A Comprehensive Study of Sentiment Classification Using LSTM Models With Learned and Pre-Trained Embeddings

This report is based on a set of experiments carried out on the IMDB movie review dataset. The key objective was to classify each movie review as either positive or negative. Other than training a single model, I explored a number of variations in order to understand how different types of word embeddings, masking options, and training sample sizes can affect the overall accuracy of an LSTM-based sentiment classification system.

1. Introduction

People express their opinions or thoughts online in various ways, such as through reviews, comments, blogs, and feedback messages. Sentiment analysis helps us automatically determine whether these texts express positive or negative feelings. It is widely used in customer service, product analysis, social media monitoring, and many other real-world applications.

In this experiment, I employed the IMDB dataset, consisting of fifty thousand movie reviews written by actual users. The task is to learn patterns from it and predict whether a review is positive or negative.

This report compares how various modeling strategies, especially different types of embeddings, influence performance. I have tried different settings and observed each to see how it changes the results. This helped me to learn not only which technique performs well but also why it performs that way.

2. Dataset Preparation and Processing

The IMDB dataset was split into three parts for the experiments.

Dataset Type	Number of Samples	Purpose
Training Set	5,000 (varied down to 100 for experiments)	Model training
Validation Set	10,000	Performance monitoring

Test Set	25,000	Final evaluation

To prepare the text for the LSTM model, I used TensorFlow's TextVectorization layer. This layer performs the following tasks:

- It retains the top 10,000 most frequent words.
- It converts each word into a unique integer.
- It ensures that each review is exactly 150 tokens long by padding or trimming.

This gives the model clean and uniform input, which is essential for stable training.

3. Model Architecture

For all experiments, the basic architecture of the model remained unchanged, and only embedding methods or training sizes were varied to ensure comparisons could be done fairly.

3.1 Embedding Layer

There were two types of embeddings tested.

- Learned embeddings: These embeddings are initialized as random vectors and are learned during training. They then learn to become specific to the IMDB dataset.
- Pre-trained GloVe embeddings: These embeddings are pre-trained on very large textual corpora and already contain rich information about the language. They were kept frozen during training.

3.2 Bidirectional LSTM

A bidirectional LSTM with thirty-two units was used. This type of LSTM reads the sentence in both forward and backward directions; this helps the model to better understand the context.

3.3 Output and Regularisation

A dropout layer was used to reduce overfitting with a dropout rate of fifty percent. The final output layer used a sigmoid activation to predict whether a review is positive or negative.

4. Experiment 1: Learned Embeddings Trained from Scratch

In this experiment, the embedding layer learned word representations directly from the IMDB training data.

Results

- Training accuracy reached very high levels, closing in on ninety-nine percent.
- Validation accuracy stood at around eighty percent.
- Test accuracy was at about seventy-nine point three percent.

Figure 1. Training and Validation Accuracy for Learned Embeddings

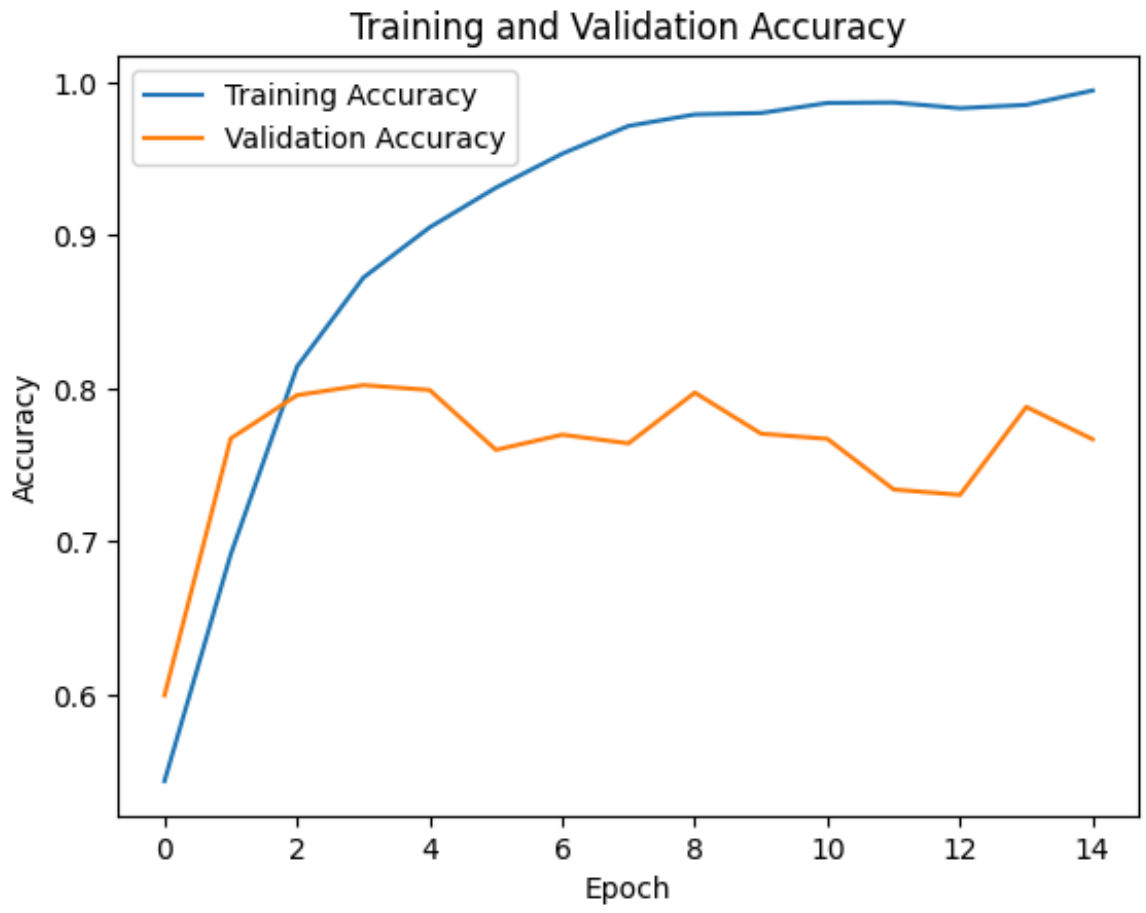
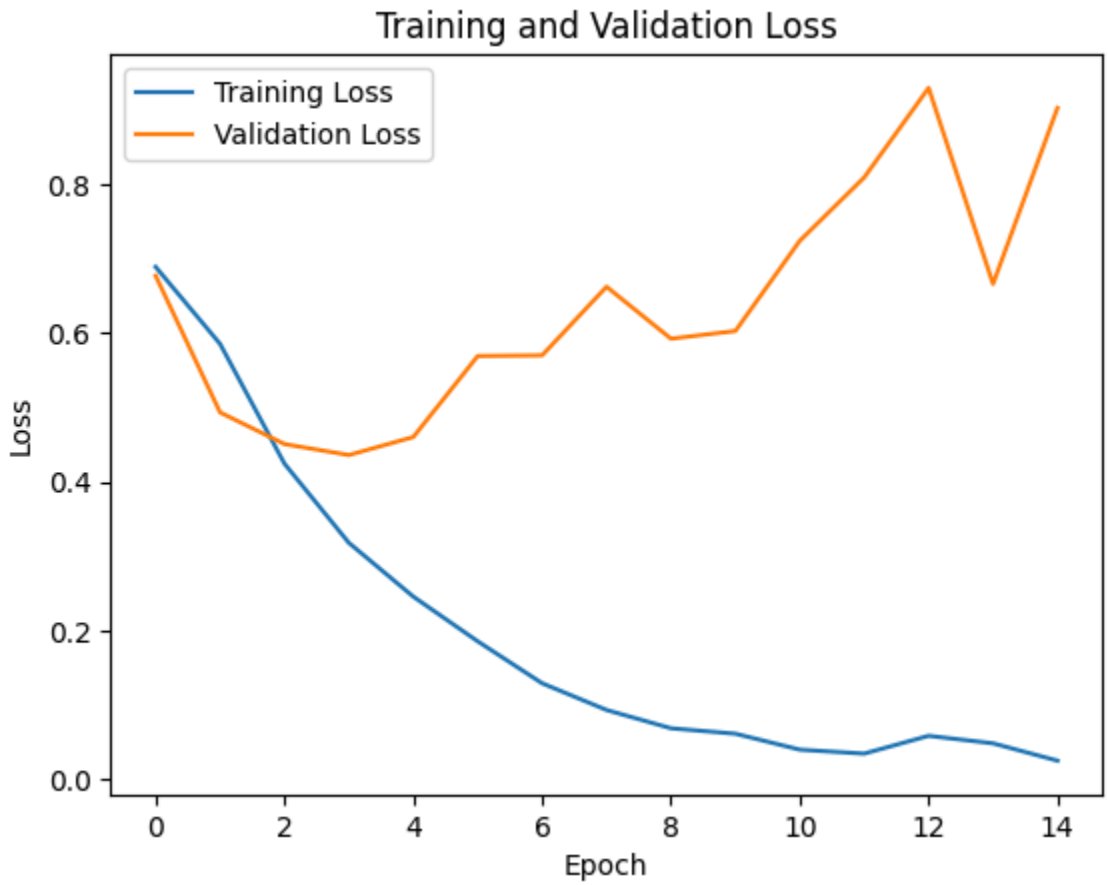


