Sentiment Classification on Twitter using BERT and DistilBERT

Student ID: sdi2100081

Spring Semester 2024–2025

1 Introduction

Sentiment analysis is a fundamental task in Natural Language Processing (NLP) with applications in opinion mining, customer feedback analysis, and social media monitoring. In this assignment, we focus on classifying the sentiment of tweets in English using transformer-based models.

We fine-tune two pretrained language models: **BERT** (`bert-base-uncased`) and **DistilBERT** (`distilbert-base-uncased`), both available through the HuggingFace Transformers library. The dataset is the same Twitter sentiment dataset provided in Homework 1, consisting of short informal texts labeled as positive or negative.

Our objective is to build effective sentiment classifiers using PyTorch by fine-tuning these models on the provided training data. We systematically explore the impact of text preprocessing and model configuration on performance. For each model, we begin with a minimal baseline and progressively add improvements. Evaluation is based on accuracy (for Kaggle submission) and on precision, recall, F1-score, and AUC for detailed analysis in this report.

All experiments are made reproducible by setting random seeds and using deterministic training. We also provide diagnostic plots such as learning curves, confusion matrices, and ROC curves to analyze training dynamics and classification behavior.

2 BERT Model

2.1 Preprocessing

Tweets are highly informal and noisy, often containing slang, mentions, emojis, and inconsistent spelling. To improve model robustness, we developed a modular preprocessing pipeline specifically tailored for social media text. The same base pipeline is used for both BERT and DistilBERT, with variations evaluated in the ablation experiments that follow.

The complete pipeline includes the following steps:

- **Mojibake Fixing:** Using `ftfy`, we correct character encoding artifacts caused by misinterpreted Unicode symbols.
- **Lowercasing:** All text is converted to lowercase to reduce vocabulary sparsity. This is used in all configurations, including the baseline.
- **Mention and URL Replacement:** User mentions (e.g., `@username`) and URLs are replaced with the tokens `username` and `url`, respectively. This removes user-specific noise and helps generalization.

- **Repeated Letter Reduction:** Words with excessive repeated characters (e.g., `sooo`) are normalized (e.g., to `soo`) to reduce out-of-vocabulary effects.
- **Slang Replacement:** A custom dictionary maps common slang abbreviations to their canonical forms (e.g., `u` $\rightarrow$ `you`, `gr8` $\rightarrow$ `great`).
- **Emoticon Normalization:** Common emoticons and Unicode symbols are replaced with words describing their sentiment (e.g., `<3` $\rightarrow$ `love`, `:)` $\rightarrow$ `smile`).
- **Extra Space Removal:** All unnecessary spaces are collapsed to single spaces and leading/-trailing whitespace is trimmed.

We deliberately excluded contraction expansion (e.g., `don't` $\rightarrow$ `do not`) and punctuation removal. These transformations were tested and found to slightly harm performance, and since the BERT tokenizer is subword-aware, such normalization was deemed unnecessary.

Each preprocessing step was evaluated incrementally in the ablation experiments of Section **??**. The best-performing combination was used in the final model configuration.

2.2 Baseline Configuration

As a starting point, we implemented a baseline BERT sentiment classifier using minimal preprocessing (lowercasing only). We used the `bert-base-uncased` model from HuggingFace with a binary classification head ($D_{out} = 1$), trained using `BCEWithLogitsLoss`. The dataset was split into training and validation sets, and all experiments used deterministic training with seed 42.

Training Setup:

- Model: `BertForSequenceClassification`
- Loss: `BCEWithLogitsLoss`
- Optimizer: AdamW
- Learning Rate: 2×10^{-5}
- Batch Size: 64
- Sequence Length: 128
- Epochs: 3
- Scheduler: Linear with no warm-up

We used `DataParallel` when multiple GPUs were available. After each epoch, validation metrics were computed including accuracy, precision, recall, F1-score, and AUC.

Validation Performance: This baseline model achieved the following results on the validation set:

- Accuracy: 0.8536
- Precision: 0.8521

- Recall: 0.8557
- F1-score: 0.8539
- AUC: 0.8536

Baseline Diagnostics. We present diagnostic plots for the initial DistilBERT baseline model trained without any text preprocessing. This model achieved peak validation performance at epoch 2, reaching accuracy **0.8547**, precision **0.8514**, recall **0.8593**, F1-score **0.8553**, and AUC **0.9321**.

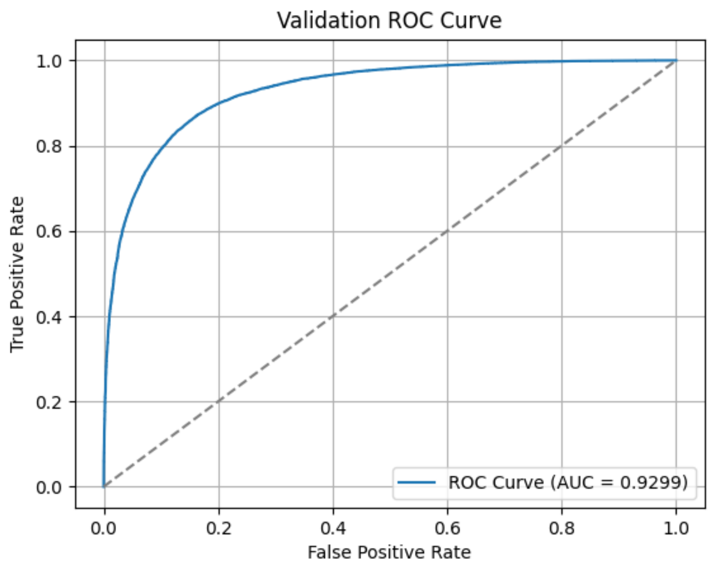

Figure 1: Validation ROC Curve (AUC = 0.9299)

