Assignment-4_Group_4
Text & Sequence Data Summary

Submitted by: Nikitha Begari & Kavyasree Bommakanti
Group Number: 4

Introduction:

Text and sequencing refers to several approaches to organizing and evaluating textual data while closely observing word order. This is important because a text's word order can have a big impact on how it is understood and interpreted. For instance, understanding the context and relationships between words frequently necessitates taking into account the sequential patterns in which they occur in natural language processing tasks like sentiment analysis or machine translation. Analysts can extract more precise insights and generate deeper conclusions from the data by preserving the text's sequential pattern.

In sentiment analysis, the significance of language and sequencing is particularly evident. Finding out if a text is neutral, positive, or negative in sentiment is the aim of this job. The words that appear and the order in which they occur have a significant impact on the overall mood.

Consider the words "good" and "not good," for example. Even though they both use the word "good," they have distinct meanings. "Good" implies positivism, while "not good" raises doubt and creates an unpleasant emotion.

Consequently, accurate sentiment analysis interpretation necessitates not just word recognition but also an understanding of word order and textual context. Sentiment analysis algorithms use word order in a sequential manner to determine the sentiment expressed in a text.

Goal:

The objective of the IMDB dataset's binary classification task is to categorize movie reviews as either positive or negative. With 50,000 reviews in the dataset, only the top 10,000 words are considered. Validation comprises 10,000 samples, whereas training samples range from 100 to 100,000. Placing the embedding layer and using a pre-trained embedding model come first in the data preparation process. To ensure the best possible model selection and tweaking, a variety of methodologies are investigated to measure performance. With the use of this method, a strong deep learning model for sentiment analysis can be developed, which will enable IMDB movie reviews to be classified as either positive or negative.

Data-Preprocessing:

In the process of preparing the dataset, each review is converted into word embeddings, representing each word by a fixed-size vector. However, there's a constraint of 10,000 samples in this meticulous procedure. Additionally, a numerical sequence is generated from the reviews, where individual numbers correspond to specific words rather than complete phrases. Yet, the neural network's input format doesn't directly support this sequential list of numbers. To tackle this issue, tensors need to be constructed using the numerical sequence. This list of integers could potentially form the basis for creating a tensor with an integer datatype and structured as (samples, word indices). However, ensuring consistent sample lengths is essential. This involves techniques like padding reviews with dummy words or numerical placeholders to standardize the length across all samples.

Method:

In this study, two different approaches were explored for generating word embeddings using the **IMDB dataset:**

- Custom-trained embedding layer
- Pre-trained word embedding layer using the GloVe model.

The GloVe model, widely used for word embeddings, was employed in our research and trained on extensive textual datasets.

To evaluate the effectiveness of different embedding strategies, two distinct embedding layers were used with the IMDB review dataset. One was a custom-trained layer, while the other utilized a pre-trained word embedding layer.

The accuracy of these two models was compared across varying training sample sizes, including 100, 5000, 1000, and 10,000.

Initially, we developed a custom-trained embedding layer using the IMDB review dataset. Each model was then trained across a range of dataset samples, and its accuracy was assessed using a dedicated testing set.

After evaluating the precision results, we compared them with those obtained from a model that underwent similar testing across varied sample sizes but incorporated a pre-trained word embedding layer.

CUSTOM-TRAINED EMBEDDING LAYER:

1. Custom-trained embedding layer with a training sample size of 100
2. Custom-trained embedding layer with a training sample size of 5000
3. Custom-trained embedding layer with a training sample size of 1000
4. Custom-trained embedding layer with a training sample size of 10000.

Depending on the size of the training sample, the accuracy of the custom-trained embedding layer varied from **97.50% to 100%.** A training sample size of 100 gave the best accuracy of 100%.

