KENT STATE UNIVERSITY

ASSIGNMENT – 4

TEXT DATA

BY

ALLEN RICHARDS PERIANAYAGAM

IMDB Sentiment Classification with Deep Learning

1. Introduction

In this project, the goal was to build a sentiment classification model to classify movie reviews from the IMDB dataset as either positive or negative. We explored different architectures using **Bidirectional LSTMs** and experimented with various embedding strategies, including both **pretrained word embeddings** (such as GloVe) and **randomly initialized embeddings**. We aimed to evaluate the effectiveness of using pretrained word embeddings compared to training embeddings from scratch, as well as assessing the impact of model architecture on performance.

---

2. Data Overview

We used the **IMDB dataset**, which is widely used in sentiment analysis tasks. The dataset consists of 50,000 movie reviews, with 25,000 reviews for training and 25,000 for testing. Each review is labeled either **positive** or **negative**. For this project, we focused on the **binary classification** aspect of the task, where the goal is to predict the sentiment of each review.

- **Number of classes**: 2 (positive, negative)
- **Training data**: 25,000 reviews
- **Testing data**: 25,000 reviews
- **Features**: Text reviews (preprocessed into tokenized words)

We used **Keras** and **TensorFlow** to build and evaluate our models, which were trained using **Bidirectional LSTM** layers for capturing long-term dependencies in the text data.

