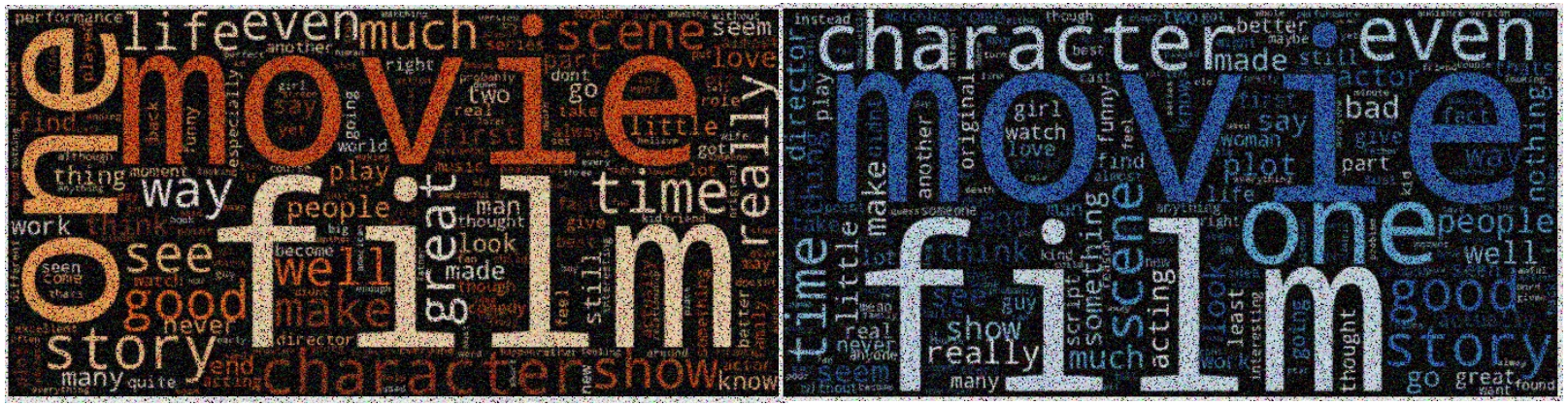


Fine-Tuned BERT for IMDb Review Sentiment Analysis

Daichi Ishikawa

GitHub Link: https://github.com/daichi6/fine-tuned-bert-imdb-sentiment-analysis

May 2024

Problem Statement

- Text sentiment analysis is widely used across various fields, such as improving business strategies by analyzing product or restaurant reviews and leveraging news sentiment in the investment sector.
- Accurately understanding sentiment is crucial for developing better approaches to business improvement. BERT (Bidirectional Encoder Representations from Transformers) has been trained on vast amounts of data, making it applicable to a wide range of domains.
- However, since it is not domain-specific, fine-tuning BERT for a particular domain can enhance its performance. In this project, I aim to create a fine-tuned BERT model specialized for IMDb movie reviews to improve the accuracy of classifying reviews as positive or negative.

EDA/Assumptions and Hypotheses about data and model

- The dataset consists of 10,000 entries (originally, the dataset had 40,000 entries, but 10,000 were randomly selected to reduce computation time).
- After removing 23 duplicate entries, 9,978 entries are used. The sentiment labels are fairly balanced, with 4,940 negative entries and 5,038 positive entries. Therefore, I assume that there will be no issues with data imbalance.
- Since the data includes relatively long reviews with complex sentiment structures, I expect BERT, which can consider context, to achieve higher accuracy than Naive Bayes.
- As mentioned later, BERT truncates at 512 tokens, so some review texts will not be fully learned. However, sentiment can be adequately judged within 512 tokens. Thus, I assume this will have minimal impact on performance.

Examples of movie reviews

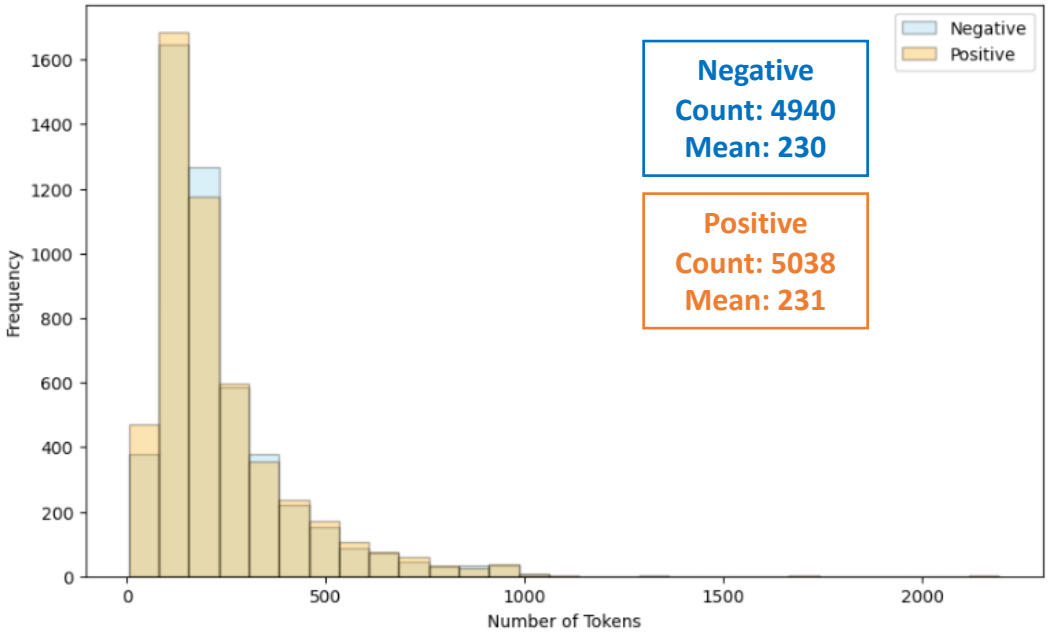
NEGATIVE

I was very excited when this series premiered in 2005. The premise was very simple and appealing: each episode would be a one-hour mini movie directed by a famous, noteworthy horror director. Then, when I finally watched them it was a bit of a letdown. Some good episodes emerged from that first season, but all in all it was a mixed bag. I attributed it to the learning curve, and figured that season 2 would be a whole lot better.Boy, was I in for a shock. At least season one had a few good stories here and there. Season 2 (with the exception of "The Black Cat" starring the excellent Jeffrey Combs) was a complete and total loss to me. The episode "Sounds Like" may very well be the worst thing I have watched on TV in the last 10 years, and most of the other episodes aren't much better. I really hope that season 3 turns this around next year, but I'm not holding my breath.

