Name: Sean Goh Ann Ray
Matric No.: G2202190G

Task 2:

Using the IMDB dataset with a train/val split from the original train data, there is now 17500 training data, 7500 validation data, and 25000 test data. The RNN model provided, which used randomly initialized embedding weights before the RNN and fully connected layer, was used to run the experiments, with the difference in the optimizer, which are SGD, Adam, and Adagrad. The common parameters were set at learning rate = 0.001 and epochs = 20, while the optimizer-specific parameters were their default values. The models were trained using the training data and evaluated using the validation data, with the loss and accuracy curves shown in Figures 1 and 2 respectively. Each figure shows 6 plots, a solid line for training and a dashed line for validation for each optimizer.

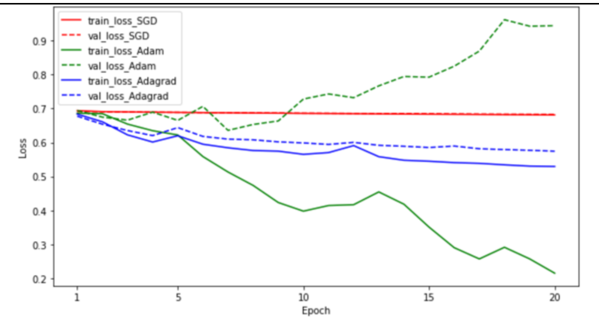

Figure 1. Loss vs Epoch

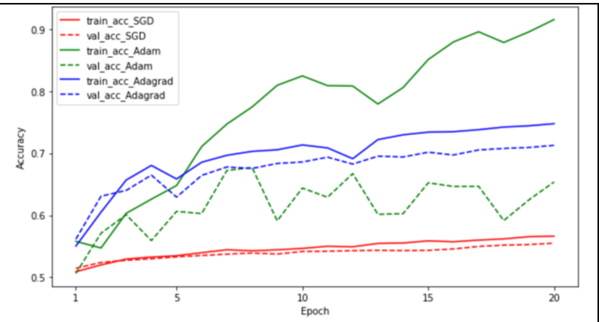

Figure 2. Accuracy vs Epoch

The trained models were then evaluated using the test dataset and the result is as follows.

Table I. Test Loss & Accuracy of Models using Different Optimizers

Optimizer	SGD	Adam	Adagrad
Test Loss	0.684	0.653	0.572
Test Accuracy (%)	55.00	66.39	71.15

As seen in Figures 1 and 2, the model with SGD optimizer improved the least over the training of 20 epochs. This is largely contributed by the small learning rate of 0.001 which results in a very small change in the weights of the embeddings. It also performed the worst on the test dataset with 55% accuracy, but this can be due to the small learning rate.

The model with Adam optimizer improved the most, but 'instabilities' can be observed as the accuracy decreased and loss increased at certain epoch steps. It can also be observed that the validation loss started to increase after epoch 7, indicating that the model is starting to overfit to the training data. This is further supported by the test accuracy of only 66.39%, placing it at 2nd place among the 3 models.

The model with Adagrad optimizer performed between the 2 earlier models discussed in terms of the training. However, this model performed the best when looking at the test accuracy of 71.15%. Since it also has the best validation accuracy among the 3 models discussed, it can be said that the Adagrad optimizer is best suited for this data.

Task 3:

The next few experiments are performed using the same training and validation data, Adam optimizer set at learning rate = 0.001, but with different number of epochs = 5, 10, 20, and 50. Figures 3 and 4 shows the loss and accuracy curves respectively. Each figure shows 8 plots, a solid line for training and a dashed line for validation for each epoch value.