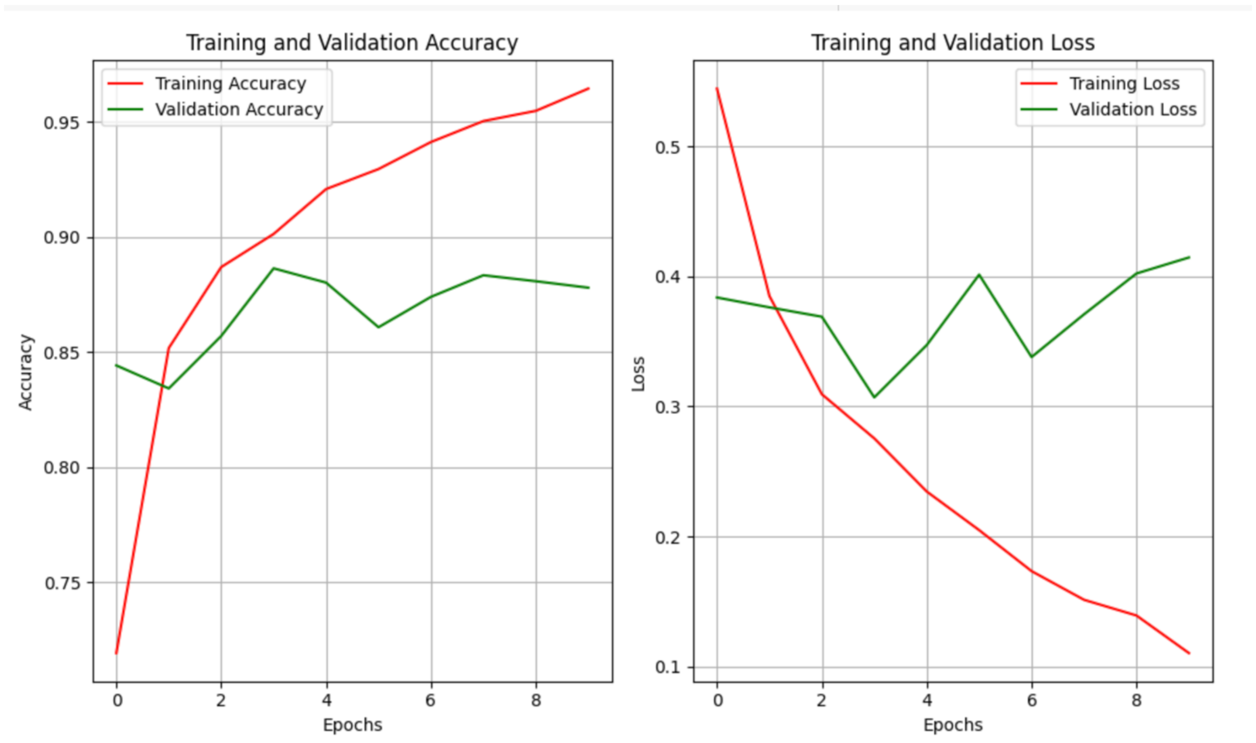
---

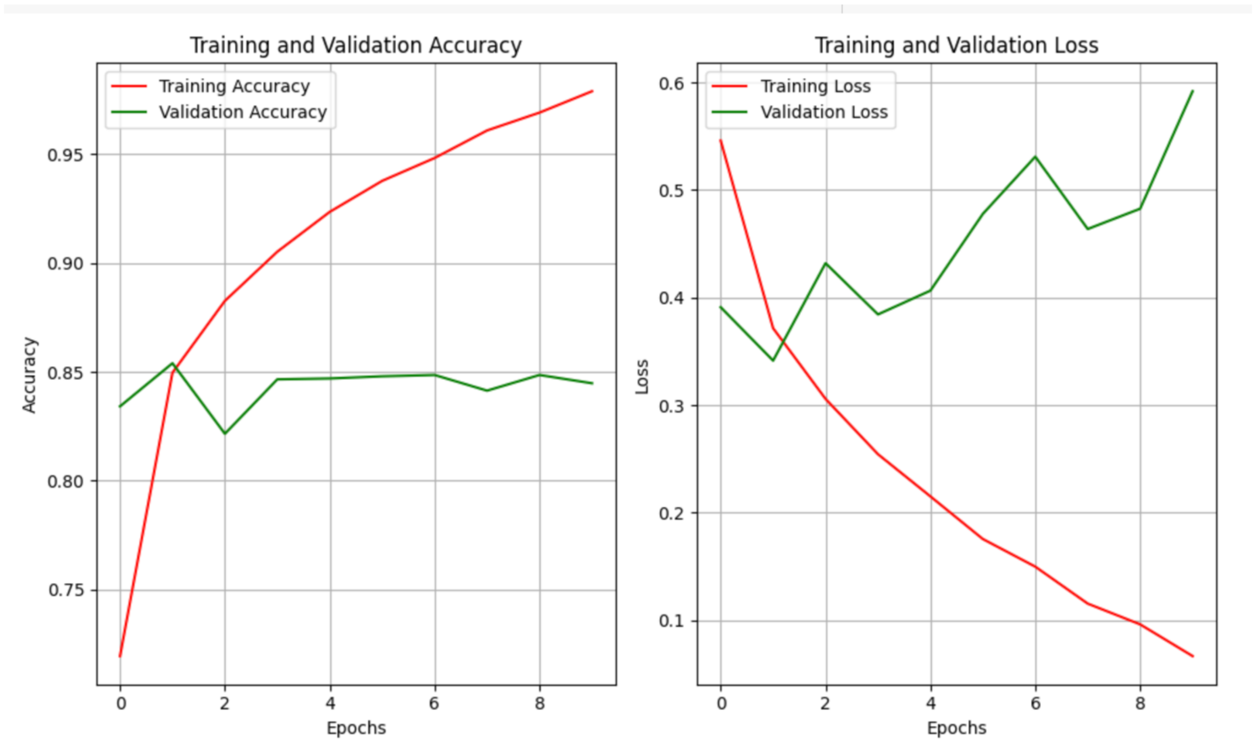
3. Base Model

A **base model** was developed to serve as a benchmark for evaluating improvements or declines in performance when compared to adjusted models. The base model utilized a **Sigmoid activation function** in the output layer and was trained on the **aclImdb/train** dataset using an **80-20 split**, where 20% was allocated for validation. The **test set** was sourced separately from **aclImdb/test**. Key architectural components included **128-dimensional embeddings**, a **Bidirectional LSTM layer with 32 units**, a **dropout rate of 0.5**, and a **final dense output layer** with a single unit and **Sigmoid activation**. The model was trained using a **vocabulary size of 20,000 tokens**, with a **maximum sequence length of 600** and a **batch size of 32**. The **RMSprop optimizer** and **binary crossentropy** loss function were employed, along with a **ModelCheckpoint callback** to save the best performing model during the training process, which spanned **10 epochs**.

The model architecture began with an **input layer** receiving sequences of integer tokens, followed by an **embedding layer** that transformed these tokens into 128-dimensional vectors. This was fed into a **Bidirectional LSTM layer** with 32 units, followed by a **dropout**

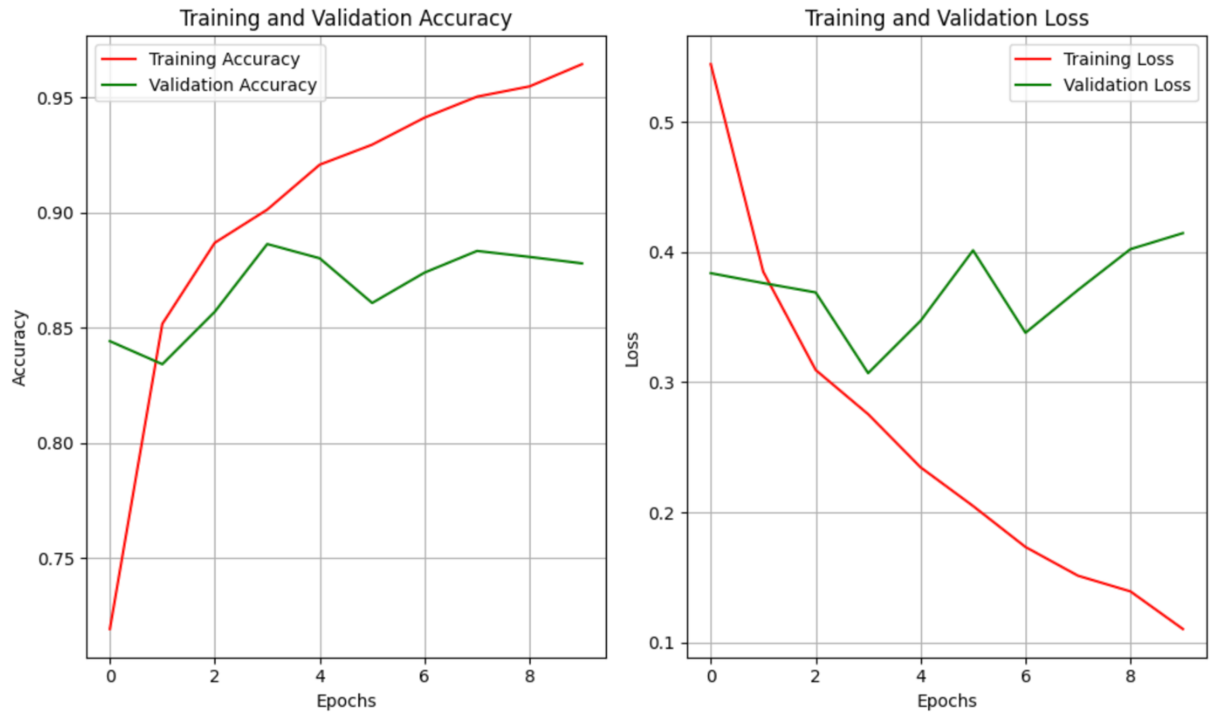
layer (0.5 rate) to prevent overfitting, and concluded with a **dense output layer** using a **Sigmoid activation**. The base model achieved an **accuracy of 0.872**.