POSITIVE

This is a very moving movie about life itself. The challenges a handicapped person must face in a land that expects perfection is brought to the forefront for all to see and hopefully understand. It should teach the bigots of society that we are all humans, and while some of us are gifted with a mind, heart and sound body, there are decent human beings that exist in the world that are not as lucky, or maybe, we're the unlucky ones. We don't always see the beauty in the world because we're wrapped up in our 'blind' ambitions, and see it only in one light "what can this world do for me!!!". Maybe we all wish we were like Radio, a loving happy individual...who loves everyone.

Token Count Distribution by Sentiment

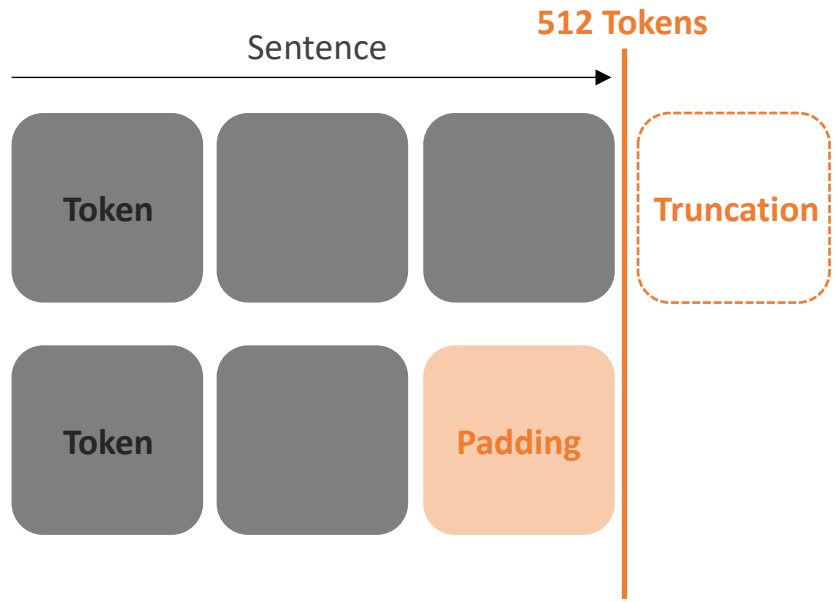


Feature Engineering and Transformations

- For the baseline model using Naive Bayes, I perform the following cleaning steps: remove special characters, remove extra spaces, tabs, and newlines, convert to lowercase, remove stopwords, and lemmatize. After cleaning, I vectorize the text using TF-IDF.
- For BERT, I use the TFAutoModelForSequenceClassification.from_pretrained('bert-base-uncased', num_labels=2) from the Transformers library. I use BertTokenizer for tokenization. Since the model has a maximum token length of 512, I apply truncation and padding to match this length. Finally, I convert the tokenized data into TensorFlow datasets.

Word Cloud(After cleaning)

Truncation and Padding



Proposed Approaches and Solution

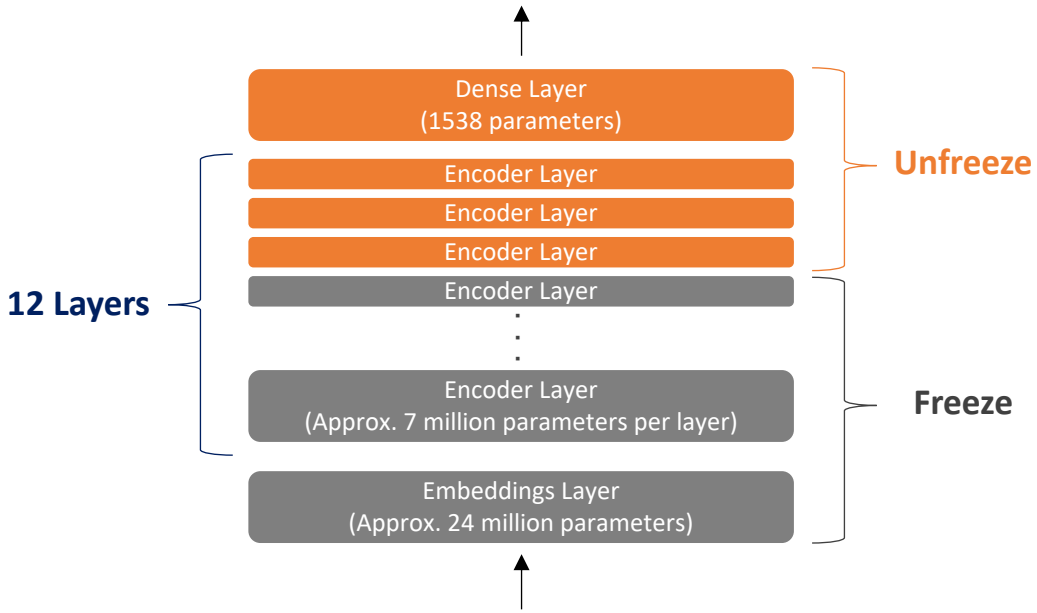
- I will gradually make the Dense layers and some Encoder layers trainable. Increasing the number of trainable layers could allow the model to specialize in the domain, but it is necessary to be cautious of the risk of overfitting. I assume that the lower layers (those closer to the input layer) will still be useful for this task, as they capture high-level features.
- The batch size is set to 32. The Adam optimizer is used for optimization. Polynomial Decay is implemented to ensure stable learning.
- Early Stopping is introduced to prevent overfitting. It is configured with a patience of 2 or 3 and restore_best_weights=True. The initial learning rate and number of epochs will be adjusted based on the results.
- I will use 20% of the entire data as test data, 20% as validation data, and the remaining 60% as training data.
- This dataset has an approximately equal number of positive and negative reviews. Given that there is no priority in classifying one over the other, I will focus on accuracy.