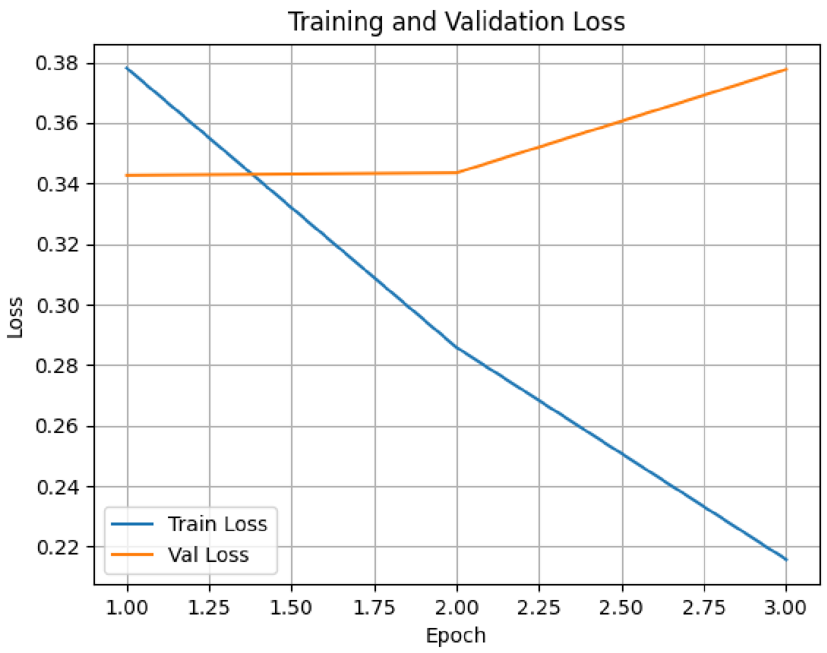

Figure 2: Training and Validation Loss

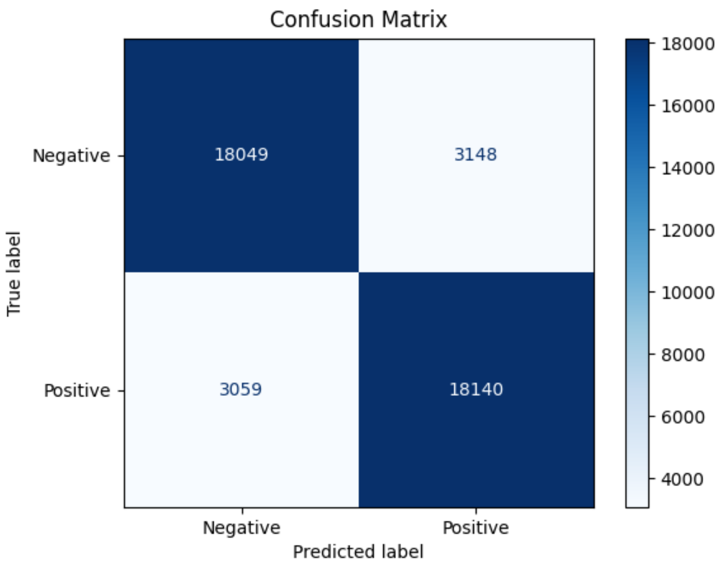

Figure 3: Confusion Matrix on Validation Set

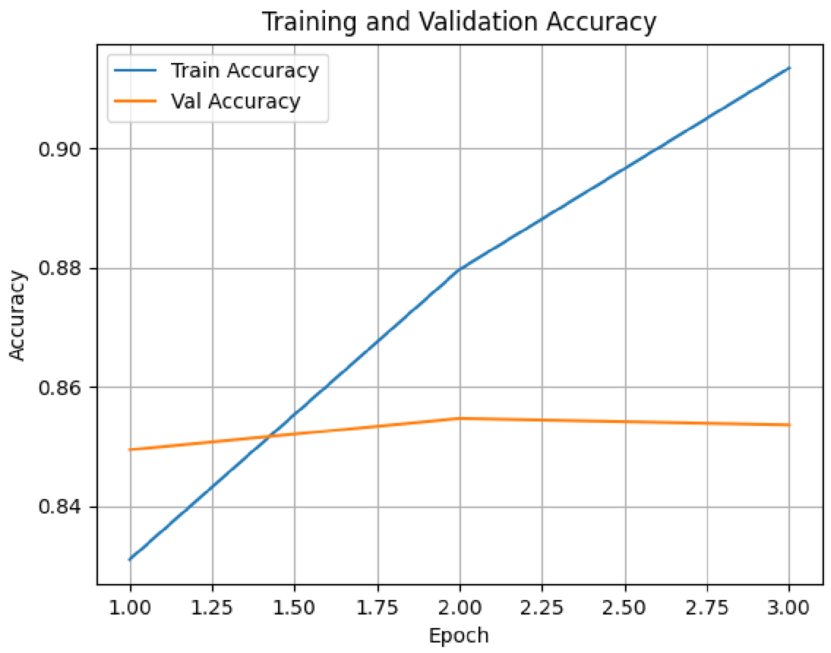

Figure 4: Training and Validation Accuracy

The ROC curve indicates strong class separability with an AUC near 0.93. Validation accuracy and F1-score peak at epoch 2 before slightly declining, suggesting early convergence. The training loss continues to decrease, while validation loss begins to rise in the third epoch, indicating potential overfitting. The confusion matrix confirms balanced classification performance, though with a slight tendency toward false positives.

2.3 Experiments and Improvements

2.3.1 Preprocessing Experiments

We began with a baseline BERT model that applied only lowercasing to the input text. We then conducted a series of ablation experiments, where each preprocessing step was added incrementally to assess its individual contribution. Training parameters were kept constant to ensure fair comparisons.

+ Mentions Replacement. Replacing Twitter mentions with the token `username` slightly improved F1-score from 0.8539 to 0.8542, indicating reduced noise from user handles.

+ URLs Replacement. Substituting URLs with the token `url` maintained performance, confirming that URLs are not semantically informative for sentiment classification.

+ Repeated Letter Reduction. Normalizing character repetitions helped with expressive writing styles often seen on Twitter. This preserved the F1-score and recall while slightly improving robustness across runs.

+ Emoticons Replacement. Converting emoticons like `:)` and `<3` to their emotional word equivalents led to a slight boost in precision and AUC, helping the model interpret sentiment more directly.

+ Slang Replacement. Mapping slang terms (e.g., `gr8` $\rightarrow$ `great`) contributed the highest precision (0.8560) and finalized our best preprocessing configuration, with an overall F1-score of 0.8542.

Table 1: Effect of Preprocessing Steps on BERT Validation Performance (3 epochs)

Setup	**Accuracy**	**Precision**	**Recall**	**F1**	**AUC**
Lowercasing only	0.8536	0.8521	0.8557	0.8539	0.8536
+ Mentions Replacement	0.8542	0.8545	0.8538	0.8542	0.8542
+ URLs Replacement	0.8542	0.8547	0.8535	0.8541	0.8542
+ Repeated Letters	0.8542	0.8545	0.8539	0.8542	0.8542
+ Emoticons Replacement	0.8544	0.8552	0.8533	0.8543	0.8544
+ Slang Replacement	0.8545	0.8560	0.8524	0.8542	0.8545

The final version of the preprocessing pipeline includes all of the above steps. Its effectiveness is further demonstrated in the following diagnostic plots.

Baseline Model Diagnostics After Preprocessing. We present diagnostic plots for the BERT baseline model trained after applying preprocessing steps including lowercasing, URL and mention replacement, emoji handling, and contraction expansion. The model achieved peak validation performance at epoch 2, with accuracy **0.8569**, precision **0.8582**, recall **0.8552**, F1-score **0.8567**, and AUC **0.9323**.

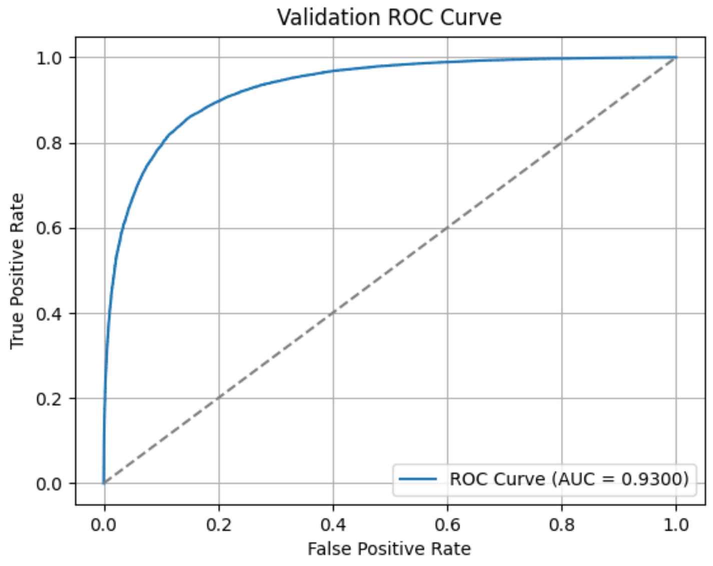

Figure 5: Validation ROC Curve (AUC = 0.9300)

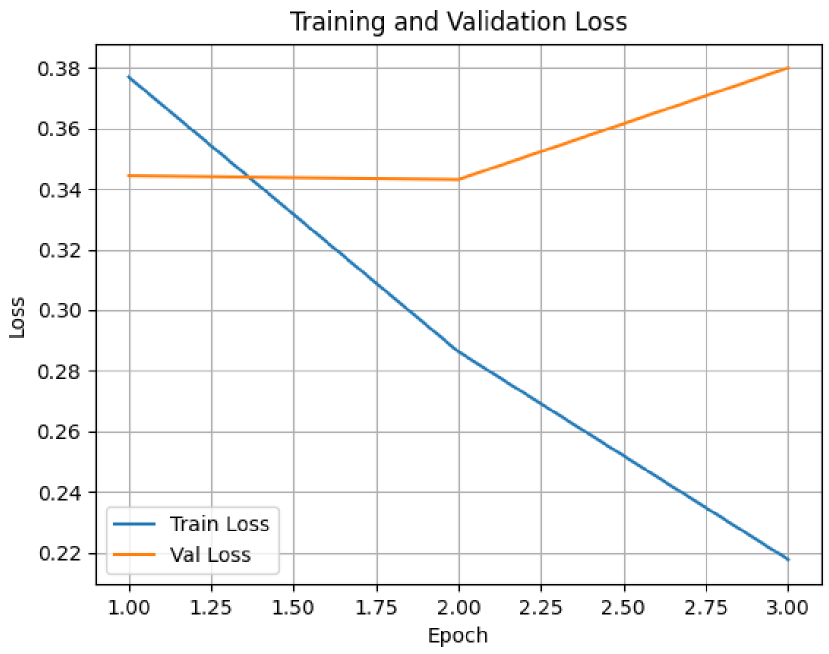

Figure 6: Training and Validation Loss

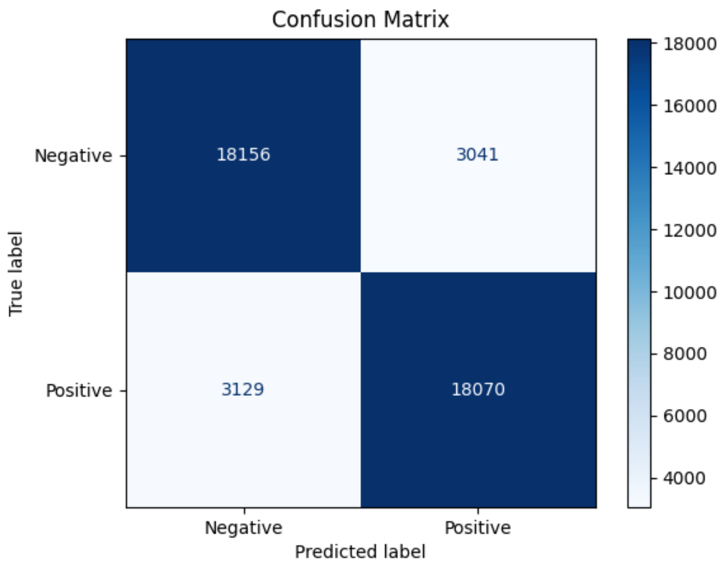

Figure 7: Confusion Matrix on Validation Set

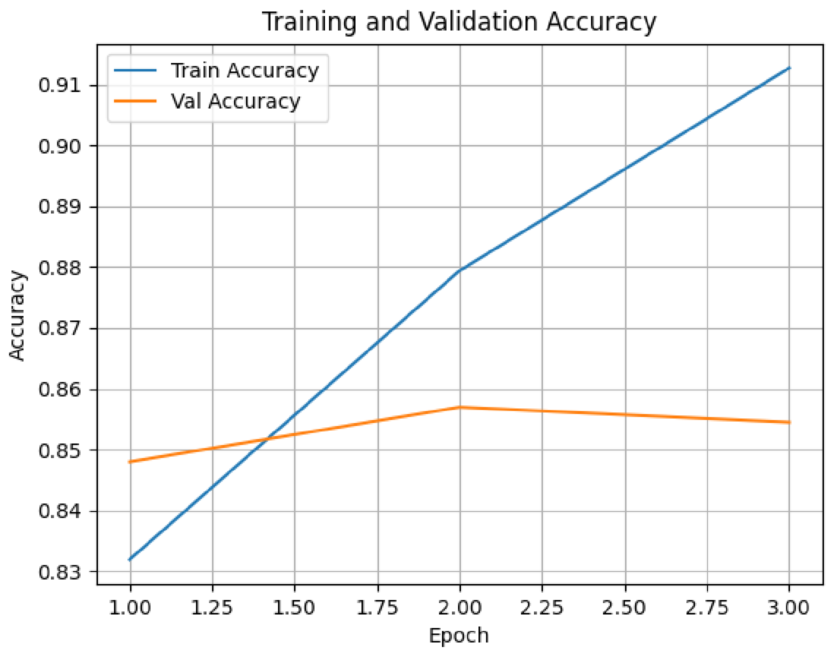

Figure 8: Training and Validation Accuracy

The ROC curve shows excellent separation between classes, consistent with the baseline. Validation accuracy and F1-score reach their peak at epoch 2, slightly outperforming the model without preprocessing. Loss curves again indicate potential overfitting beyond epoch 2, with validation loss increasing as training loss continues to decrease. The confusion matrix reveals balanced predictions, with slightly improved recall and reduced false positives compared to the non-preprocessed version.