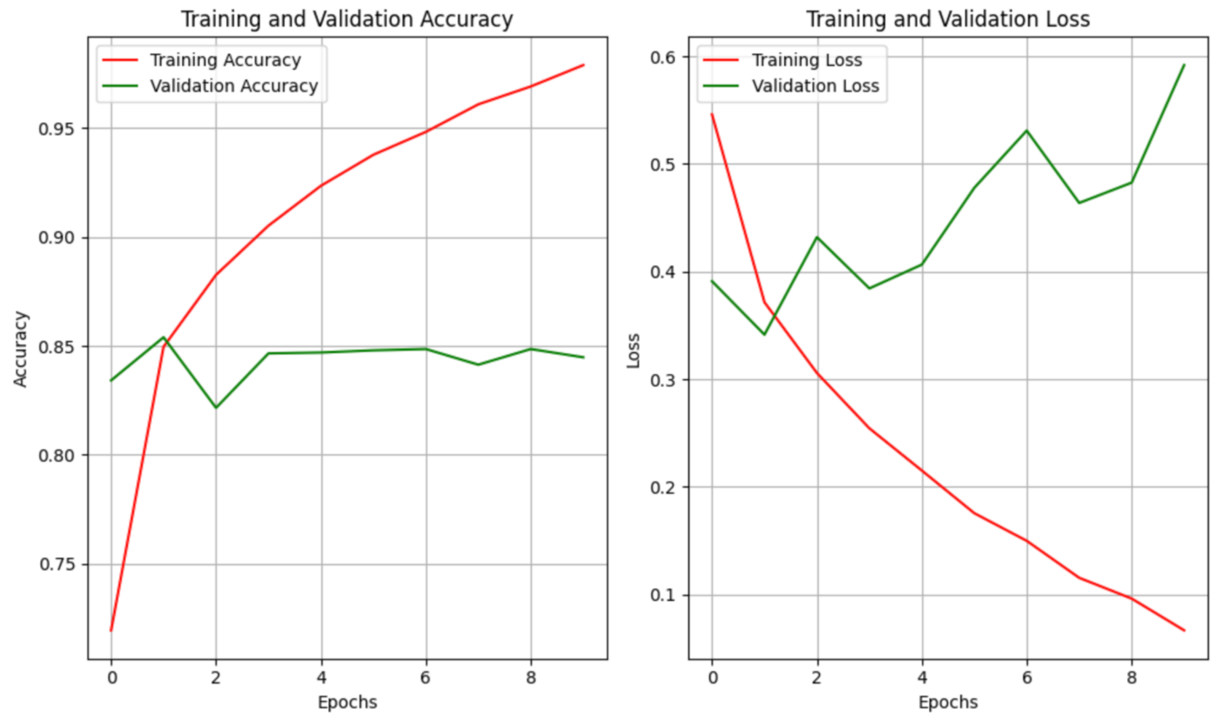


NEXT GRAPHS - COMPARISON

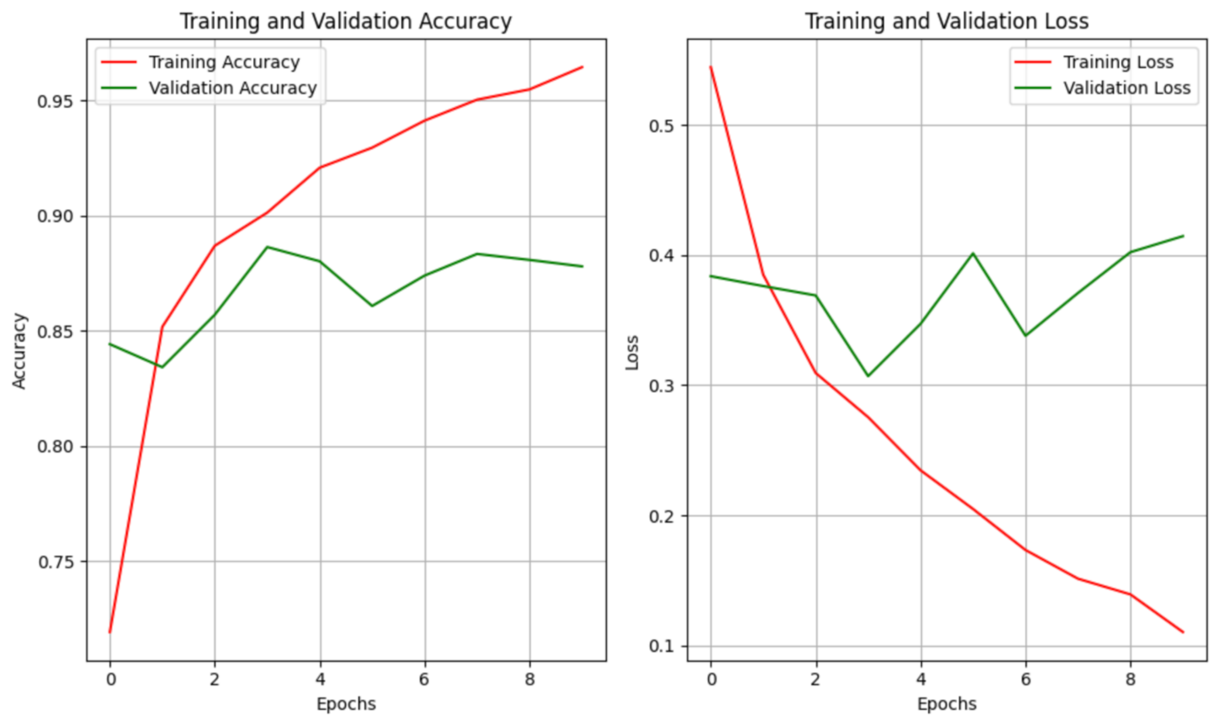
CUT OFF REVIEW 150 WORDS



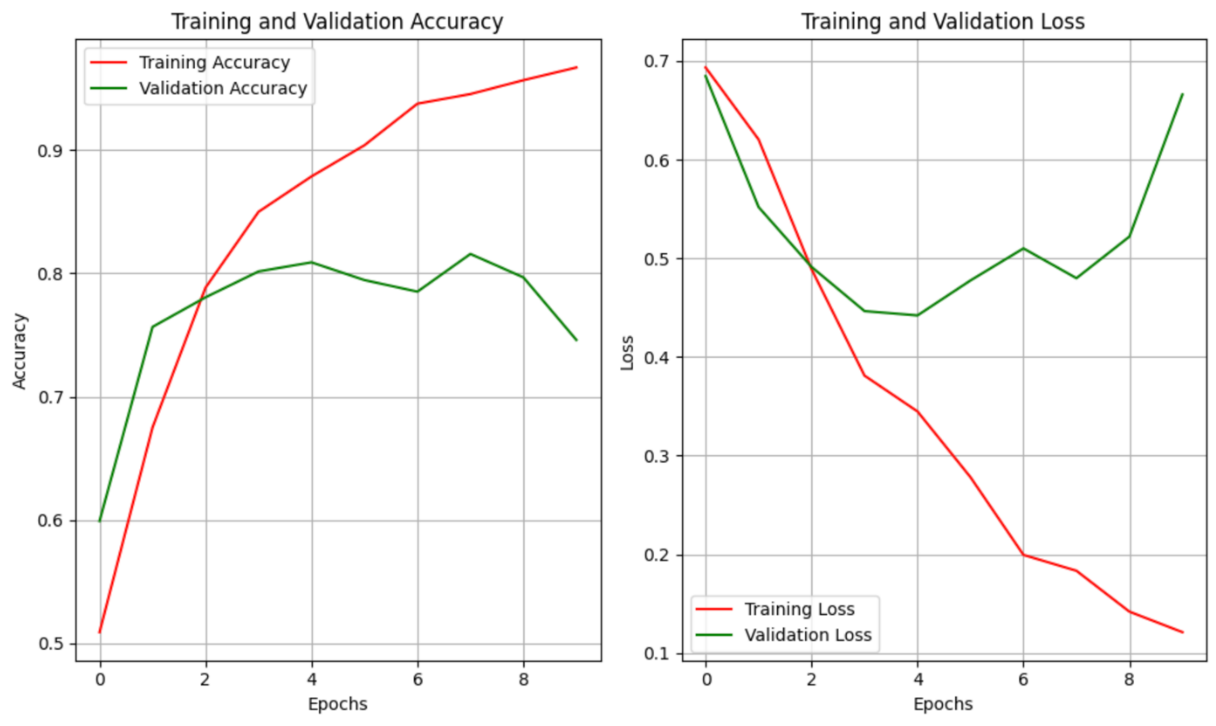
