Applied Deep Learning Project (WS2025)

IMDB Sentiment Analysis Using NLP

Name: Aman Bhardwaj
Student ID: 12333472

Introduction

In this project, I re-implemented a deep learning-based system to classify movie reviews from the IMDB dataset as positive or negative. Using a RoBERTa transformer model, I implemented an end-to-end pipeline from data preprocessing to model training, evaluation, and deployment in a web-based Streamlit application.

Approach: Bring Your Own Method – Re-implemented and fine tuned the model
Dataset: IMDB Large Movie Reviews – 50,000 labeled reviews (binary sentiment).
Model: RoBERTa-base Transformer for contextual text understanding.
Tokenizer: RoBERTa Byte-Pair Encoding (BPE).
Loss Function: Cross-Entropy Loss for binary classification.
Deliverables: Pipeline + interactive Streamlit web app for real-time sentiment prediction.

Data Preprocessing

The IMDB reviews were preprocessed minimally and tokenized to feed into a RoBERTa transformer model, which excels at understanding contextual language. The RoBERTa tokenizer converts text into sub-word tokens, while the model is trained using Cross-Entropy Loss for binary sentiment classification.

Dataset Split: 80% Train / 10% Validation / 10% Test

Minimal Text Cleaning: Removed HTML tags and normalized whitespaces; no lowercasing or stopwords, as RoBERTa is pretrained on raw text. Saved cleaned CSVs

RoBERTa Model:

- Pretrained transformer encoder for contextual language understanding
- Fine-tuned for sentiment classification with 2 output classes (Positive / Negative)

Tokenizer:

- Byte-Pair Encoding (BPE)
- Converts text → Input IDs & Attention Masks
- Max sequence length = 256
- Saved .pt files for train/val/test

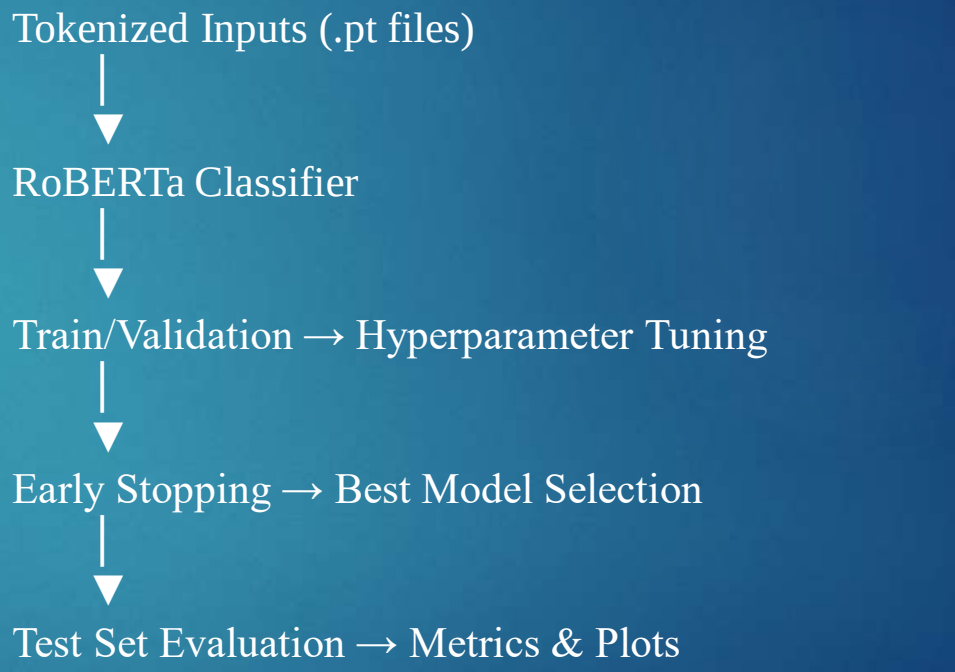
Model Training & Optimization

After preprocessing and tokenization, the RoBERTa model was trained on the IMDB dataset. We implemented both a baseline and an optimized fine-tuned model, experimenting with hyperparameters like learning rate, batch size, dropout, weight decay, and pooling to maximize performance.

Baseline Model:
- Trained only the classification head for 5 epochs
- Metrics established a reference performance

Optimized Model:
- Fine-tuned all layers of RoBERTa
- Hyperparameter tuning across 8 configurations
- Early stopping used to prevent overfitting

Hyperparameters Considered:
- Learning rate: 1e-5, 2e-5, 3e-5
- Batch size: 16, 32
- Dropout: 0.1, 0.2
- Weight decay: 0, 0.01
- Pooling: CLS / Mean
- Scheduler: linear / cosine



