cs577 Assignment 5 - Report

Girish Rajani-Bathija
A20503736
Department of Computer Science
Illinois Institute of Technology
November 22, 2022

Programming Question 1 (IMDB):

Problem Statement

Load the IMDB movie reviews dataset from Keras and split it into training, validation, and testing subsets. A network should be built with an embedding layer and two fully connected layers followed by training and evaluation of the model. The GloVe embedding file should be downloaded and loaded into an embedding matrix and initialized into the embedding layer while freezing the weights. The trained embedding layer should be used, and the new model should be trained and evaluated. Finally, a third model should be created whereby replacing one of the two fully connected layers with an LSTM layer and training should be repeated.

Proposed Solution

The proposed solution is to build multiple models that will incorporate embedding layers (one model having pretrained embedding layer, another model having trainable embedding layer, and the third model having either one) and fully connected layers. The final model will contain an LSTM layer in replacement of one of the fully connected layers. Training and evaluation of each of the three models will be done, and the results will be compared with each other to see how well pretrained vs trainable embedding layer performs and how adding an LSTM layer improves the performance of the model on the IMDB dataset.

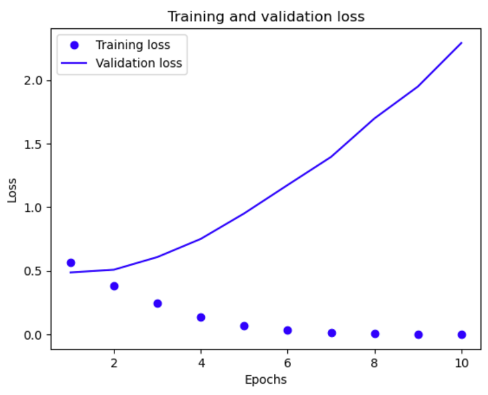
Implementation Details

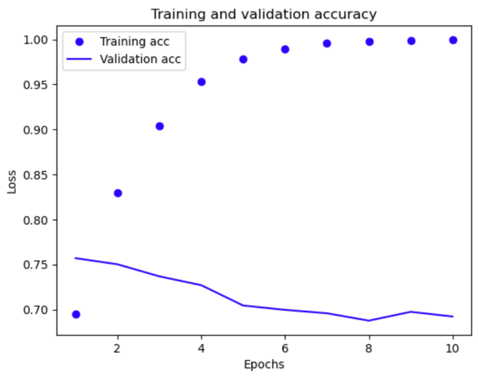
After loading the IMDB dataset, when turning the lists of integers into a 2D integer tensor using preprocessing.sequence.pad_sequences(), the following error occurred:
module ‘keras.preprocessing.sequence’ has no attribute ‘pad_sequences’.
It was found that in the newer version of tensorflow, the pad_sequences was under keras.utils.

When parsing the GloVe word-embeddings file, the following error occurred:
‘charmap’ codec can’t decode byte 0x9d in position 2776: character maps to <undefined>
To solve this problem, I had to specify the encoding when opening the glove.6B.100d.txt file. utf8 encoding was specified and that was able to solve the problem and open the file.

Results and Discussion

From the IMDB dataset, a subset of 20,000 training samples, 5,000 validation samples, and 25,000 testing samples were created. The first model consisted of an embedding layer with an input dimension of 10,000, output dimension of 100, and input length of 20. The model then used a flatten along with 2 dense layers, with the first layer containing 16 units with a relu activation and the output layer containing 1 unit with a sigmoid activation. The rmsprop optimizer and binary crossentropy loss was used to measure the accuracy metric. The model was trained for 10 epochs using a batch size of 32 and the results can be seen below.

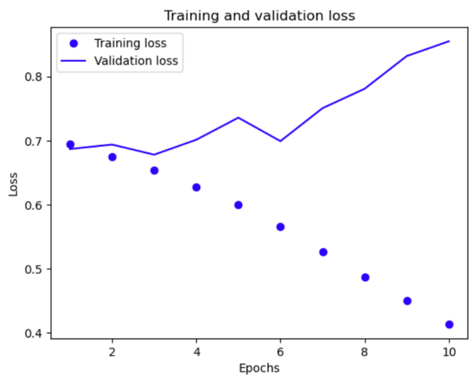


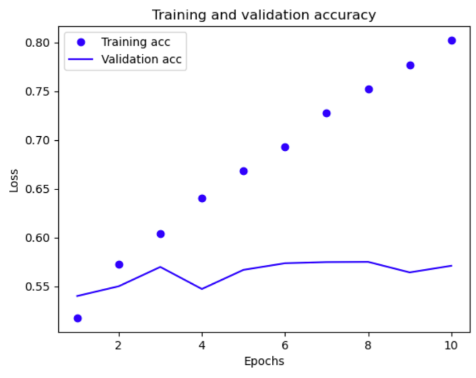


```
Test loss: 2.3817713260650635
Test accuracy: 0.6732000112533569
```

We can see from above that the model does not perform well. The validation loss increases instead of decreasing and the validation accuracy decreases instead of increasing. Although when training, the training accuracy reaches almost 100%, the test accuracy only gets to 67% with a very high test loss of 2.38 which shows that the model significantly overfits.