PRE-TRAINED WORD EMBEDDING LAYER:

1. A pre-trained word embedding layer with a training sample size of 100.
2. A pre-trained word embedding layer with a training sample size of 5000
3. A pre-trained word embedding layer with a training sample size of 1000.
4. Pre-trained word embedding layer with a training sample size of 10000

- Depending on the size of the training sample, the pre-trained word embedding layer (GloVe) has accuracy levels ranging from **90% to 100%.**
- A training sample size of 100 produced the best accuracy, which was 100%. But the model with pre-trained embeddings tended to overfit faster with larger training sample numbers, which resulted in lower accuracy.
- These findings demonstrate how difficult it is to choose the best course of action with confidence because it depends so much on the particular requirements and limitations of the work at hand.

Results:

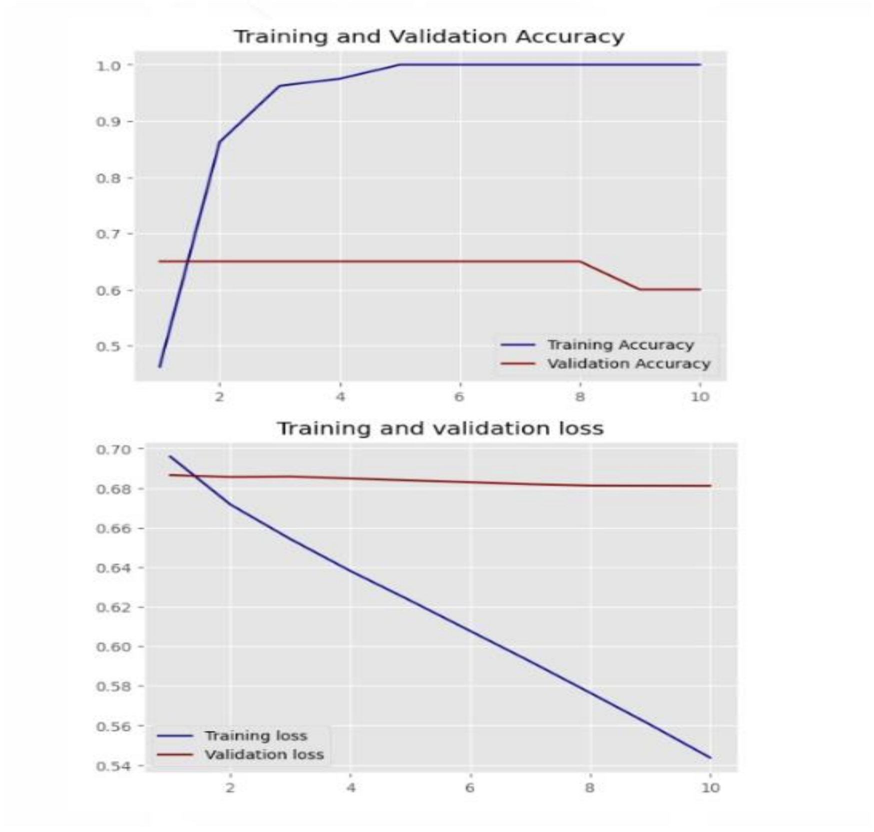
CUSTOM-TRAINED EMBEDDING LAYER:

Embedding Technique	Training Sample Size	Training Accuracy (%)	Test loss
Custom-trained Embedding layer	100	100	0.69
Custom-trained Embedding layer	5000	98.33	0.37
Custom-trained Embedding layer	1000	97.50	0.66
Custom-trained Embedding layer	10000	97.85	0.34