BASE MODEL



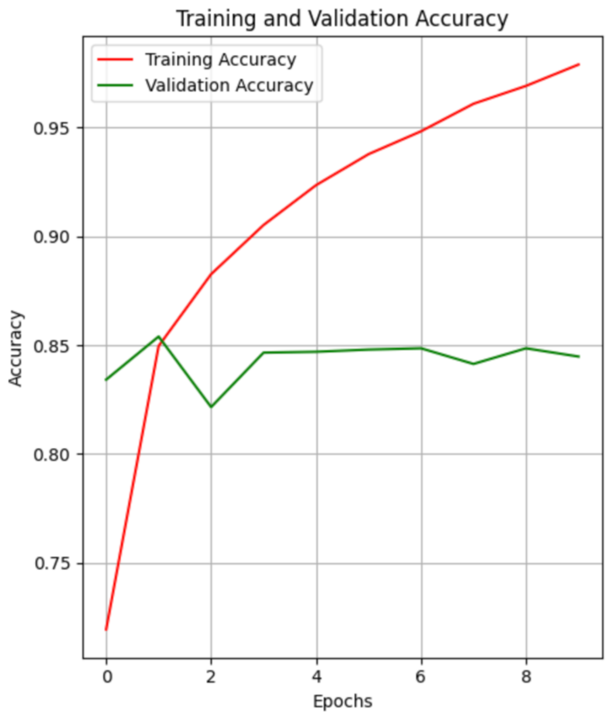
BASE MODEL

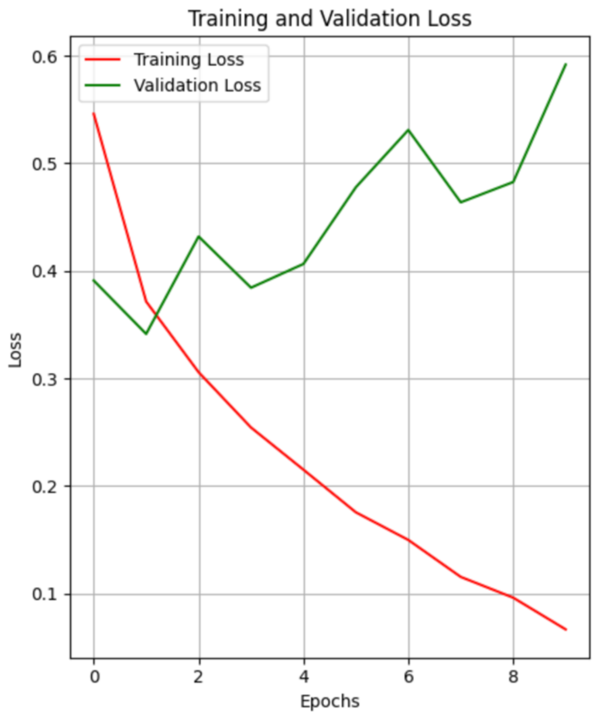


RESTRICT TRAINING SAMPLE – 100

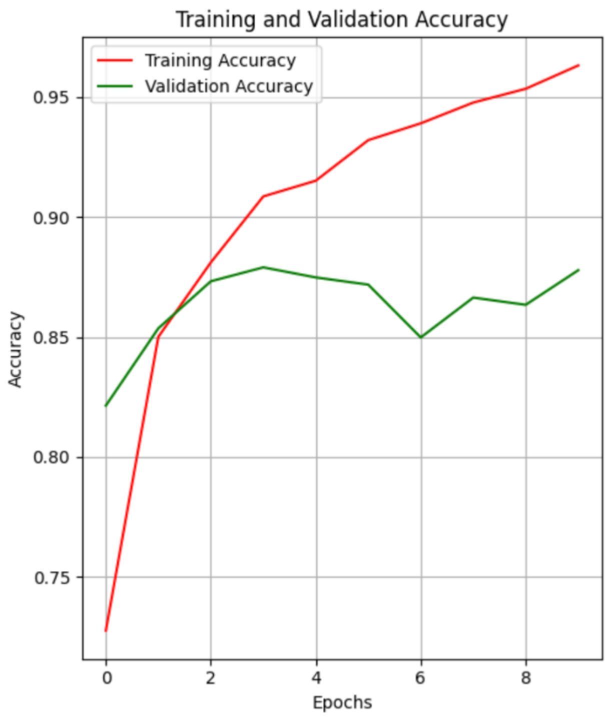


