CSE641 - Deep Learning

Assignment 3

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

Methodology:

➢ **Preprocessing and Visualisation**

- The input definitions were present in a text file from where they were read into a pandas dataframe.
- Each of the train, val, test sets were present in separate txt, so were separately read into their respective dataframes.
- Unrequired columns like ‘Entity’, ‘Four-class labels’ were removed.
- Some values from the train/val/test set were displayed through matplotlib for getting an idea of the data set.
- The average lengths of sentences and average word count were calculated for each of the three sets to get an idea about the dataset composition.
- Further, cosine similarity for the training set was computed to check the semantics similarity between the pair of paraphrased definitions and pair of un-paraphrased definitions.

➢ **Learning**

- A ‘MyModel’ 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 MyModelDataset class was implemented extending the torch.utils.data.Dataset after being wrapped around by torch.utils.data.DataLoader.
- Weight Initialisation [He initialisation] of the model was done through the torch.nn.Module.apply the method to which functions taking a model argument was passed.
- 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, the class ‘MyModelClass’ 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 Loss’ and ‘Epochs v/s Accuracy’ plots were generated.
- For the testing set, additional metrics like Precision, Recall and F1 Score were also computed on top of Accuracy.
- The architecture of the model is described as followed: A starting embedding layer to embed the input data, it was then followed by an attention module, followed by a bidirectional lstm and ultimately sigmoid in the last layer.

➢ **Saving Results**

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

Observation:

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

➔ The model performed slightly better when all layers were frozen in comparison to when some layers were frozen.

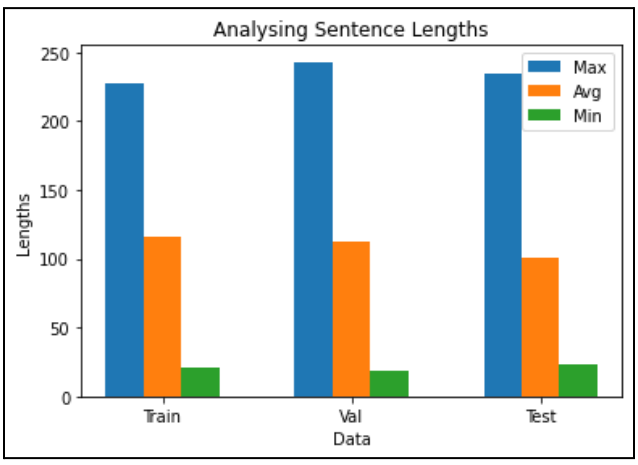
➔ Further, it can be seen that model on the same parameters with no layers frozen performed slightly poorer than some/all layers frozen.

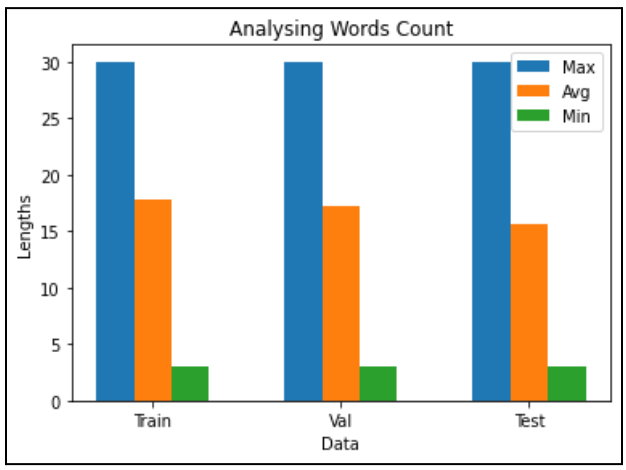
➔ In case of different combinations of learning rate (0.001, 0.01, 0.1) and SGD/Adam, Adam appeared to perform better for all values of learning rate.