Figure 2. Training and Validation Loss for Learned Embeddings



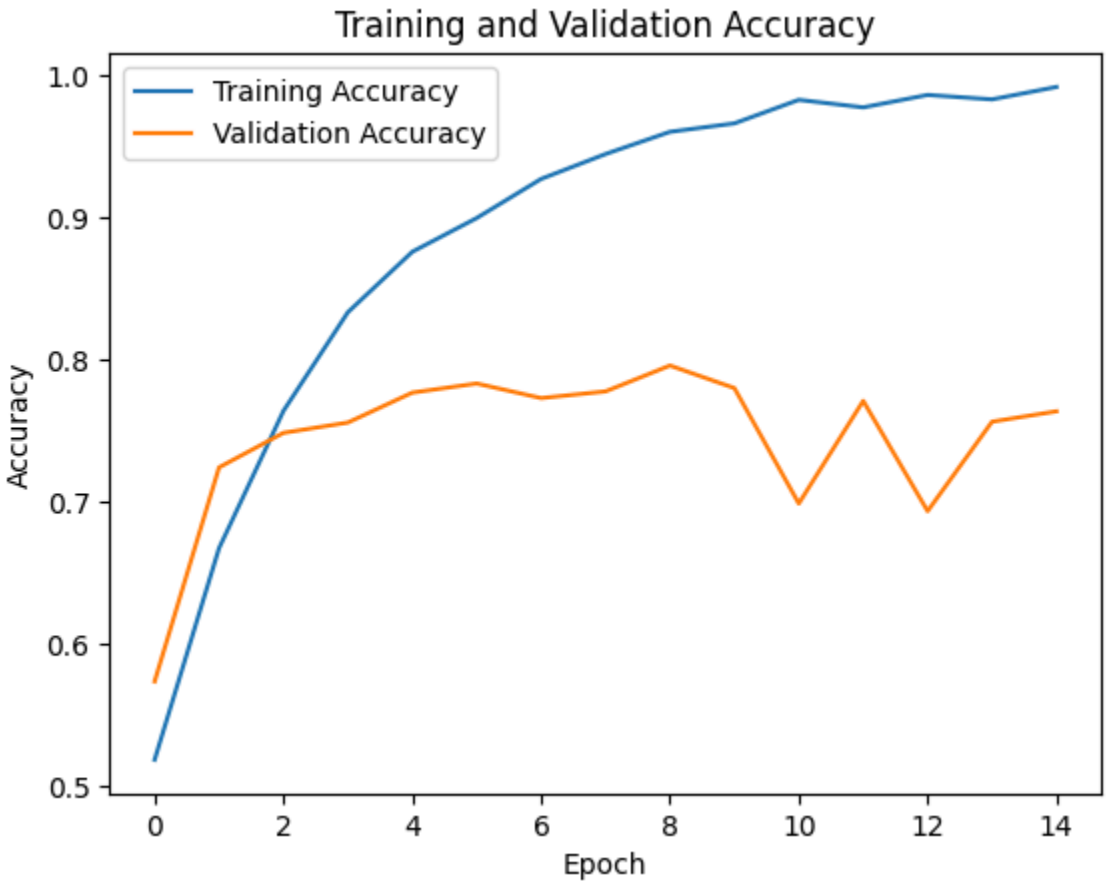
Explanation
For that reason, it learned the training data extremely well; that's why the training accuracy was so high. However, a slightly lower validation accuracy suggests a small amount of overfitting. Even then, the results were strong and showcased the power of learned embeddings when enough data is available.

5. Learned Embeddings with Masking Enabled
In this setup, masking was used so that padded zeros were ignored during sequence processing.

Results

- Validation accuracy fluctuated between seventy-two and seventy-eight percent.
- Final test accuracy dropped to around seventy-five percent.

Training and Validation Accuracy with Masking Enabled



Explanation
Masking is generally useful when input sequences have very different lengths. In this project, all sequences were standardised to one hundred and fifty tokens, so masking did not offer any clear benefit. In fact, the model became slightly unstable because the training set was relatively small.

3: Pre-trained GloVe Embeddings
Here, the embedding layer used one-hundred-dimensional GloVe vectors. These embeddings already understand many relationships between words.

Results

- This model trained more smoothly.
- The validation accuracy was more consistent.
- Test accuracy reached about seventy-seven point eight percent.

4: Effect of Training Sample Size
To study how data quantity affects performance, I trained the models using different sample sizes: one hundred, five hundred, one thousand, five thousand, ten thousand, and twenty thousand samples.