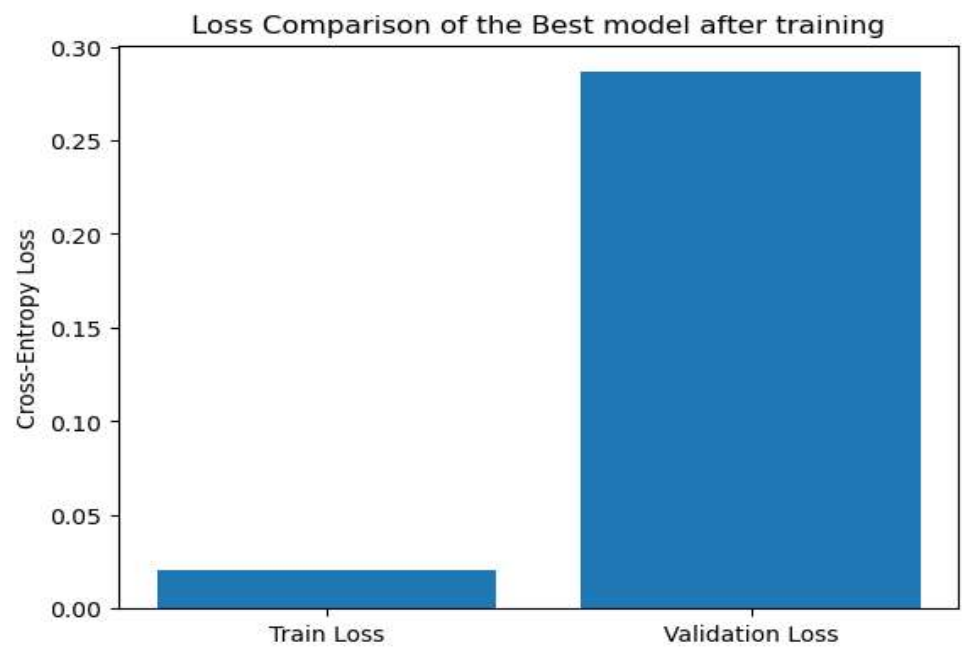
Training time: ~16 hours for optimized configurations

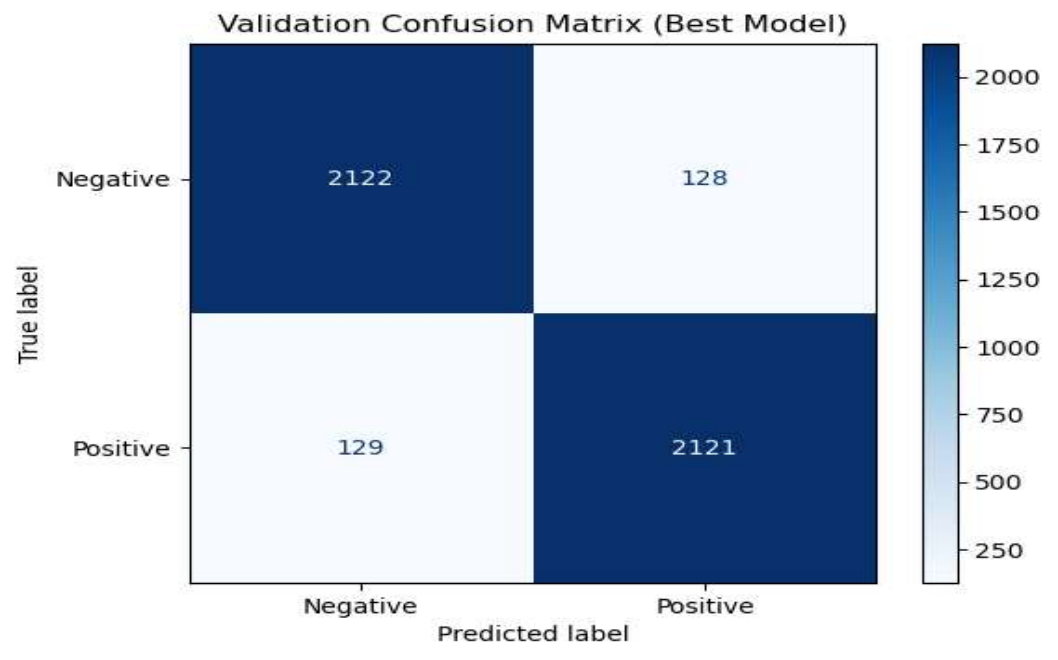
Results Table (Best Model)

Metric	Train	Validation	Test
Accuracy	0.9862	0.9429	0.9430
Precision	0.9850	0.9431	0.9355
Recall	0.9875	0.9427	0.9516
F1-score	0.9862	0.9429	0.9435
Loss (CE)	0.0209	0.2864	0.2727
Runtime (s)	5722.76	5722.76	197.21

Best configuration after training: { "lr": 1e-05, "batch_size": 32, "dropout": 0.1, "weight_decay": 0.0, "scheduler": "cosine", "warmup_ratio": 0.06, "pooling": "cls" }

The validation set showed a bit less performance than the training set which is understandable as the size of the validation set is 8 times smaller than training set. Below are the plots for the training of the model, validation confusion matrix and runtimes:

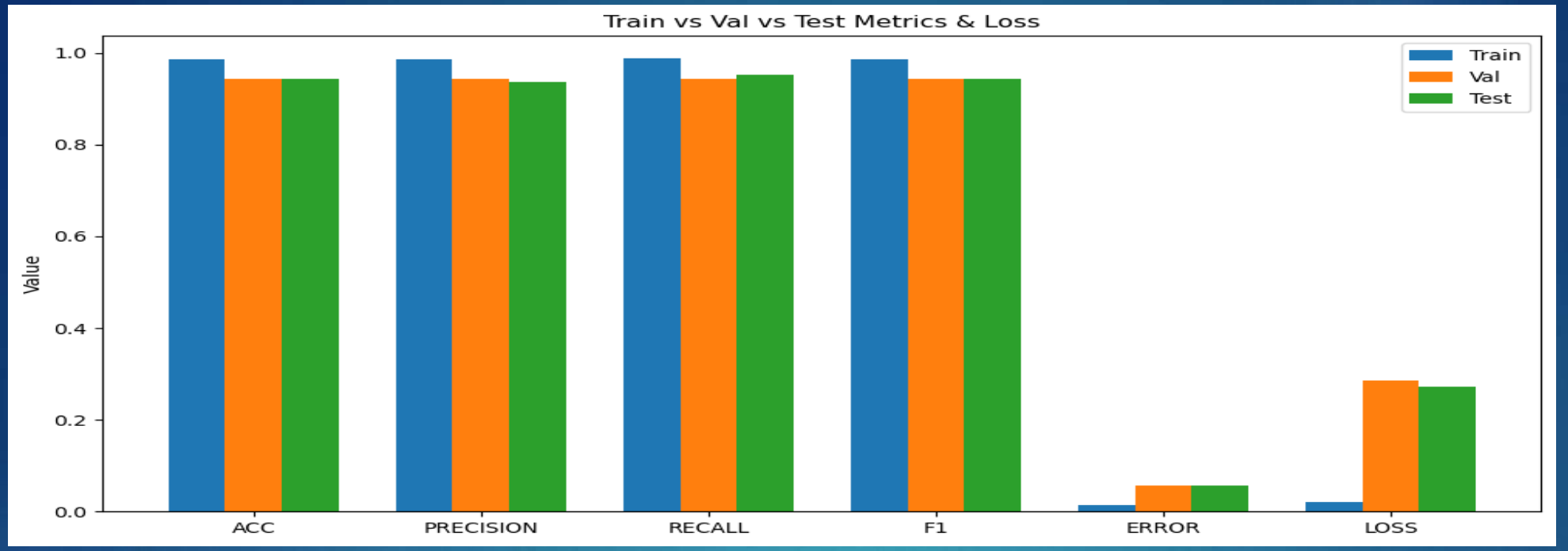


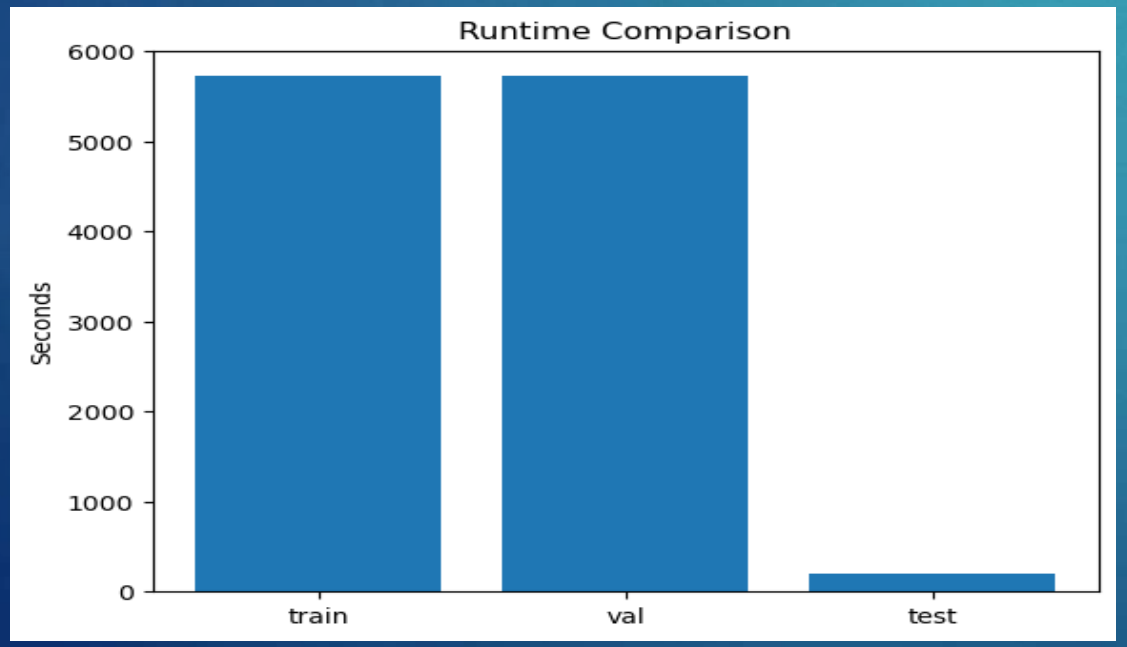


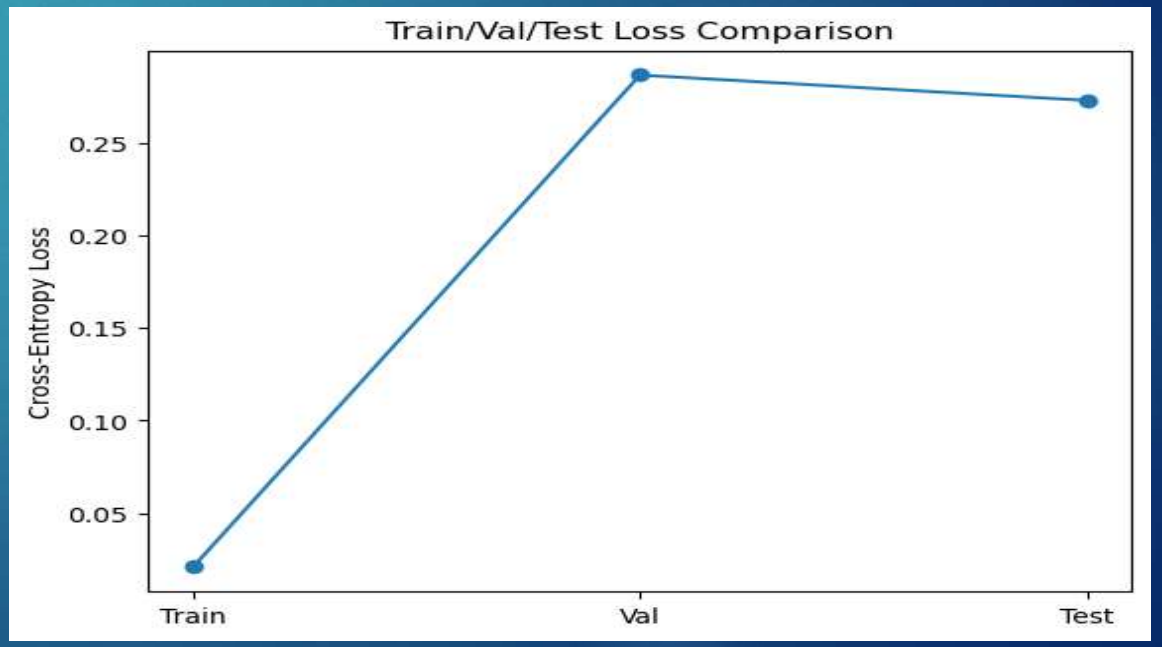
Test Results (Best Model)

Split	Accuracy	Precision	Recall	F1-score	Error	Loss	Runtime (s)
Test	0.9430	0.9355	0.9516	0.9435	0.0570	0.2727	197.21

So, from the above tables it can be seen that the F1 score improved a bit for the test set and Cross Entropy Loss also improved a little.. Below are the plots for the testing of model:


Train vs Val vs Test Metrics & Loss
Train
Val
Test
Value
1.0
0.8
0.6
0.4
0.2
0.0
ACC
PRECISION
RECALL
F1
ERROR
LOSS