Proposed Models

Model	Trainable Layer	Initial Learning Rate	Epoch	Patience	Batch Size
Naive Bayes (Baseline Model)	-	-	-	-	-
BERT	Dense	5e-5	4	2	32
BERT	Dense	1e-3	10	2	32
BERT	Dense + 1 Encoder	1e-3	10	2	32
BERT	Dense + 1 Encoder	1e-4	10	2	32
BERT	Dense + 3 Encoders	1e-4	10	2	32
BERT	Dense + 3 Encoders	1e-5	10	3	32

BERT (TFAutoModelForSequenceClassification) Architecture

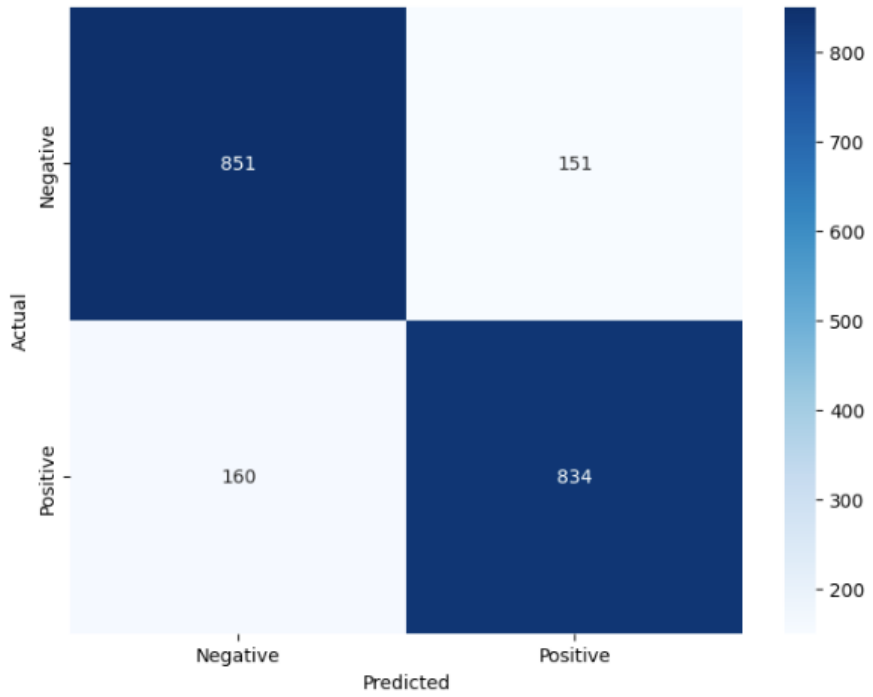
A total of approximately 110 million parameters



Naive Bayes(Baseline Model)

- The accuracy was 0.844, which is fairly good. The positive and negative samples are roughly equal in number, and there is no noticeable bias in the classification results.
- However, since the model is simple and based on word frequency (TF-IDF), it likely does not account for context. There is a higher likelihood of misclassification, especially when positive and negative words are mixed within the same review.

Confusion Matrix



Classification Report

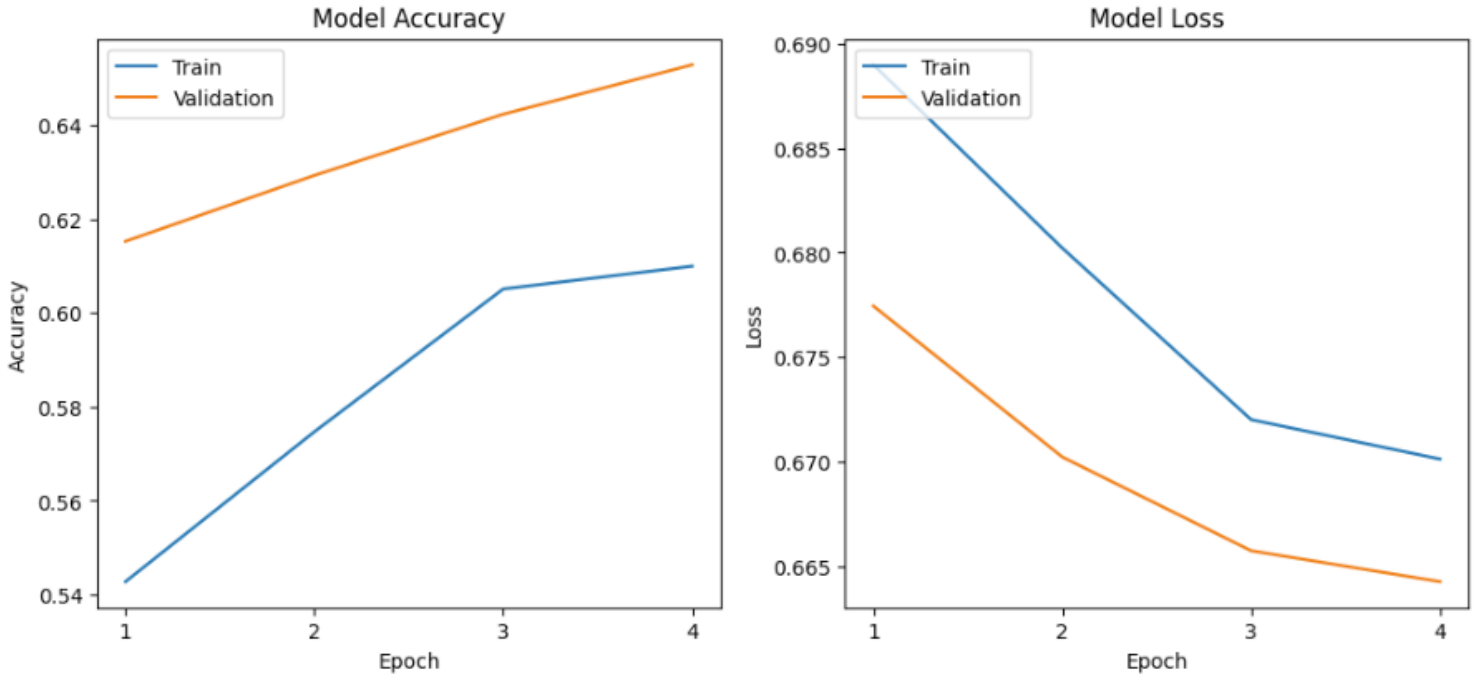
Accuracy: 0.844

Class	Precision	Recall	F1-Score	Support
Negative	0.84	0.85	0.85	1002
Positive	0.85	0.84	0.84	994
Macro Avg	0.84	0.84	0.84	1996
Weighted Avg	0.84	0.84	0.84	1996

Dense Layer Only(Learning Rate = 5e-5, Epoch = 4)