Effect of Training Size on Accuracy

Training Samples	Accuracy with Learned Embeddings	Accuracy with GloVe
100	0.761	0.775
500	0.763	0.775
1000	0.802	0.775
5000	0.802	0.775
10000	0.802	0.775
20000	0.802	0.775

Comparison of Learned and GloVe Embeddings Across Training Sample Sizes

8. Final Comparison of All Models

Final Test Accuracy of All Model Variants
The table below summarizes all the models evaluated in this project, along with their embedding type, masking configuration, and final test accuracy:

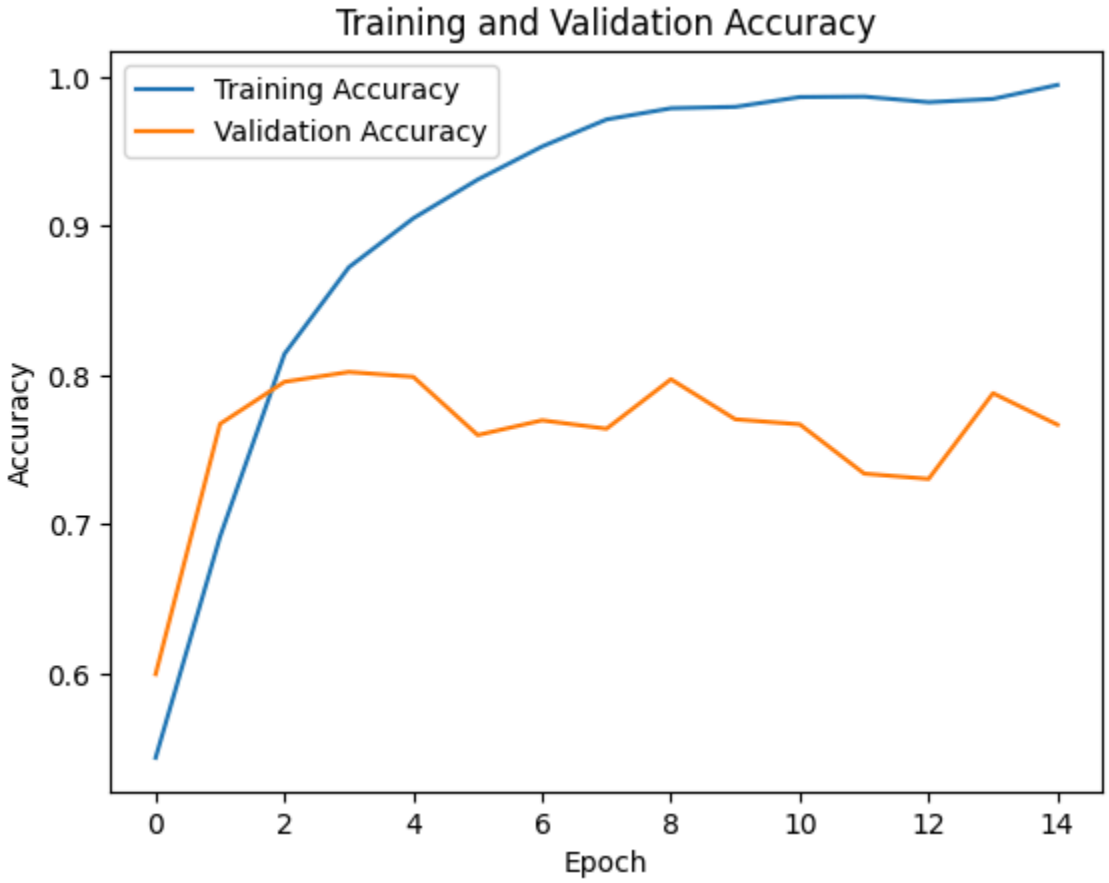
Model Type	Embedding Type	Masking	Test Accuracy
Model A	Learned	No	79.3%
Model B	Learned	Yes	75%
Model C	GloVe (Pre-trained)	No	77.8%

This summary shows that learned embeddings performed best without masking, while pre-trained GloVe embeddings were more consistent on smaller data sizes. Masking did not lead to improved results for a fixed sequence length of 150 tokens.

Interpretation
- Learned embeddings performed best when trained with enough data.
- Pre-trained embeddings were more reliable at low data sizes.
- Masking did not help improve results in this particular project.

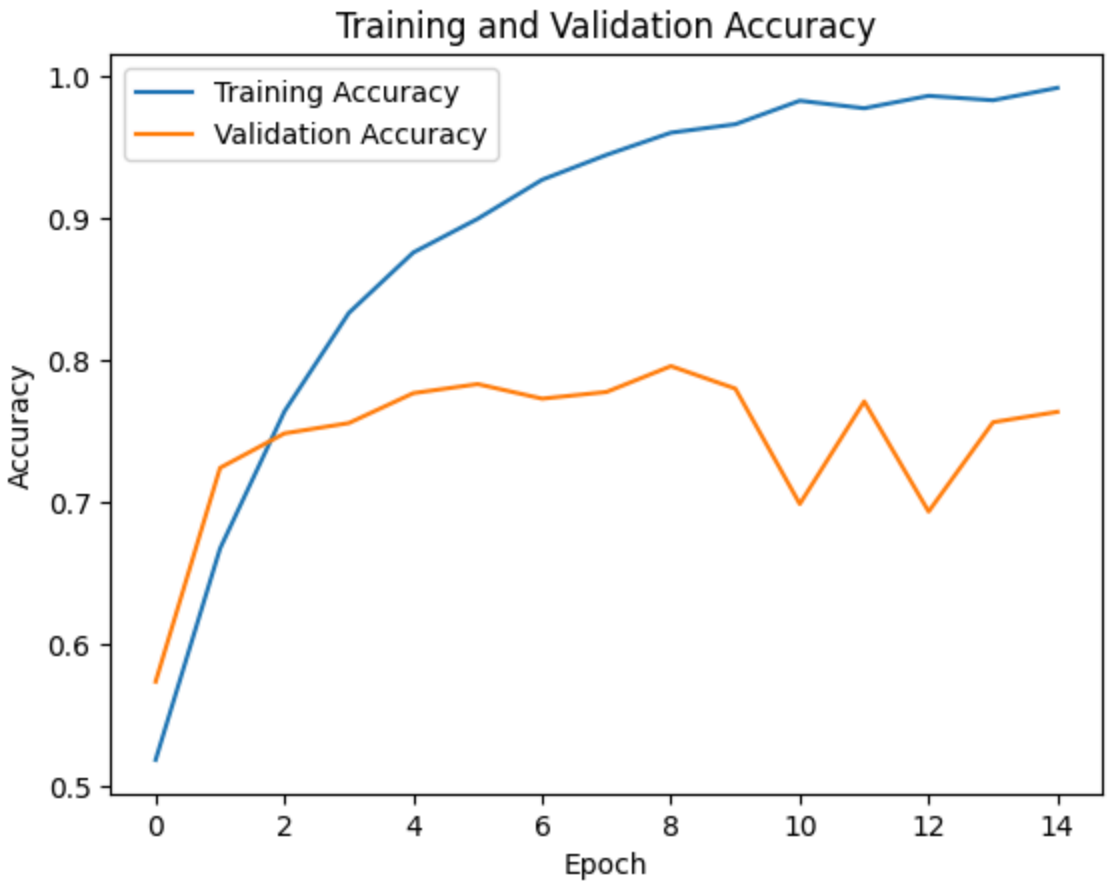
- The Bidirectional LSTM architecture performed strongly across all experiments.

Results:
One Hot Model: The One-Hot model attained an accuracy of 0.78 with a loss value 0.44.