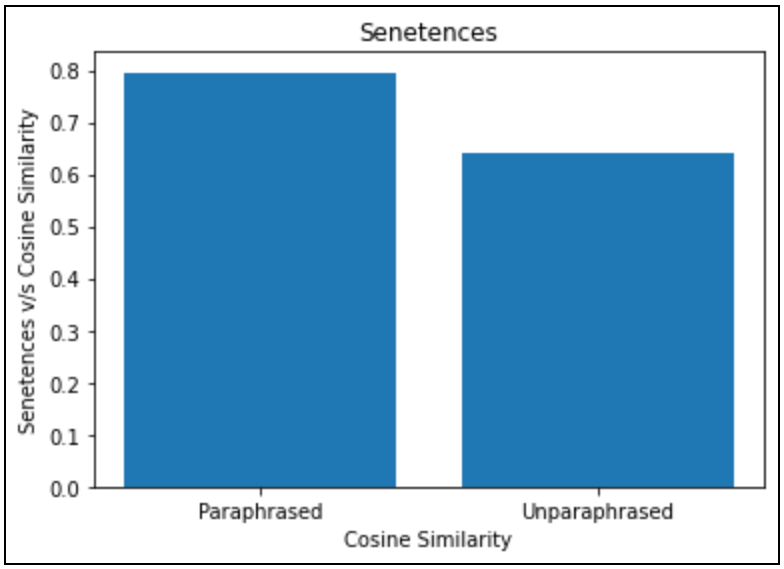
➔ Further, with increased learning rate, model tend to perform slightly poorer.

Outputs and Plots:

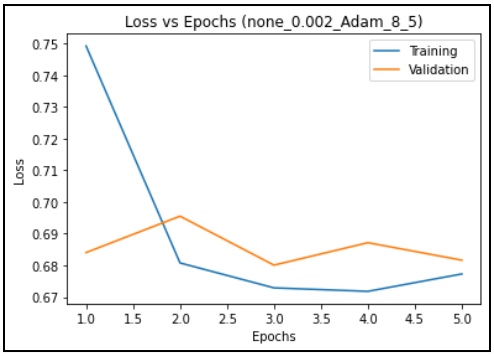
➢ **Visualisation:**

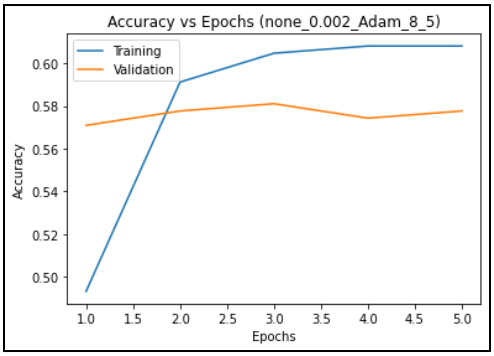






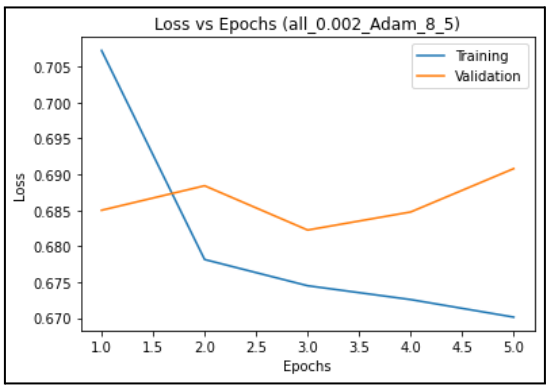
➢ Model with no freezing and Adam optimizer

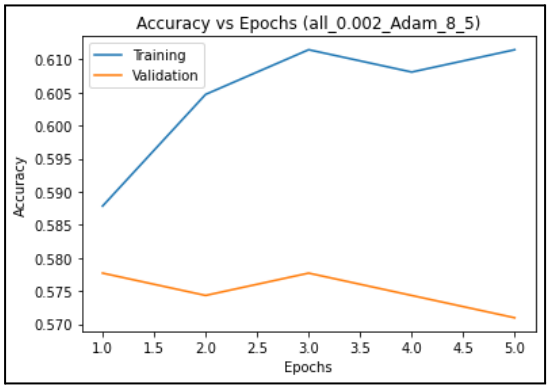





```
For Testing:
Accuracy is 0.5472972972972973
Precision is 0.5472972972972973
Recall is 0.5472972972972973
F1 Score is 0.5472972972972973
```