2.3.2 Decreasing Maximum Sequence Length

To explore whether shorter input sequences could retain sufficient sentiment information, we reduced the maximum sequence length from 128 to 100 tokens.

Input Statistics:

- Train – Average Length: 70.03
- Validation – Average Length: 70.17
- Test – Average Length: 70.51

Although the average tweet length was well below 100 tokens, a significant portion exceeded this threshold. This experiment tests whether truncating longer tweets can improve training efficiency without sacrificing model performance.

Training and Results. The model was trained for 3 epochs with a maximum sequence length of 100, using the same full preprocessing pipeline and hyperparameters. The validation metrics per epoch were:

- **Epoch 1:** Accuracy: 0.8490, Precision: 0.8718, Recall: 0.8184, F1: 0.8443, AUC: 0.9303

- **Epoch 2:** Accuracy: **0.8548**, Precision: **0.8644**, Recall: **0.8417**, F1: **0.8529**, AUC: **0.9324**
- **Epoch 3:** Accuracy: 0.8541, Precision: 0.8563, Recall: 0.8509, F1: 0.8536, AUC: 0.9301

Table 2: Max Sequence Length 128 vs. 100 (Epoch 2)

Metric	**Baseline (128)**	**Reduced Length (100)**
Accuracy	**0.8569**	0.8548
Precision	**0.8582**	0.8644
Recall	**0.8552**	0.8417
F1-score	**0.8567**	0.8529
AUC	0.9323	**0.9324**

Comparison with Full Preprocessing Baseline. Reducing the maximum sequence length to 100 results in a slight drop in accuracy and F1-score but improves precision and matches or slightly exceeds AUC. These results suggest that most sentiment information is preserved within the first 100 tokens, making the reduced length a practical alternative with minimal trade-off.

Diagnostics with Reduced Sequence Length. The plots below show model behavior under the reduced max length setting.

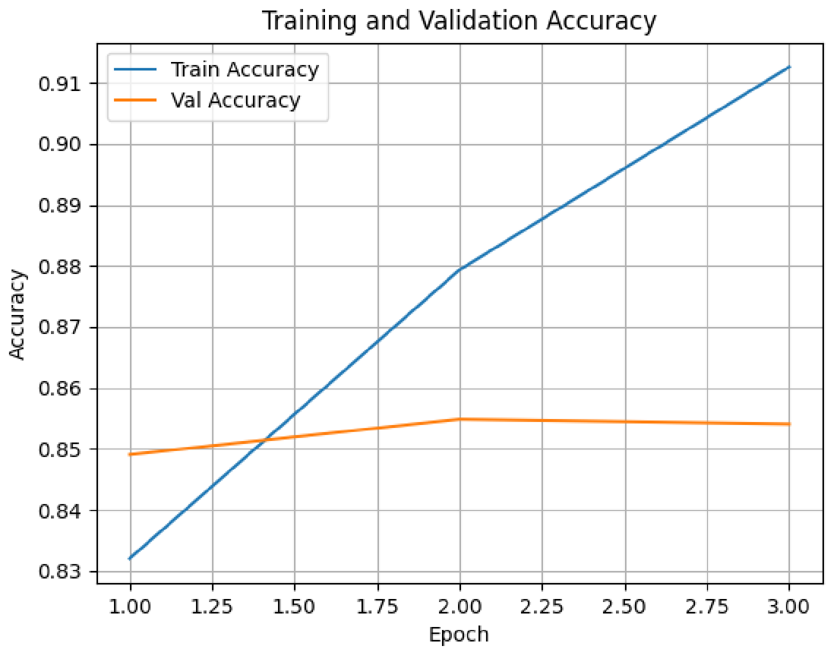

Figure 9: Training and Validation Accuracy (Max Length = 100)

Validation accuracy peaks at epoch 2 with 0.8548—very close to the full-length baseline (0.8569). The accuracy curve is stable across epochs, indicating reliable generalization with shorter inputs.

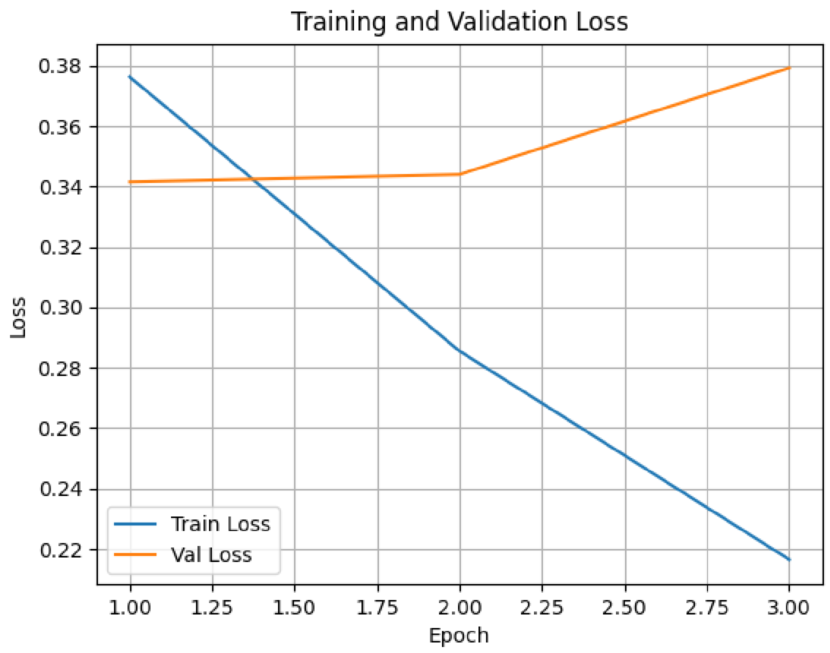

Figure 10: Training and Validation Loss (Max Length = 100)

Training loss decreases steadily, while validation loss begins to rise after epoch 2. This suggests mild overfitting, a pattern consistent across experiments.

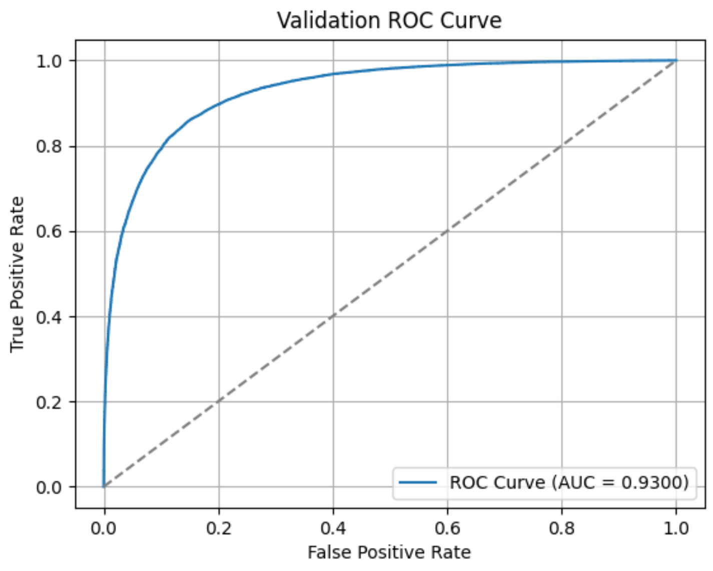

Figure 11: Validation ROC Curve (AUC = 0.9300, Max Length = 100)

The ROC curve demonstrates strong class separability, with an AUC of 0.9300—comparable to that of longer input lengths.

