**Assignment 4**

**Text And Sequence Data**

**Summary:**

The goal of the binary classification project based on the IMDB dataset is to decide whether the give movie will have favorable review or not. The dataset consists of 50,000 reviews. 10,000 most popular words are extracted. Training samples on 100, 500, 1000, and 100000 are trained and validated on 10,000 samples with a cutoff after 150 words. The data goes through pre - processing. Then for the next step include data fed into the pre-trained embedding model and the embedding layer and contrast the results from different methods for measuring performance.

The RNN model is then trained on the IMDB dataset, which contains the reviews and at the end a binary classification is made concerning the positive and negative reviews. This is the system of a sequential neural network method that functions and processes with one direction only. RNN features the recurrent connection which helps to deal with the sequential data.

**Word Embedding:**

Unlike one-hot encodings, word embeddings are a technique for encoding words as vectors in very high-dimensional spaces. These vectors represent the semantic connections between words and make machine algorithms to understand and interpret concept of words and sentences of them more effectively.

**Pre trained word embedding:**

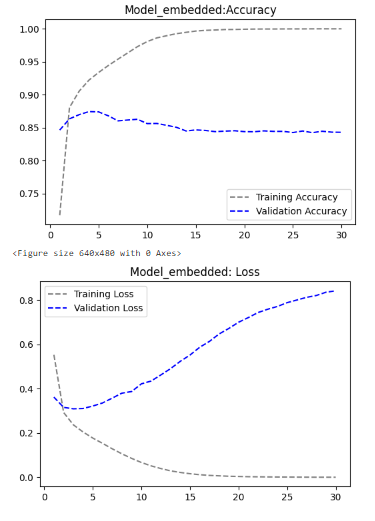
Resembling these models of neural networks are word embeddings that have been learned on a huge corpus of unlabeled textual data. Words2Vec, GloVe, and FastText are models that are frequently used to train these methods and can process huge text fragments. Initially, pre-trained embeddings can be used as the set of features in different sort of natural language processing tasks and the timing and resources needed to train the model from the beginning on the same dataset is completely avoided.

**Results:**

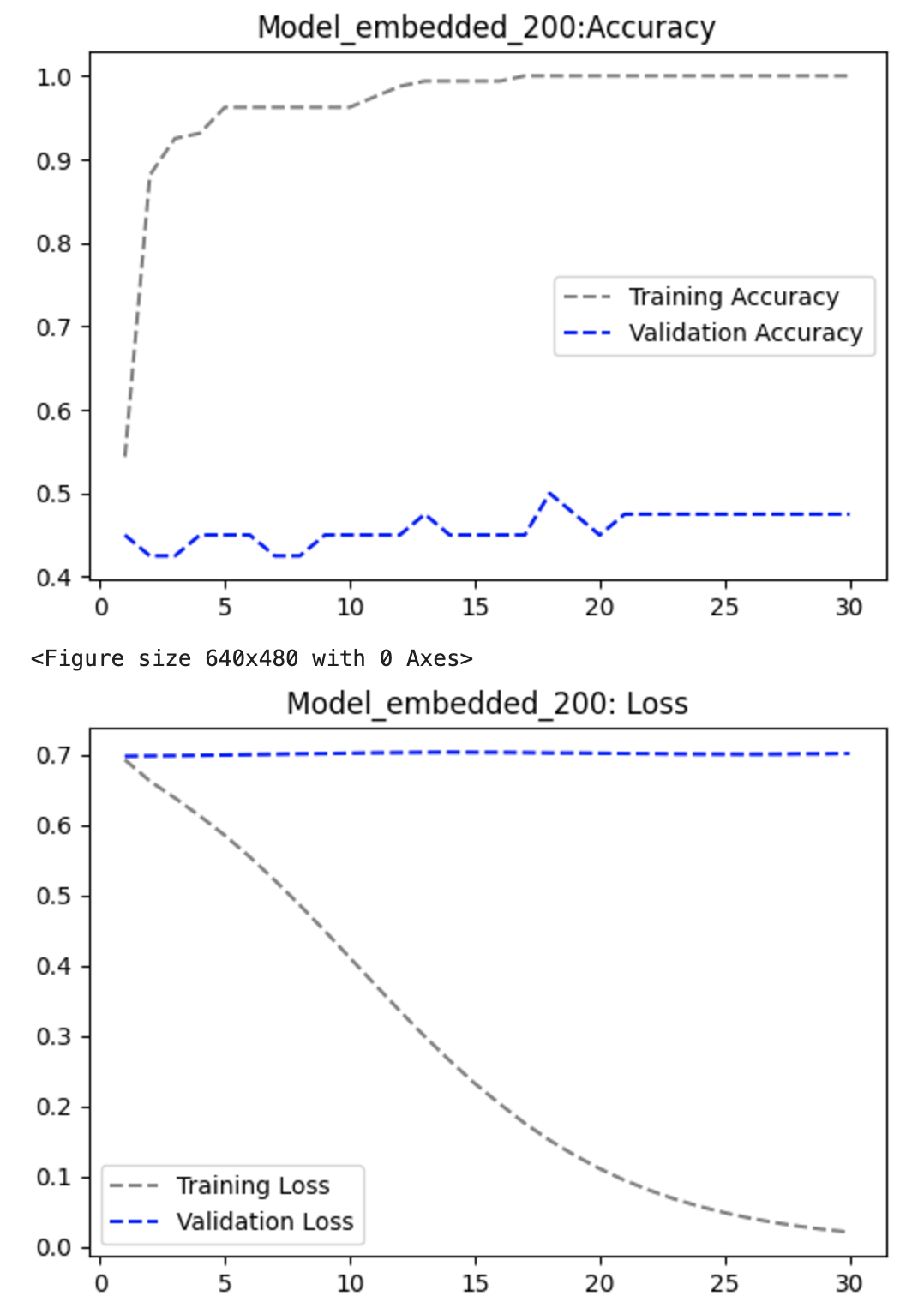
Initially using only, the embedded layer with different training sizes:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Review Words | Max Length | Training Size | Accuracy | Loss |
| 150 | 10000 | Minimum | 84.50 | 81 |
| 150 | 10000 | 200 | 47.50 | 70.20 |
| 150 | 10000 | 500 | 54.00 | 84.89 |
| 150 | 10000 | 1000 | 55.00 | 96.40 |
| 150 | 10000 | 2000 | 72.50 | 68.90 |

