningful collectively. Similarly, for random initialization the process of initialization is so random, because of which it is difficult to make anything sensible of it.

➔ In case of regularization, there seems to slight increase testing set accuracy over plain optimizer as much upto 1% in case of both L1 and L2. The reason for this can be inferred from Epochs v/s Loss graph for L1 and L2 where we can see the difference between training and validation loss is narrow (that is model does not overfit easily) as compared to no regularization case.

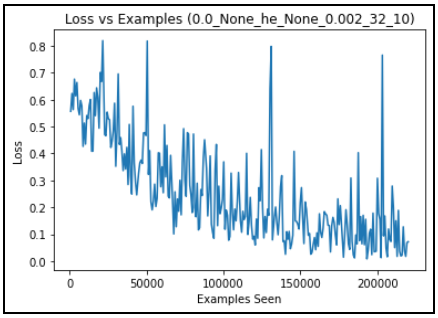
➔ Dropout between Convolution Layers seemed to perform better than Dropout after Fully Connected Layers. This might be happening as the information holded in input images is centered around a few pixels only for ‘Parasitized’; thus no significant information is lost when performing Dropout just after convolution layers.

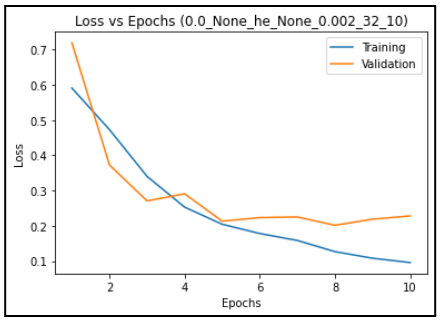
➔ Overall, as expected both Regularization and Dropout tend to perform better than simple model, with performance of both being more or less comparable.

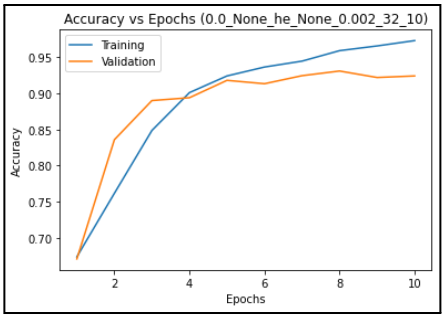
Outputs and Plots:

➢ **Model without Dropout and Regularization:**

- He Initialization

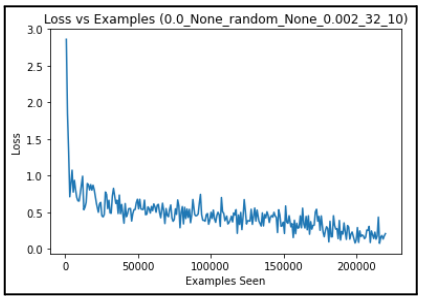


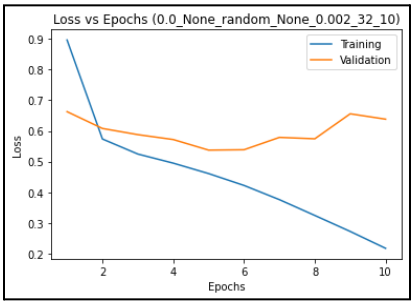


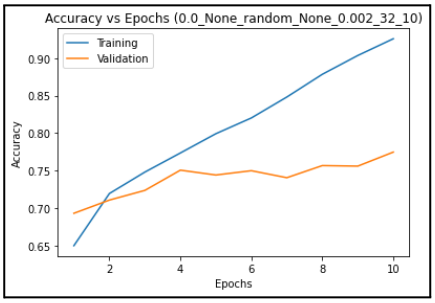



```
Testing Accuracy is 0.9245689655172413 with loss being 0.22885362033186288
```

- Random Initialization

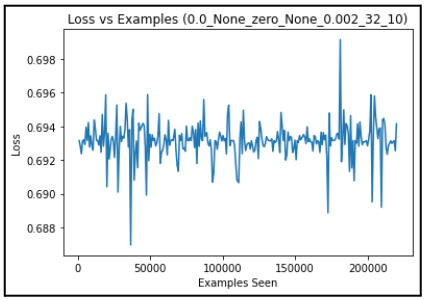


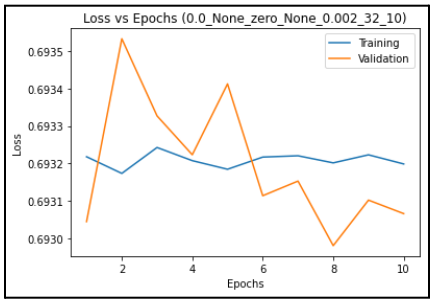


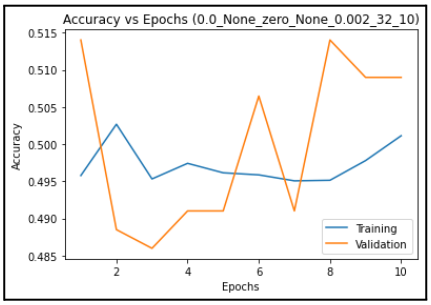



```
Testing Accuracy is 0.7704741379310345 with loss being 0.6931997277270788
```