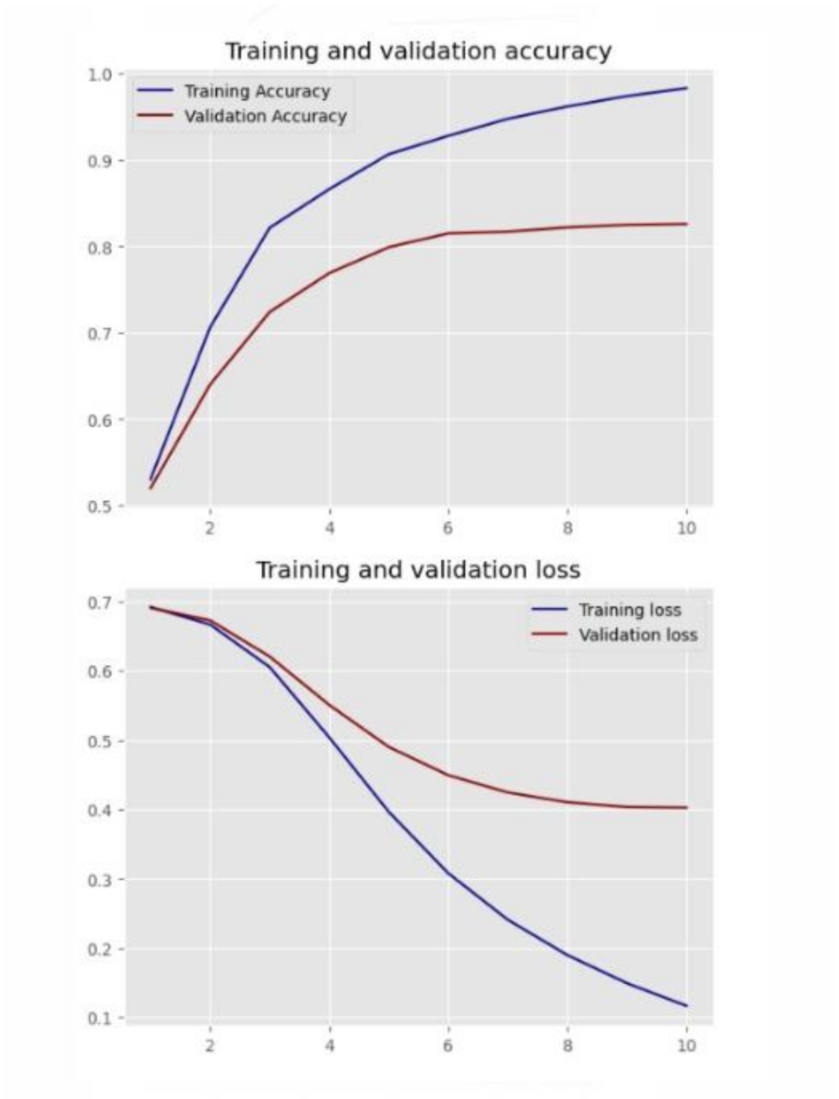
PRE-TRAINED WORD EMBEDDING LAYER:

Embedding Technique	**Training Sample Size**	**Training Accuracy (%)**	**Test loss**
Pretrained word Embedding layer (GloVe)	100	100	0.80
Pretrained word Embedding layer g (GloVe)	5000	94.28	1.43
Pretrained word Embedding layer (GloVe)	1000	95.40	0.90
Pretrained word Embedding layer (GloVe)	10000	90.19	1.51