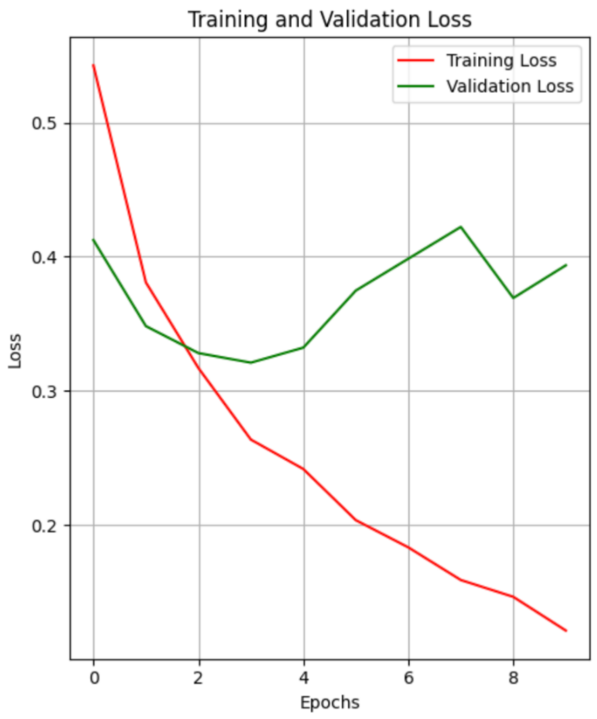
BASE MODEL



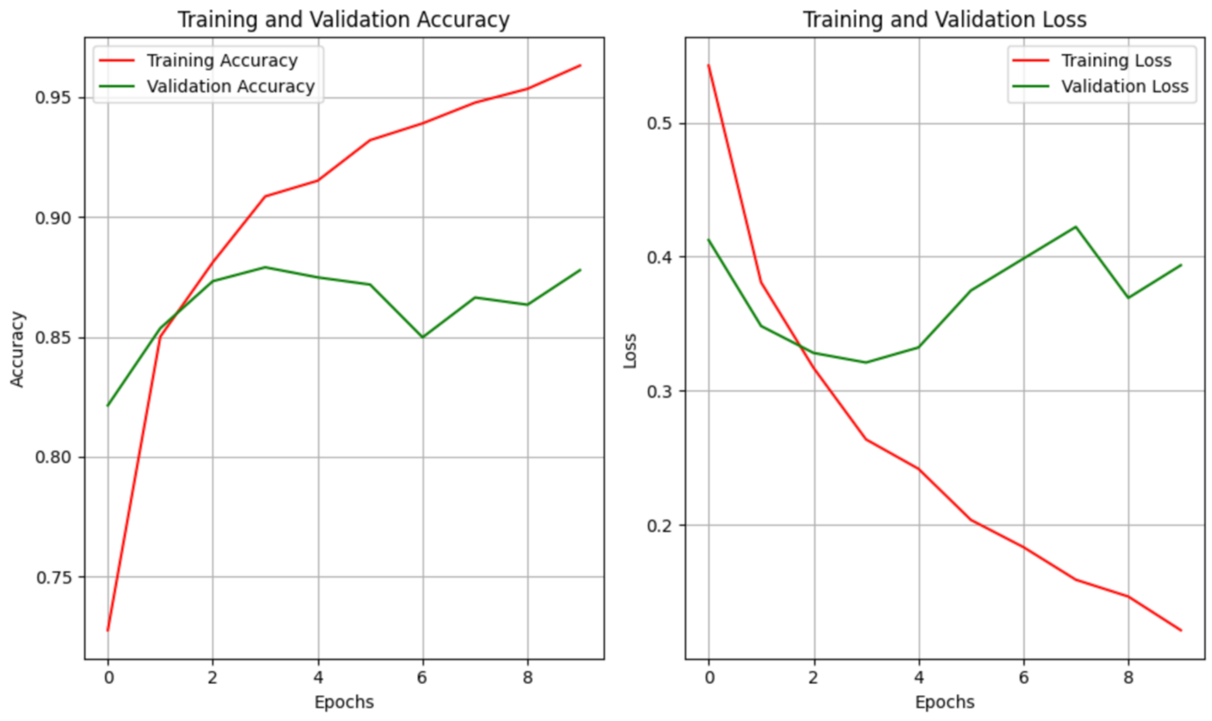


VALIDATE SAMPLE - 10000

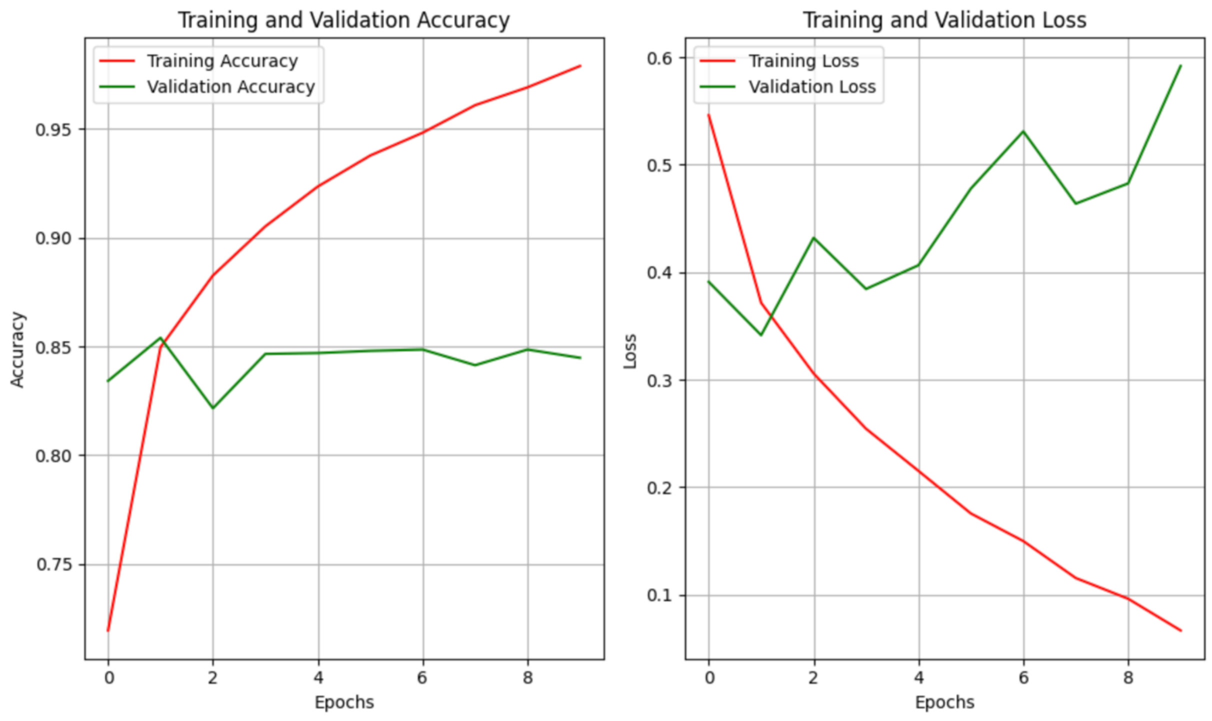




