SENTIMENT ANALYSIS ON IMDB REVIEWS

Objective:

The goal of the binary classification task for the IMDB dataset is to divide movie reviews into positive and negative categories. The dataset comprises 50,000 reviews; 10,000 words out of the top 10,000 are analyzed; training samples are restricted to 100, 5000, 1000, and 100,000 samples; 10,000 samples are validated. All of the data has been prepared. After then, a pre-trained embedding model and the data are combined, and various strategies are tested to gauge performance.

Data Preprocessing:

- As part of the dataset preparation process, every review is transformed into a set of word embeddings, where each word is represented by a fixed-size vector. The maximum sample size that is applicable is 10,000. In addition, rather than using a string of words, a set of integers representing individual words was generated from the reviews. Even if I have the list of numbers, the input of the neural network is inappropriate for it.
- Tensors need to be constructed using the numbers. One might create a tensor with integer data type and form (samples, word indices) using the integer list. In order for me to do that, I have to make sure that every sample is the same length, which means I have to use dummy words or numbers to ensure that every review is the same length.

Procedure:

For this IMDB dataset, I looked into two distinct methods for generating word embeddings:

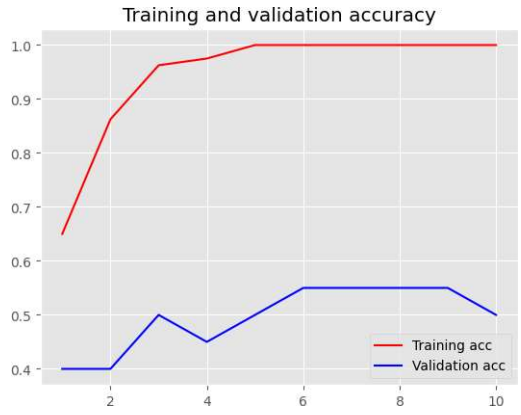
1. Custom-trained embedding layer
2. pre-trained word embedding layer using the GloVe model.

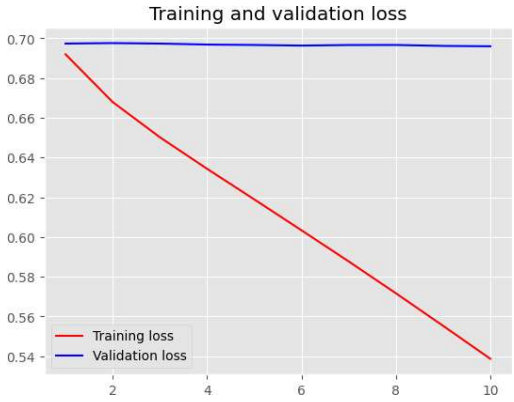
The widely used pre-trained word embedding model GloVe, which we utilized in our work, istrained on large amounts of textual data.

- To evaluate the effectiveness of various embedding tactics, I used the IMDB review dataset and two distinct embedding layers, one with a custom-trained layer and the other with a pre-trained word embedding layer. I contrasted the two models' accuracy using training sample sizes of 100, 5000, 1000, and 10,000.
- Using the IMDB review dataset, we started by building a specially trained embedding layer. After training each model on many dataset samples, we used a testing set to determine its accuracy. Next, we compared these precisions to a model with a pre-trained word embedding layer that had already been evaluated on various sample sizes.

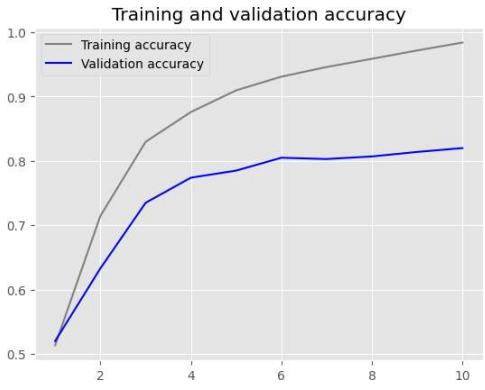
CUSTOM-TRAINED EMBEDDING LAYER

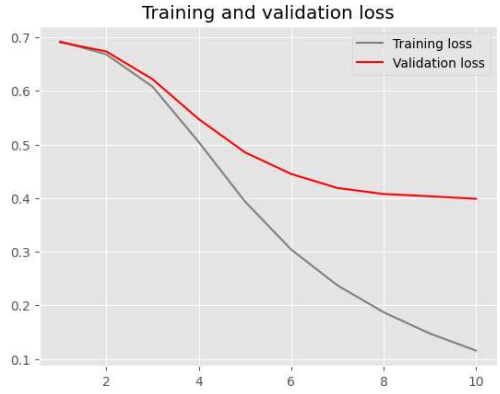
1. Custom-trained embedding layer with training sample size = 100