- Zero Initialization



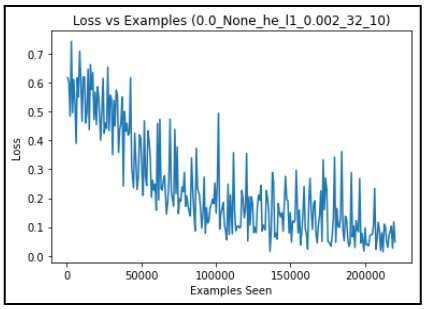


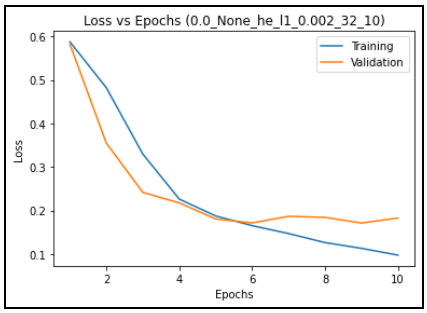


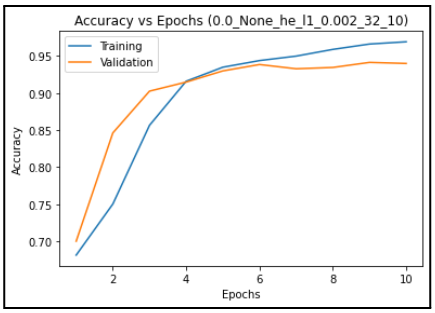

```
Testing Accuracy is 0.4885057471264368 with loss being 0.6932826425837374
```

➢ **Model with only Regularization:**

- L1 Regularization with He initialization

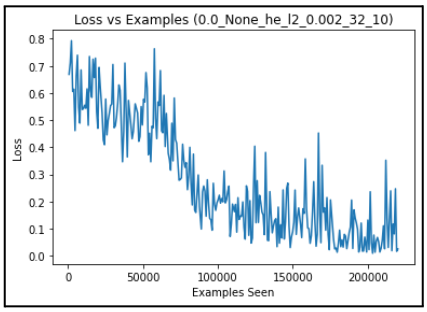


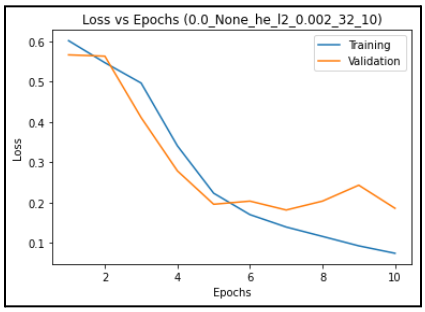


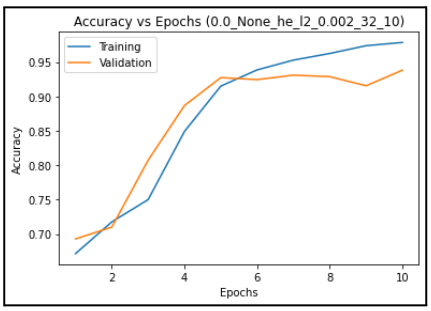



```
Testing Accuracy is 0.9371408045977011 with loss being 0.18802603359880118
```