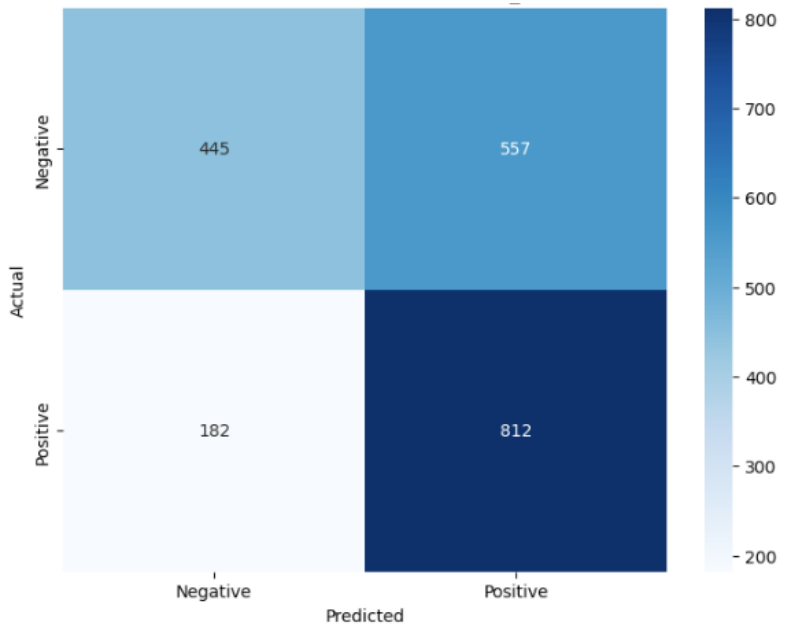
- As the epochs progressed both training and validation accuracy, as well as loss improved, but the training ended with low accuracy and high loss.
- This is likely due to a low learning rate. Although a learning rate of 5e-5 is recommended for fine-tuning multiple layers, a higher learning rate would have been more appropriate in this case since I am only fine-tuning the dense layer.
- The classification of negative sentiment is particularly ineffective.

Model Accuracy and Loss (Training and Validation Data)



Confusion Matrix (Test Data)

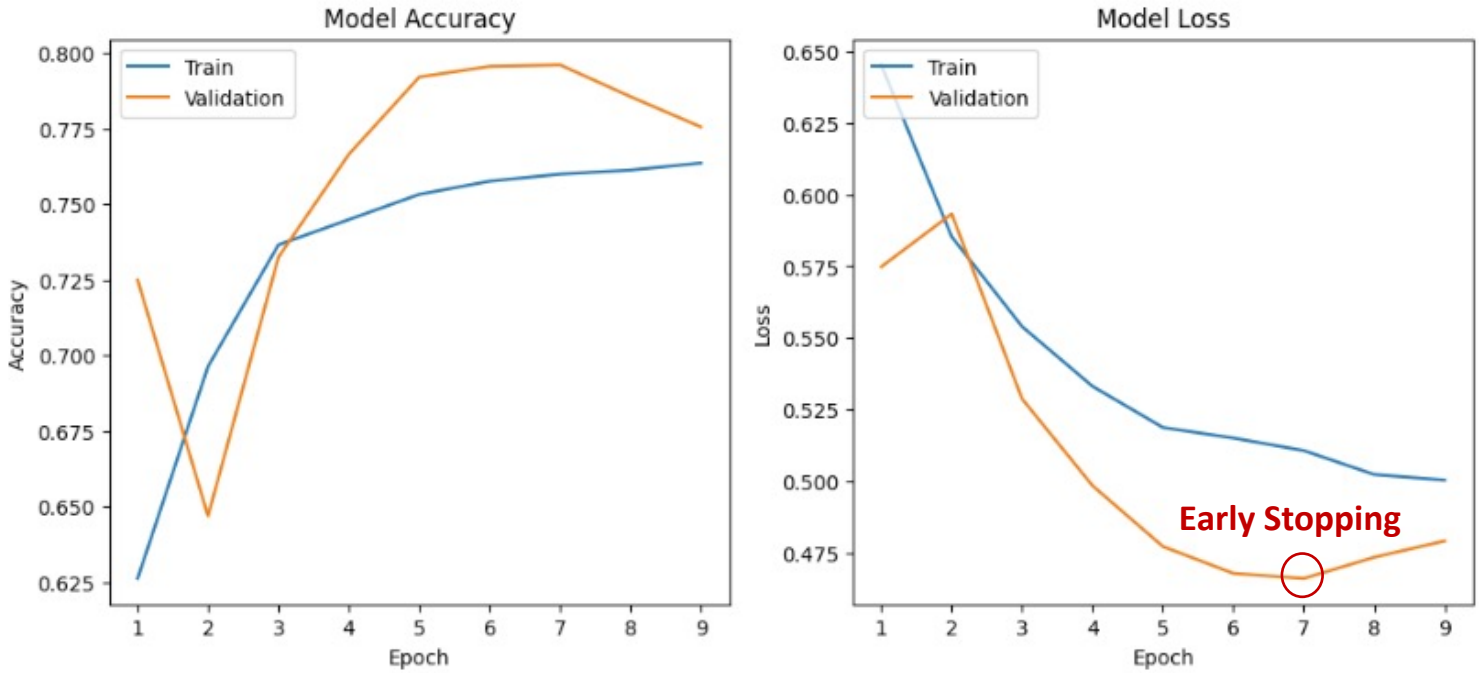
Accuracy: 0.630



Dense Layer Only(Learning Rate = 1e-3, Epoch = 10)

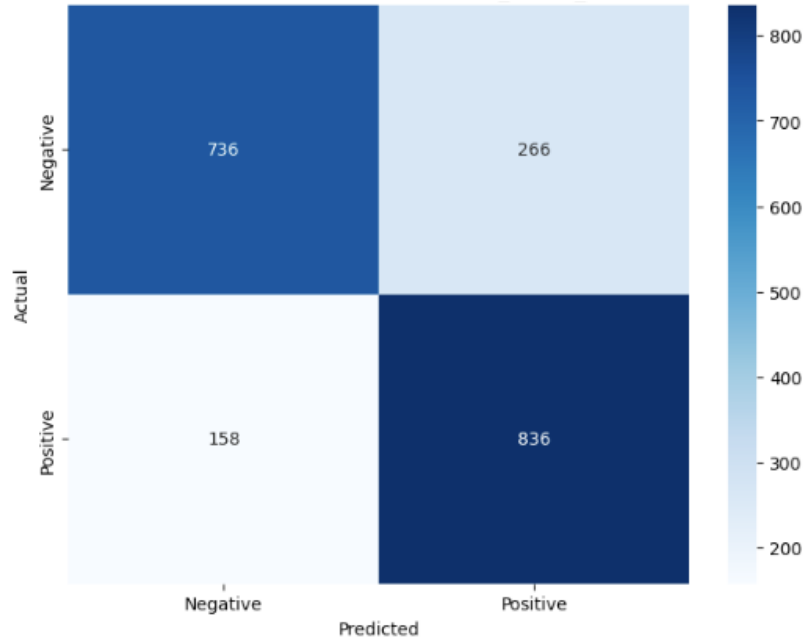
- Based on the previous results, the learning rate was increased to 1e-3. Additionally, the number of epochs was set to 10 to observe the training progress over a longer period.
- The results show that both training and validation accuracy, as well as loss, improved up to epoch 7. However, beyond that, while training accuracy and loss continued to improve, validation accuracy and loss deteriorated, indicating a trend of overfitting. Due to early stopping, the best model was obtained at epoch 7.
- The test data accuracy was 0.788, still below the baseline model, suggesting the limitations of using only the dense layer to improve accuracy.
- The classification of negative sentiment continues to be relatively ineffective.