**A single embedded layer:**



**A single embedded layer with 200 training size:**



**A single embedded layer with 500 training size:**

A graph of a graph of a training loss

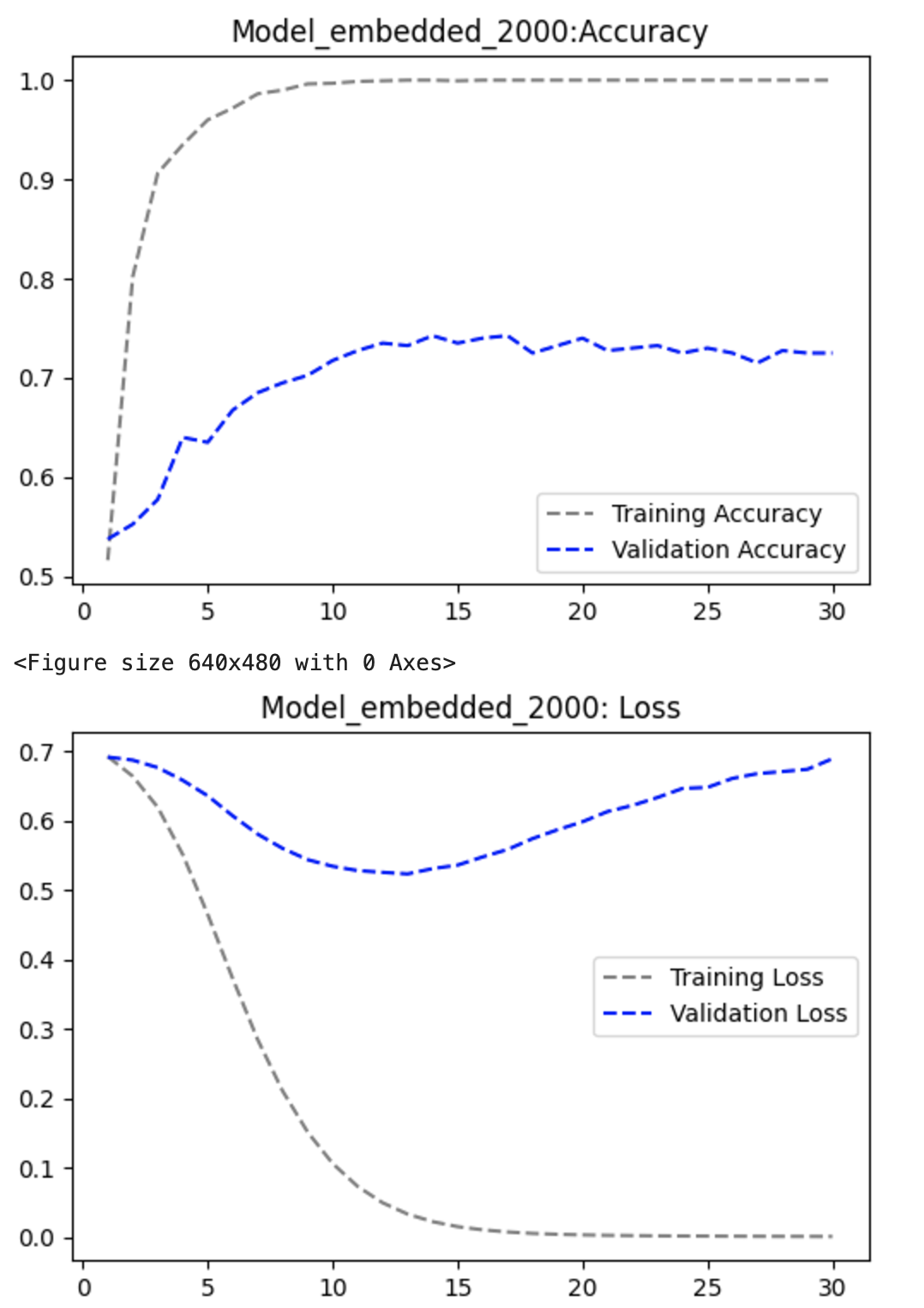
Description automatically generated with medium confidence