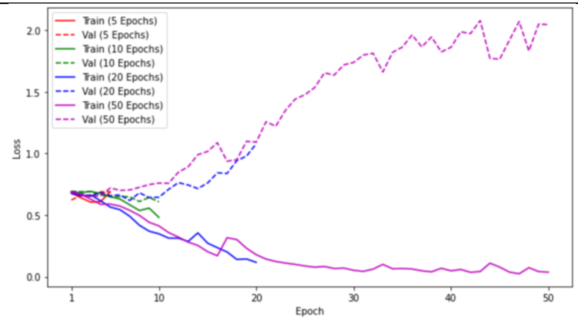

Figure 3. Loss vs Epoch

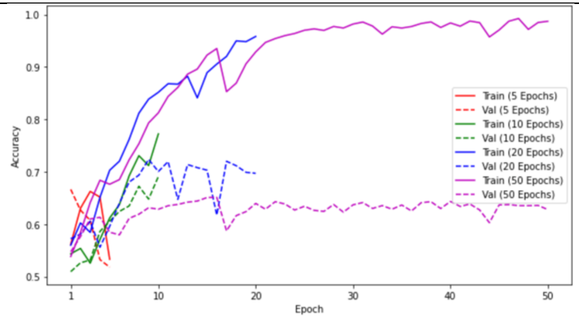

Figure 4. Accuracy vs Epoch

The trained models were then evaluated using the test dataset and the result is as follows.

Table II. Training Duration, Test Loss and Accuracy of Models with Different Number of Epoch

No. of Epochs	5	10	20	50
Training Duration (s)	22.41	53.57	102.65	224.71
Test Loss	0.631	0.614	0.618	0.652
Test Accuracy (%)	65.24	68.30	68.87	60.91

As seen in Figure 3, a larger epoch number would generally result in better (smaller) training loss, but validation loss starts to increase after 10 epochs, which hints at overfitting of the training data. In Figure 4, both training and validation accuracies tend to improve more as epoch number increases, but validation accuracy tend to stagnate after 10 epochs.

Additionally, from Table II, the model that achieves the best test accuracy is the model trained with 20 epochs, but if taking the training duration and validation loss into consideration, the model with 10 epochs may provide the best balance between performance and training duration as it also avoids overfitting of the training data.

Task 4:

The next experiments were conducted using Adam optimizer set at learning rate = 0.001 and epochs = 50, while using the pretrained Word2Vec embeddings as the initialization of the model. Figures 5 and 6 shows the loss and accuracy curves respectively, with each figure also showing the model performance of the randomly initialized weights from Task 3.

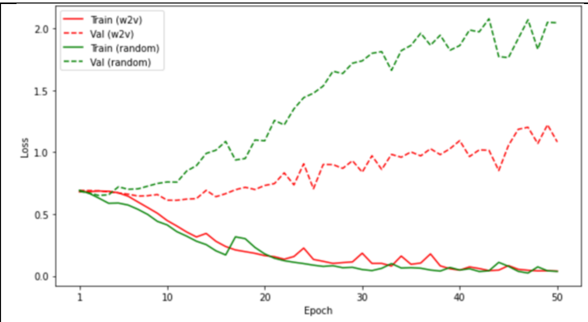

Figure 5. Loss vs Epoch

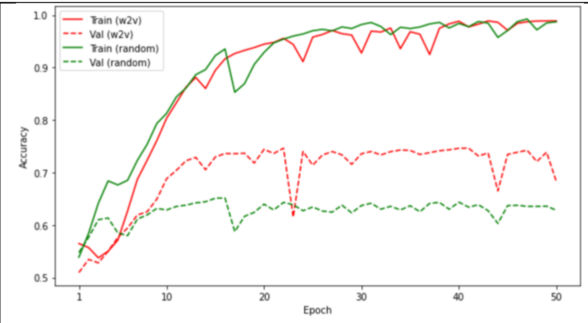

Figure 6. Accuracy vs Epoch

The trained model with the pretrained Word2Vec embeddings was evaluated using the test dataset and the result is shown in Table III, alongside the result for the model with the randomly initialized embeddings from Task 3.

Table III. Test Loss and Accuracy of Model with Different Embedding Weights

Embedding Weights	Pretrained Word2Vec	Random
Test Loss	0.534	0.652
Test Accuracy (%)	76.72	60.91

As seen in Figure 5, both models have comparable training losses, but the model with the pretrained Word2Vec embedding weights has a better (smaller) validation loss, although both the training and validation losses increase after epoch 10. In Figure 6, both models have comparable training accuracies, but the model with the pretrained Word2Vec embedding weights has a better (higher) validation accuracy.

In Table III, the model which performs better on the test dataset is clearly the one with the pretrained Word2Vec embedding, with smaller loss and higher accuracy which is comparable to their respective validation counterpart.