Model Accuracy and Loss (Training and Validation Data)



Confusion Matrix (Test Data)

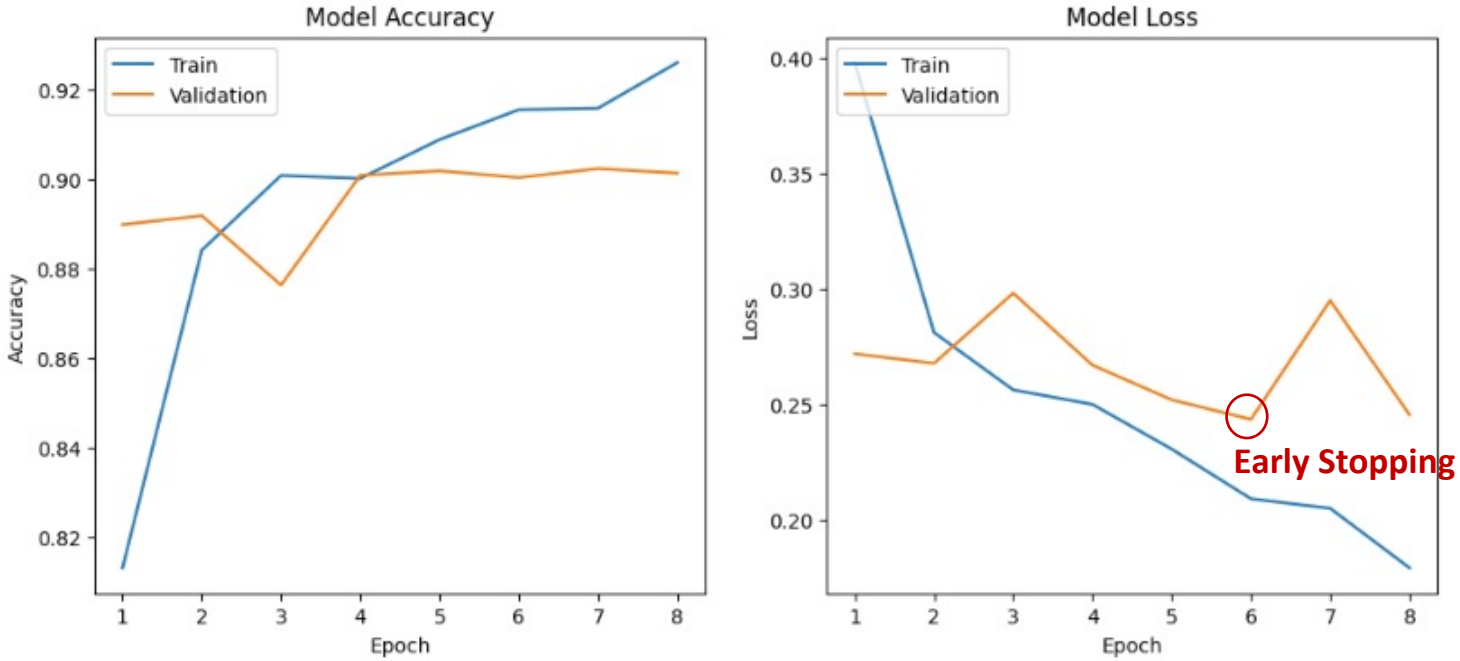
Accuracy: 0.788



Dense Layer and 1 Encoder Layer(Learning Rate = 1e-3, Epoch = 10)

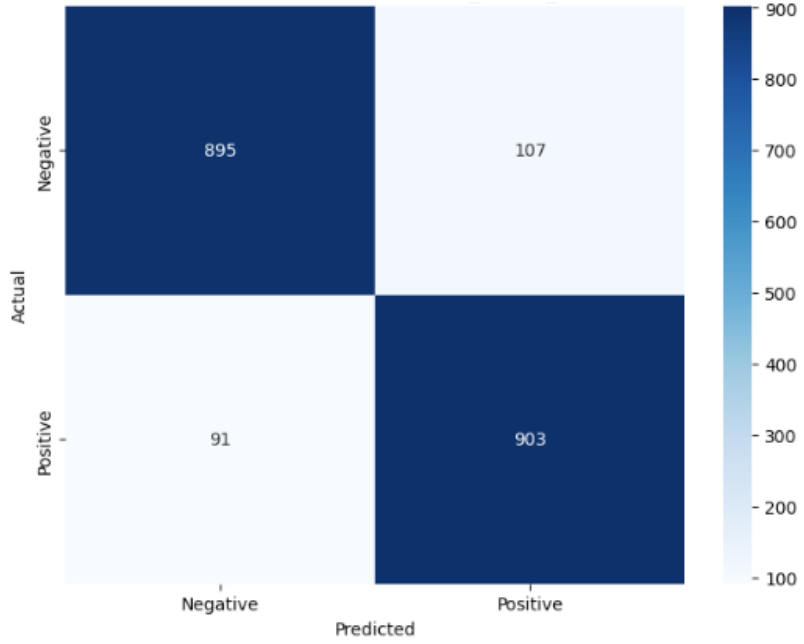
- By making the encoder layer closest to the output layer trainable, the model became more specialized for movie reviews, resulting in a significant improvement in test data accuracy to 0.901.
- However, the training process showed some instability, indicating that the optimal parameters may not have been achieved. This instability is likely due to the learning rate of 1e-3 being too high, given the substantial increase in trainable parameters.
- Early stopping identified the best model at epoch 6, but considering the instability in training, there is likely room for further improvement.
- Making the encoder layer trainable particularly improved the accuracy of negative sentiment classification.