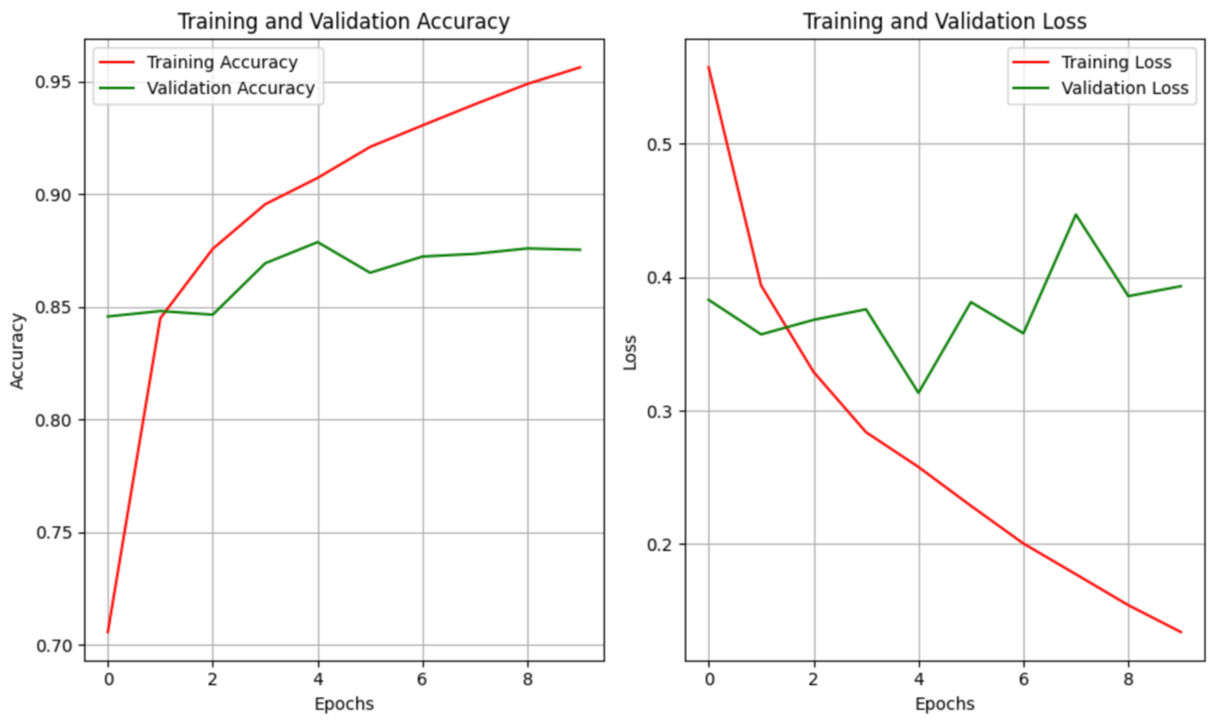
FIRST 10,000 WORDS



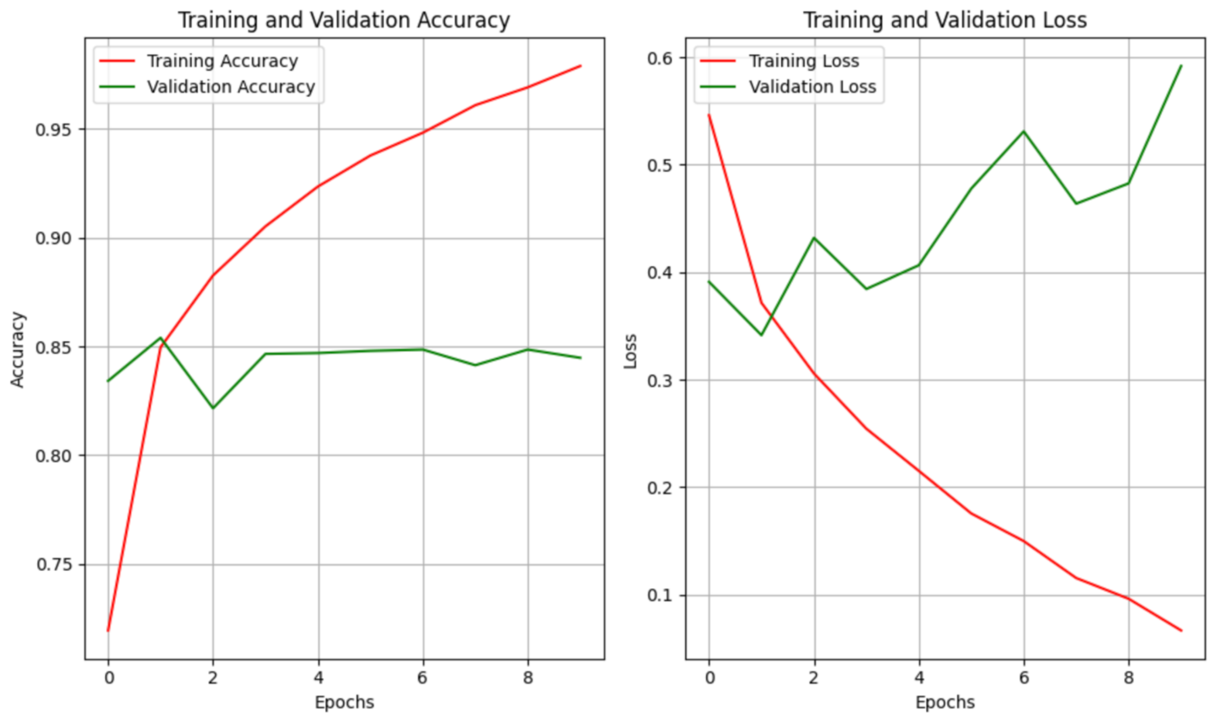
BASE MODEL



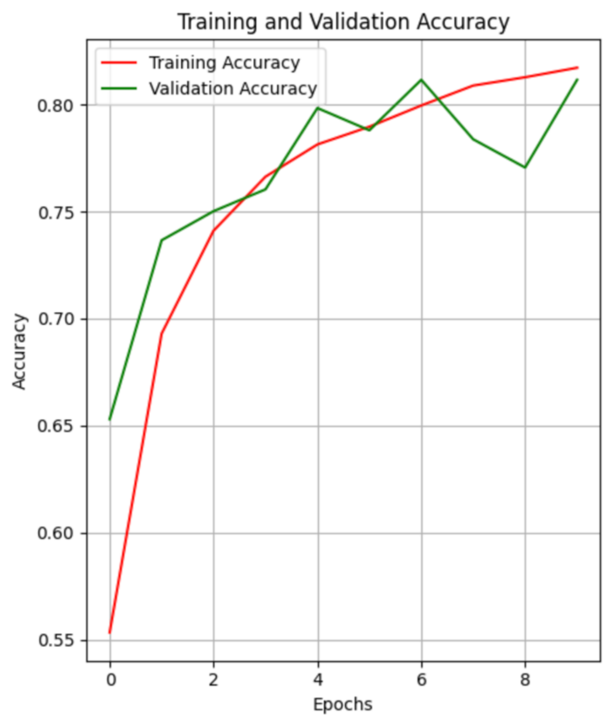
EMBEDDING LAYER (BIDIRECTIONAL LAYER)