➢ Model with all layers frrozen

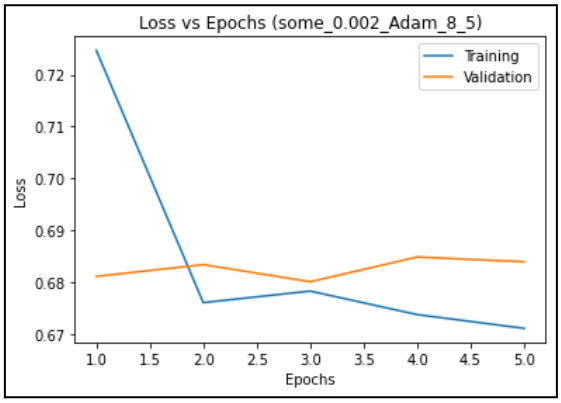


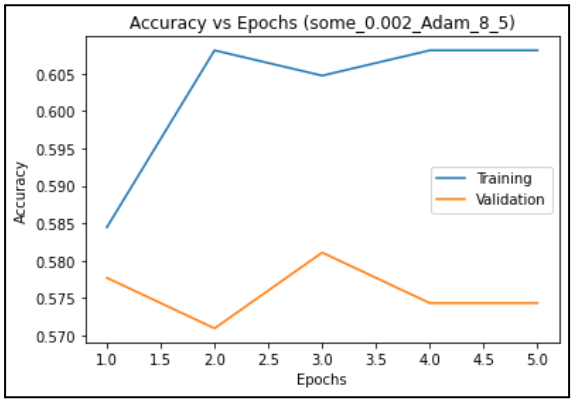



```
For Testing:
Accuracy is 0.5506756756756757
Precision is 0.5506756756756757
```

```
Recall is 0.5506756756756757
F1 Score is 0.5506756756756757
```

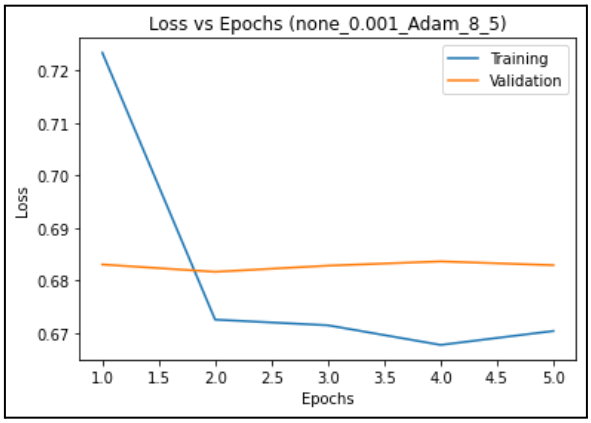
➢ **Model with some layers frozen**

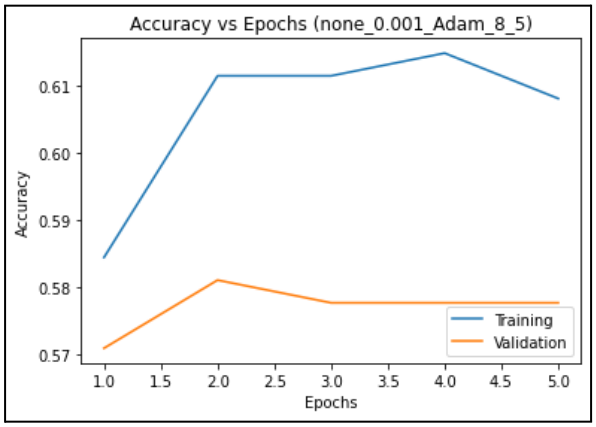





```
For Testing:
Accuracy is 0.5540540540540541
Precision is 0.5540540540540541
Recall is 0.5540540540540541
F1 Score is 0.5540540540540541
```

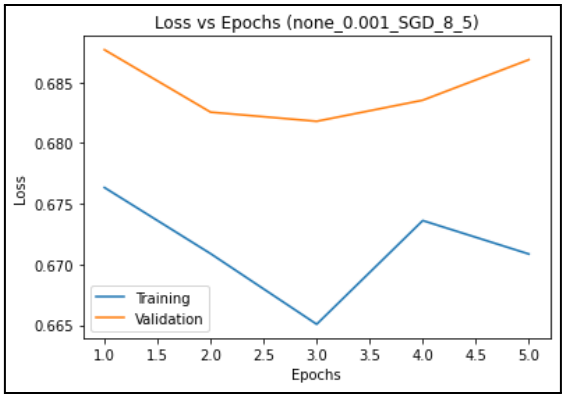
➢ **Model with learning rate 1e-3 and Adam optimizer**