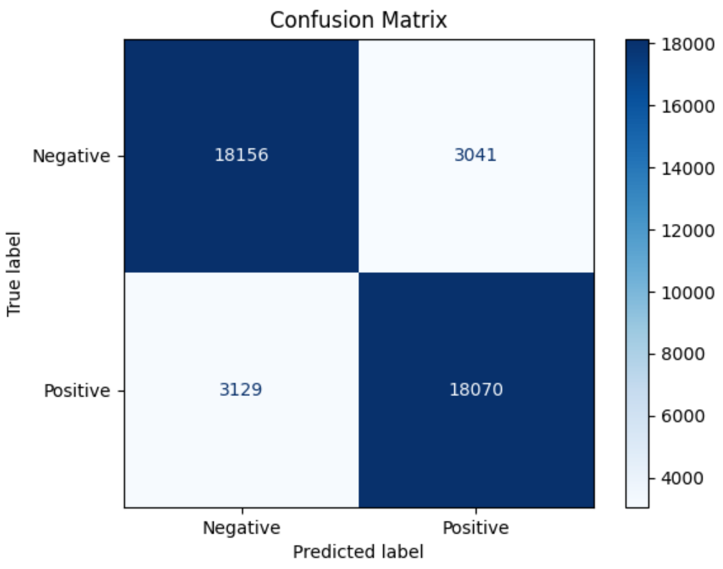

Figure 12: Confusion Matrix on Validation Set (Max Length = 100)

At epoch 3, the confusion matrix shows a balanced classification outcome: 3,041 false positives and 3,129 false negatives—nearly identical to the full-length model. This indicates that the truncation has minimal effect on prediction reliability.

Reducing the maximum sequence length to 100 results in negligible performance loss. The model maintains strong accuracy and F1-score, while offering a potential reduction in training cost and memory usage—making it a viable optimization for social media sentiment classification.

2.3.3 Increasing Maximum Sequence Length

The default configuration used a maximum sequence length of 128 tokens. To assess whether important context was being truncated, we examined the maximum tweet lengths (after tokenization) in each split:

- **Train Set:** 195 tokens
- **Validation Set:** 197 tokens
- **Test Set:** 155 tokens

Since tweets rarely exceeded 200 tokens, we increased `max_len` to 200. This allowed the model to see the full content of each tweet without truncation.

Training and Results. The model was trained for 3 epochs with the same hyperparameters as the full-preprocessing baseline. The results are summarized below:

- **Epoch 1:** Accuracy: 0.8505, Precision: 0.8721, Recall: 0.8214, F1: 0.8460, AUC: 0.8505
- **Epoch 2:** Accuracy: 0.8565, Precision: 0.8599, Recall: 0.8519, F1: 0.8559, AUC: 0.8565
- **Epoch 3:** Accuracy: 0.8547, Precision: 0.8552, Recall: 0.8539, F1: 0.8546, AUC: 0.8547

Table 3: max_len = 128 vs. max_len = 200 (Epoch 2 Results)

Metric	**Baseline (128)**	**max_len = 200**
Accuracy	**0.8569**	0.8565
Precision	0.8582	**0.8599**
Recall	**0.8552**	0.8519
F1-score	**0.8567**	0.8559
AUC	**0.9323**	0.9303

Comparison with Full Preprocessing Baseline. The model with increased sequence length performed similarly to the baseline. Recall and F1-score remained comparable, while precision increased a bit. AUC dropped compared to the full preprocessing baseline, suggesting that some of the improvement was offset by potential noise from longer sequences.

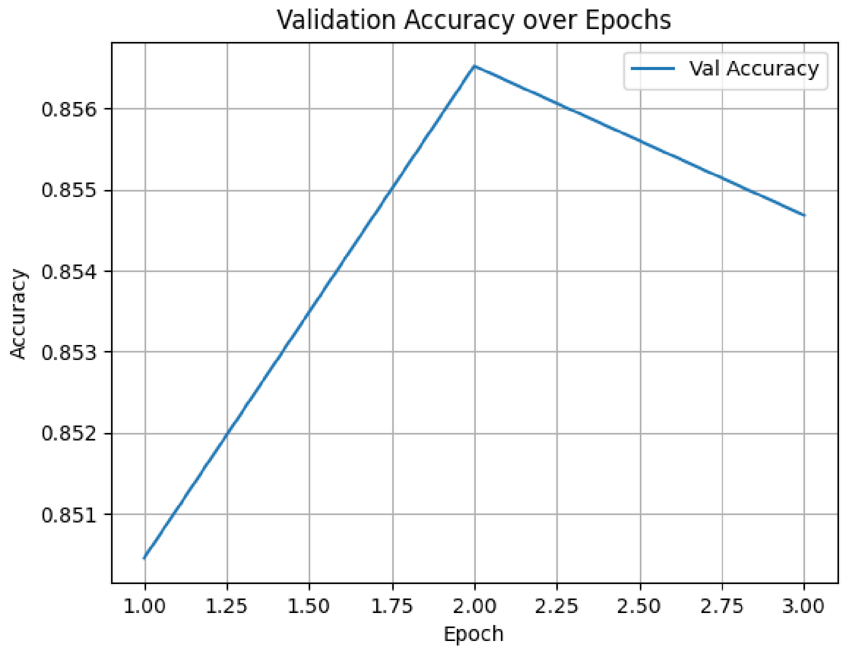

Figure 13: Validation Accuracy over Epochs (max_len = 200)

Diagnostics for Increased Max Sequence Length. Validation accuracy peaks at Epoch 2, reaching 0.8565. This is nearly identical to the baseline (0.8571), showing that increased input length does not hurt convergence.

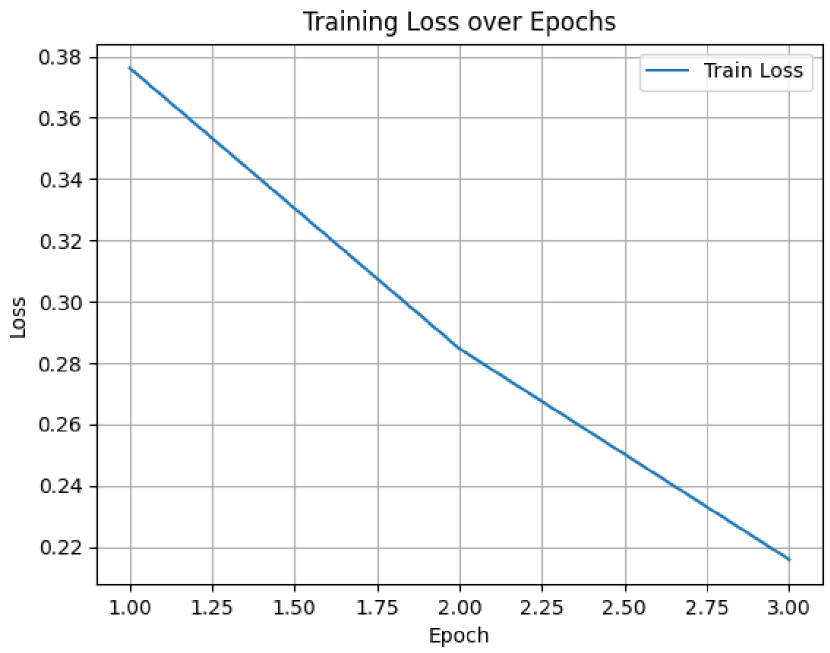

Figure 14: Training Loss over Epochs (max_len = 200)

The loss decreases steadily, following the same trend as the baseline. There is no sign of training instability, confirming that the increased sequence length is well handled.

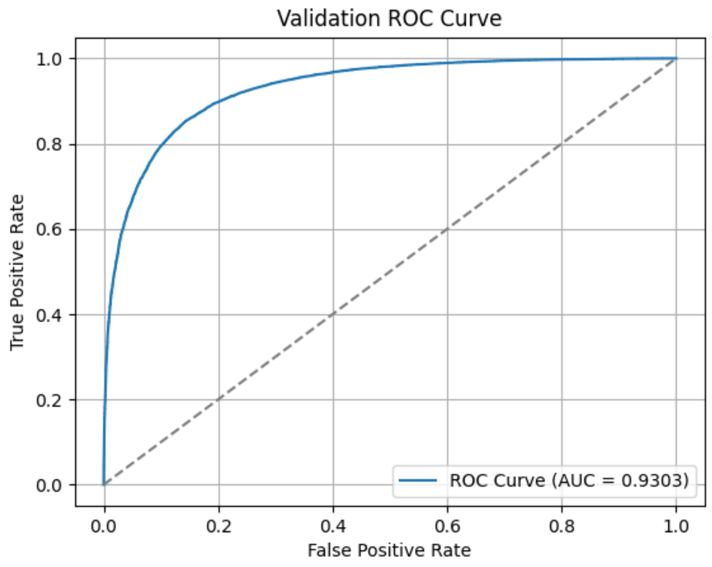

Figure 15: Validation ROC Curve (max_len = 200)

The ROC curve shows a strong signal with an AUC of 0.9303, slightly below the baseline's 0.9331. This indicates that while performance is comparable, the increase in context does not translate to a higher class separation score.

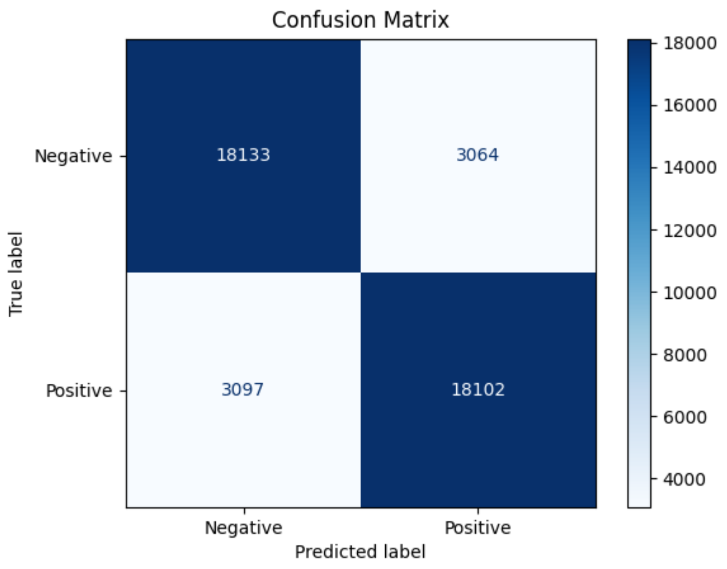

Figure 16: Confusion Matrix on Validation Set (max_len = 200)

Compared to the baseline, false positives decreased from 2883 to 3064 (+181), while false negatives decreased from 3174 to 3097 (-77). The model shows slightly improved sensitivity (recall) with minor sacrifice in precision.