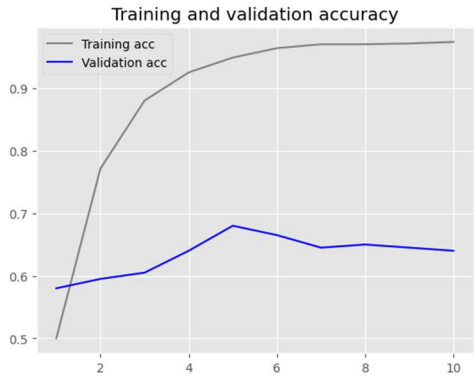


2. Custom-trained embedding layer with training sample size = 5000

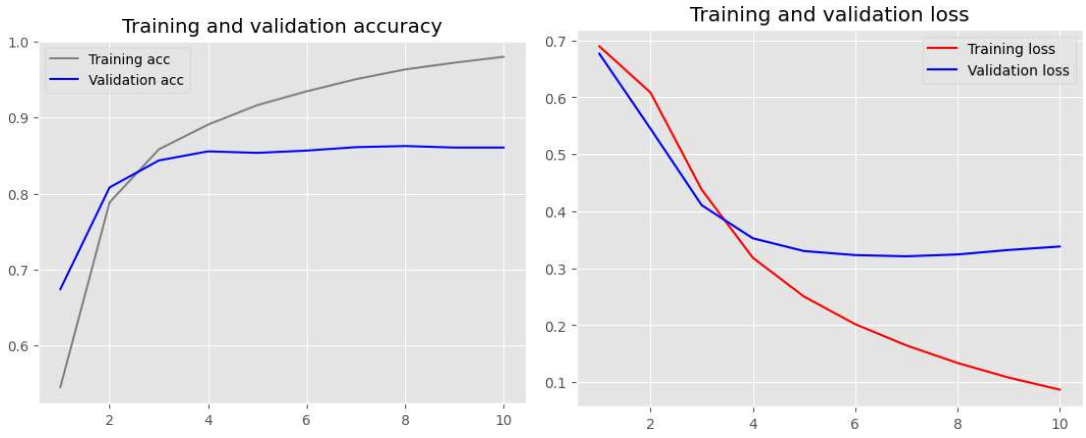




3. Custom-trained embedding layer with training sample size = 1000



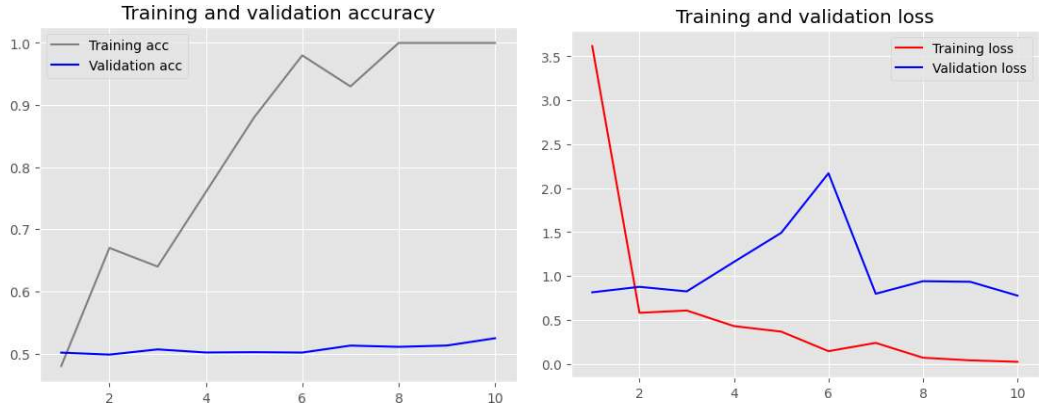
4. Custom-trained embedding layer with training sample size = 10000



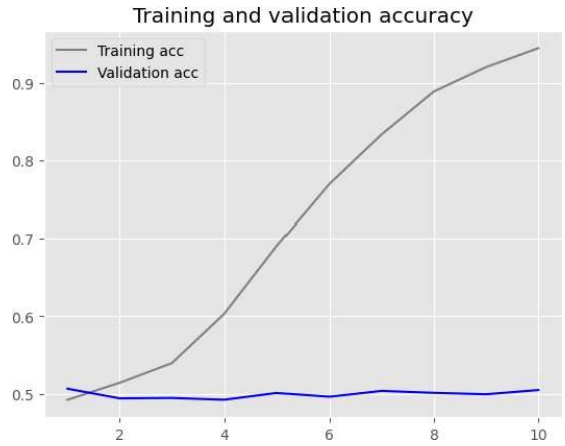
Depending on the size of the training sample, the accuracy of the custom-trained embedding layer varied from 97.3% to 100%. The training sample size of 100 produced the best accuracy.