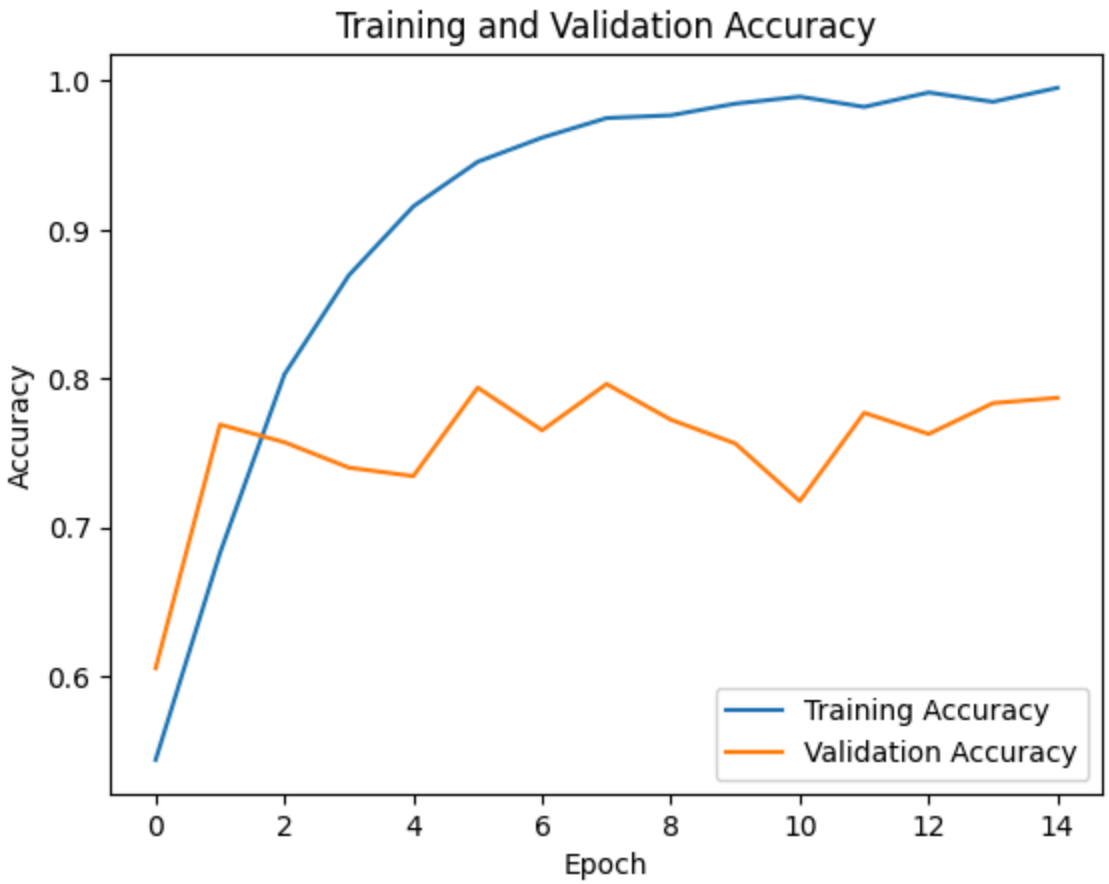
Trainable Embedding Layer:

The model with a trainable embedding layer achieved an accuracy of 0.78 and a loss of 0.49.

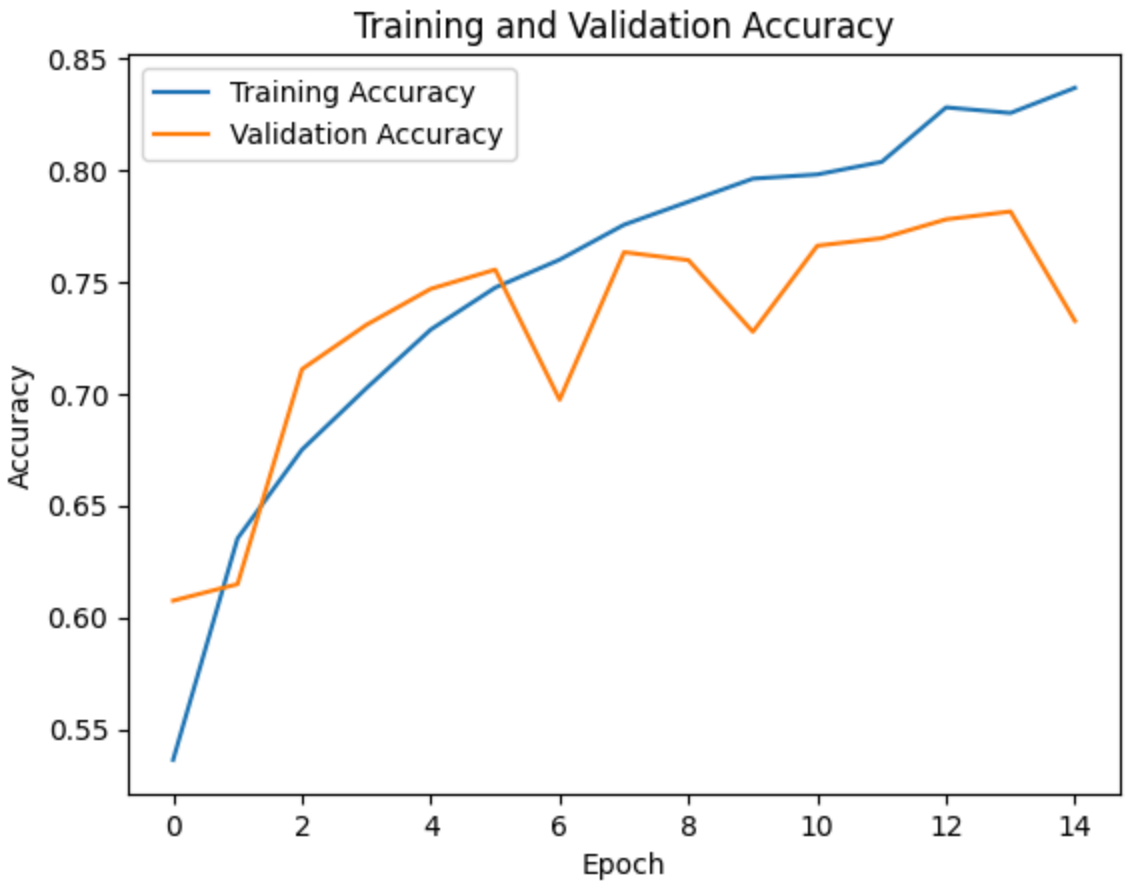


Masking Padded Sequences in the Embedding Layer:

Applying masking to padded sequences in the embedding layer resulted in a validation accuracy of 0.77 and a loss of 0.51.