To try to combat this, instead using the pretrained embedding layer, a trainable embedding layer was made by using the GloVe word embedding file and loading it into an embedding matrix and initializing it into the embedding layer. This new model is identical to the previous one but just the embedding layer was changed from pretrained to a trainable layer. After training for 10 epochs with a batch size of 32, the results can be seen below:

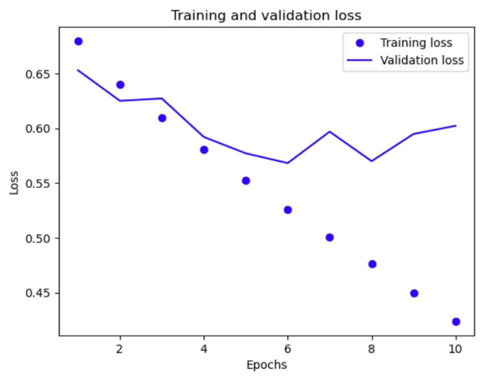


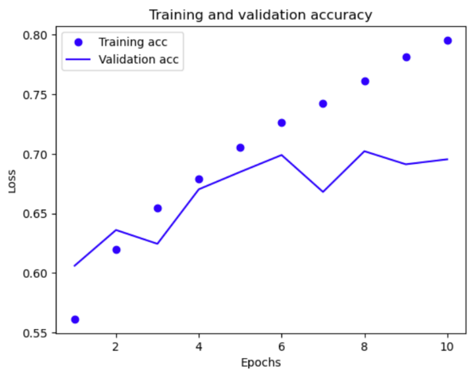


```
Test loss: 0.8738839626312256
Test accuracy: 0.5623199939727783
```

The weights of the trainable embedding layer were frozen and based on the results above, it can be observed from the new model that the network still overfits and does not perform well. Although when evaluated on test data, the test accuracy does drop from 67% to 56%, the test loss significantly drops from 2.38 to 0.87 so this means that the model did perform better than the initial implementation.

In hope of getting better results, a small modification was made to the model. A new identical model was made but the first fully connected layer with an output size of 16 and relu activation was removed and instead an LSTM layer was added. The trainable embedding layer was kept from the 2nd model with the weights frozen. The results of this final implementation can be seen below:





```
Test loss: 0.6110551357269287
Test accuracy: 0.6942399740219116
```

We can observe that even though the model still overfits, it performed the best out of all three implementations. It was able to reach an all time high test accuracy of 69% and reached the lowest test loss of 0.61 when compared to the other models. In conclusion, introducing a trainable embedding layer instead of a pretrained one was able to generate slightly better results however, the biggest impact was seen when implementing an LSTM layer along with an embedding layer.

Programming Question 2 (Reuters):

Problem Statement

Load the Reuters movie reviews dataset from Keras and split it into training, validation, and testing subsets. A network should be built with an embedding layer and two fully connected layers followed by training and evaluation of the model twice (once using trainable embedding layer and second time using loaded pretrained embedded layer). The results of both models should be compared and appropriate conclusions should be drawn. In a third model, an LSTM layer should replace the fully connected layer. Training and testing of this model should also be done and finally, a fourth model should be made, this time, just adding an additional LSTM layer.

Proposed Solution

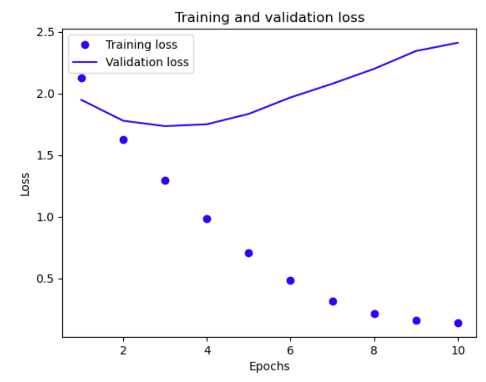
The proposed solution is to build multiple models that will incorporate embedding layers (both trainable and loaded embedding layer) and fully connected layers. The final two models will contain LSTM layer/s (1 model will contain 1 LSTM layer, and the final model will contain 2 LSTM layers) in replacement of one of the fully connected layers. Training and evaluation of each of the these models will be done, and the results will be compared with each other to see how well pretrained vs trainable embedding layer performs and how adding LSTM layers affects the performance of the model on the Reuters dataset.