- L2 Regularization with He initialization



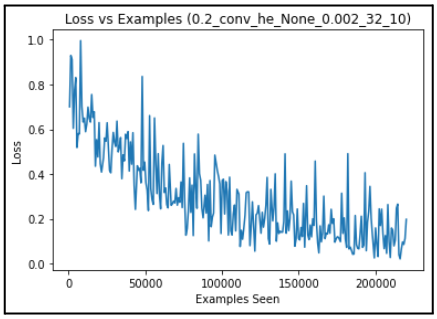


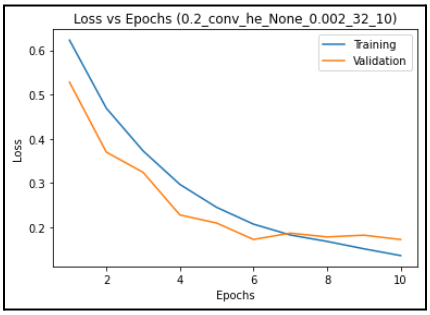


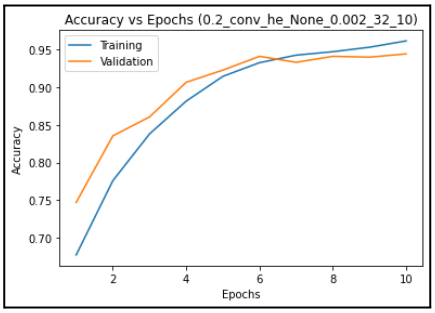

```
Testing Accuracy is 0.9324712643678161 with loss being 0.2061675937696435
```

➢ **Model with only Dropout**

- 0.2 Dropout between Convolution layers

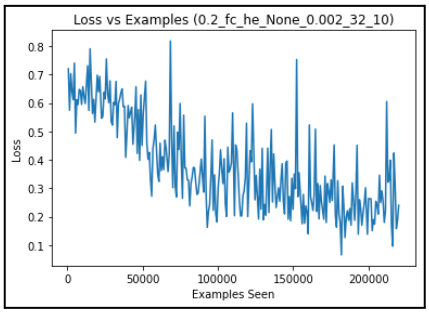


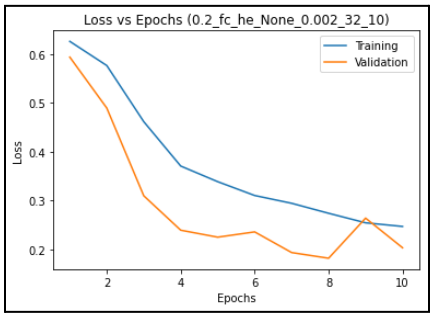


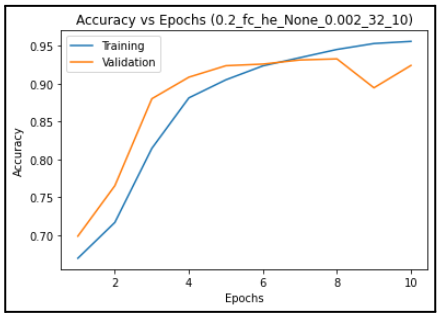



```
Testing Accuracy is 0.9407327586206896 with loss being 0.17139916036320833
```

- 0.2 Dropout after Fully Connected Layers

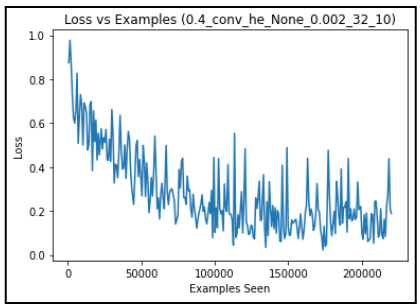


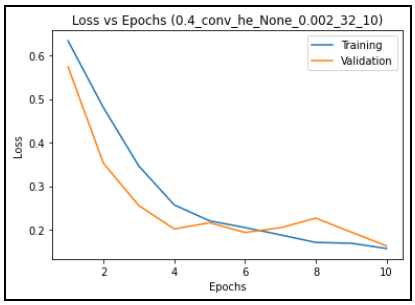


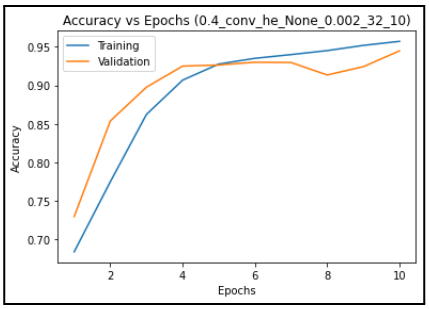



```
Testing Accuracy is 0.9267241379310345 with loss being 0.19565766433189655
```