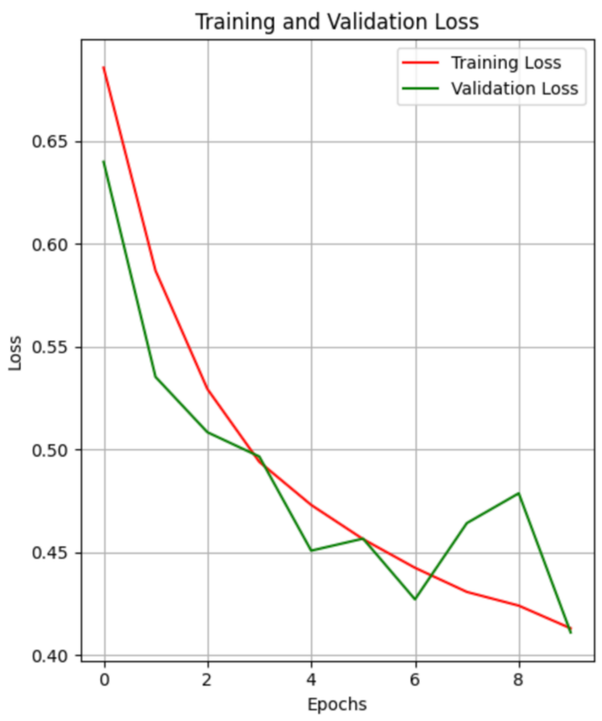


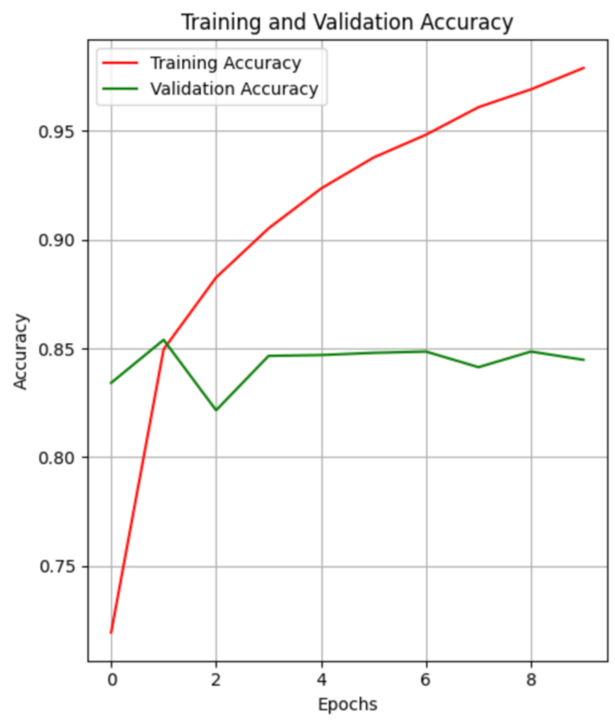
BASE MODEL

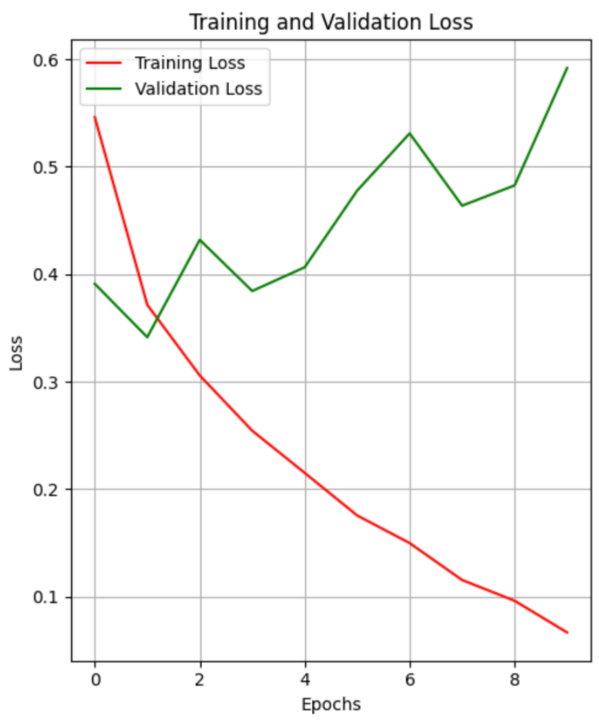


A **pretrained model** was leveraged to enhance performance by utilizing prior learned representations, providing a more informed starting point compared to training from scratch. The use of **pretrained embeddings** or a previously trained language model allowed for faster convergence and improved generalization on the target task. This approach aimed to capture deeper semantic patterns in the data and, in most cases, led to a noticeable improvement in validation and test accuracy.









BETTER APPROACH

The **Base Model** stands out for its strong and consistent performance, driven by well-optimized characteristics tailored to the task. Among the different model configurations tested—including those using learned or pretrained embedding layers—the base model consistently outperforms the rest in key evaluation metrics.

Test Accuracy: With a test accuracy of **0.856**, it achieves the highest score among all compared setups, highlighting its effectiveness in understanding and classifying the data without relying on external pretrained knowledge. This result demonstrates that, for this specific task and dataset, a well-tuned architecture with standard embeddings can match or even exceed the performance of more complex alternatives.

Stability: Another notable strength of the base model is its **robust generalization**. Regardless of the training size used, it maintains a steady and reliable performance, suggesting that it does not overfit to specific samples and can handle variations in data availability effectively. This level of consistency is particularly important for real-world applications where training data quantity can vary, and stability becomes critical for deployment.

Overall, the base model serves as a solid benchmark, illustrating that thoughtfully constructed models using traditional methods can still deliver exceptional performance, even when compared to more modern or computationally expensive alternatives.

SUPERIOR PERFORMANCE

Several key architectural and design choices contribute to the **superior performance** of the base model, particularly in comparison to models that incorporate embedding layers.

Direct Feature Learning: Unlike models that rely on embedding layers for initial word representation, the base model processes **integer-encoded sequences directly through an LSTM layer**. This enables it to learn features that are highly specific to the task and dataset, rather than depending on generalized semantic structures from external sources.