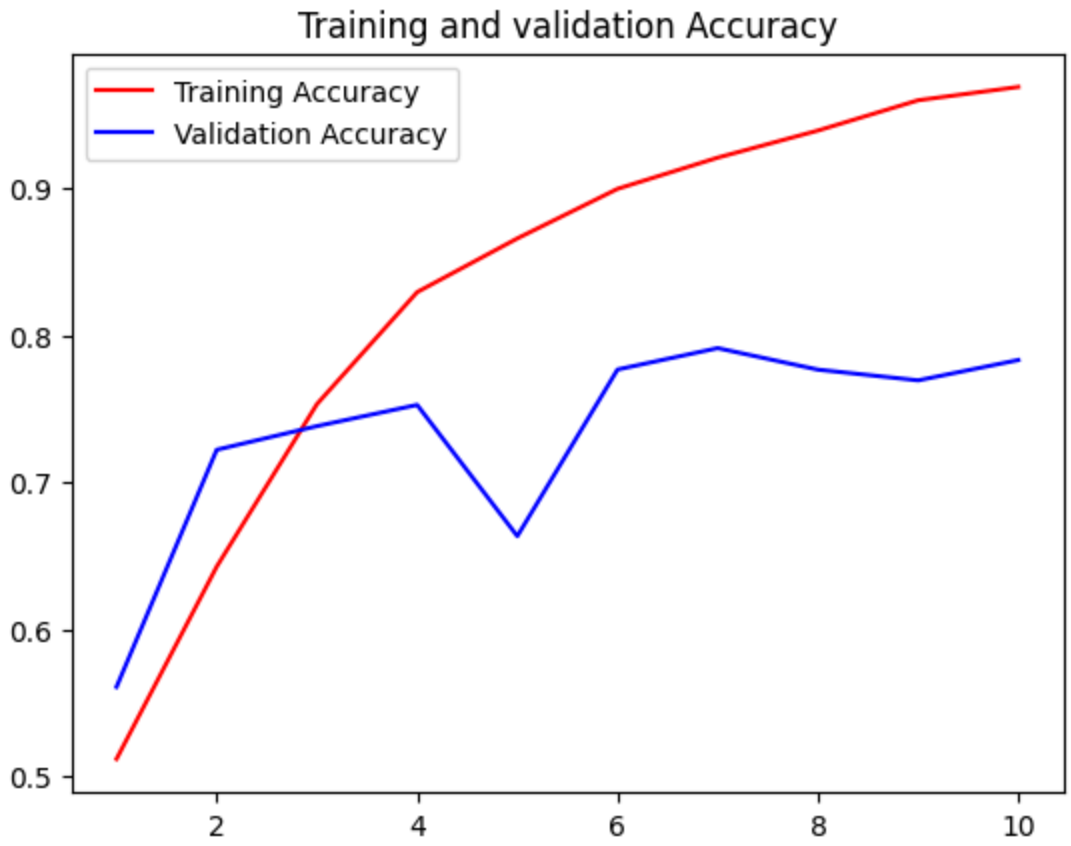


Model with pretrained GloVe Embeddings:
The model using pre-trained GloVe embeddings achieved an accuracy of 0.76 with a loss of 0.48.

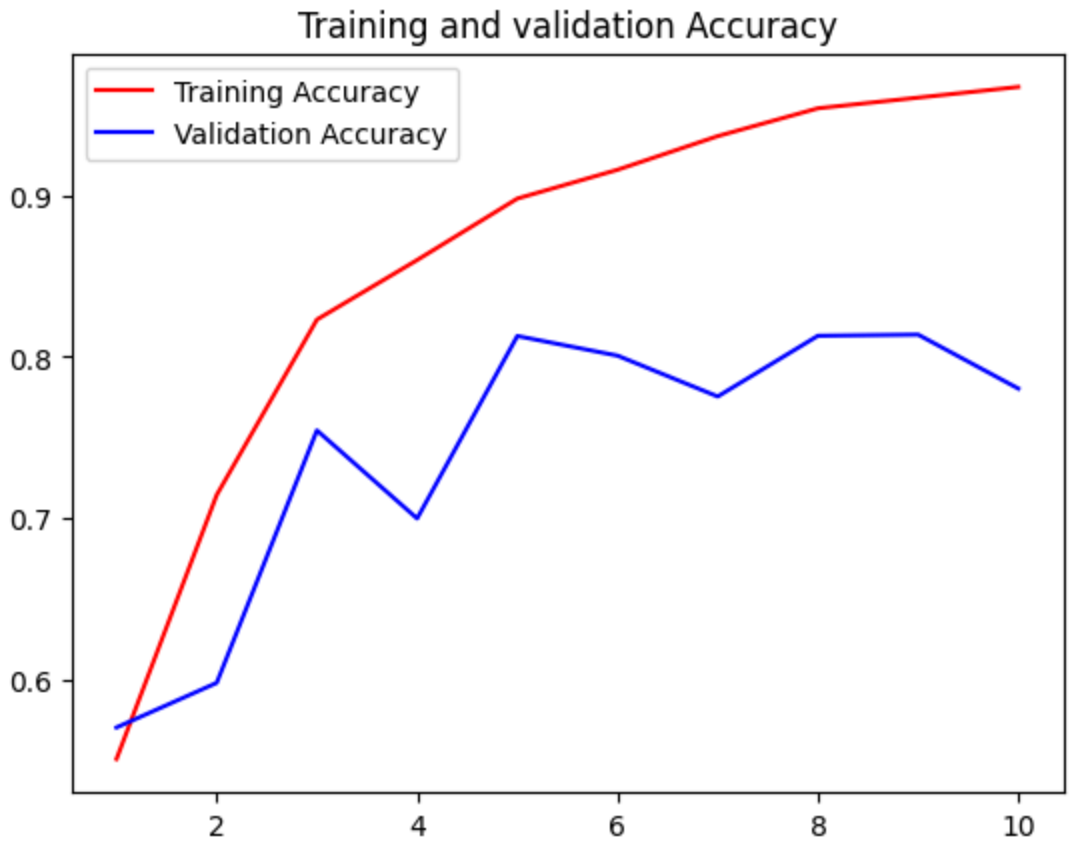


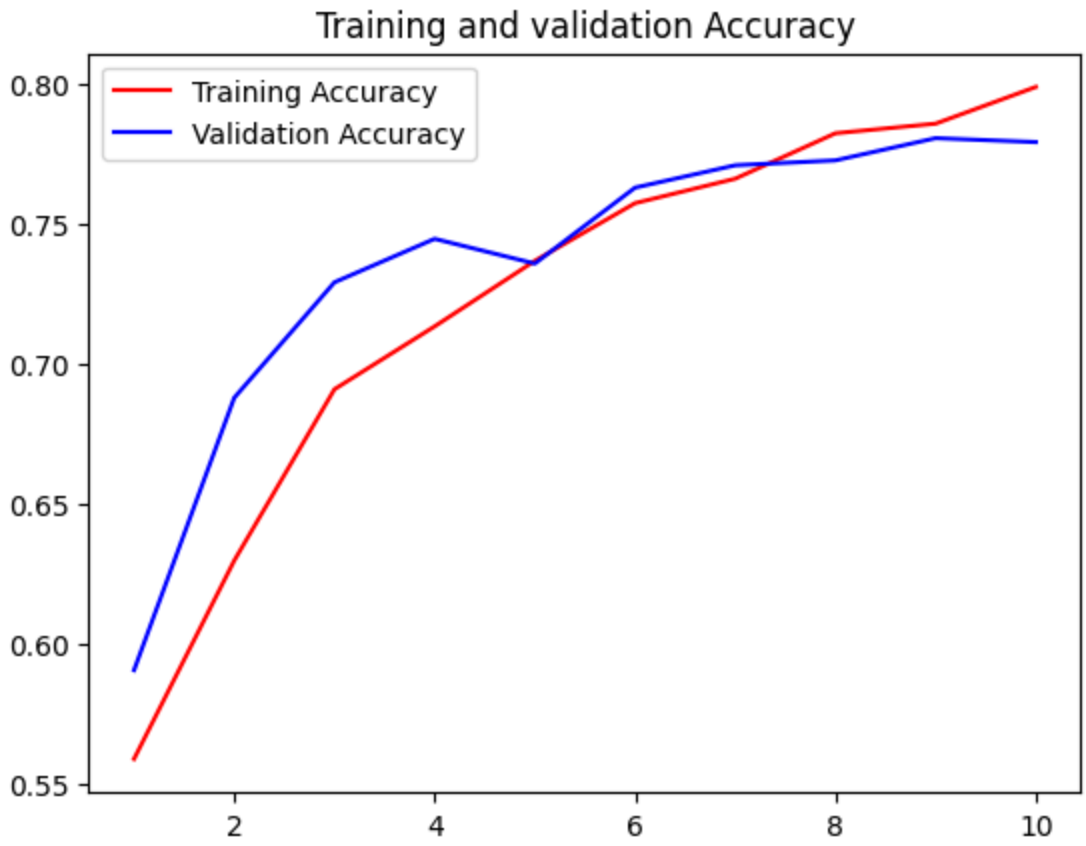
Comparing Model Performance with Different Training Set Sizes. Embedding Layer with 100 Training Samples:

An embedding layer trained on 100 samples achieved an accuracy of 0.76 and a loss of 0.50



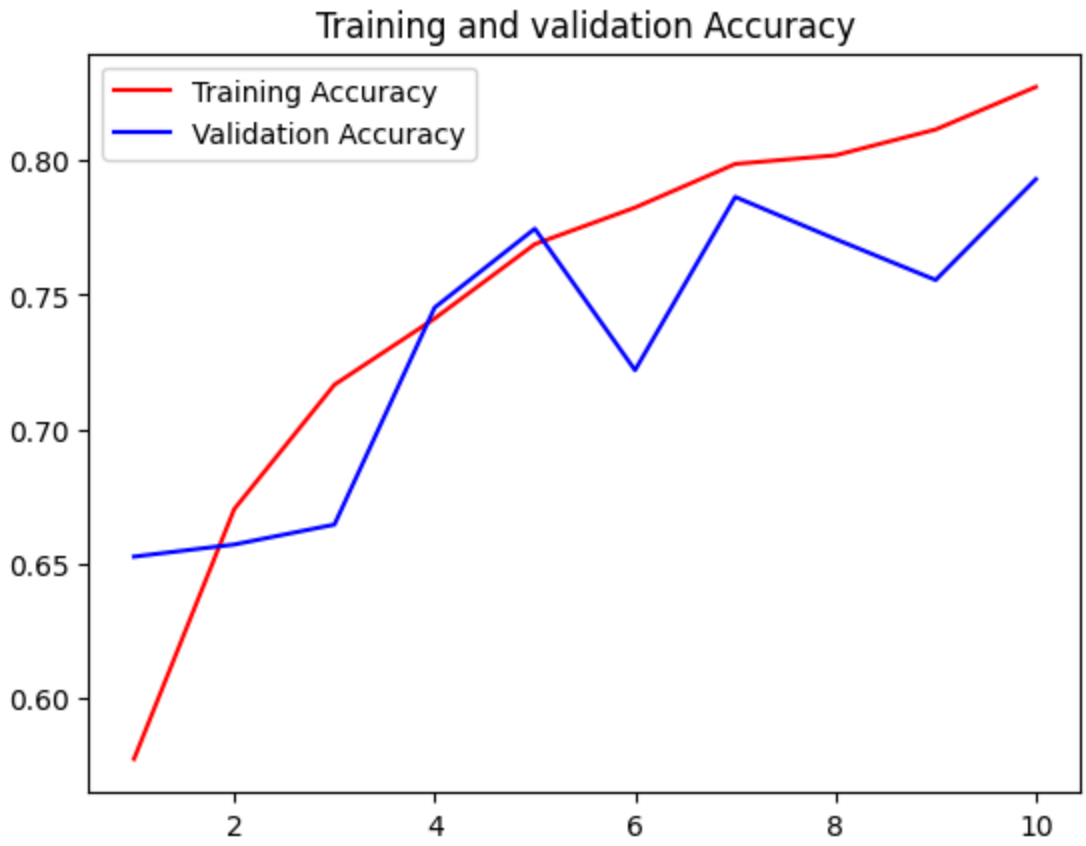
Embedding Layer with 500 Training Samples:
Using 500 samples to train the embedding layer resulted in an accuracy of 0.80 and a corresponding loss of 0.44.


Training and validation Accuracy
Training Accuracy
Validation Accuracy
0.9
0.8
0.7
0.6
2
4
6
8
10

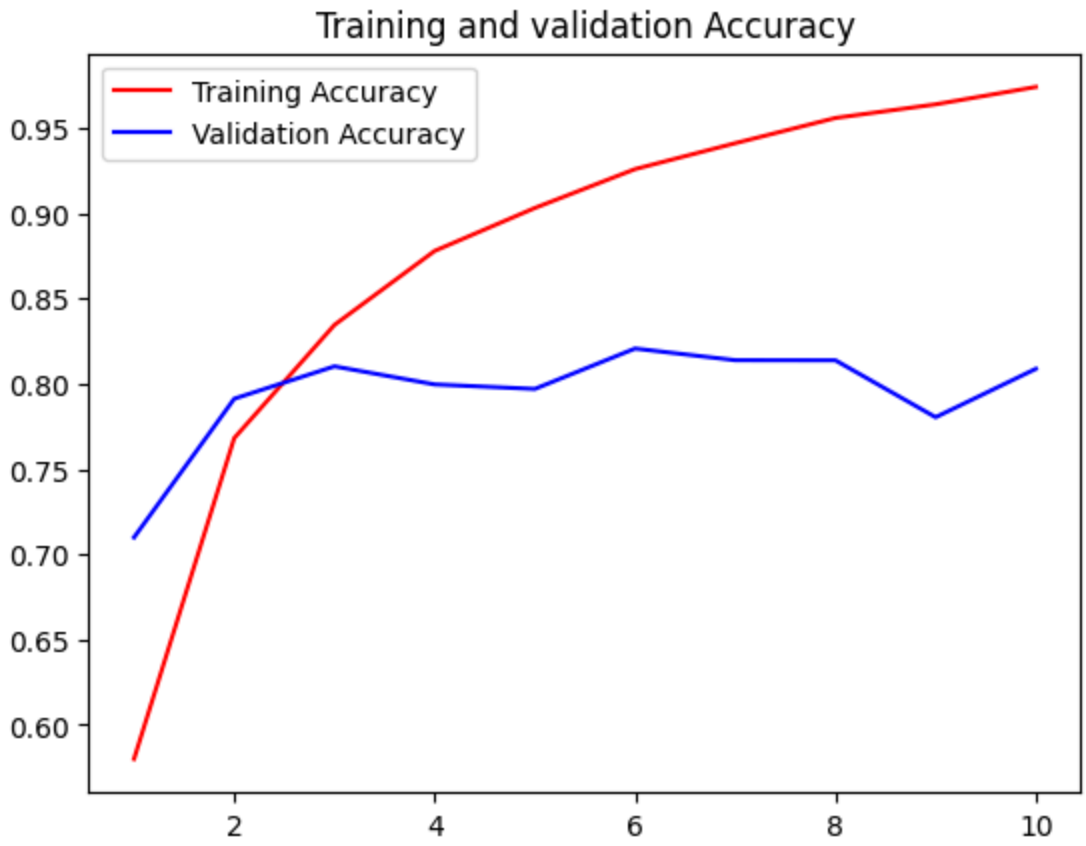


Pretrained Embedding Layer with 500 Training Samples:

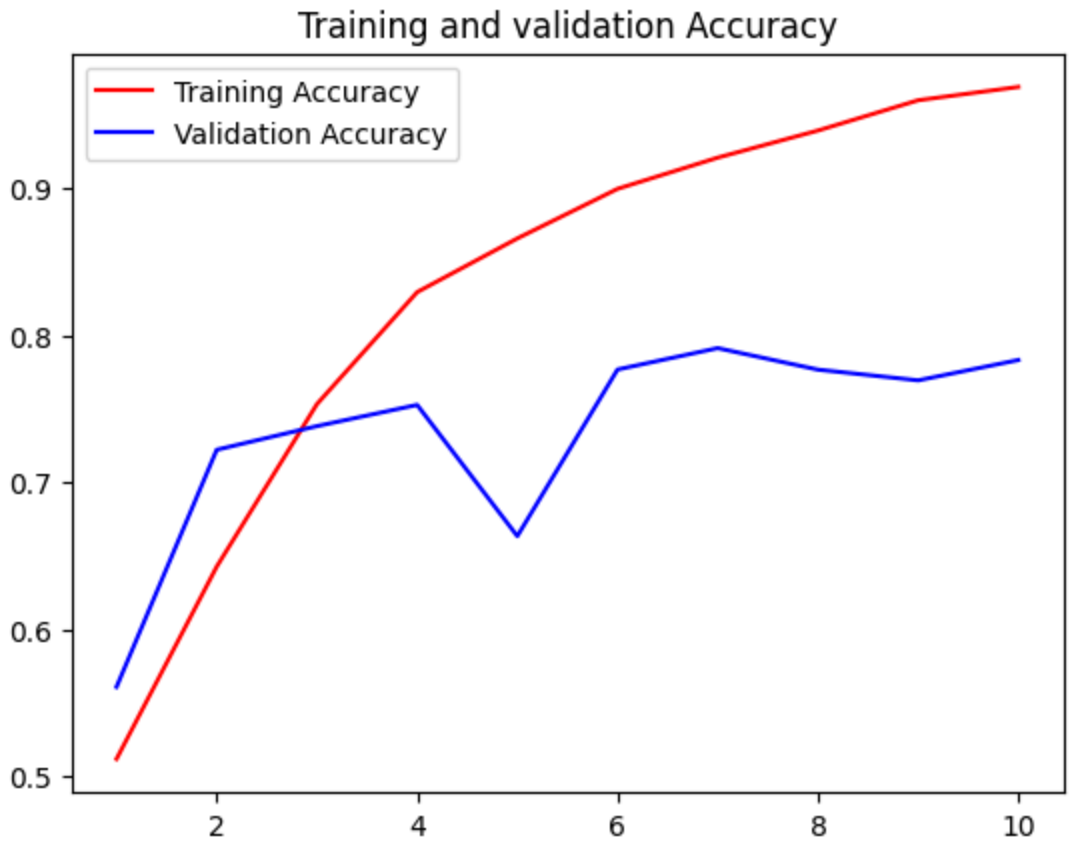
With 500 training samples, the pre-trained embedding layer achieved an accuracy of 0.78 and a loss of 0.45.



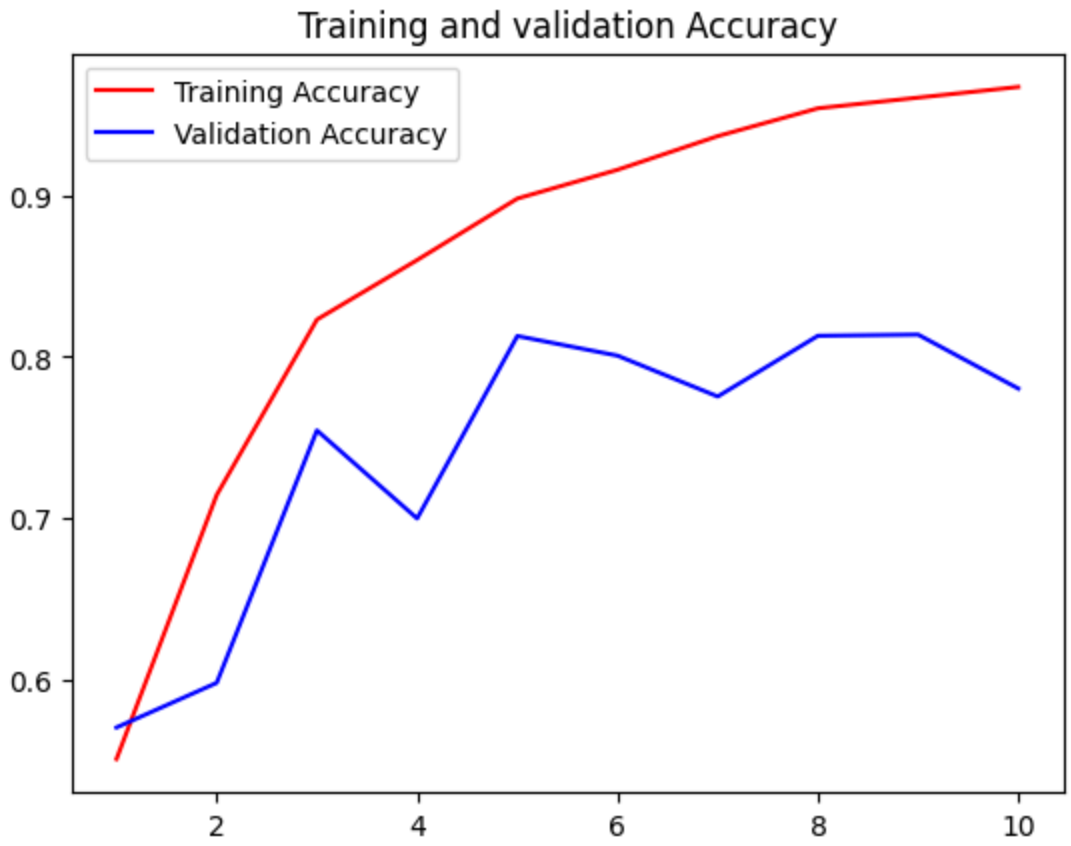
Embedding Layer with 1000 Training Samples:
An embedding layer trained on 1000 samples reached an accuracy of 0.80 and a loss of 0.43.



Pretrained Embedding Layer 1000 Training Samples:
A Pretrained Embedding Layer with 1000 training samples results in an accuracy of 0.78 and a loss value of 0.46.