Runtime Comparison
Seconds
6000
5000
4000
3000
2000
1000
0
train
val
test

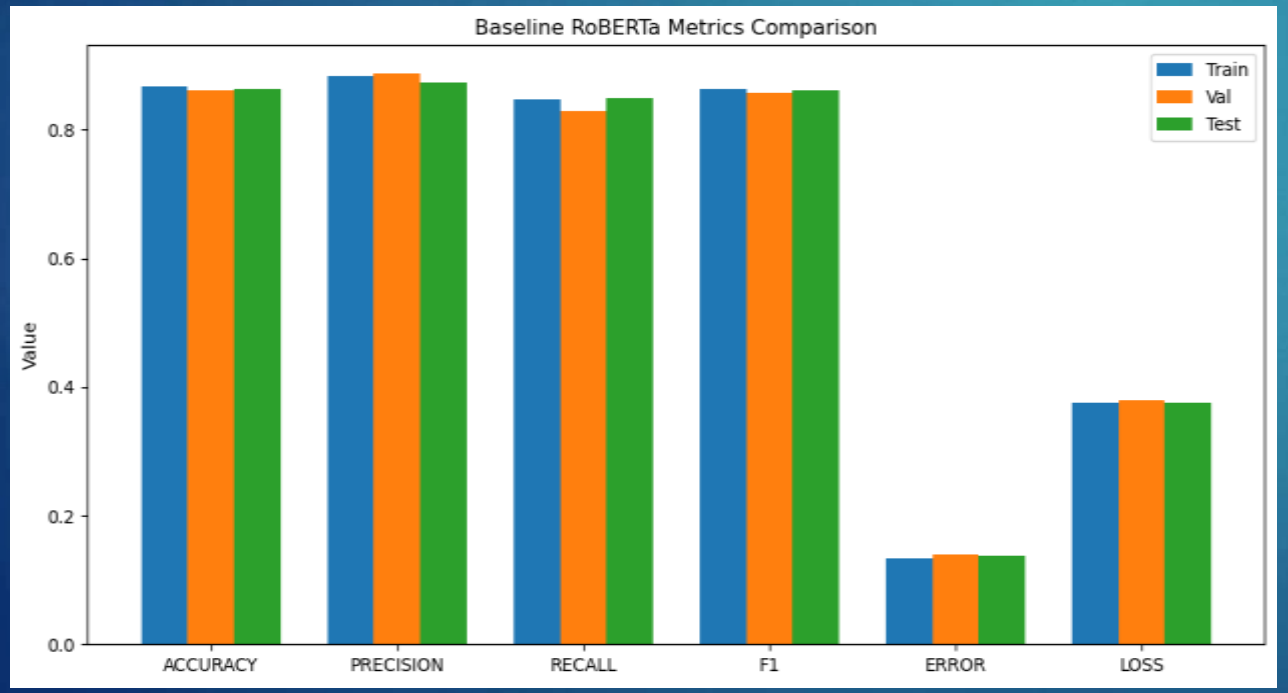

Train/Val/Test Loss Comparison
Cross-Entropy Loss
0.25
0.20
0.15
0.10
0.05
Train
Val
Test

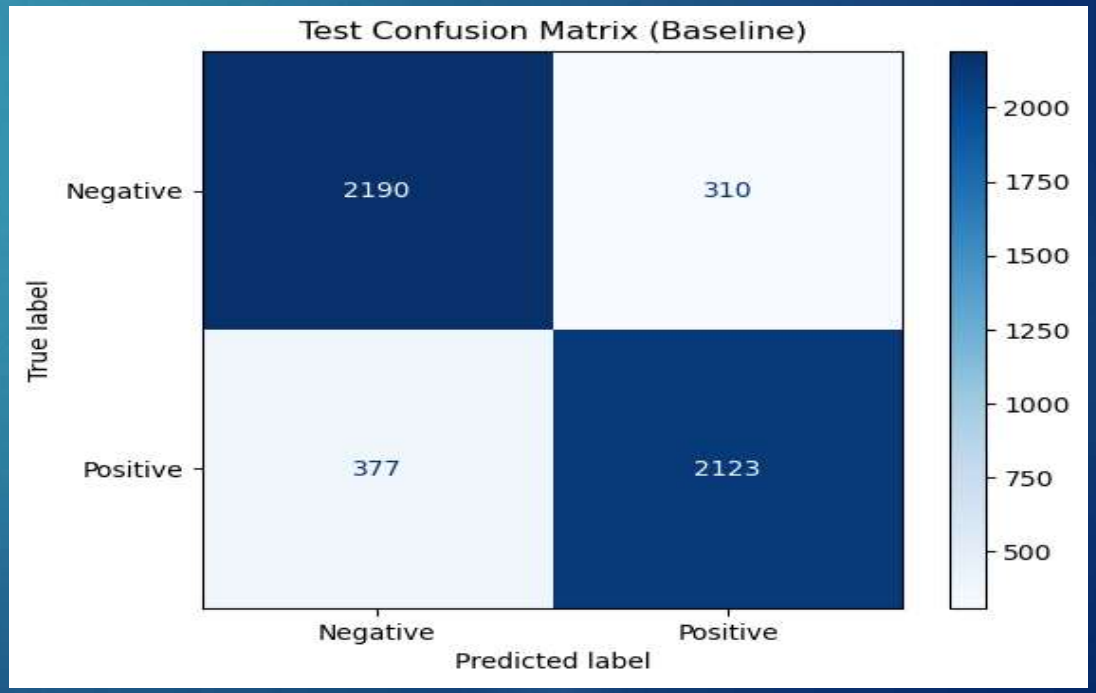
Baseline Model

RoBERTa with linear classification head. Trained head only for 5 epochs with Default hyperparameters

Baseline Results Table:

Split	Accuracy	F1-score	Loss	Runtime (s)
Train	0.8667	0.8639	0.3757	492.02
Val	0.8611	0.8564	0.3786	54.66
Test	0.8626	0.8607	0.3762	60.74