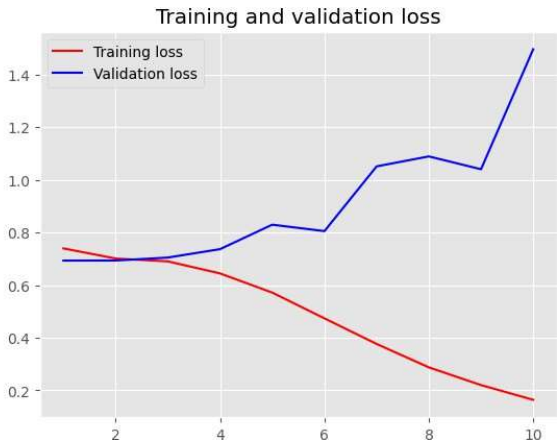
PRETRAINED WORD EMBEDDING LAYER

1. pre-trained word embedding layer with training sample size = 100

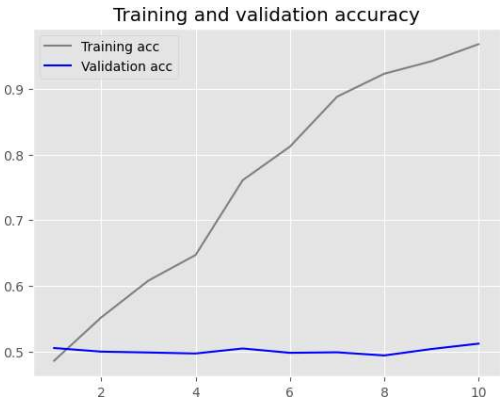


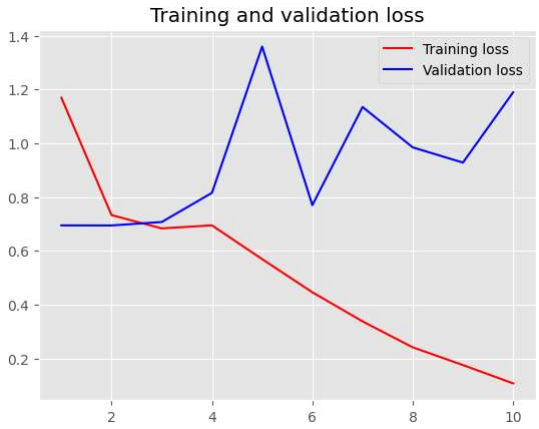
2. pretrained word embedding layer with training sample size = 5000



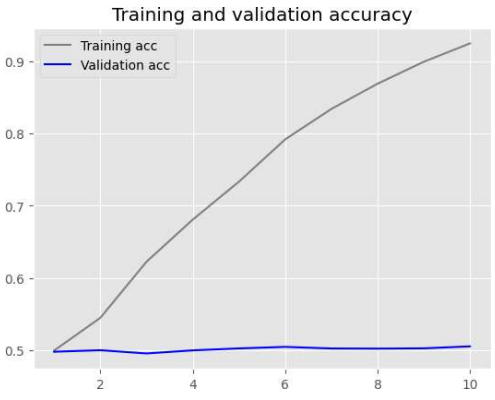


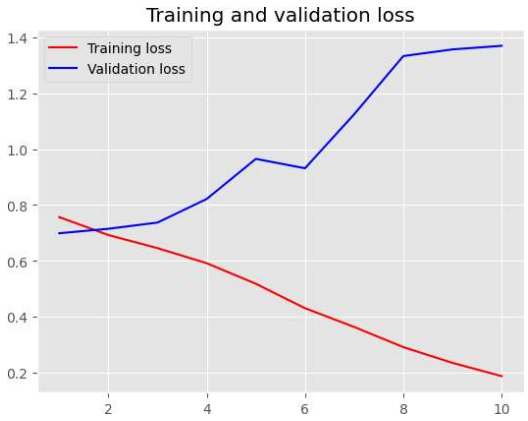
3. pretrained word embedding layer with training sample size = 1000





4. pretrained word embedding layer with training sample size = 10000





The accuracy of the pretrained word embedding layer (GloVe) varied according on the size of the training sample, ranging from 92% to 100%. With 100 training samples, the most accurate result was achieved. Furthermore, employing the pretrained embeddings with larger training sample sizes causes the model to rapidly overfit, which lowers accuracy. These results make it challenging to decide which strategy is the "best" to utilize with confidence because it depends on the requirements and limitations of the task at hand.

Results:

Embedding Technique	Training Sample Size	Training Accuracy (%)	Test loss
Custom-trained embedding layer	100	100	0.69
Custom-trained embedding layer	5000	98.40	0.37
Custom-trained embedding layer	1000	97.3	0.68
Custom-trained embedding layer	10000	98	0.34
Pretrained word embedding (GloVe)	100	100	0.79
Pretrained word embedding (GloVe)	5000	94.48	1.43
Pretrained word embedding (GloVe)	1000	96.80	1.10
Pretrained word embedding (GloVe)	10000	92.48	1.41

Conclusion:

Nevertheless, in this experiment, the custom-trained embedding layer performed better than the pre-trained word embedding layer, especially when training with more training sample numbers. If computational resources are restricted and a modest training sample size is required, the pre-trained word embedding layer might be a "better choice" despite the risk of overfitting.