**A single embedded layer with 1000 training size:**

A graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of

Description automatically generated

**A single embedded layer with 2000 training size:**



High accuracy might be achieved rather quickly which is an indication of overfitting, especially for sample sizes that are much smaller for the only embedded layers models, with the model with 500 training size having slower but more consistent convergence.

Now we will be using pretrained word embedding; that is using an embedded layer and a convolution 1D with different embedding dimensions:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Review Words | Max Length | Embedding Dimension | Accuracy | Loss |
| 150 | 10000 | 10 | 52.66 | 73.96 |
| 150 | 10000 | 50 | 81.48 | 41.94 |
| 150 | 10000 | 10000 | 92.24 | 26.00 |

**Pretrained model with embedded layer and convolution 1D:**

A graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of

Description automatically generated

**Pretrained model with 50 embedded dimensions:**

A comparison of a graph

Description automatically generated

**Pretrained model with 10000 embedded dimensions:**

A graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of

Description automatically generated

A pretrained word embedding with both Embedding and Conv1D layers seems to be suffering from the problem of an overfit, this can be caused by the network complexity and the smaller size of our dataset.

Later for embedding dimension we Incorporated simple architectural designs or use dropout approach to achieve regularization and hence reduction in overfitting, thus improving in accuracy. But still we didn’t get the optimal accuracy with least loss. Because the loss should be least and accuracy should be high for a best performing model.

A large embedding dimension of 10,000 is utilized for word representation. Dropout is applied after each convolutional layer to mitigate overfitting, and

We also tried two different models:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Review Words | Max Length | Model | Accuracy | Loss |
| 150 | 10000 | RNN | 82.32 | 42.55 |
| 150 | 10000 | LSTM | 50.86 | 69.30 |

RNN analyzes the series of word vectors to identify features crucial for sentiment analysis. With 64 RNN units, it can discern 64 distinct aspects of the sequential data. The model efficiently categorizes IMDB movie reviews as either positive or negative. However, Simple RNN struggles with long sequences in our dataset, prompting the utilization of LSTM and GRU layers to address the vanishing gradient problem.

As we can see by using LSTM model, there is a slight increase in loss and decrease in accuracy.

**Conclusion:**

* This high accuracy on both the training and the validation dataset shows a well-balanced complexity of the model and its generalization capacity. The pre-trained model performance with containing dimensions of 10000 again shows the model's ability to get the unseen data summarizing and regularization to be better.
* The accuracy value kept getting better with the increasing of the training size and the less the loss was which depicts that as we increase the training size the model is learning and improving.
* The training loss of the model reduced with the increase in the number of training samples, evident of the model being able to learn better with more data.
* Standard regularization methods, like masking and the try of different dimension numbers of the embeddings with the application of dropout or fine-tuning the model can enhance the model's performance.
* The tested models of two have the similar results, means there is not one method which works for all the models of embedding layer. To determine the best model for a specific dataset quite a number of models should be evaluated. Regularization being used, changing embedding sizes experimentally, building LSTM layers, training pre-trained embeddings with a given provided dataset, and increasing the training sample to a particular level all can be used to boost performance.