- 0.4 Dropout between Convolution layers

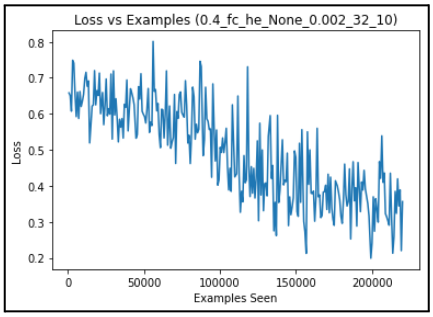


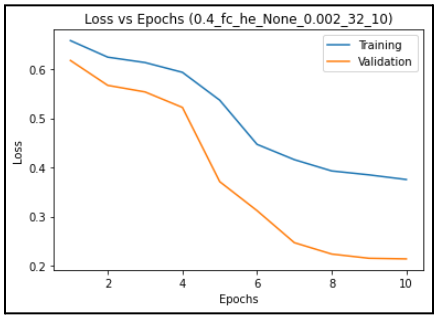


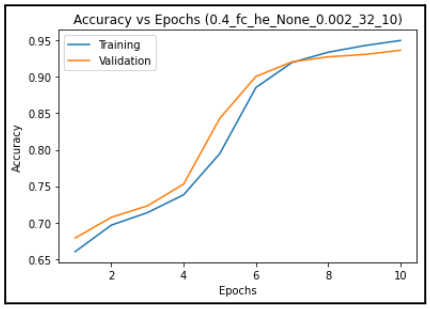



```
Testing Accuracy is 0.9392959770114943 with loss being 0.15834580344715338
```

- 0.4 Dropout after Fully Connected Layers

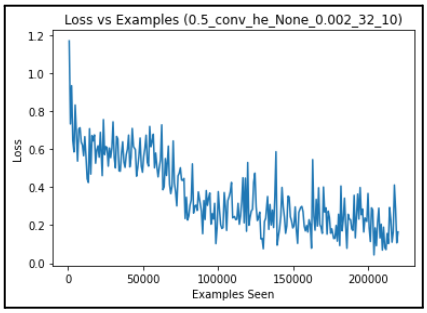


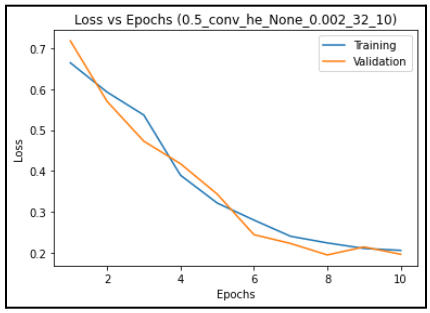


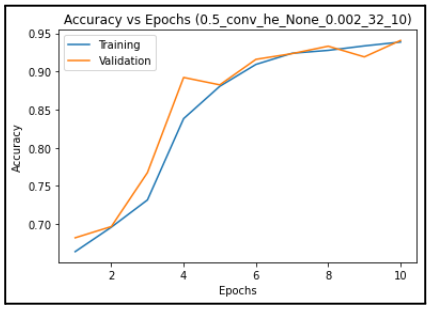



```
Testing Accuracy is 0.9375 with loss being 0.21532733961083422
```

- 0.5 Dropout between Convolution layers

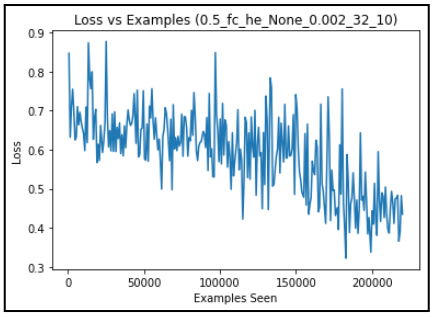


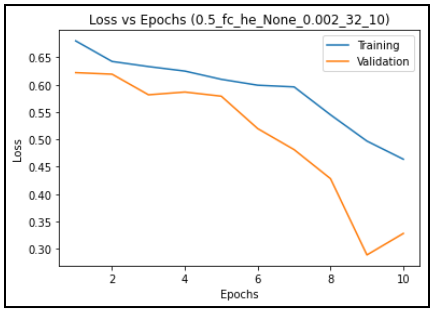


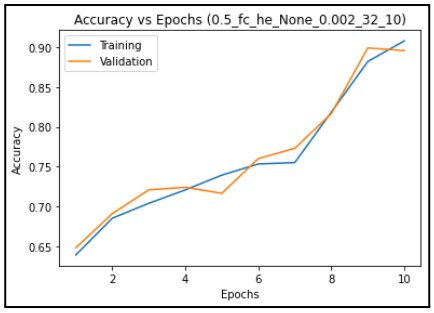



```
Testing Accuracy is 0.9335488505747126 with loss being 0.1928626531842111
```

- 0.5 Dropout after Fully Connected Layers








```
Testing Accuracy is 0.8951149425287356 with loss being 0.32266621205998564
```

