Assignment 4

INTRODUCTION

Recurrent Neural Networks (RNNs) are a class of neural networks designed for sequential data, where the model retains a hidden state that summarizes information from previous time steps. This allows RNNs to capture temporal dependencies in tasks such as language processing, time series forecasting, and speech recognition. This assignment is focused specifically on the IMDB movie review dataset.

DATA PREPARATION

Training Set Limitation: Only 100 entries were used from the training data to streamline the process.
Review Length Restriction: Reviews were cut off after reaching a limit of 150 words.
Validation Set Creation: A separate set of 10,000 samples was reserved for validation purposes.
Vocabulary Curation: The vocabulary was limited to the 10,000 most frequent words in the dataset.

METHODOLOGY

Baseline Model

The baseline model combined a Recurrent Neural Network (RNN) with an embedding layer. The RNN was responsible for processing text in a sequential manner, while the embedding layer converted words into dense vector representations. Key hyperparameters included the number of RNN units and the dimensionality of the embedding vectors. To gauge performance, both accuracy and loss metrics were monitored on the validation and test datasets.

Pretrained Word Embeddings

Next, pretrained word embeddings (like GloVe) were introduced to leverage semantic features acquired from large-scale corpora. These embeddings were integrated into the model, enabling it to capitalize on existing word vector representations for more robust text understanding. As before, accuracy and loss were recorded on the validation and test sets to evaluate the impact of using pretrained embeddings.

Varying Training Set Size

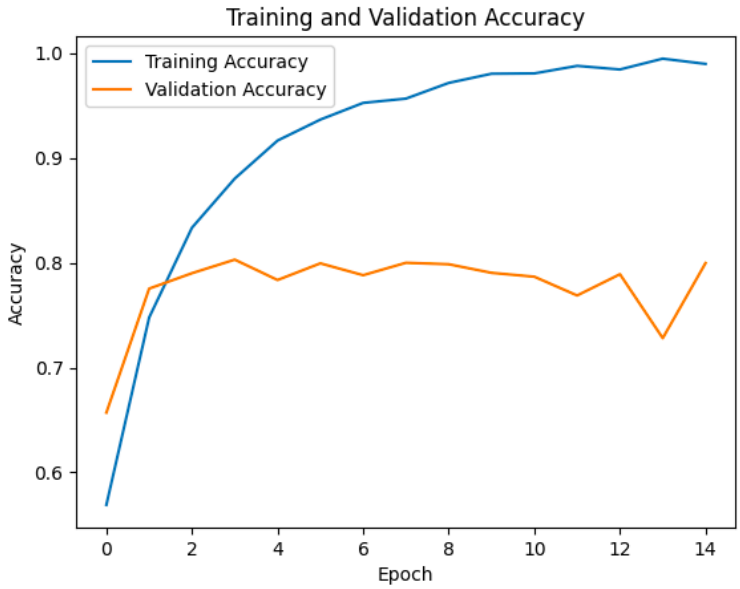
To explore how different amounts of data influence the model's performance, the number of samples in the training set was varied. Accuracy and loss values were collected for each training set size on both validation and test sets. A direct comparison was then made between models that relied solely on an embedding layer and those that employed pretrained embeddings, examining how each approach scaled as the volume of training data changed.

TABLE RESULTS:

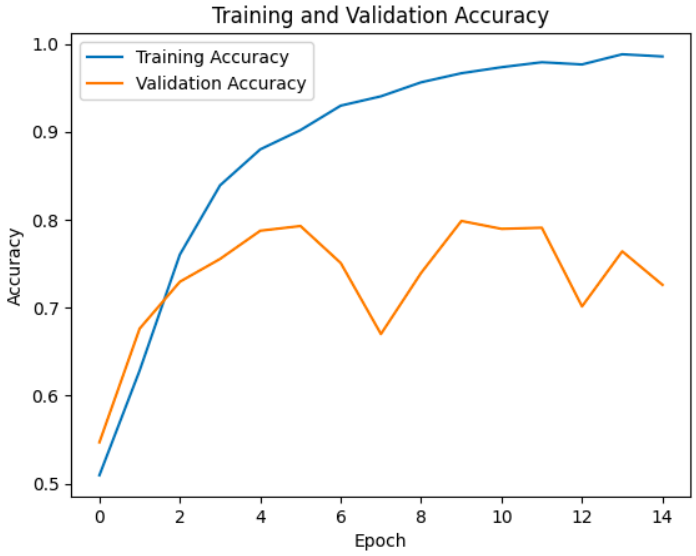
Model	Accuracy	Loss
One Hot model	0.7924	0.7911
Trainable Embedding Layer	0.7844	0.4987
Masking Padded Sequences in the Embedding Layer	0.7791	0.4730
Model with Pretrained GloVe Embeddings	0.7977	0.4440
Embedding Layer of 100 Training Samples	0.7773	0.4934
Pretrained Embedding Layer of 100 Training Samples	0.7859	0.4555
Embedding Layer of 500 Training Samples	0.7862	0.4605
Pretrained Embedding Layer of 500 Training Samples	0.7892	0.4463
Embedding Layer of 1000 Training Samples	0.7941	0.4676
Pretrained Embedding Layer of 1000 Training Samples	0.7694	0.4775
Embedding Layer of 5000 Training Samples	0.7871	0.4673
Pretrained Embedding Layer of 5000 Training Samples	0.7699	0.4708
Embedding Layer of 10000 Training Samples	0.7905	0.4473
Pretrained Embedding Layer of 10000 Training Samples	0.7846	0.4524
Embedding Layer of 20000 Training Samples	0.7718	0.4758
Pretrained Embedding Layer of 20000 Training Samples	0.7885	0.4483

RESULTS

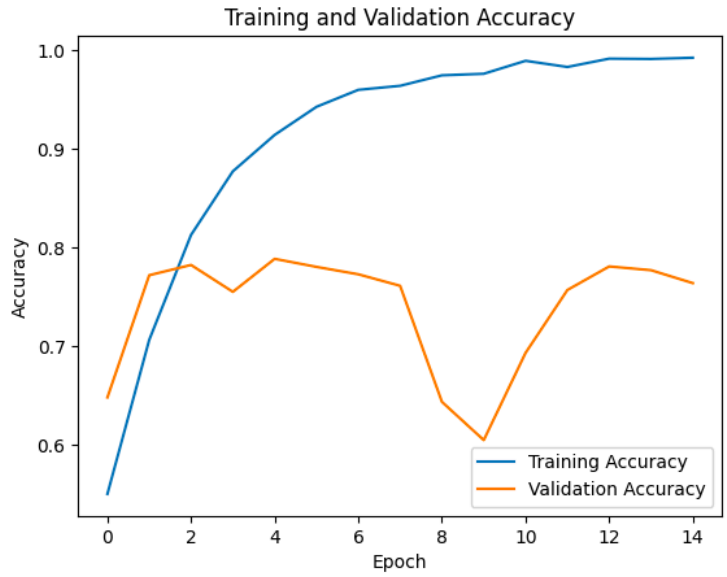
One Hot Model: One Hot model achieves an Accuracy of 0.7924 and a loss of 0.7911.



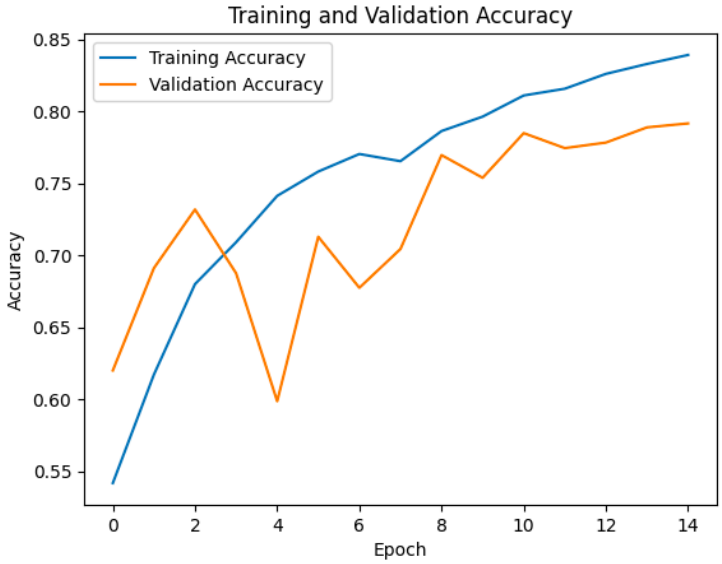
Trainable Embedding Layer: A Trainable Embedding Layer achieves an Accuracy of 0.7844 and a loss of 0.4987.