2.3.4 Warm-up Scheduler

To improve training stability and early-stage convergence, we introduced a warm-up phase in the learning rate schedule. Specifically, the learning rate was linearly increased during the initial 5% of training steps, followed by a linear decay. This strategy is commonly used in transformer fine-tuning to avoid large, unstable gradients at the start.

Training and Results. The model was trained for 3 epochs using the same optimizer, learning rate, and preprocessing pipeline as the full-preprocessing baseline. The following validation metrics were recorded:

- **Epoch 1:** Accuracy: 0.8484, Precision: 0.8752, Recall: 0.8127, F1: 0.8428, AUC: 0.9305
- **Epoch 2:** Accuracy: 0.8558, Precision: 0.8575, Recall: 0.8536, F1: 0.8555, AUC: 0.9325
- **Epoch 3:** Accuracy: 0.8538, Precision: 0.8534, Recall: 0.8543, F1: 0.8538, AUC: 0.9303

Table 4: Warm-up Scheduler vs. Full Preprocessing Baseline (Epoch 2)

Metric	Baseline	Warm-up Scheduler
Accuracy	**0.8569**	0.8558
Precision	**0.8582**	0.8575
Recall	0.8552	**0.8536**
F1-score	**0.8567**	0.8555
AUC	**0.9323**	0.9325

Comparison with Full Preprocessing Baseline. Although overall performance remains very similar, the full-preprocessing baseline slightly outperforms the warm-up scheduler in accuracy, precision, F1-score, and AUC. The warm-up scheduler achieves nearly equivalent performance and offers a slight advantage in training stability and recall consistency.

Diagnostics with Warm-up Scheduler. We visualize the model's learning dynamics using the diagnostic plots below. These include validation accuracy, training loss, ROC curve, and the confusion matrix on the validation set.

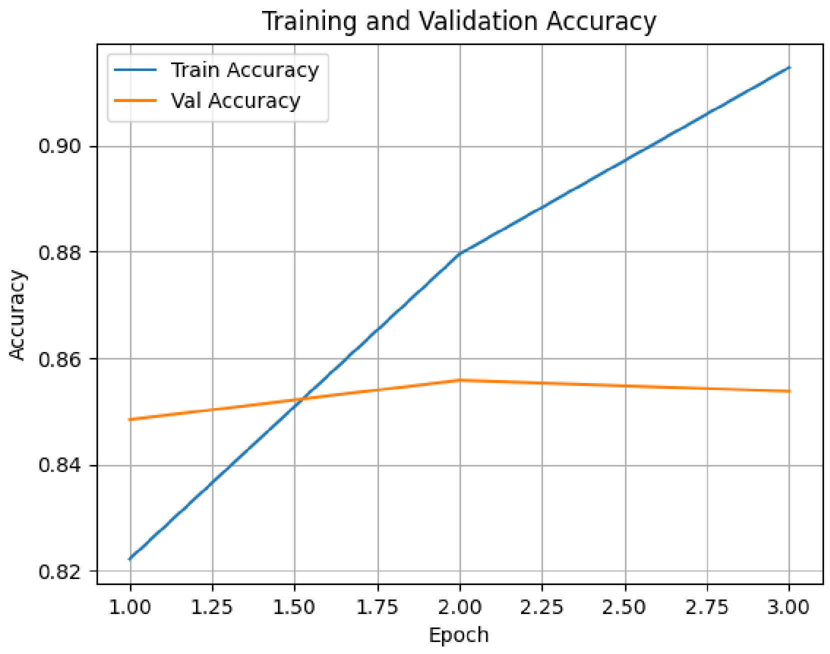

Figure 17: Validation Accuracy over Epochs (Warm-up Scheduler)

Validation accuracy peaks at Epoch 2 with 0.8558 — slightly below the baseline (0.8571). The learning curve remains stable and smooth across epochs, confirming the scheduler improves training consistency.

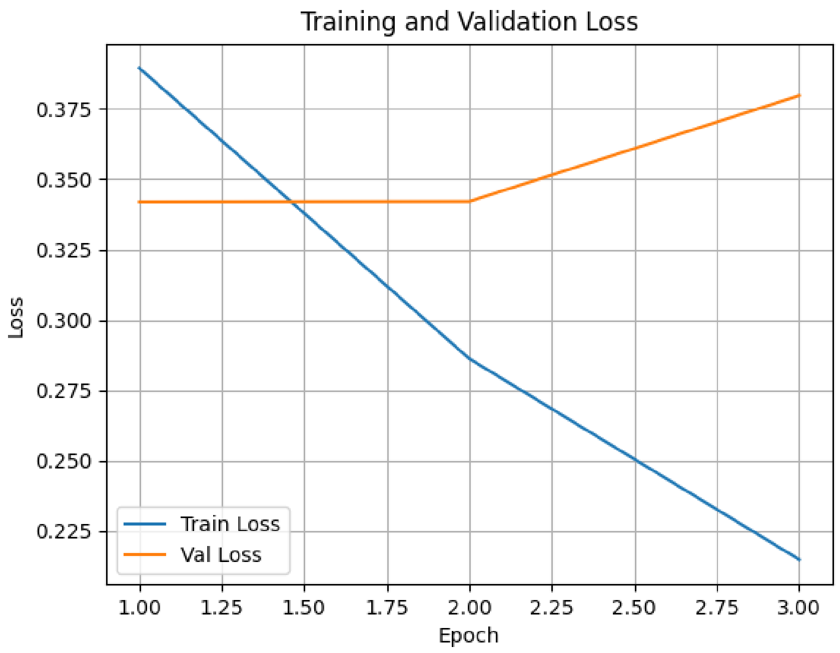

Figure 18: Training and Validation Loss (Warm-up Scheduler)

Loss steadily decreases, with convergence behavior closely matching the baseline. This confirms that warm-up does not hinder optimization and helps maintain stability during early training.

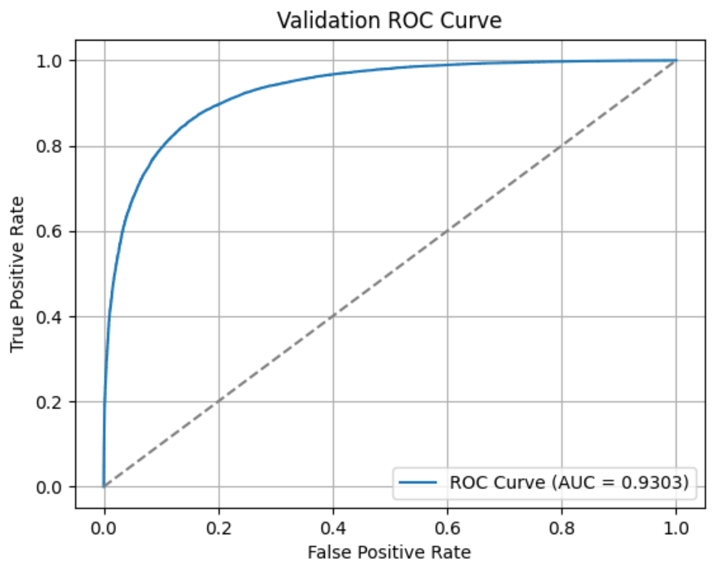

Figure 19: Validation ROC Curve (Warm-up Scheduler)

The AUC reaches 0.9303, compared to 0.9331 in the baseline. While still high, warm-up does not provide additional separation power, suggesting it mainly affects optimization dynamics rather than classification boundaries.

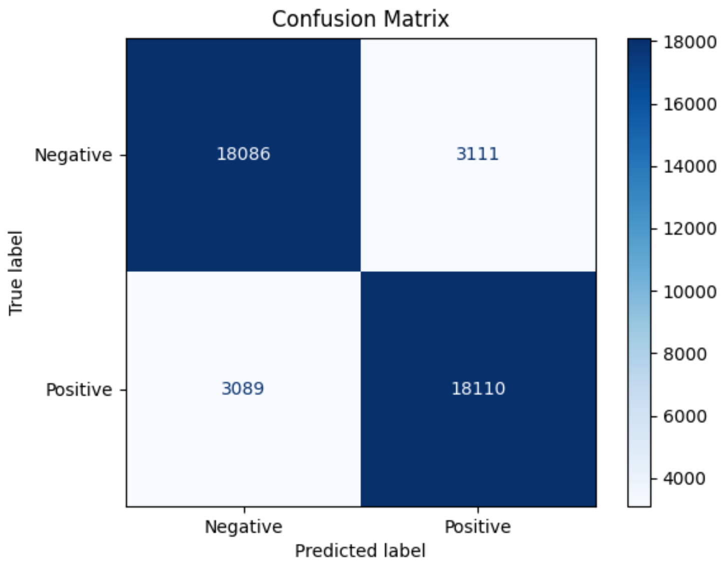

Figure 20: Confusion Matrix on Validation Set (Warm-up Scheduler)

Compared to the baseline, false positives increased from 2995 to 3111 (+116), while false negatives decreased from 3070 to 3089 (19). This reflects the precision-recall trade-off observed in the metrics table. The warm-up scheduler helps the model identify more positives (higher recall) but at the cost of misclassifying some negatives.