Part 2

Part 1: EDA

1.1. Visualisation

```
[10] train_data.shape, test_data.shape
```

```
((5090, 2), (722, 2))
```

```
[11] train_data.head(10)
```

	utterance	act
0	Say , Jim , how about going for a few beers af...	3
1	You know that is tempting but is really not g...	4
2	What do you mean ? It will help us to relax .	2
3	Do you really think so ? I don't . It will ju...	2
4	I guess you are right.But what shall we do ? ...	2
5	I suggest a walk over to the gym where we can...	3
6	That's a good idea . I hear Mary and Sally of...	4
7	Sounds great to me ! If they are willing , we...	1
8	Good.Let ' s go now .	3
9	All right .	4

```
[12] test_data.sample(10)
```

	utterance	act

Q1)

```
[12] test_data.sample(10)
```

	utterance	act
566	Hey , Karen . Look like you got some sun this ...	1
541	It is ? Oh , that's nice .	2
480	Could you lend your bike to me for several day...	3
315	I'm ready .	4
270	When can we expect you for dinner ? Can you co...	3
139	I don't have an aspirin . Perhaps you should ...	3
670	On my first trip I went to Tokyo , and on my ...	1
349	Maybe he is trying to find a job . When he fi...	1
664	It's too late .	4
44	Hey , Ann . You don't have a pen , do you ?	3

```
[13] train_data.groupby("act").count()
```

act	utterance
1	2859
2	1497
3	426
4	308

```
[14] test_data.groupby("act").count()
```

5s completed at 11:51 PM

```
[14] test_data.groupby("act").count()
```

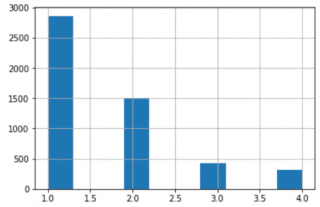
act	utterance
1	363
2	217
3	83
4	59

```
[15] train_data.shape, test_data.shape
```

```
((5090, 2), (722, 2))
```

```
[16] train_data["act"].hist()
```

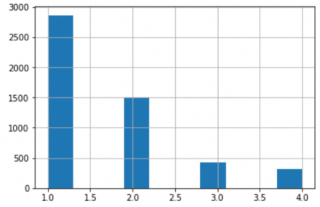
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f57276d6090>
```




```
[17] test_data.act.hist()
```

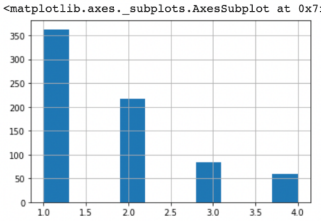

```
[16] train_data["act"].hist()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f57276d6090>
```