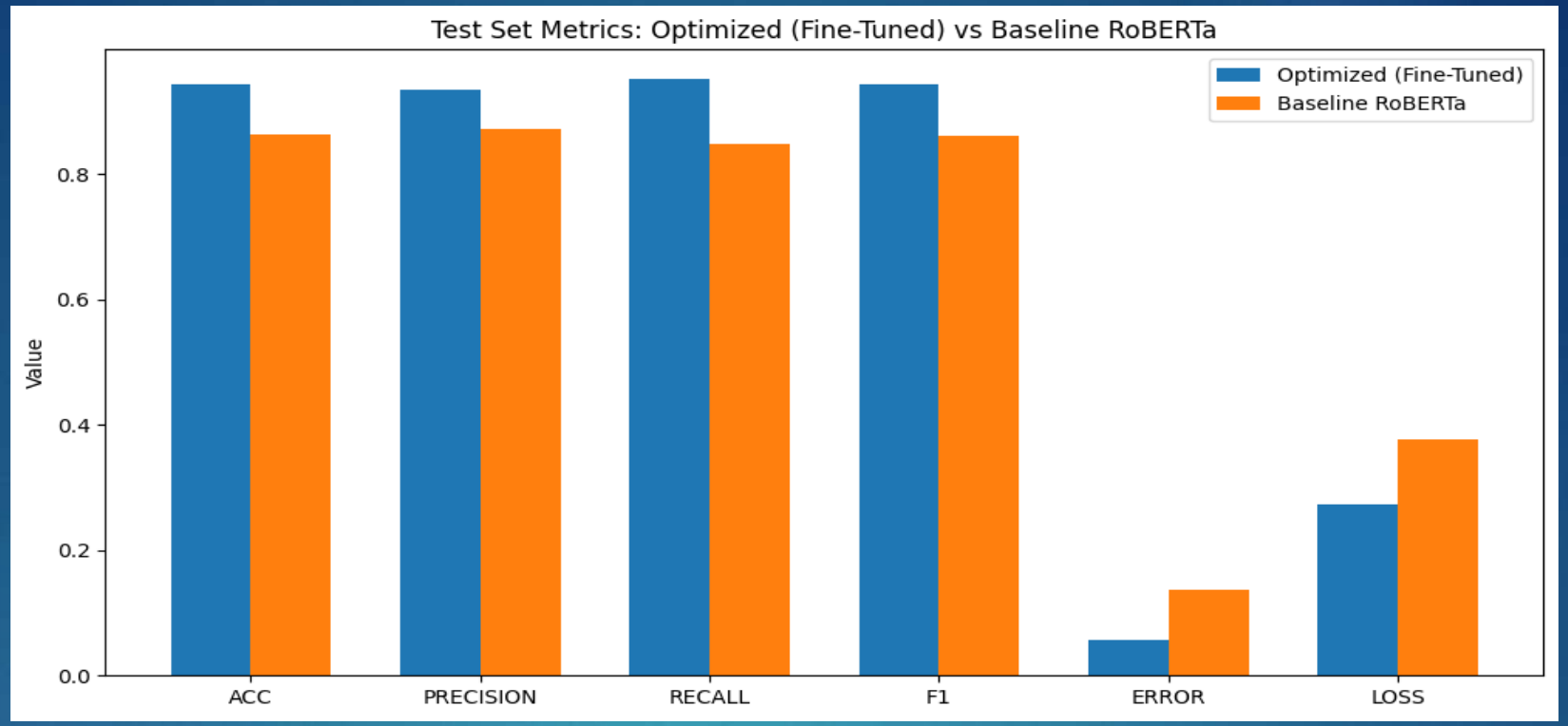


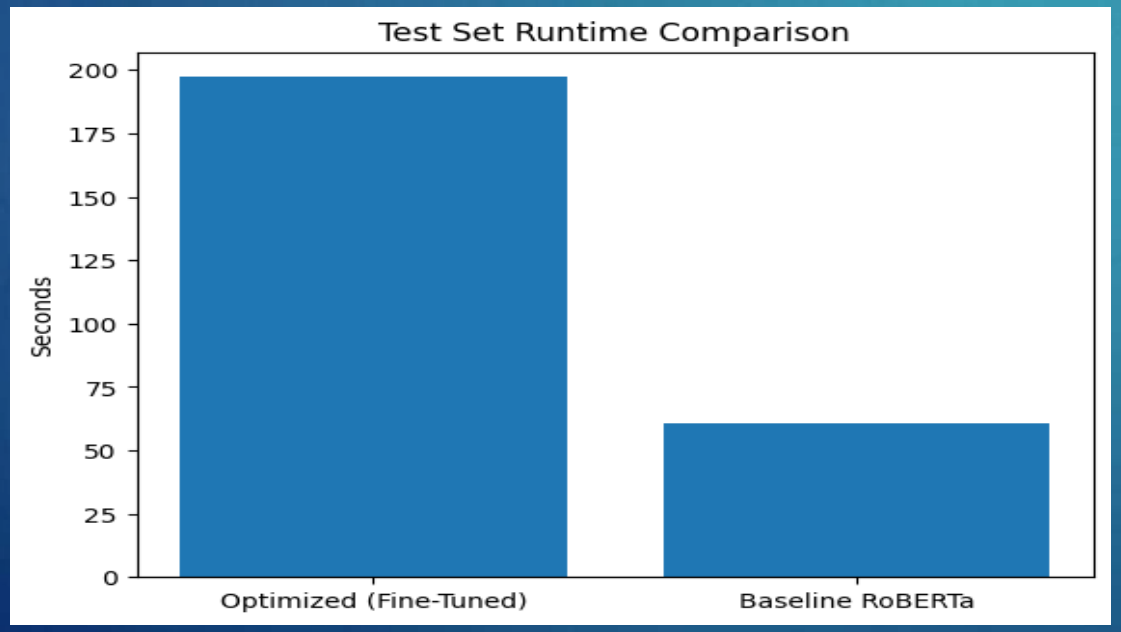


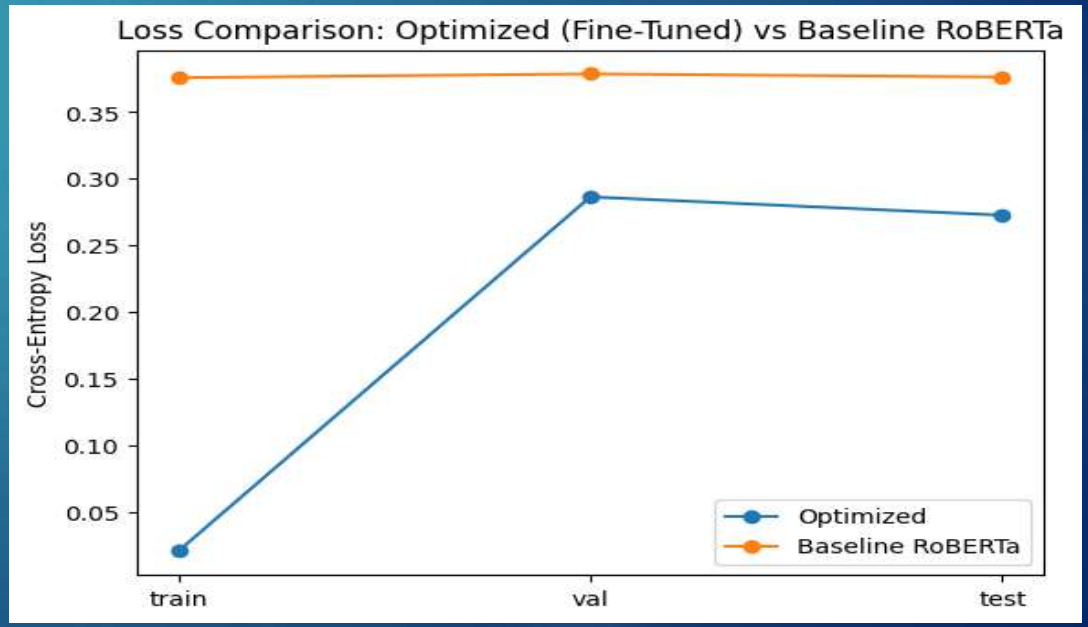
Comparison: Optimized vs Baseline

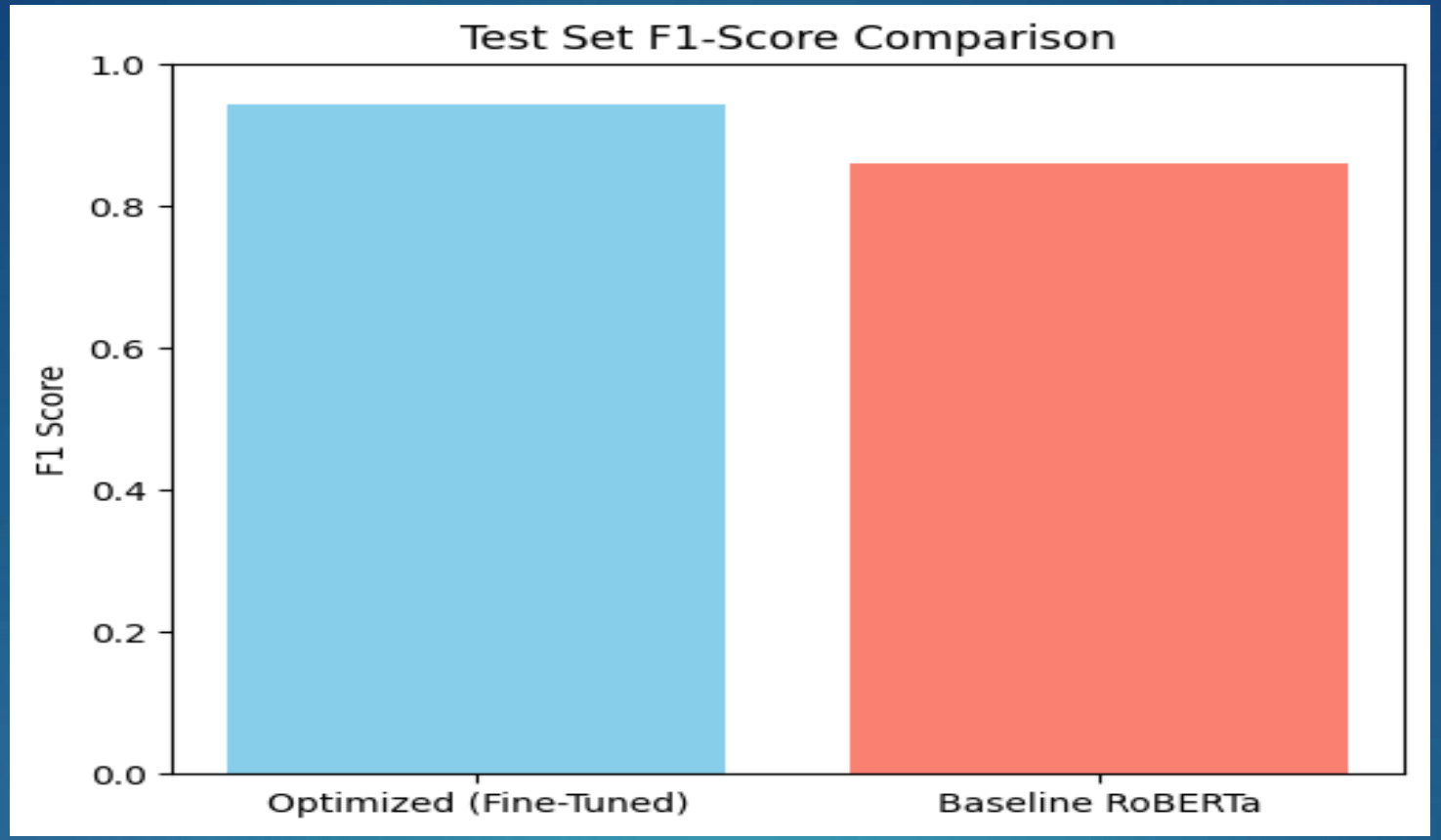
Metric	Optimized (Fine-Tuned)	Baseline RoBERTa
Accuracy	0.9435	0.8626
Precision	0.9355	0.8726
Recall	0.9516	0.8492
F1-score	0.9435	0.8607
Error	0.0570	0.1374
Loss	0.2727	0.3762
Runtime (s)	197.21	60.74

So, from the above table it can be clearly seen that my reimplemented and fine Tuned RoBERTa model performed quite better than the SOTA baseline RoBERTa model. The cross-entropy loss for optimized model is very less as compared to the base model. The Primary metric for comparison I took was F1 score and Optimized model has higher F1 score as compared to the baseline model. So, the optimized model significantly outperforms the baseline across all metrics. Fine-tuning all layers and hyperparameter tuning led to higher F1-score (0.9435 vs ~0.8607).