Masking Padded Sequences in the Embedding Layer: A Masking Padded Sequences in the Embedding Layer achieves a validation Accuracy of 0.7791 and a loss of 0.4730.

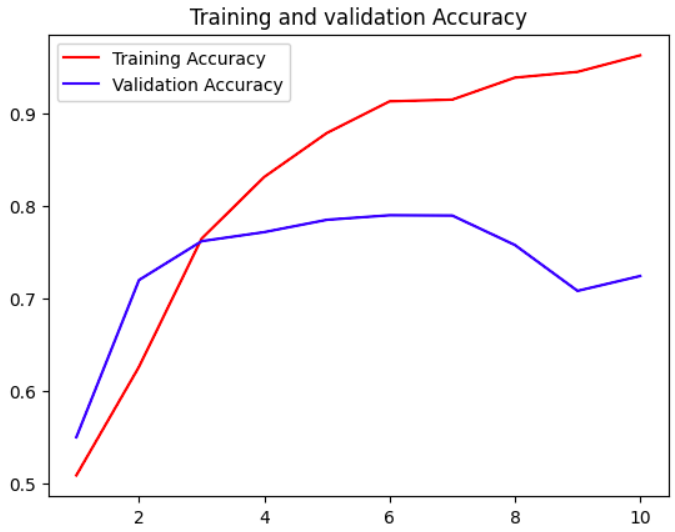


Model with Pretrained GloVe Embeddings: A Model with Pretrained GloVe Embeddings achieves Accuracy of 0.7977 and a loss of 0.4440.

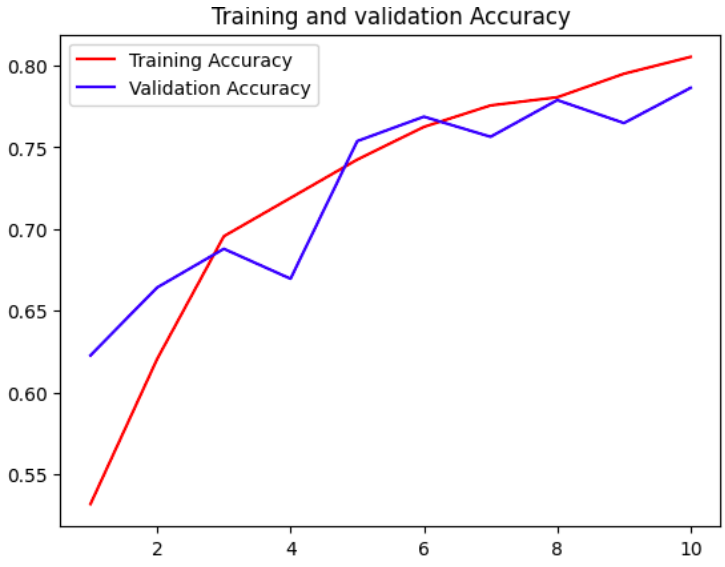


Comparing Model Performance with Different Training Set Sizes

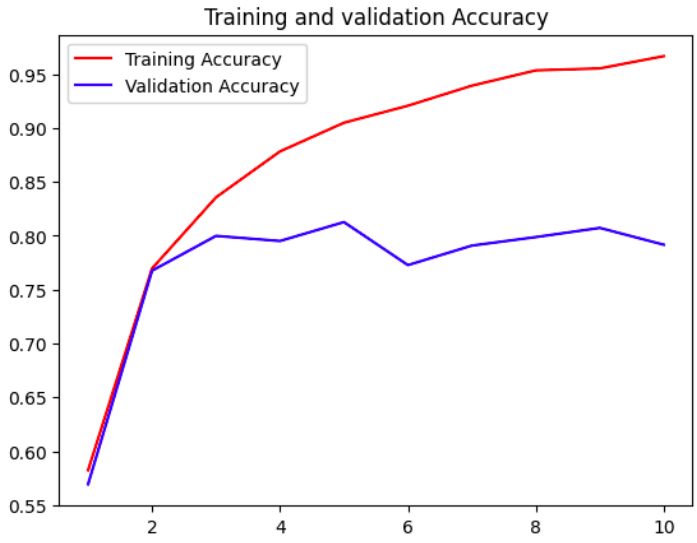
Embedding Layer 100 Training Samples: A Embedding Layer with 100 Training Samples achieves Accuracy of 0.7773 and a loss of 0.4934.