```
[17] test_data.act.hist()
```


```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5727191e90>
```

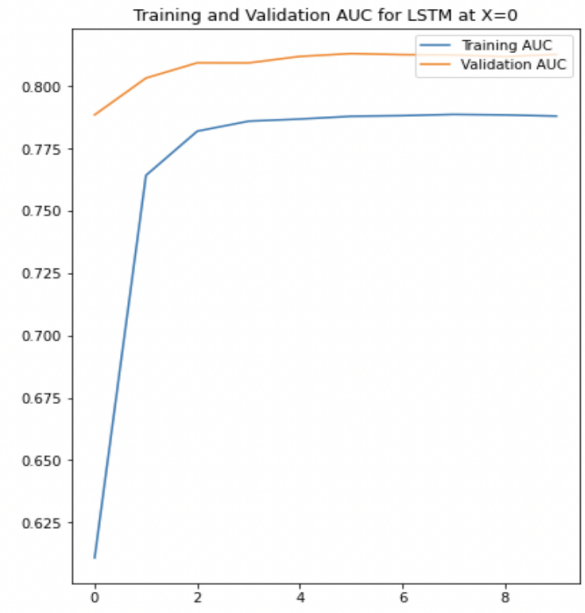


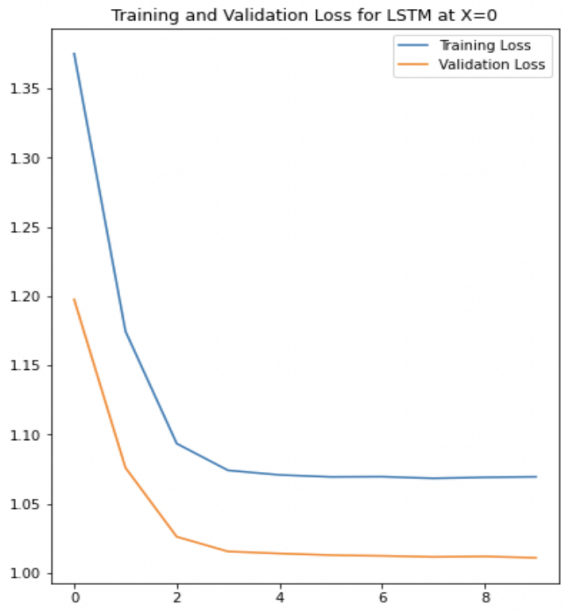
1.2. Data Cleaning

Q3)

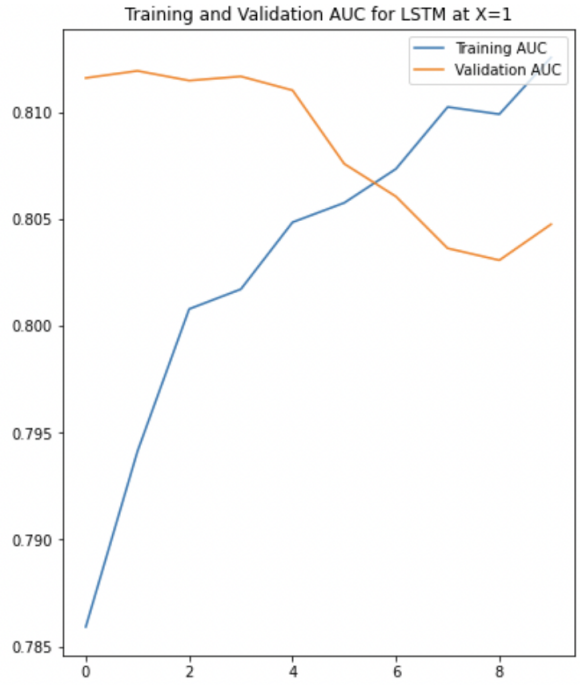
AOC curve and accuracy plotted in the form of Loss

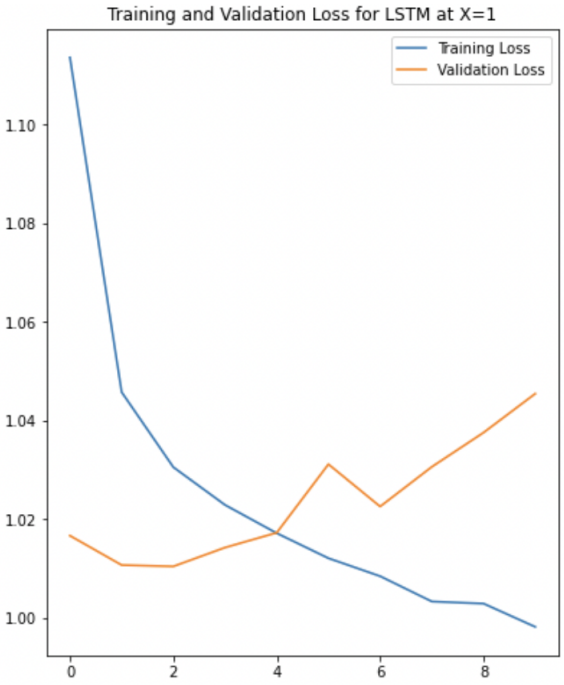
X=0



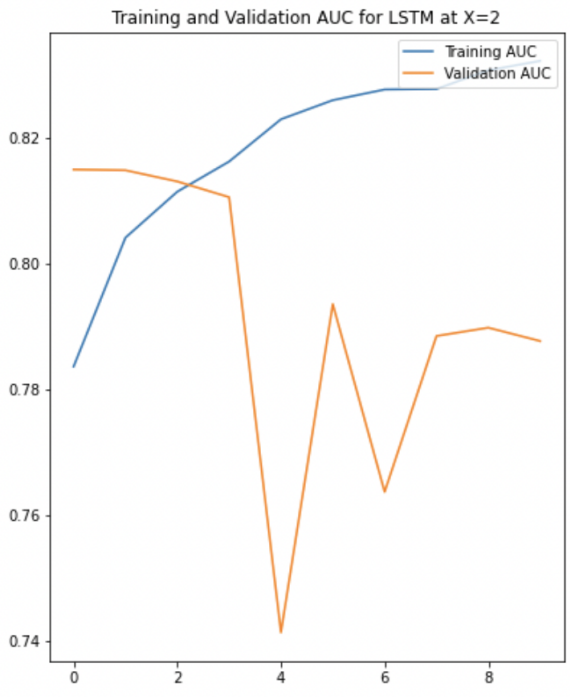


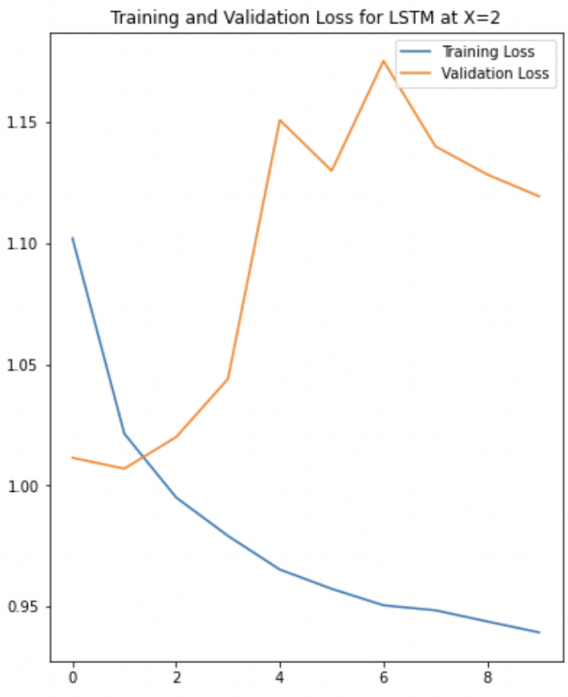
X=1



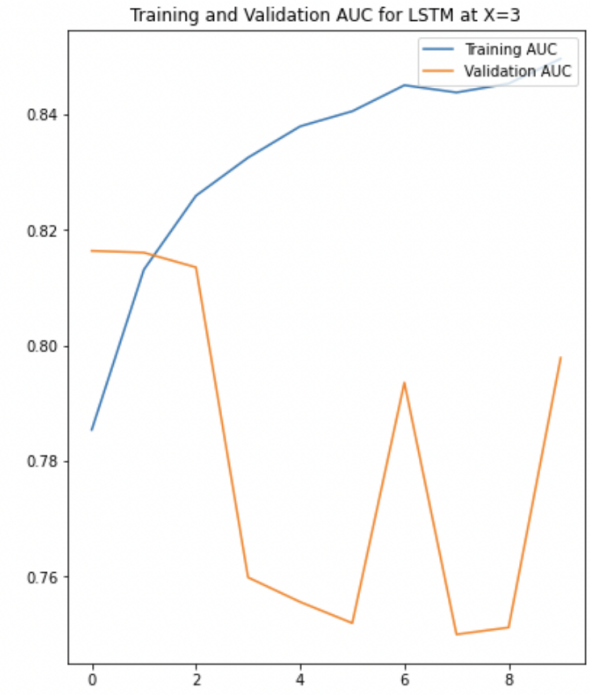


X=2


Training and Validation AUC for LSTM at X=2
Training AUC
Validation AUC
0.82
0.80
0.78
0.76
0.74
0
2
4
6
8

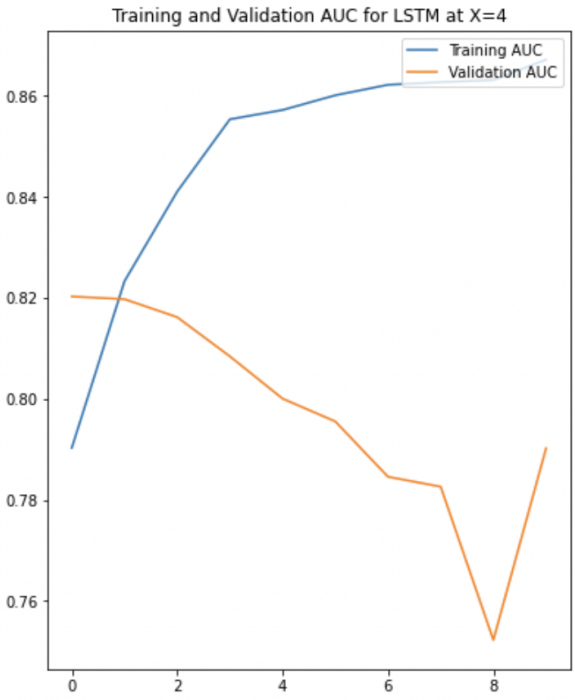

Training and Validation Loss for LSTM at X=2
Training Loss
Validation Loss
1.15
1.10
1.05
1.00
0.95
0
2
4
6
8

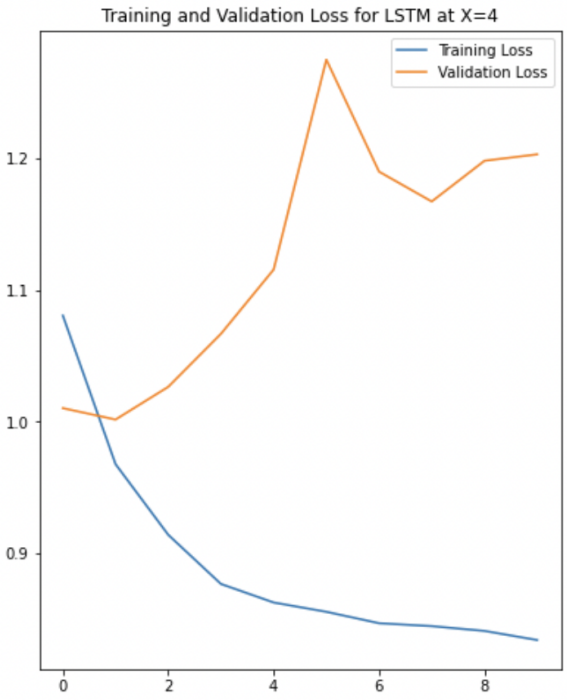
X=3


Training and Validation AUC for LSTM at X=3
Training AUC
Validation AUC
0.84
0.82
0.80
0.78
0.76
0
2
4
6
8


Training and Validation Loss for LSTM at X=3
Training Loss
Validation Loss
1.15
1.10
1.05
1.00
0.95
0.90
0
2
4
6
8

X=4





F1 classification reports

X=0

Training Evaluation

	precision	recall	f1-score	support
1	0.56	1.00	0.72	2859
2	0.00	0.00	0.00	1497
3	0.00	0.00	0.00	426