Model Accuracy and Loss (Training and Validation Data)



Confusion Matrix (Test Data)

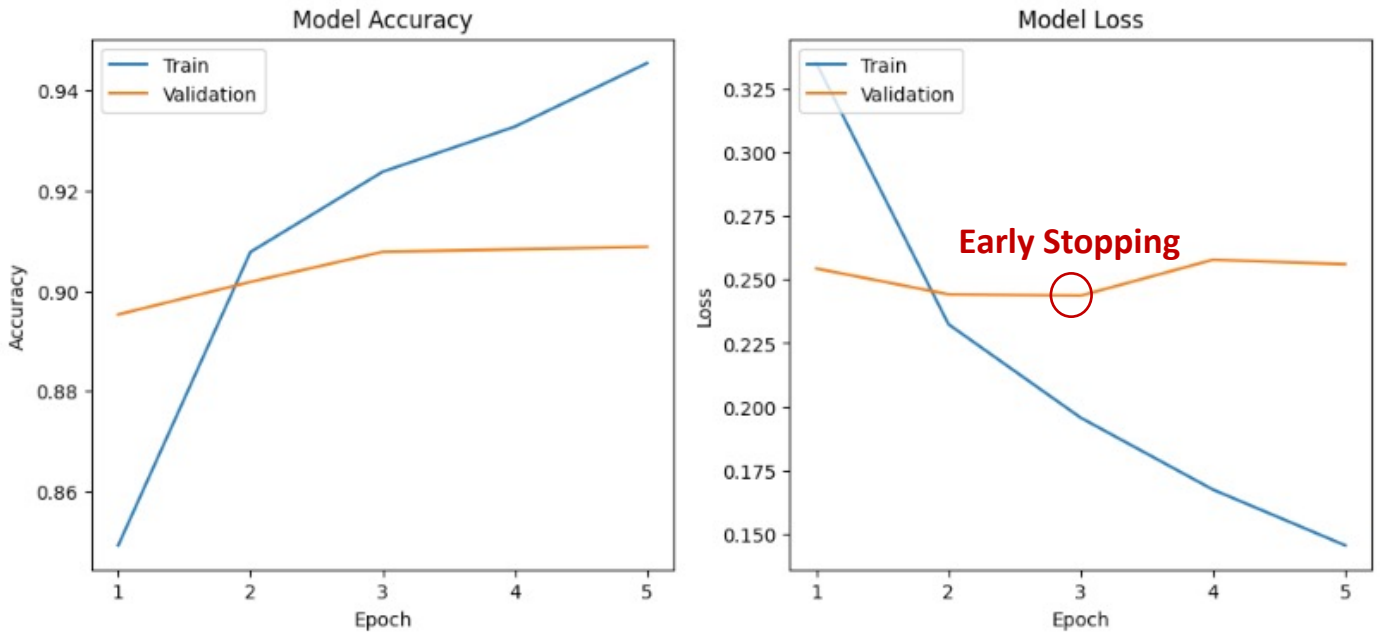
Accuracy: 0.901



Dense Layer and 1 Encoder Layer(Learning Rate = 1e-4, Epoch = 10)

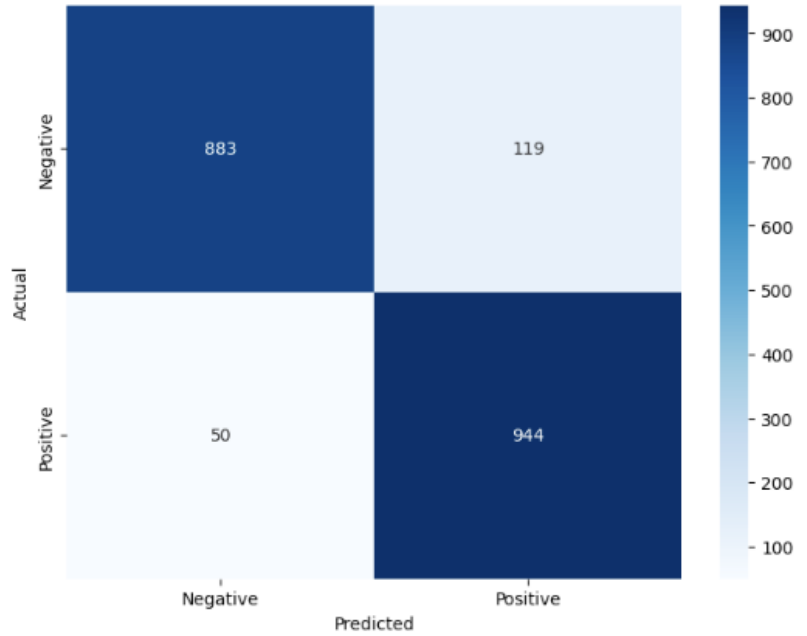
- Based on the previous result, the learning rate was reduced to 1e-4, resulting in a more stable training process. The performance on the test data improved to 0.915.
- After the second epoch, the accuracy and loss on the validation data remained relatively stable, indicating early convergence.
- With early stopping, the best model was selected at epoch 3, allowing me to avoid overfitting and skip the unnecessary processes beyond epoch 6.

Model Accuracy and Loss (Training and Validation Data)



Confusion Matrix (Test Data)

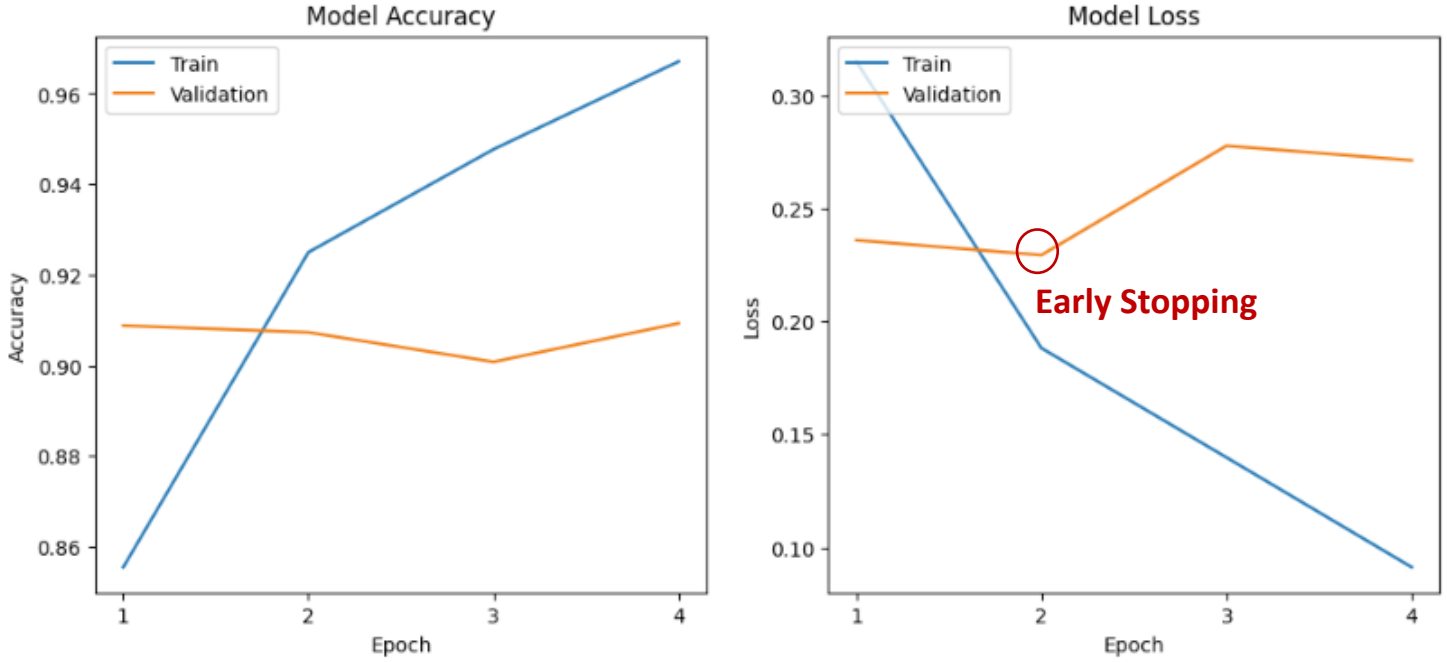
Accuracy: 0.915



Dense Layer and 3 Encoder Layers(Learning Rate = 1e-4, Epoch = 10)