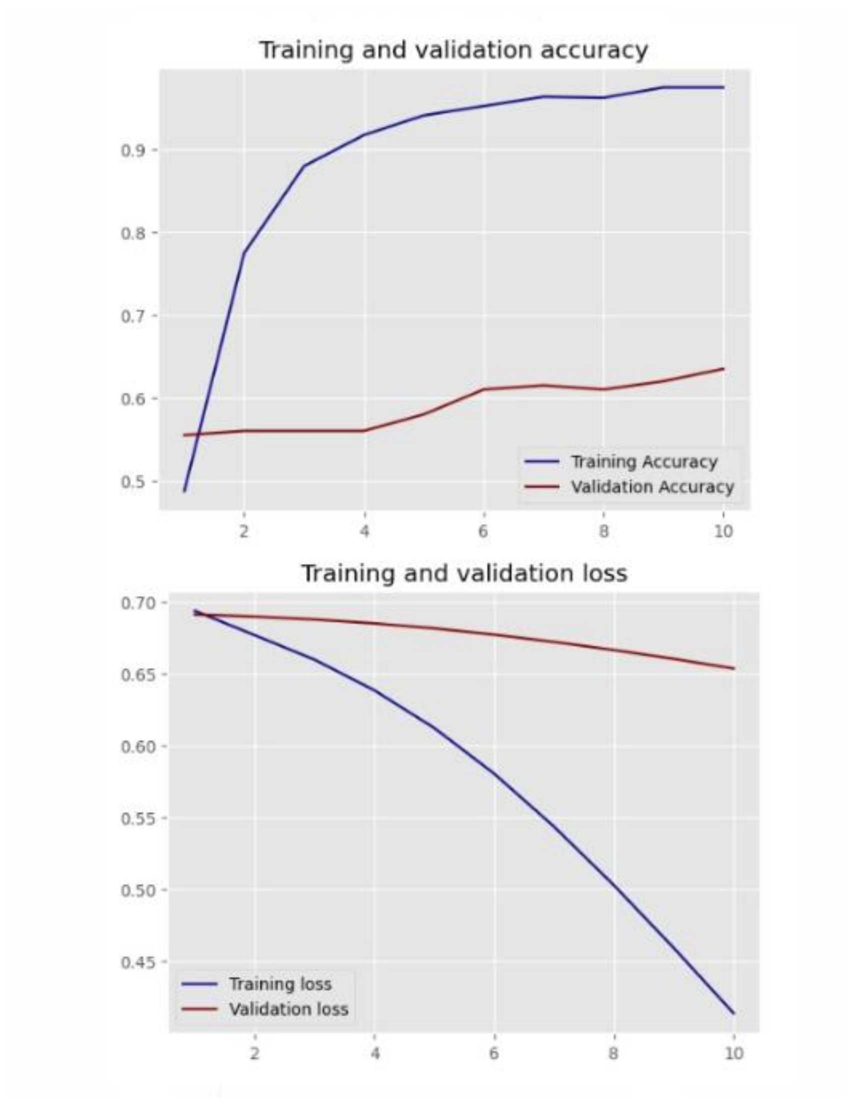
Custom-trained embedding layer- Training Sample Size-100



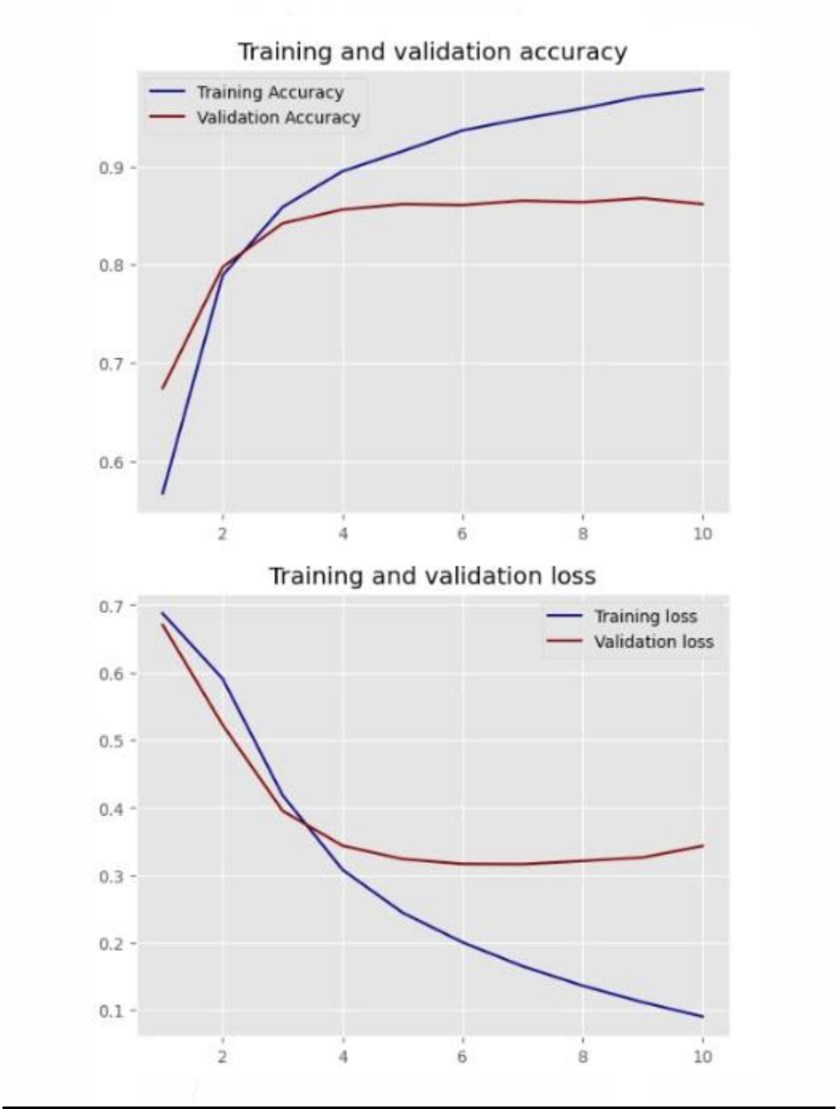
Custom-trained embedding layer- Training Sample Size-5000