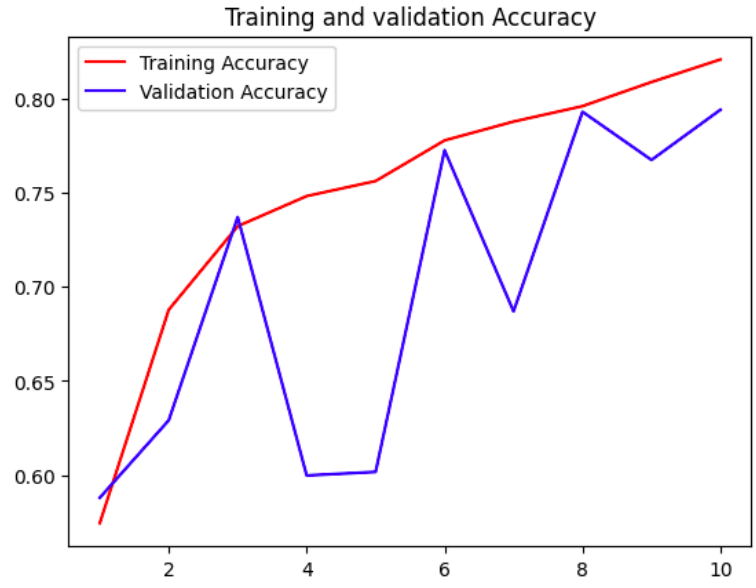
Pretrained Embedding Layer 100 Training Samples: A Pretrained Embedding Layer with 100 Training Samples achieves Accuracy of 0.7859 and a loss of 0.4555.



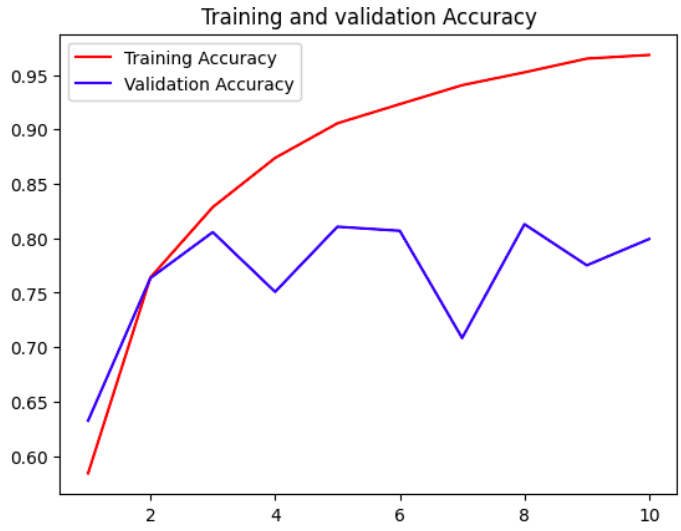
Embedding Layer 500 Training Samples: A Embedding Layer with 500 Training Samples achieves Accuracy of 0.7862 and a loss of 0.4605.



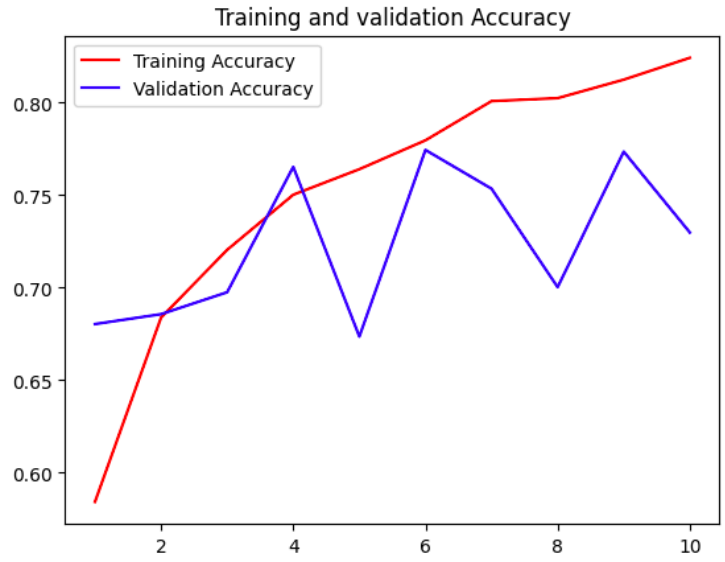
Pretrained Embedding Layer 500 Training Samples: A Pretrained Embedding Layer with 500 Training Samples achieves Accuracy of 0.7892 and a loss of 0.4463.