4	0.00	0.00	0.00	308
accuracy			0.56	5090
macro avg	0.14	0.25	0.18	5090
weighted avg	0.32	0.56	0.40	5090

Testing Evaluation

	precision	recall	f1-score	support
1	0.50	1.00	0.67	363
2	0.00	0.00	0.00	217
3	0.00	0.00	0.00	83
4	0.00	0.00	0.00	59
accuracy			0.50	722
macro avg	0.13	0.25	0.17	722
weighted avg	0.25	0.50	0.34	722

X=1

Training Evaluation

	precision	recall	f1-score	support
1	0.57	0.99	0.72	2859
2	0.74	0.02	0.05	1497
3	0.58	0.10	0.18	426
4	0.00	0.00	0.00	308
accuracy			0.57	5090
macro avg	0.47	0.28	0.24	5090
weighted avg	0.59	0.57	0.43	5090

Testing Evaluation

	precision	recall	f1-score	support
1	0.50	0.99	0.66	363
2	0.00	0.00	0.00	217
3	0.00	0.00	0.00	83
4	0.00	0.00	0.00	59
accuracy			0.50	722
macro avg	0.12	0.25	0.17	722
weighted avg	0.25	0.50	0.33	722

X=2

Training Evaluation

	precision	recall	f1-score	support
1	0.58	0.99	0.73	2859
2	0.90	0.05	0.09	1497
3	0.73	0.19	0.30	426
4	0.85	0.07	0.13	308
accuracy			0.59	5090
macro avg	0.76	0.33	0.31	5090
weighted avg	0.70	0.59	0.47	5090

Testing Evaluation

	precision	recall	f1-score	support
1	0.50	0.96	0.66	363
2	0.43	0.01	0.03	217
3	0.14	0.02	0.04	83
4	0.00	0.00	0.00	59
accuracy			0.49	722
macro avg	0.27	0.25	0.18	722