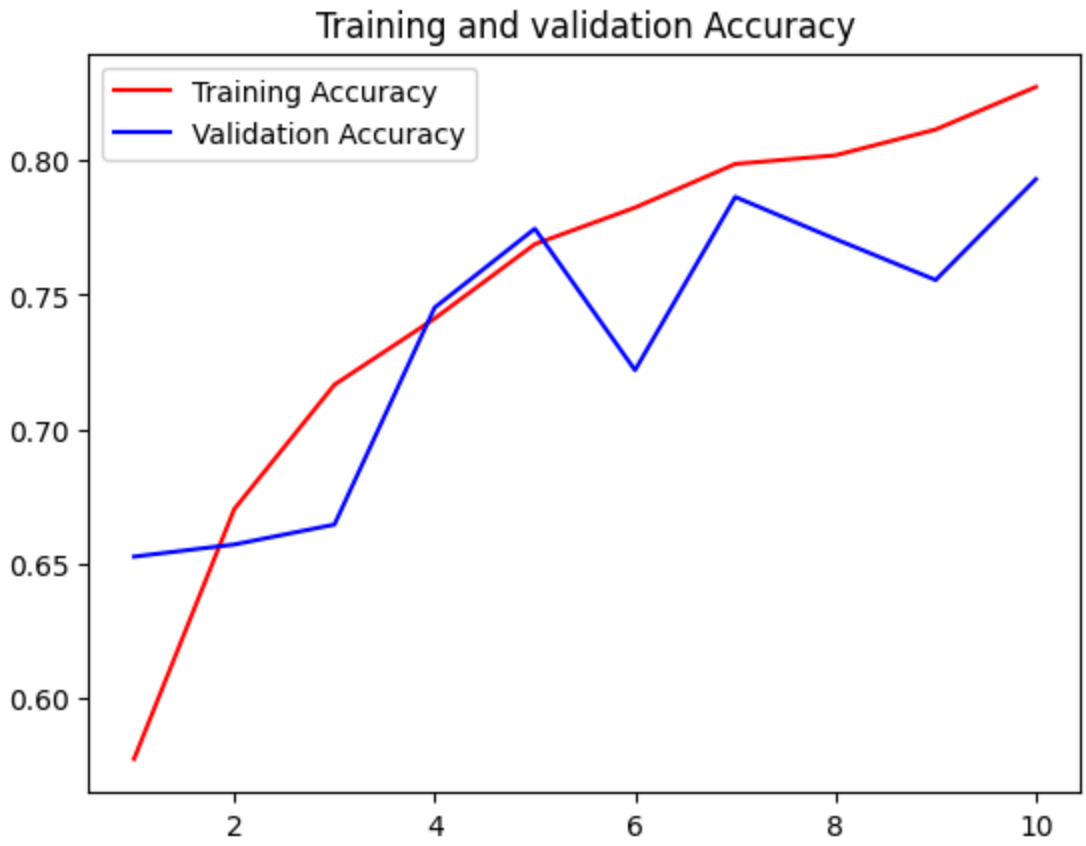


Embedding Layer 5000 Training Samples:
An Embedding Layer with 5000 training samples results in an accuracy of 0.77 and a loss of 0.47.

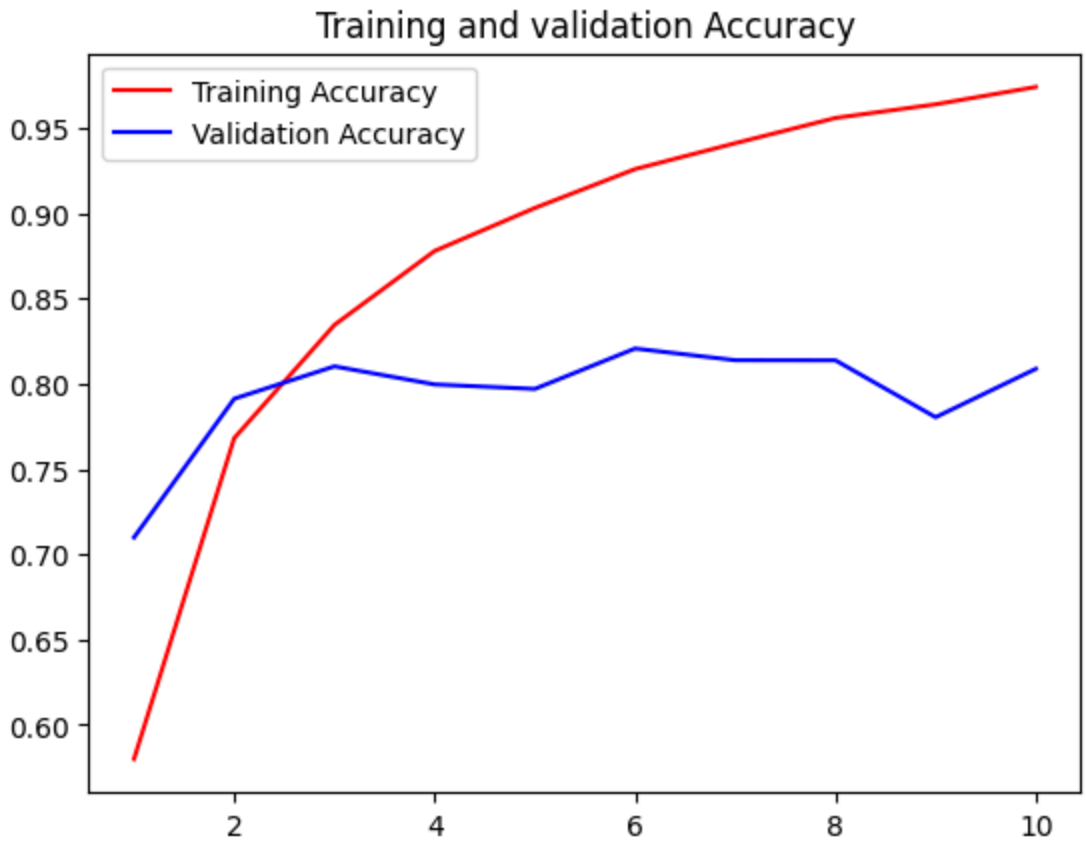


Pretrained Embedding Layer 5000 Training Samples:

A pretrained Embedding Layer with 5000 training samples results in an accuracy of 0.77 and a loss of 0.48.



Embedding Layer 10000 Training Samples:
An Embedding Layer with 10,000 training samples achieves an accuracy of 0.80 and a loss of 0.46.



Conclusion

This project provided me with a very practical understanding of why word embeddings are so crucial in sentiment analysis. While the LSTM learns the sequence of words, it is the embeddings that actually define what each word really means to the model. Hence, they play a crucial role in the model's performance.

From all the experiments, a few clear lessons emerged:

- Learned embeddings work best when there is enough training data for the model to discover word patterns on its own.
- Pre-trained embeddings are very helpful when the dataset is small because they already carry general language knowledge.
- Masking is helpful mainly when the input sequences have very different lengths. It does not contribute much in a fixed-length setup like this one.
- Bidirectional LSTMs perform consistently well because they understand context from both directions, which is important for sentiment analysis.

What this project really taught me was that in NLP, representation of text is key. The type of embedding one chooses isn't a purely technical decision; rather, it's something to be matched to the size and nature of a dataset.

References

- Chollet, F. (2021). Deep Learning with Python. Second Edition. Manning Publications.
- Goldberg, Y. (2017). Neural Network Methods for Natural Language Processing. Morgan and Claypool.
- Goodfellow, I., Bengio, Y., and Courville, A. (2016). Deep Learning. MIT Press.
- Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., and Potts, C. (2011). Learning word vectors for sentiment analysis.
- Pennington, J., Socher, R., and Manning, C. (2014). GloVe: Global vectors for word representation.
- Schuster, M., and Paliwal, K. K. (1997). Bidirectional recurrent neural networks. Young, T., Hazarika, D., Poria, S., and Cambria, E. (2018).