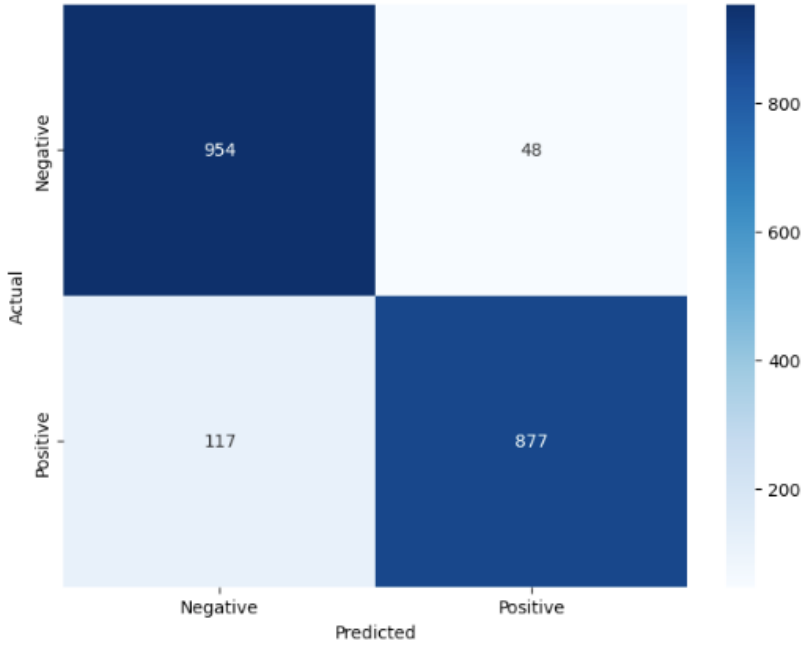
- To further specialize the model for this movie review dataset, I made the three encoder layers closest to the output layer trainable. The validation loss reached its lowest level so far, and the accuracy on the test data improved slightly to 0.917.
- Examining the training process, we observed a trend of overfitting after epoch 3, with early stopping identifying epoch 2 as the best model.
- Making additional encoder layers trainable particularly improved the accuracy of negative sentiment classification.

Model Accuracy and Loss (Training and Validation Data)



Confusion Matrix (Test Data)

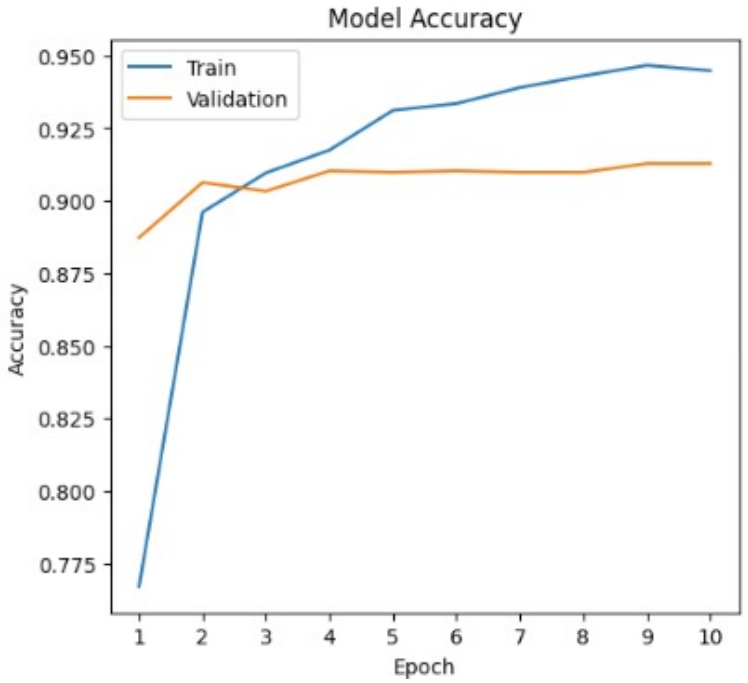
Accuracy: 0.917

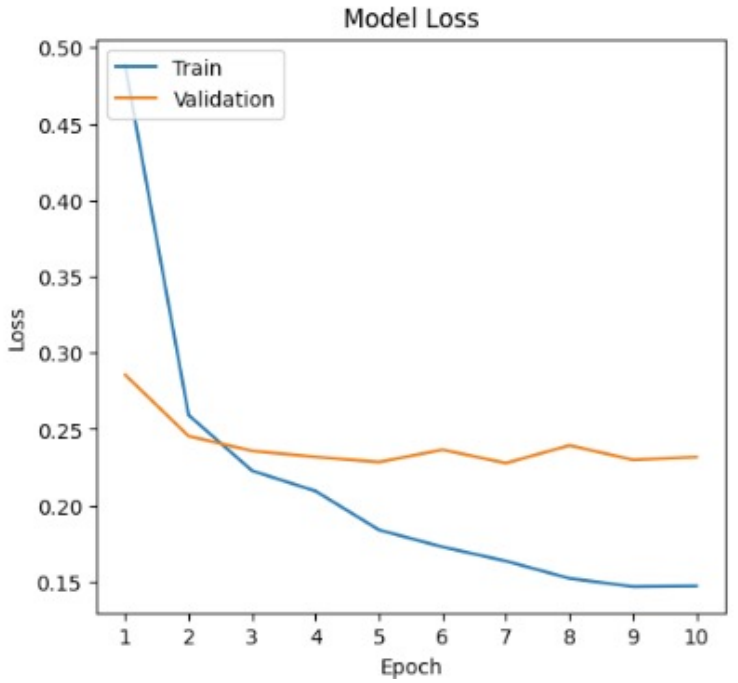


Dense Layer and 3 Encoder Layers(Learning Rate = 1e-5, Epoch = 10)

- Since the model may not have converged to the optimal solution, I lowered the learning rate to 1e-5 to enable finer learning.
- At epoch 7, the validation loss reached its lowest level so far and then stabilized. The accuracy on the test data slightly improved to 0.919, marking the highest level recorded.