weighted avg	0.40	0.49	0.34	722

X=3

Training Evaluation

	precision	recall	f1-score	support
1	0.59	0.99	0.74	2859
2	0.78	0.08	0.14	1497
3	0.83	0.24	0.37	426
4	0.93	0.08	0.15	308
accuracy			0.60	5090
macro avg	0.78	0.35	0.35	5090
weighted avg	0.68	0.60	0.49	5090

Testing Evaluation

	precision	recall	f1-score	support
1	0.50	0.94	0.65	363
2	0.31	0.02	0.04	217
3	0.07	0.01	0.02	83
4	0.00	0.00	0.00	59
accuracy			0.48	722
macro avg	0.22	0.24	0.18	722
weighted avg	0.35	0.48	0.34	722

X=4

Training Evaluation

	precision	recall	f1-score	support
1	0.60	0.98	0.74	2859
2	0.87	0.09	0.17	1497
3	0.77	0.31	0.45	426
4	0.85	0.11	0.20	308
accuracy			0.61	5090
macro avg	0.77	0.38	0.39	5090
weighted avg	0.71	0.61	0.52	5090

Testing Evaluation

	precision	recall	f1-score	support
1	0.49	0.93	0.64	363
2	0.00	0.00	0.00	217
3	0.13	0.04	0.06	83
4	0.00	0.00	0.00	59
accuracy			0.47	722
macro avg	0.16	0.24	0.18	722
weighted avg	0.26	0.47	0.33	722

Q4)

- We see that the model performance does not increase linearly with X.
- Beyond X=2, the model starts overfitting.
- This may be because of use of only Linear Layers in LSTM.
- To increase accuracy, we need to add Bidirectional LSTM to increase model context for learning.