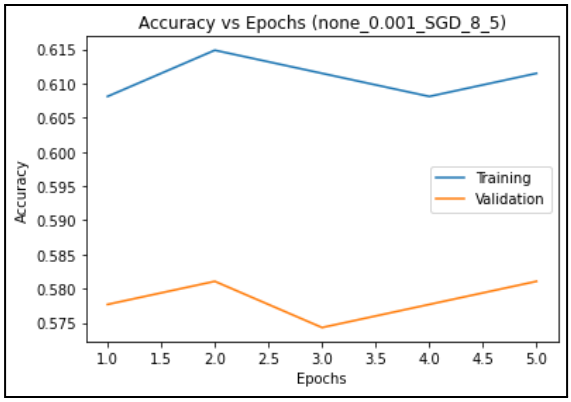





```
For Testing:
Accuracy is 0.5574324324324325
Precision is 0.5574324324324325
Recall is 0.5574324324324325
F1 Score is 0.5574324324324325
```

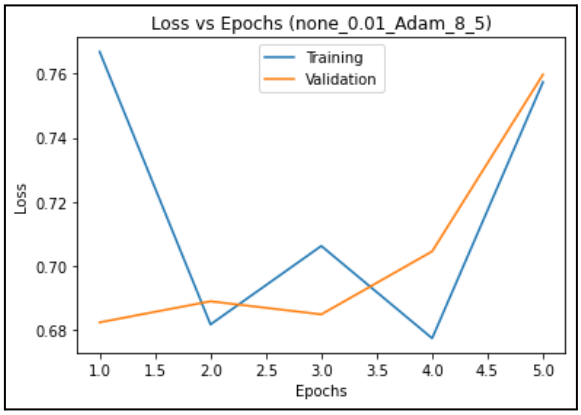
➢ **Model with learning rate 1e-3 and SGD optimizer**

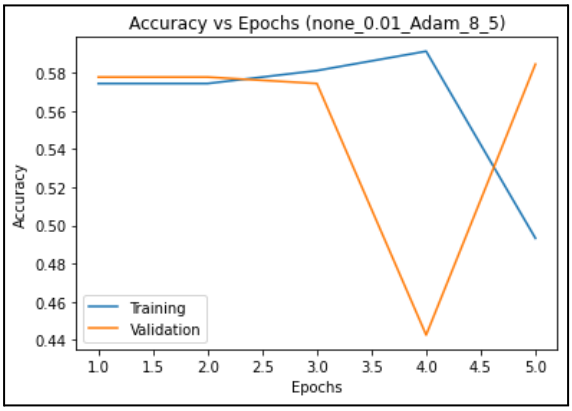





```
For Testing:
Accuracy is 0.5506756756756757
Precision is 0.5506756756756757
Recall is 0.5506756756756757
F1 Score is 0.5506756756756757
```

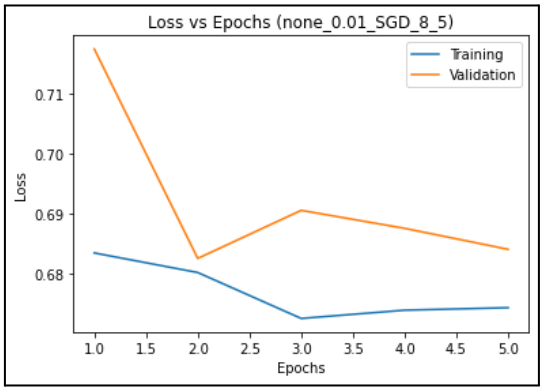
➢ **Model with learning rate 1e-2 and Adam optimizer**

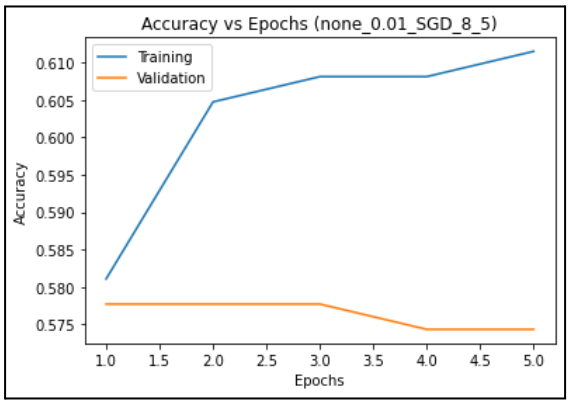





```
For Testing:
Accuracy is 0.5540540540540541
Precision is 0.5540540540540541
Recall is 0.5540540540540541
F1 Score is 0.5540540540540541
```