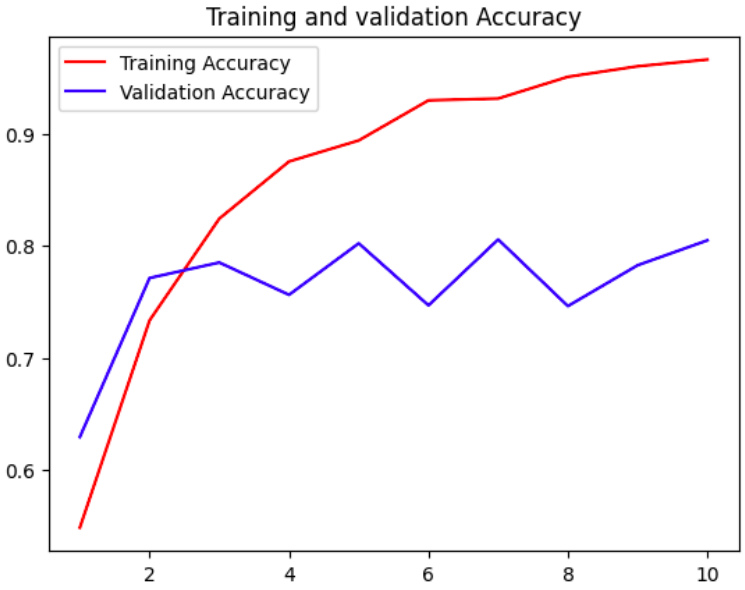
Embedding Layer 1000 Training Samples: A Embedding Layer with 1000 Training Samples achieves Accuracy of 0.7941 and a loss of 0.4676.



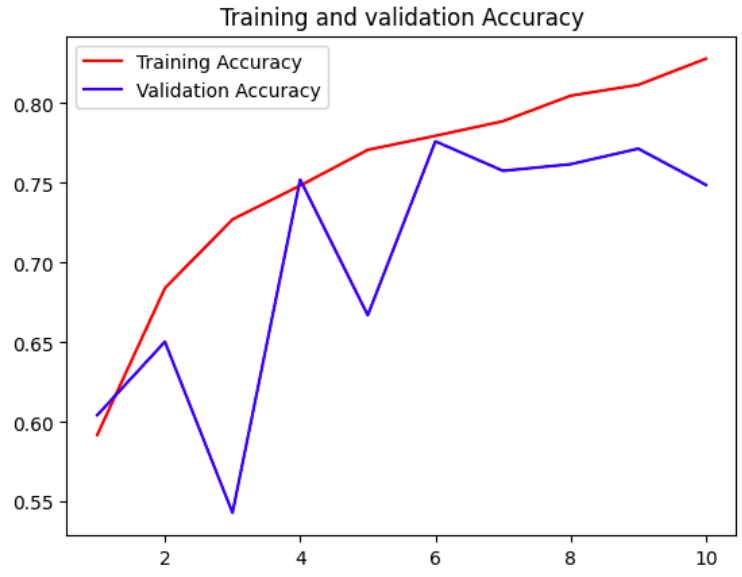
Pretrained Embedding Layer 1000 Training Samples: A Pretrained Embedding Layer with 1000 Training Samples achieves an Accuracy of 0.7694 and a loss of 0.4775.