This is likely due to the pretrained Word2Vec embedding which have weights that represent the meaning of each word in the vocabulary, and hence require less changes than that of randomly initialized embedding. This can also be observed from the smaller changes at the start of the training process. In Figure 5, the red curves are almost flat up till epoch 5, while the green curves already have a slope to them. In Figure 6, the red curves also have smaller gradients than the green counterparts up till epoch 5.

Task 5:

The final experiments were conducted using 6 different models (1-layer feed forward neural network, 2-layer feed forward network, 3-layer feed forward neural network, CNN with 3 feature maps, LSTM, and bi-directional LSTM), all using Adam optimizer set at learning rate = 0.001 and epochs = 50 with randomly initialized embeddings.

The feed forward neural networks (FNNs) were modeled such that fully connected layers replaced the RNN layer, and had [500], [500, 300], and [500, 300, 200] dimensions for the 1-layer, 2-layer, and 3-layer FNN respectively. Due to the structure of the fully connected linear layers, the mean of the embedding weights was utilized instead. The FNN then connects to the last fully connected layer to output the sentiment (positive or negative) of the sentence.

The CNN model was modeled such that the RNN layer was replaced by the CNN layer with 3 1-dimensional convolution layers with kernel sizes 1, 2, and 3 respectively. The 1-max pooling of each convolution layer was concatenated together to be used as the input for the last fully connected layer.

The uni-directional LSTM model was modeled such that the RNN layer was replaced by a uni-directional LSTM layer, while the bi-directional LSTM model used a bi-directional LSTM layer, as well as a final fully connected layer having twice the size for the input.

Figures 7 and 8 shows the loss and accuracy curves respectively.

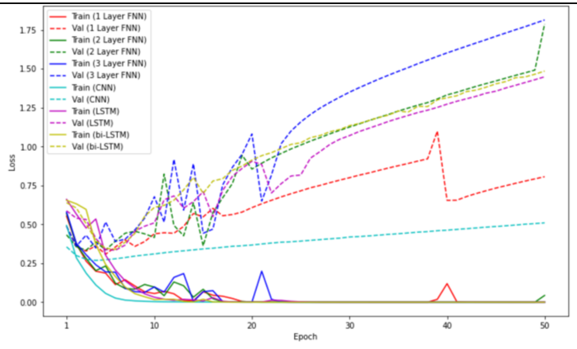

Figure 7. Loss vs Epoch

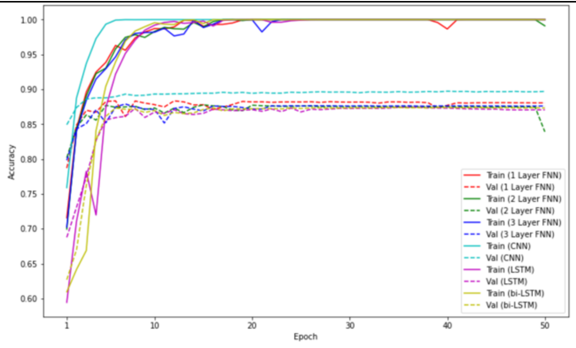

Figure 8. Accuracy vs Epoch

All 6 trained models were then evaluated using the test dataset and the results shown in Table IV below.

Table IV. Test Loss and Accuracy of Different Models

Model	1-Layer FNN	2-Layer FNN	3-Layer FNN	CNN	LSTM	Bi-LSTM
Test Loss	0.338	0.350	0.358	0.272	0.352	0.361
Test Accuracy (%)	86.84	85.43	86.39	88.75	85.49	86.01

As seen in Figure 7, all the models achieved near zero training loss, with the validation loss increasing after epoch 5, thus hinting at the possibility of overfitting to the training data. The CNN model performs the best here as it proved to have the smallest validation loss among the 6 different models. In Figure 8, all models achieved near 100% training accuracy, but the model with the best validation accuracy is also the CNN model. It can also be observed that the CNN model is the fastest to achieve near zero loss and near 100% accuracy among the 6 models.

In Table IV, the model with the best performance on the test data is also the CNN model. This is likely due to the use of 3 different kernel sizes and thus can carry both short and long term spatial context of the words, thus outperforming all other models.