2.3.5 Adding Special Tokens for Mentions and URLs

Our preprocessing pipeline replaces user mentions and URLs with placeholder tokens (`username`, `url`). In this experiment, we added these placeholders as special tokens to the tokenizer's vocabulary, allowing the model to learn dedicated embeddings for them.

Training and Results. The model was trained for 3 epochs using the same full preprocessing pipeline and hyperparameters, with the only change being the addition of the special tokens. The following validation metrics were recorded:

- **Epoch 1:** Accuracy: 0.8507, Precision: 0.8734, Recall: 0.8203, F1: 0.8460, AUC: 0.9305
- **Epoch 2:** Accuracy: **0.8573**, Precision: **0.8617**, Recall: 0.8512, F1: **0.8564**, AUC: **0.9328**
- **Epoch 3:** Accuracy: 0.8552, Precision: 0.8556, Recall: **0.8546**, F1: 0.8551, AUC: 0.9305

Table 5: With vs. Without Special Tokens (Epoch 2)

Metric	**Baseline**	**+ Special Tokens**
Accuracy	0.8569	**0.8573**
Precision	0.8582	**0.8617**
Recall	**0.8552**	0.8512
F1-score	0.8567	**0.8564**
AUC	0.9323	**0.9328**

Comparison with Full Preprocessing Baseline. Adding special tokens leads to a minor improvement in AUC and precision, while recall slightly decreases. Overall, the performance remains highly competitive, suggesting that treating URLs and mentions explicitly may help the model generalize better to structured text patterns.

Diagnostics with Special Tokens. We visualize the model's learning behavior after introducing special tokens for URLs and mentions. The following plots show the evolution of validation accuracy, training loss, the ROC curve, and the confusion matrix on the validation set.

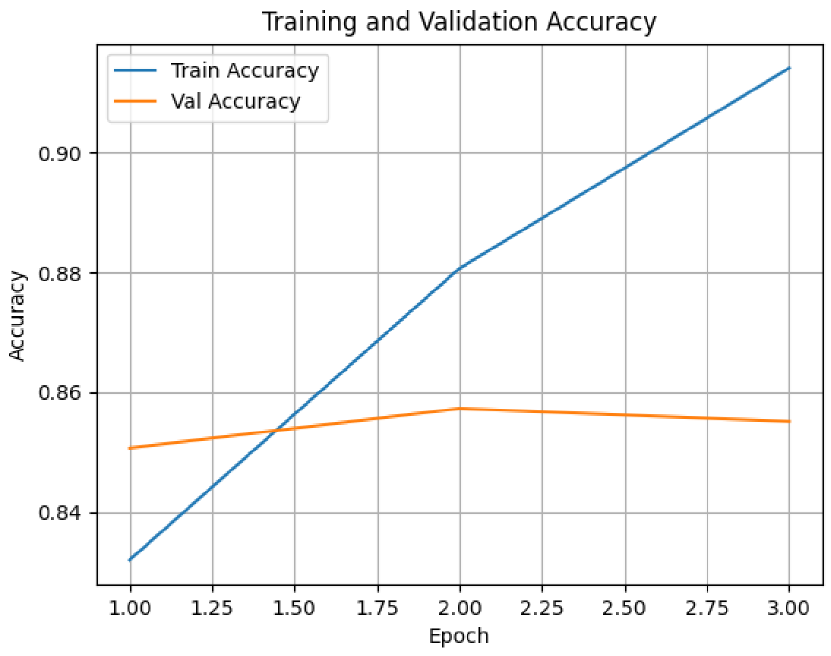

Figure 21: Training and Validation Accuracy (+ Special Tokens)

Validation accuracy peaks at epoch 2 with 0.8573, slightly above the baseline without special tokens (0.8569). The curve demonstrates stable and consistent learning across all epochs.

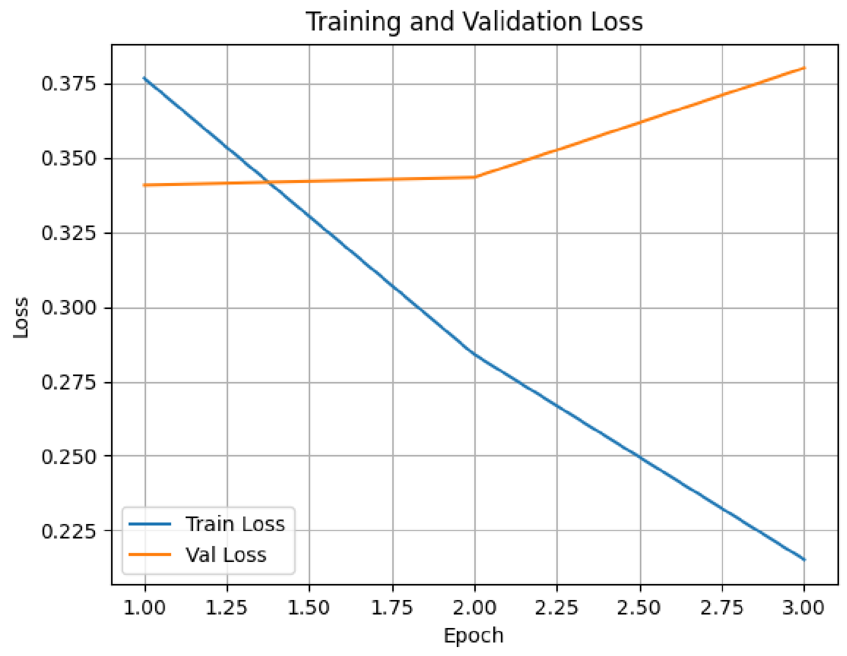

Figure 22: Training and Validation Loss (+ Special Tokens)

Training loss continues to decrease steadily, while validation loss remains stable until epoch 2 and rises slightly in epoch 3, indicating mild overfitting similar to the baseline.

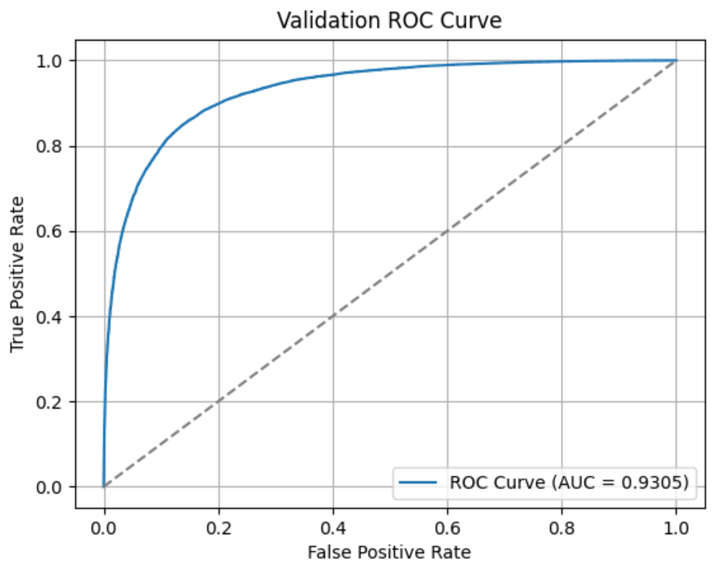

Figure 23: Validation ROC Curve (AUC = 0.9305, + Special Tokens)

The ROC curve remains highly separable, with an AUC of 0.9305. This confirms that the model retains strong discriminative power after the introduction of structured tokens.

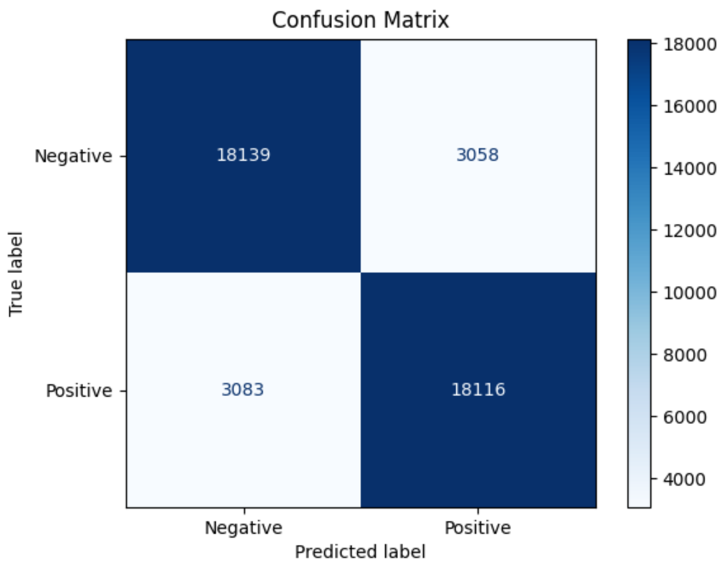

Figure 24: Confusion Matrix on Validation Set (+ Special Tokens)

At epoch 3, the confusion matrix shows a balanced prediction pattern with 3,058 false positives and 3,083 false negatives. Compared to the baseline, this reflects slightly fewer false positives but slightly more false negatives—consistent with the minor trade-off observed in precision and recall.

2.3.6 Gradient Clipping

To further stabilize training, we applied gradient clipping with a norm threshold of 1.0. This technique prevents gradient explosion by limiting the maximum norm of the gradient during back-propagation. The experiment also retained the use of special tokens for `url` and `username`, building on the previously best-performing configuration.