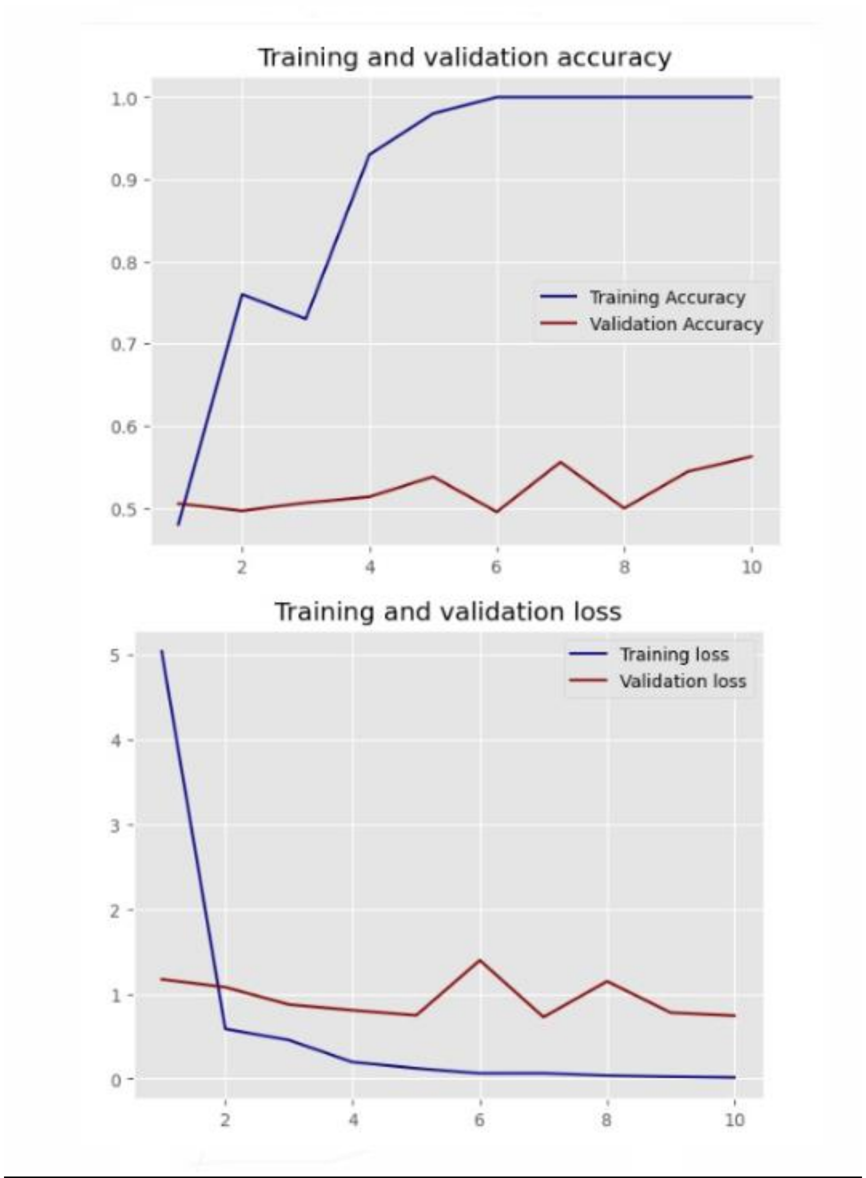
<u>Custom-trained embedding layer- Training Sample Size-1000</u>