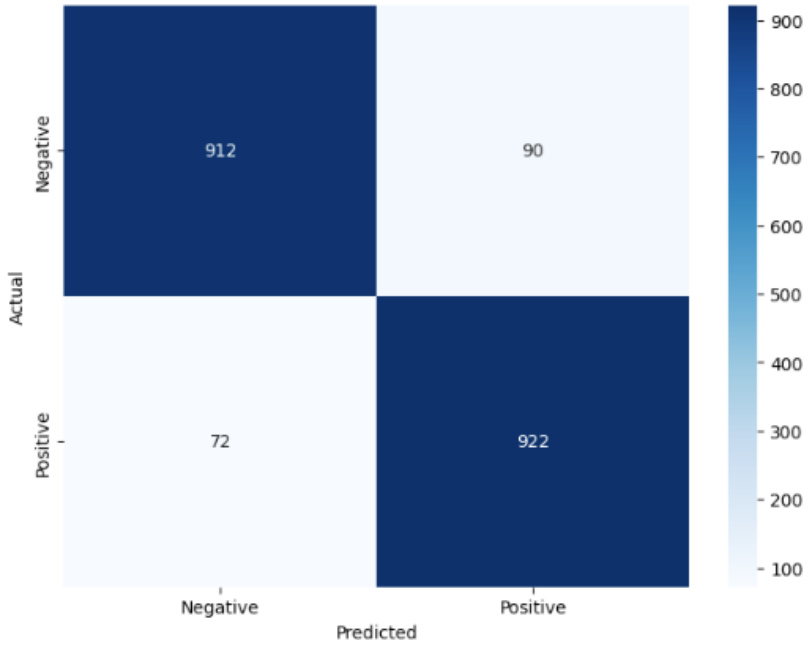
Model Accuracy and Loss (Training and Validation Data)





Confusion Matrix (Test Data)

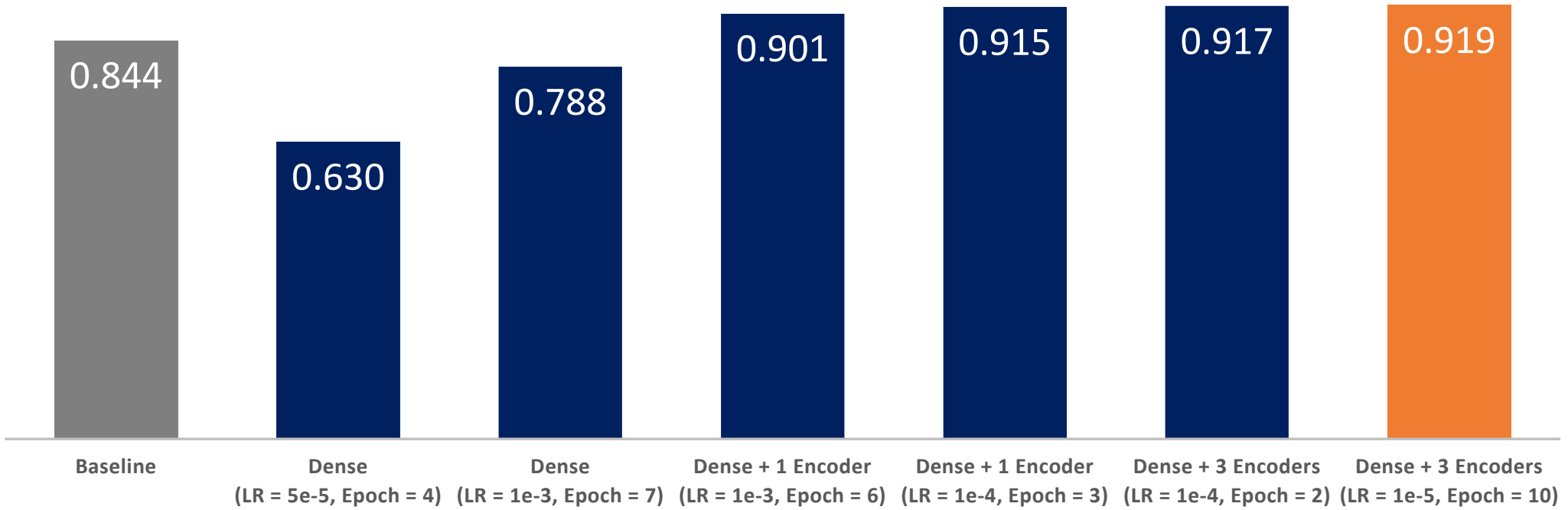
Accuracy: 0.919



Results (Accuracy) and Learnings from the methodology

- By expanding the fine-tuning from only the dense layer to include the encoder layers, I was able to create a classifier specialized for IMDb movie reviews, gradually improving its accuracy.
- Unfreezing the encoder layers improved the classification accuracy overall, especially for negative sentiment, which the dense layers alone struggled to classify effectively.
- However, the improvement in accuracy from expanding from 1 encoder layer to 3 encoder layers was minimal. Therefore, for this dataset, fine-tuning just one encoder layer was sufficient to achieve a high accuracy, considering the computational resources and time.
- Additionally, I found that adjusting the learning rate and using early stopping to avoid overfitting improved accuracy. This also highlighted the importance of choosing an appropriate learning rate based on the number of parameters being trained.

Comparison of Accuracy Across All Models (Test Data)



* Note: The number of epochs used is based on early stopping.
* LR: Learning Rate

Future Work

- **Expanding Trainable Layers**: While significant accuracy improvements may not be expected, I would like to explore the performance trends as the number of trainable layers is increased further.
- **Custom Layers**: I also plan to investigate how the addition of custom layers affects the model's accuracy.
- **Increasing Dataset Size**: This analysis was conducted with a dataset of 10,000 instances. I aim to examine how accuracy changes when the dataset size is increased.
- **Parameter Adjustments**: I also want to observe the effects of changing various parameters, such as batch size, optimizer type, and learning rate schedule, on the model's performance.
- **Data Transformation**: For the project, truncation and padding to 512 tokens were used, which led to excessive padding for some short texts and information loss for some long texts. Given studies suggesting that important information often appears at the beginning and end of reviews and articles, I intend to explore more efficient data transformation methods.
- **Specialized Text/Document Classification**: In the context of classifying more specialized texts, such as legal documents, I am interested in comparing how the performance improvement with varying numbers of trainable layers differs from the current project case. While movie reviews contain unique expressions, they lack the specialized vocabulary found in more domain-specific texts, making this a relatively general case.