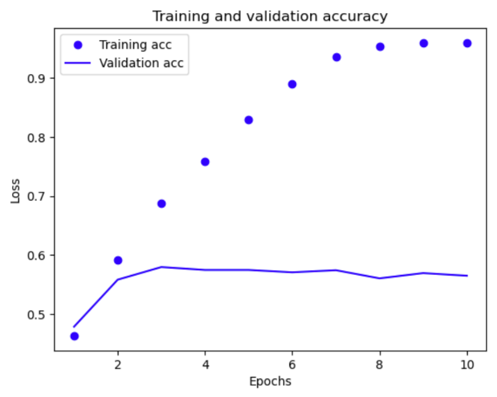
Implementation Details

The implementation and program design followed a similar implementation from the previous question. The same problems mentioned in the previous implementation were also encountered and solved in the same way.

Results and Discussion

From the Reuters dataset, a subset of 6,736 training samples, 2,246 validation samples, and 2,246 testing samples were created. The first model consisted of an embedding layer with an input dimension of 10,000, output dimension of 100, and input length of 20. The model then used a flatten along with 2 dense layers with the first layer containing 64 units with a relu activation and the output layer containing 46 units with a sigmoid activation. The rmsprop optimizer and categorical crossentropy loss was used to measure the accuracy metric. The model was trained for 10 epochs using a batch size of 32 and the results can be seen below.

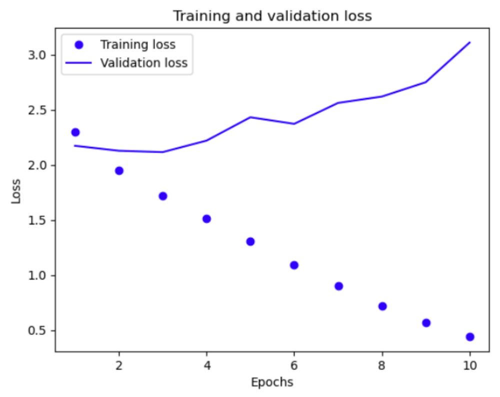


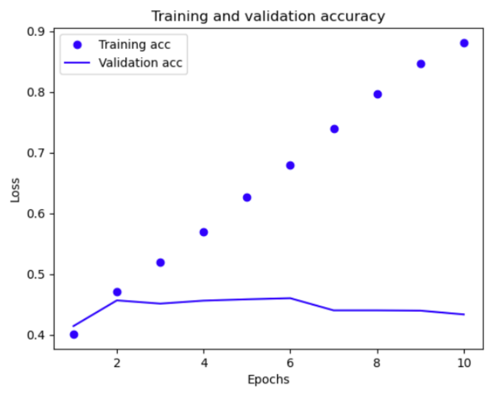


```
Test loss: 2.458040714263916
Test accuracy: 0.5569902062416077
```

We can see from above that the model does not perform well as it overfits after 3-4 epochs. The validation loss starts to increase and the validation accuracy stops increasing after a few epochs. Although when training, the training accuracy reaches almost 100%, the test accuracy only gets to 56% with a very high test loss of 2.46, which shows that the model significantly overfits.

To try to combat this, instead of using the pretrained embedding layer, a trainable embedding layer was made by using the GloVe word embedding file and loading it into an embedding matrix and initializing it into the embedding layer. This new model is identical to the previous one but just the embedding layer was changed from pretrained to a trainable layer. After training for 10 epochs with a batch size of 32, the results can be seen below:

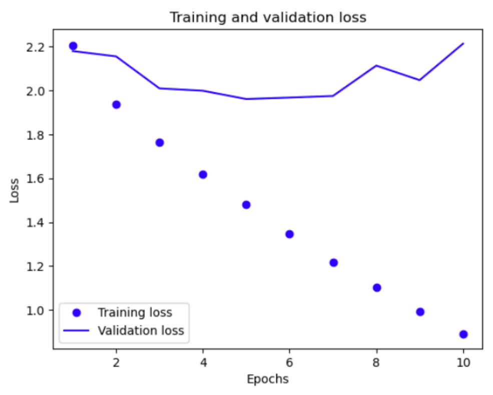


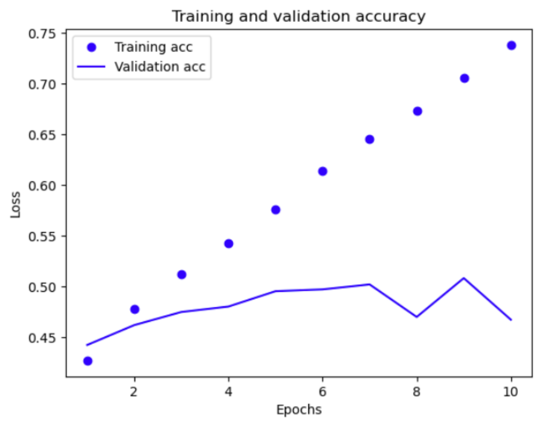


```
Test loss: 3.1166772842407227
Test accuracy: 0.4336598515510559
```