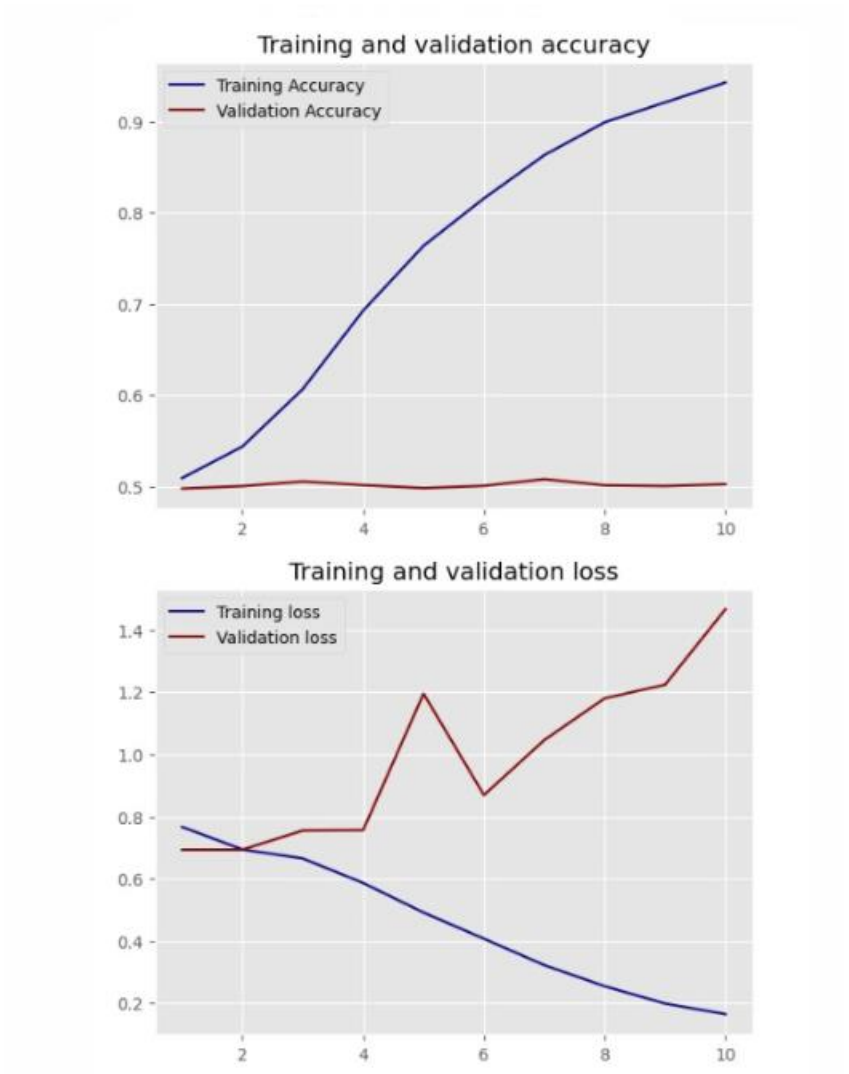
<u>Custom-trained embedding layer- Training Sample Size-10000</u>