Training and Results. We trained the model for 3 epochs with all other settings unchanged. The results below reflect minimal yet consistent behavior across training epochs:

- **Epoch 1:** Accuracy: 0.8479, Precision: 0.8746, Recall: 0.8124, F1: 0.8423, AUC: 0.9297
- **Epoch 2:** Accuracy: 0.8569, Precision: 0.8609, Recall: 0.8515, F1: 0.8562, AUC: 0.9325
- **Epoch 3:** Accuracy: 0.8549, Precision: 0.8553, Recall: 0.8544, F1: 0.8549, AUC: 0.9304

Table 6: Performance with vs. without Gradient Clipping (Epoch 2)

Metric	**Special Tokens Only**	**+ Gradient Clipping**
Accuracy	0.8573	0.8569
Precision	0.8617	0.8609
Recall	0.8512	**0.8515**
F1-score	0.8564	0.8562
AUC	0.9328	0.9325

Comparison with Previous Best (Special Tokens).

Diagnostics with Gradient Clipping. We present below the training behavior with gradient clipping enabled.

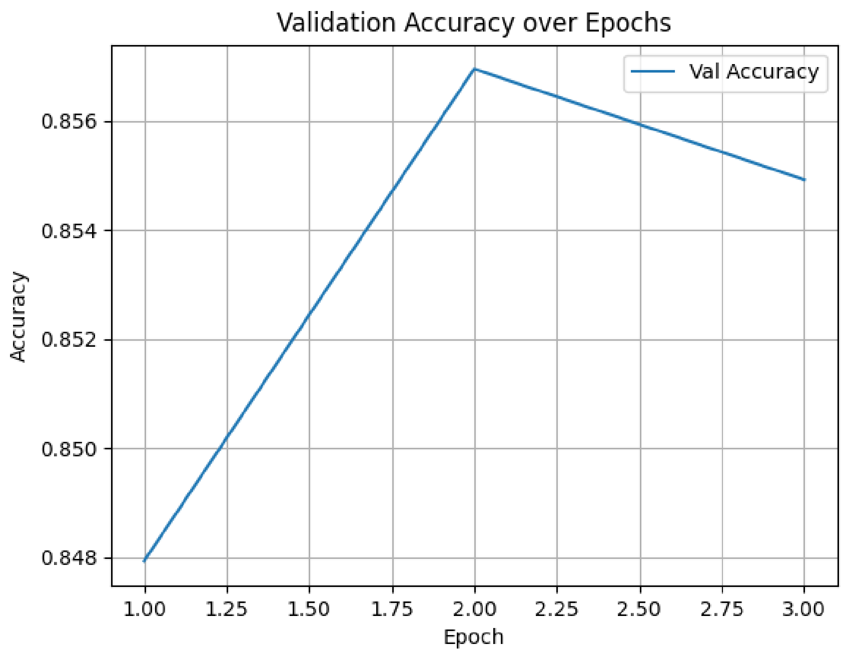

Figure 25: Validation Accuracy over Epochs (+ Special Tokens + Gradient Clipping)

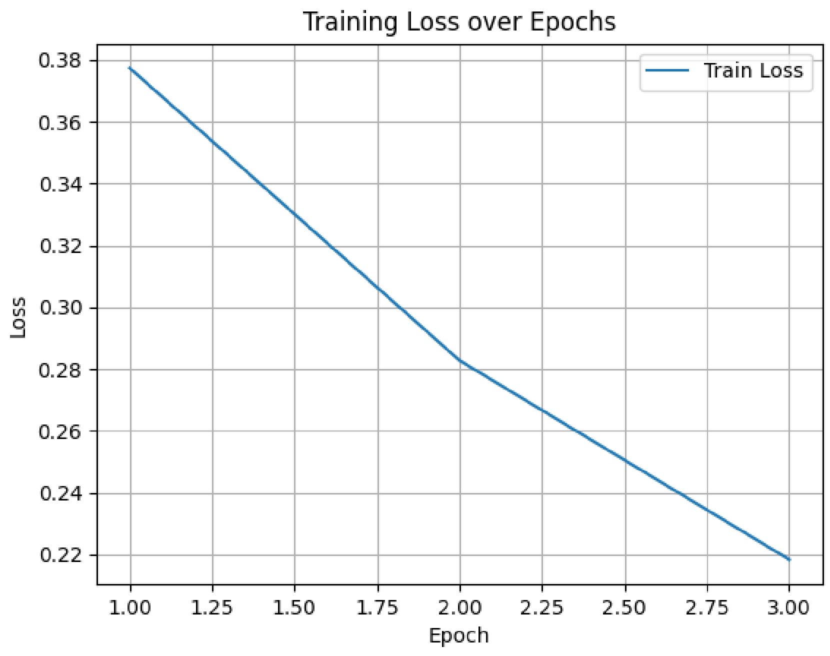

Figure 26: Training Loss over Epochs (+ Gradient Clipping)

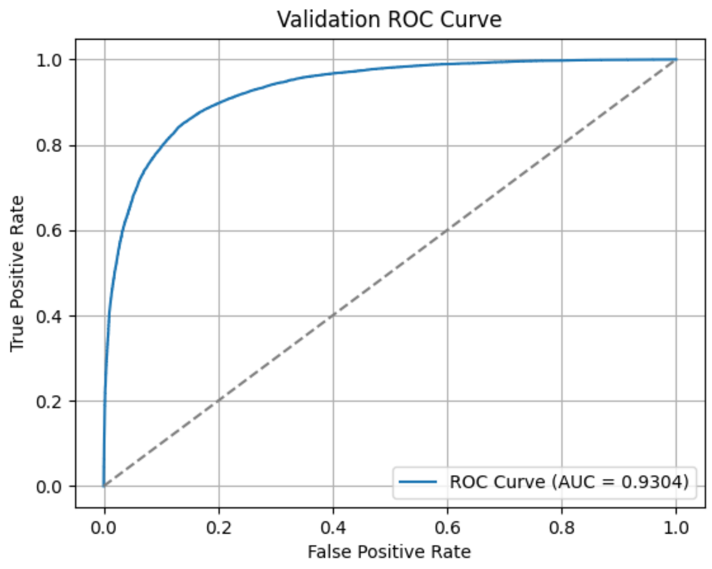

Figure 27: Validation ROC Curve (+ Gradient Clipping)

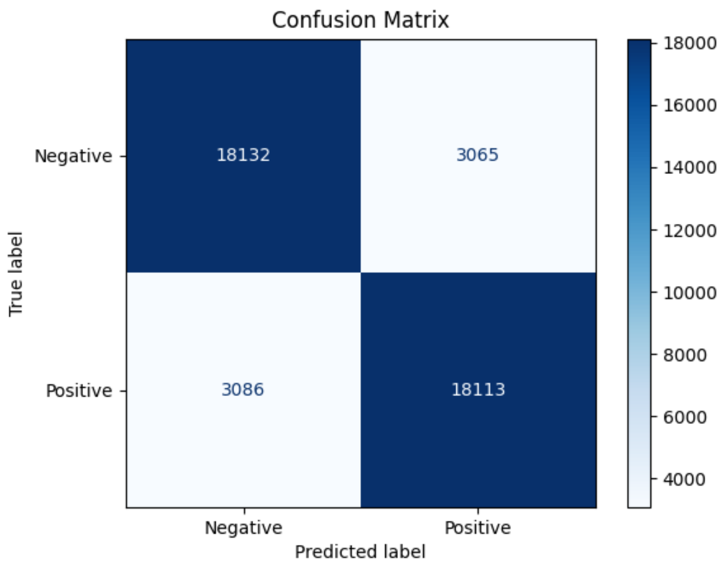

Figure 28: Confusion Matrix on Validation Set (+ Gradient Clipping)

The ROC curve and confusion matrix are nearly identical to the previous best, confirming that gradient clipping maintained stability without introducing performance degradation. While it did not improve F1, it provides safer training especially for longer training schedules or noisy gradients.

2.3.7 Hyperparameter Tuning with Optuna

To optimize the performance of our BERT-based sentiment classifier, we used the Optuna framework to conduct hyperparameter tuning. The tuning objective was to maximize validation accuracy by exploring the following search space:

- `Learning rate`: log-uniform in $[1e{-}5, 5e{-}5]$
- `Weight decay`: log-uniform in $[1e{-}6, 1e{-}2]$

Each trial trained the model for 2 epochs on the training set and evaluated it on the validation set using accuracy. Pruning was enabled based on intermediate results to avoid wasting computation on underperforming configurations. Tokenizer special tokens (`url`, `username`) were included in all trials.

2.4 Best Model

After conducting a series of experiments with architecture and training refinements, our final and best-performing model was discovered via Optuna hyperparameter tuning. This model builds upon the full preprocessing pipeline and includes additional training optimizations.

Setup. The model is based on `bert-base-uncased` and fine-tuned for binary classification using the following components:

- **Preprocessing:** lowercasing, mention/URL replacement, emoticon/slang/contraction expansion, repeated character normalization

- **Tokenizer:** extended with special tokens `url` and `username`
- **Training:** 2 epochs, batch size 64, BCEWithLogitsLoss, gradient clipping at 1.0
- **Scheduler:** Linear schedule without warm-up

Best Hyperparameters. These were found using Optuna on the full training and validation sets:

- **Learning Rate:** `4.71e-5`
- **Weight Decay:** `3.28e-4`
- **Max Sequence Length:** 128

Validation Performance. The model achieved its best metrics at epoch 2:

- **Accuracy:** 0.8577
- **Precision:** 0.8610
- **Recall:** 0.8532
- **F1-score:** 0.8571
- **AUC:** 0.9326

Test Set Accuracy. Evaluated on the test set, the model achieved **85.51%** accuracy.