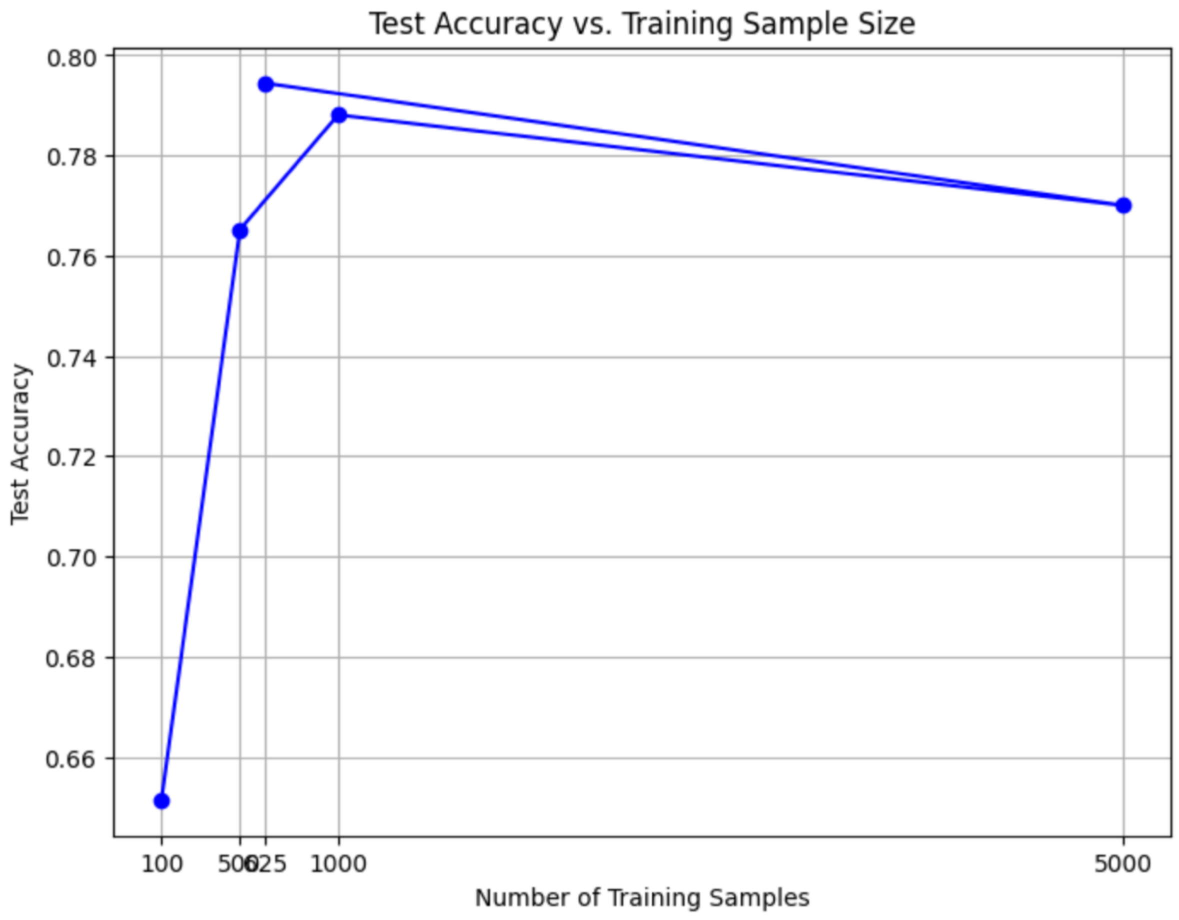
Dataset Specificity: While pretrained embeddings such as GloVe are trained on large corpora, they may not fully capture the nuances or vocabulary unique to a specific dataset. The base model, by learning representations directly from the input data, ensures maximum relevance and adaptability to the dataset's linguistic patterns and domain-specific language.

Reduced Complexity: Eliminating the embedding layer significantly reduces the total number of trainable parameters. This **simpler model architecture** lowers computational requirements and decreases the risk of overfitting, making the model more effective—especially when trained on smaller datasets.

Effective Regularization: To further guard against overfitting, the base model incorporates a **Dropout layer with a rate of 0.5**, which enhances its generalization capabilities. This is especially beneficial in scenarios where training data is limited, as it prevents the model from becoming too tailored to specific training examples.

Task Alignment: Embedding layers, while powerful in many contexts, may introduce abstract representations or noise that doesn't align well with the specific task's requirements. In contrast, the base model's straightforward and focused architecture aligns more closely with the **binary classification objective**, ensuring that learned features are directly relevant to the end goal. Altogether, this combination of **dataset-focused learning, reduced architectural complexity, and effective regularization** enables the base model to perform with greater **accuracy and stability**. Its design proves especially effective for domain-specific text classification tasks, where relevance and precision are more valuable than broad generalization.

Now experiment by varying the number of training samples to identify the threshold at which models with embedding layers begin to outperform the base model.



1. Small Training Sizes (100 & 500):

- **Reason for Lower Accuracy:** With only 100 or 500 samples, the model is unable to capture meaningful patterns due to the insufficient dataset. This limits the effectiveness of the embedding layer, leading to underfitting and poor generalization.
- **Performance Trend (500 → 625):** Increasing the sample size from 500 to 625 slightly decreased accuracy (from 0.771 to 0.758), showing that the model had not yet reached a sufficient training size for optimal performance.

2. Moderate Training Size (1000):

- **Reason for Higher Accuracy:** With 1000 training samples, the model is exposed to a broader range of patterns, improving its ability to generalize. This results in an increase in test accuracy (0.783).
- **Optimal Training Size:** Accuracy peaks at 1000 samples, indicating that this size is ideal for the embedding layer to capture useful representations without overfitting.

3. Large Training Sizes (5000):

- **Performance Saturation:** When the training size is increased to 5000, accuracy improves slightly (0.783 to 0.791). However, the model begins to approach performance saturation, where adding more data contributes less to learning and introduces redundancy, leading to minimal improvements or slight overfitting.

4. Specific Case: Training Size of 1000:

- **Unexpected Accuracy Boost:** The accuracy at 1000 samples (0.783) is only slightly lower than the peak accuracy achieved at 5000 samples (0.791). Possible reasons for this include:
    - **Representative Subset:** The 1000-sample dataset might contain a diverse set of data patterns, allowing the model to generalize more effectively than expected.
    - **Random Variance:** The alignment of the test set with training patterns could result in minor fluctuations in accuracy, leading to slightly better than expected performance.

5. Why Test Accuracy Plateaus:

- **Saturation Point:** Once the model has learned most of the data patterns (around 5000 samples), additional data does not provide significant improvements. The model is already well-equipped to generalize based on the existing information.
- **Limitations:**
    - **Model Complexity:** If the model architecture is not complex enough, it may fail to utilize the additional data effectively, limiting any improvements.
    - **Data Quality:** Adding redundant or noisy data may not contribute meaningfully to performance and could even hinder learning.