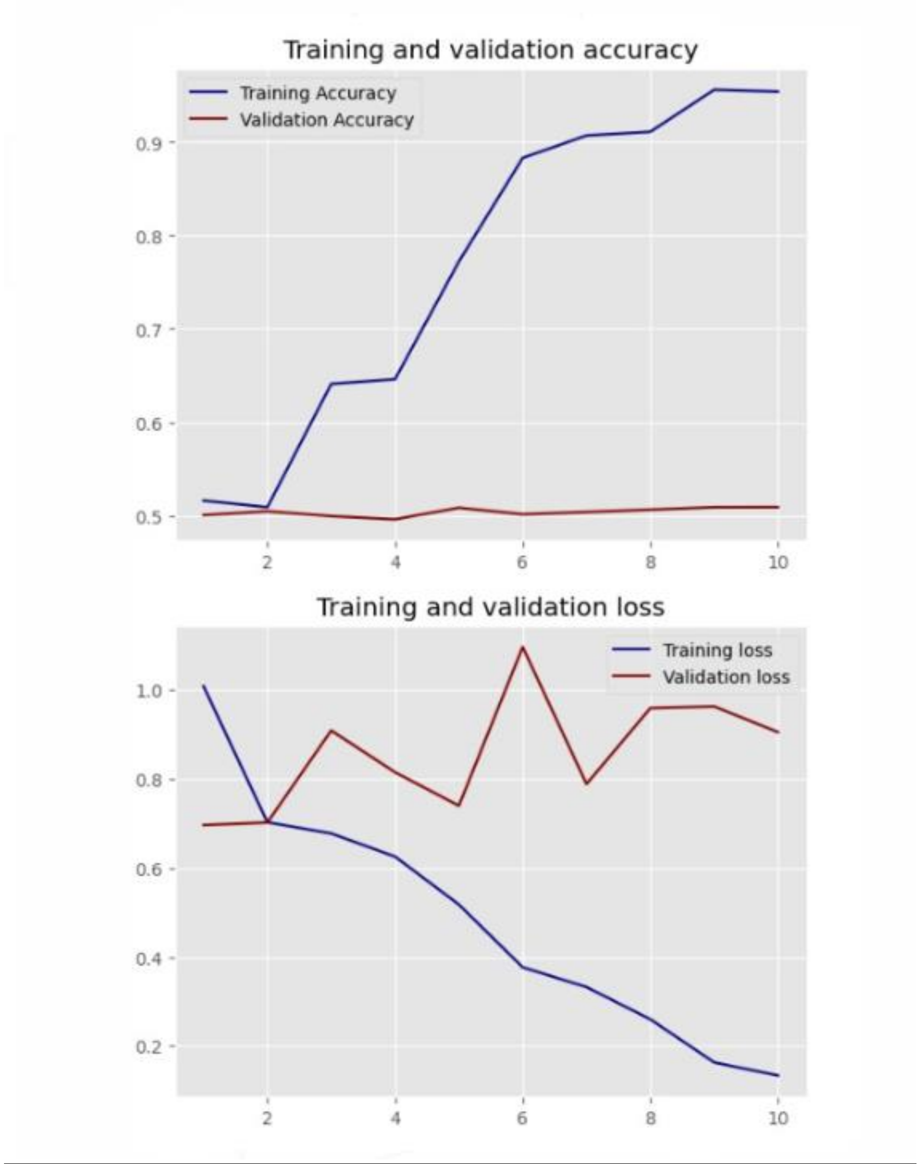
Pretrained word embedding (GloVe)- Training Sample Size-100