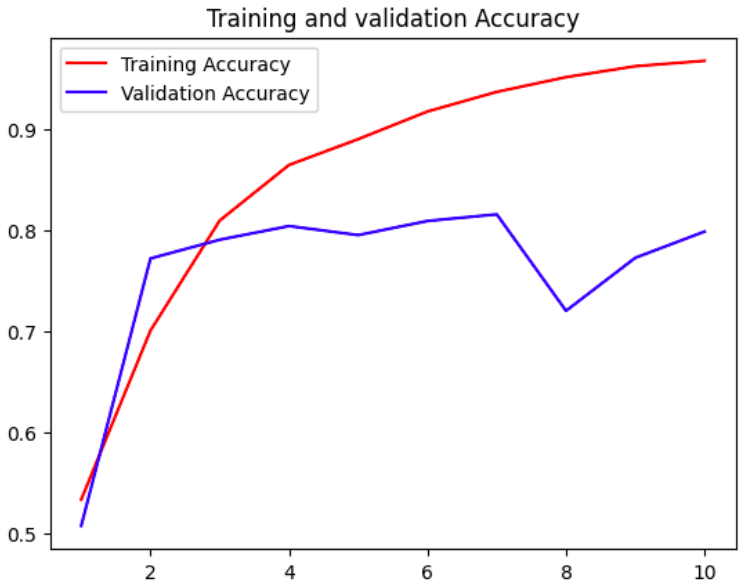
Embedding Layer 5000 Training Samples: A Embedding Layer with 5000 Training Samples achieves an Accuracy of 0.7871 and a loss of 0.4673.



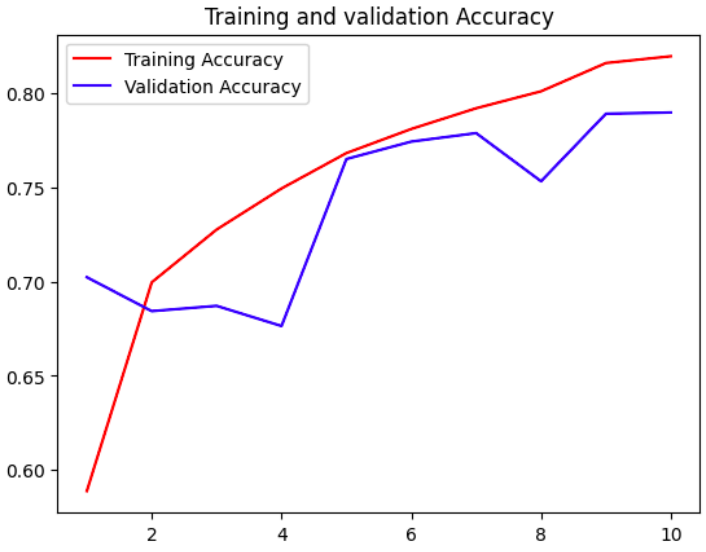
Pretrained Embedding Layer 5000 Training Samples: A Pretrained Embedding Layer with 5000 Training Samples achieves Accuracy of 0.7699 and a loss of 0.4708.



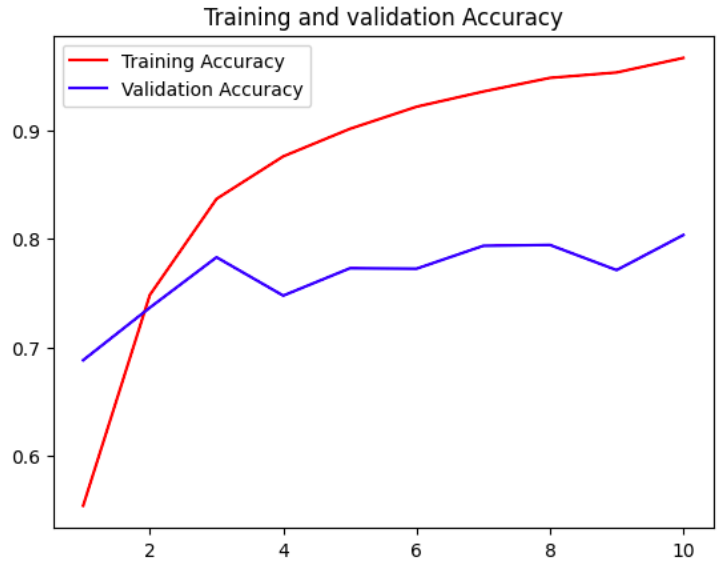
Embedding Layer 10000 Training Samples: A Embedding Layer with 10000 Training Samples achieves Accuracy of 0.7905 and a loss of 0.4473.