➢ **Model with learning rate 1e-2 and SGD optimizer**

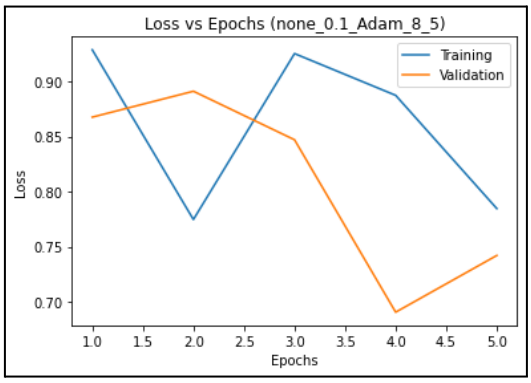


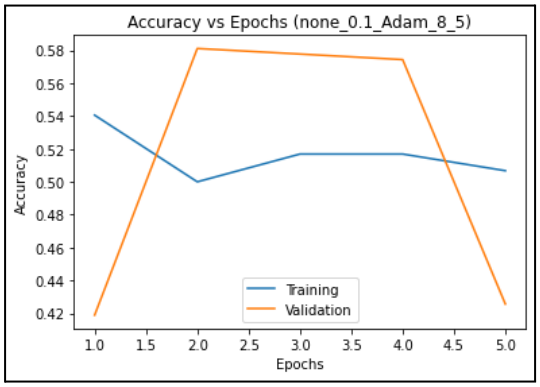



```
For Testing:
```

```
Accuracy is 0.5574324324324325
Precision is 0.5574324324324325
Recall is 0.5574324324324325
F1 Score is 0.5574324324324325
```

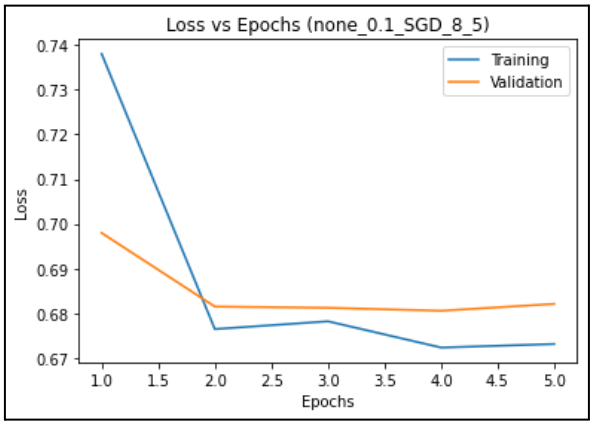
➢ **Model with learning rate 1e-1 and Adam optimizer**

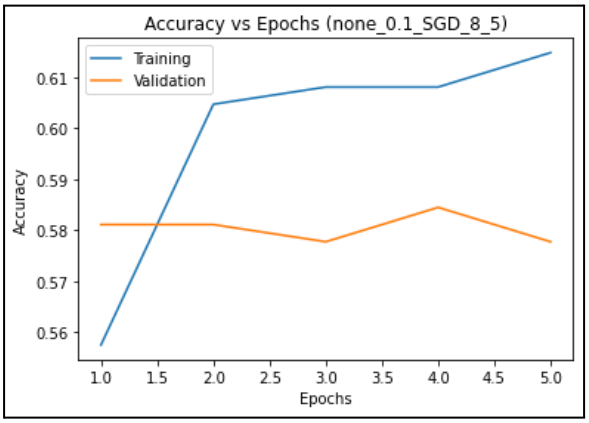





```
For Testing:
Accuracy is 0.44594594594594594
Precision is 0.44594594594594594
Recall is 0.44594594594594594
F1 Score is 0.44594594594594594
```

➢ **Model with learning rate 1e-1 and SGD optimizer**






```
For Testing:
Accuracy is 0.5540540540540541
Precision is 0.5540540540540541
Recall is 0.5540540540540541
F1 Score is 0.5540540540540541
```