The weights of the trainable embedding layer were frozen and based on the results above, it can be observed from the new model that the network still overfits and actually performs worse than the previous implementation. When evaluated on test data, the test accuracy does drop from 56% to 44%, and the test loss increases from 2.46 to 3.12 which demonstrates a loss in performance after implementing the trainable embedding layer.

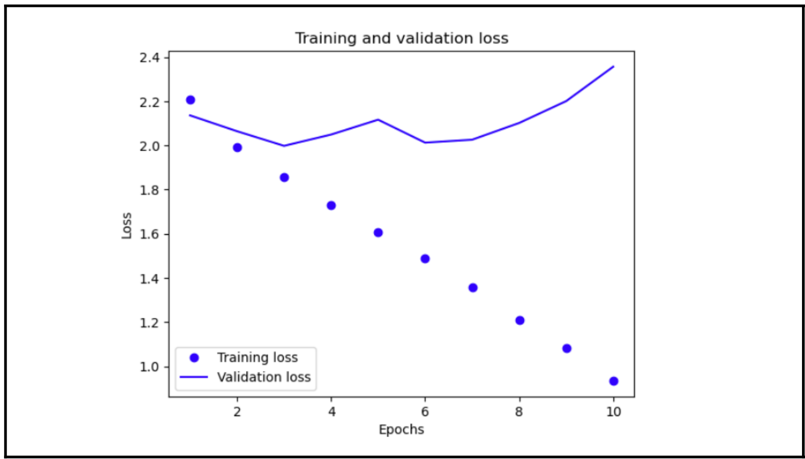
In hope of getting better results, the model was modified. A new identical model was made but the first fully connected layer with an output size of 64 and relu activation was removed and instead an LSTM layer was added. The trainable embedding layer was kept from the 2nd model with the weights frozen. The results of this implementation can be seen below:

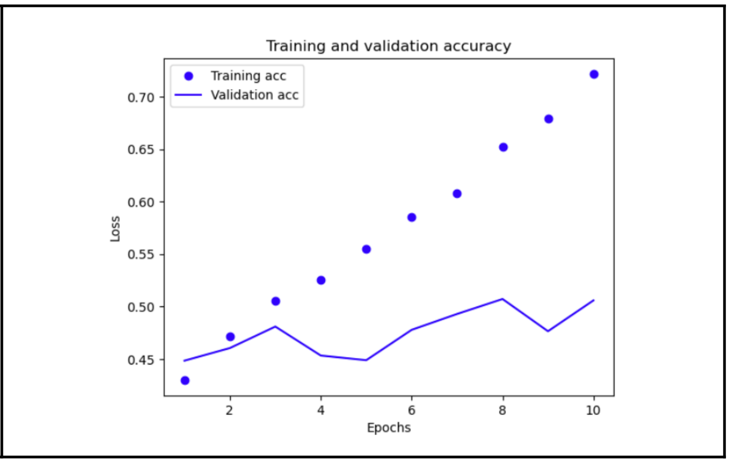




```
Test loss: 2.195302963256836
Test accuracy: 0.46660730242729187
```

It can be observed from above that after introducing an LSTM layer, the model now overfits after 7 epochs compared to 3-4 epochs from the previous models. Although when evaluated on test data, the test accuracy remained relatively the same, the test loss was reduced from 3.12 to 2.20 which shows an improvement in the model and the capability to slightly generalize better. For the final model, an additional LSTM layer was added to see if the model can generalize better. The results of this can be seen below:





```
Test loss: 2.3712334632873535
Test accuracy: 0.5102404356002808
```

It can be observed that the model still overfits and is able to reach a training accuracy of appx 74% but when evaluated on test data, achieves a test accuracy of 51% which is slightly higher than the previous implementation. However, the test loss does slightly increase when compared to the previous implementation, and therefore, in conclusion, using a trainable embedding layer instead of a pretrained embedding layer did not improve the performance of the model on the Reuters dataset. Replacing the fully connected layer with an LSTM layer did improve the performance by reducing the test loss and increasing the test accuracy and adding a second LSTM layer did not have much effect on the results.