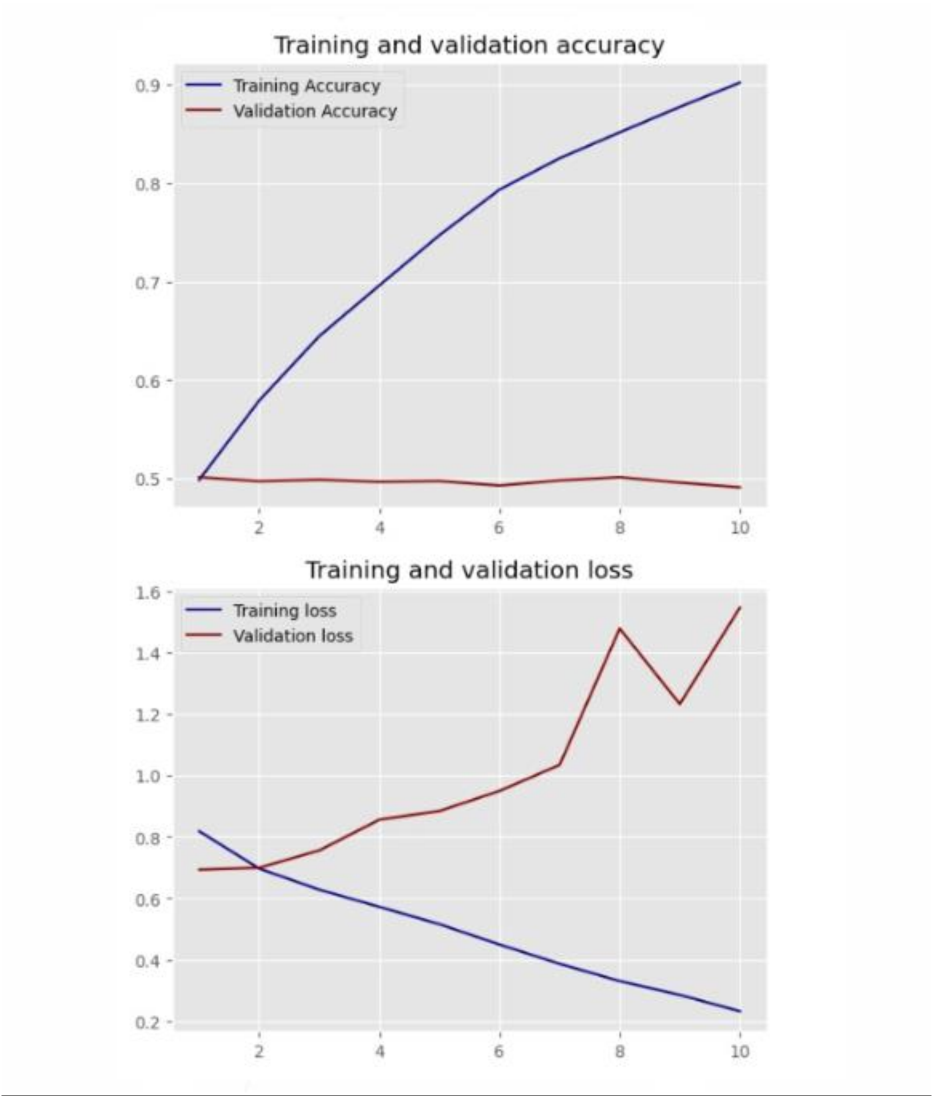
<u>Pretrained word embedding (GloVe)- Training Sample Size-5000</u>



<u>Pretrained word embedding (GloVe)- Training Sample Size-1000</u>



<u>Pretrained word embedding (GloVe)- Training Sample Size-10000</u>



<u>Conclusion:</u>

- In this study, the pre-trained word embedding layer was regularly outperformed by the custom-trained embedding layer, especially when training with bigger sample numbers.
- The model's great accuracy on both the training and validation datasets demonstrates how well-balanced its complexity and generalizability are. The pre-trained model's performance of 95.40% with 10000 dimensions demonstrates the model's capacity to improve the summarization and regularization of unseen data.

- The accuracy value improved as the training size was increased, and the loss decreased as well, indicating that the model is learning and getting better as we raise the training size.
- As the number of training samples increased, the model's training loss decreased, indicating that it could learn more effectively with larger amounts of data.
- Using dropout or fine-tuning the model, standard regularization techniques like masking and experimenting with different embedding dimension values might improve the model's performance.
- The findings of the two tested models are comparable, indicating that no single technique is effective for all embedding layer models. Many models should be assessed in order to choose the optimal model for a given dataset. Performance can be increased by using regularization, varying embedding sizes experimentally, creating LSTM layers, training pre-trained embeddings with a given dataset, and increasing the training sample to a certain point.
- This suggests that the model with better accuracy levels was trained on the particular dataset, which allowed it to pick up on more subtle patterns and traits found in the IMDB ratings. However, it's important to recognize that the pre-trained word embedding layer is still useful and might be a "better choice" in some circumstances.
- For example, the pre-trained embeddings provide a practical and effective option when working with a short training sample size or when computational resources are constrained.
- Although greater sample sizes may increase the danger of overfitting, the pre-trained embeddings offer a reliable foundation and can aid in bootstrapping the model's learning process.
- As a result, the decision between pre-trained and custom-trained embeddings ultimately comes down to the particular needs, limitations, and objectives of the task at hand.
- This emphasizes how crucial it is to take a variety of factors into account when developing and deploying deep learning models for tasks involving natural language processing.