Test Set Metrics: Optimized (Fine-Tuned) vs Baseline RoBERTa
Optimized (Fine-Tuned)
Baseline RoBERTa
Value
0.0
0.2
0.4
0.6
0.8
ACC
PRECISION
RECALL
F1
ERROR
LOSS


Test Set Runtime Comparison
Seconds
0
25
50
75
100
125
150
175
200
Optimized (Fine-Tuned)
Baseline RoBERTa


Loss Comparison: Optimized (Fine-Tuned) vs Baseline RoBERTa
Cross-Entropy Loss
0.05
0.10
0.15
0.20
0.25
0.30
0.35
Optimized
Baseline RoBERTa
train
val
test



The above plot shows that the F1 score of the Optimized model is significantly higher and thus shows better performance than the base model.

Deployed Model and Inference Setup

- **Web Application Implementation** (Streamlit-Based Deployment)
- The sentiment analysis system is deployed as a Streamlit web application, offering:
- Text input for user reviews
- Real-time prediction
- Confidence score display
- Decision strength visualization
- Warning messages for ambiguous sentiment

The deployed pipeline follows these steps:

1. User enters a movie review in the web interface

2. Text is tokenized using the RoBERTa tokenizer

3. Tokenized inputs are passed through the fine-tuned model

4. Model outputs raw logits for positive and negative sentiment

5. Temperature scaling is applied to logits

6. Final prediction, confidence score, and decision strength are displayed on the web app

Conclusion

- This project demonstrates a complete end-to-end deep learning workflow: from dataset selection and preprocessing to model training, optimization, evaluation, deployment, and reflection. The fine-tuned RoBERTa model achieved strong performance on the IMDB dataset and was successfully transformed into a usable web application.
- The project highlights both the strengths and limitations of deep learning for sentiment analysis and provides practical insights into real-world NLP system design.