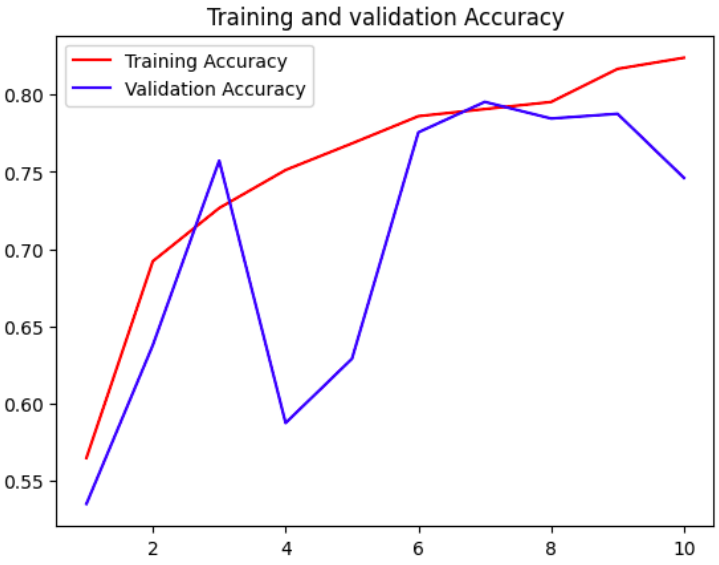
Pretrained Embedding Layer 10000 Training Samples: A Pretrained Embedding Layer with 10000 Training Samples achieves an Accuracy of 0.7846 and a loss of 0.4524.



Embedding Layer 20000 Training Samples: An Embedding Layer with 20000 Training Samples achieves an Accuracy of 0.7718 and a loss of 0.4758.



Pretrained Embedding Layer 20000 Training Samples: A Pretrained Embedding Layer with 20000 Training Samples achieves Accuracy of 0.7885 and a loss of 0.4483.



CONCLUSION AND RECOMMENDATIONS

From this research, it appears that using pretrained word embeddings (like GloVe) provides a distinct edge when the amount of training data is limited. Models that incorporated pretrained embeddings consistently matched or slightly outperformed those equipped with their own trainable embedding layers when the training set included between 100 and 500 samples. For instance, a model employing GloVe embeddings reached a validation accuracy of 0.7977 with a loss of 0.4440, whereas a trainable embedding model recorded a slightly lower accuracy of 0.7844 with a loss of 0.4987. These findings underscore the usefulness of pretrained embeddings for small datasets, given their higher validation accuracy and reduced loss values.

However, as training set sizes expanded—especially once reaching 1000 samples or more—the gap between the two approaches narrowed. A clear illustration of this is at the 1000-sample mark, where both the pretrained model (accuracy 0.7694, loss 0.4775) and the trainable model (accuracy 0.7941, loss 0.4676) registered fairly similar metrics. In other words, with enough data, a model can learn its own meaningful word representations without needing to rely on external embeddings. Moreover, trainable embedding layers showed they could effectively capture specific features when given a sufficiently large dataset.

Notably, pretrained embeddings performed at their best with 1000 training samples, achieving an accuracy of 0.7694 and a loss of 0.4775. Yet the model with a trainable embedding surpassed those figures slightly, scoring 0.79 in accuracy and 0.46 in loss. Beyond that point—at 5000, 10000, and 20000 samples—the difference in performance between pretrained and trainable approaches became almost negligible, with both delivering comparable accuracy and loss results.

In the end, the choice between pretrained or trainable embeddings is largely a matter of how much training data you have. Pretrained embeddings seem beneficial for smaller datasets, while their advantage diminishes once sufficient data is available for the model to learn robust representations independently. Ultimately, the most fitting decision should factor in the particular objectives of the task, the resources at hand, and how important it is to balance model complexity with performance.

RECOMMENDATIONS:

- Use pretrained embeddings (e.g., GloVe) to improve performance with small datasets.
- Allow the model to learn its own embeddings when ample data is available.
- Adjust key parameters like RNN units and embedding dimensions based on data size.
- Choose model complexity that aligns with available training data and computational limits.

REFERENCES

1. Bengio, Y., Courville, A., & Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 35*(8), 1798–1828.

2. Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., & Potts, C. (2011). Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies* (pp. 142–150).

3. Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. In *Proceedings of the Workshop at the International Conference on Learning Representations (ICLR)*.

4. Pennington, J., Socher, R., & Manning, C. D. (2014). GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 1532–1543).