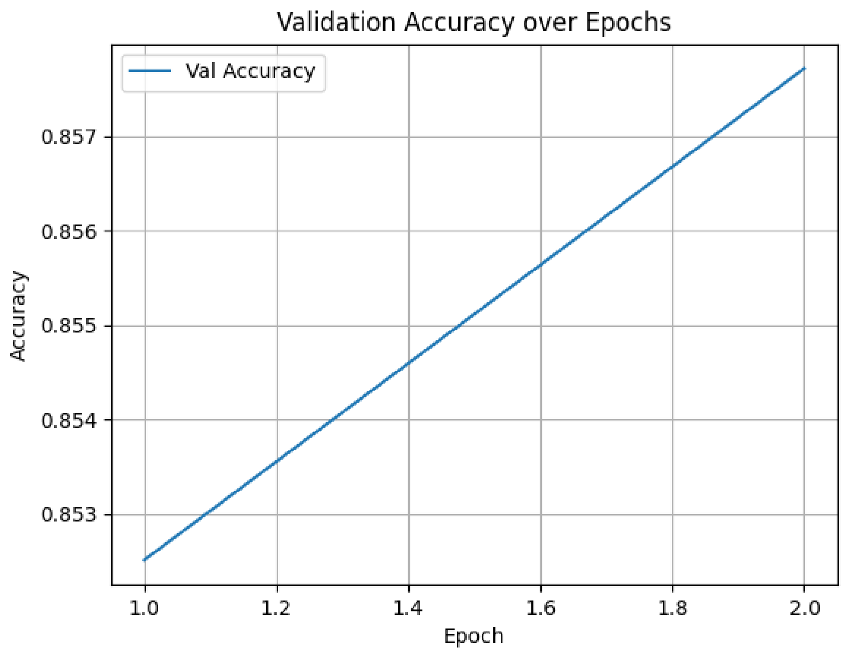

Figure 29: Validation Accuracy over Epochs (Optuna-Tuned BERT)

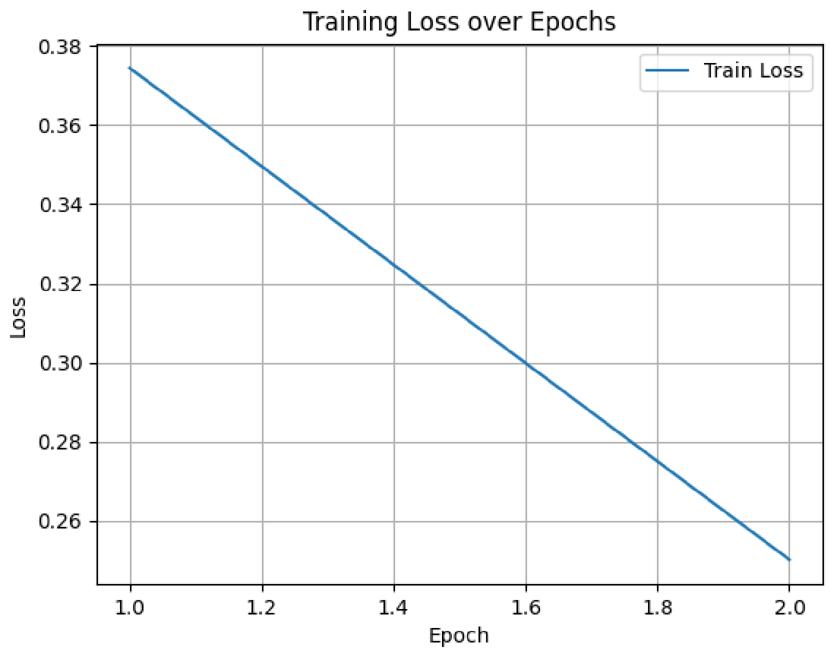

Figure 30: Training Loss over Epochs (Optuna-Tuned BERT)

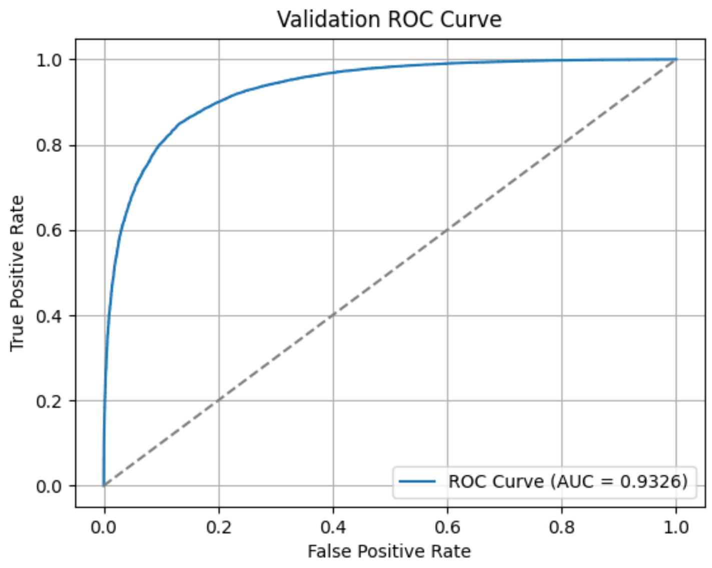

Figure 31: Validation ROC Curve (AUC = 0.9326)

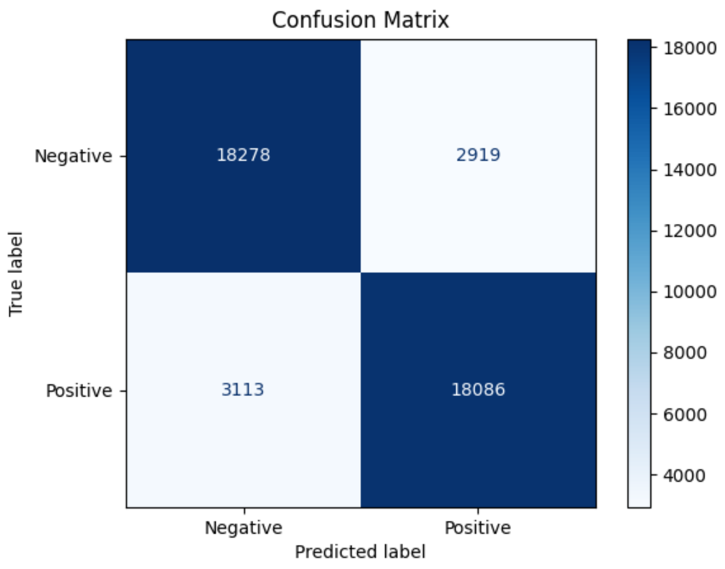

Figure 32: Confusion Matrix on Validation Set

Training Diagnostics. The model exhibits a strong balance between false positives and false negatives, with an AUC nearing 0.933 and a consistently high F1-score. These results demonstrate that Optuna-guided fine-tuning led to a robust and generalizable classifier, outperforming all previous configurations.

3 Conclusion

Through systematic experimentation, we significantly improved BERT-based sentiment classification on tweets. Preprocessing steps and architectural enhancements like increased sequence length, warm-up scheduling, and added special tokens contributed to performance gains. Final tuning with Optuna yielded our best model, achieving a validation F1-score of **0.8571** and test accuracy of **85.51%**. These results highlight the effectiveness of targeted improvements and optimization in fine-tuning transformer models.

4 DistilBERT Model

4.1 Preprocessing

To improve the quality of input for fine-tuning, we applied a lightweight yet effective preprocessing pipeline tailored to informal Twitter content. This setup was designed to preserve semantic cues while reducing noise, and was shared between both BERT and DistilBERT models.

The steps used in the final preprocessing function are:

- **Mojibake Fixing:** We applied the `ftfy` library to automatically fix garbled characters caused by incorrect Unicode encoding (mojibake). This ensures the model receives correctly interpreted symbols and text.

- **Lowercasing:** All tweets were converted to lowercase to reduce vocabulary sparsity, which is especially effective in transformer models that are case-insensitive when using uncased tokenizers.
- **Mention Replacement:** Twitter mentions (e.g., `@user123`) were replaced with the token `username`. This abstracts away user-specific content while retaining the structural cue of a mention.
- **Repeated Letter Reduction:** Words with excessive repeated characters for emphasis (e.g., `soooo`) were normalized to two repetitions (e.g., `soo`). This reduces sparsity without eliminating emotional expression.
- **Emoticon Replacement:** A curated dictionary of emoticons (e.g., `:-)`, `<3`) was used to substitute them with their semantic meaning (e.g., `smile`, `love`), helping the model better interpret informal sentiment signals.
- **Extra Space Removal:** All consecutive spaces were reduced to a single space, and leading/trailing whitespace was removed for normalization.

This pipeline was deliberately chosen for its balance between effectiveness and simplicity. It preserves most of the tweet structure and sentiment cues while eliminating noise. More aggressive transformations such as punctuation stripping or number removal were avoided to maintain compatibility with the tokenizer's subword units.

4.2 Baseline Configuration

As a baseline, we fine-tuned the `distilbert-base-uncased` model using only lowercased tweet text. No other preprocessing was applied. The model was trained for 3 epochs with a linear learning rate schedule and no warm-up steps.

Training Setup:

- **Model:** `DistilBertForSequenceClassification` with a single output neuron ($D_{out} = 1$)
- **Tokenizer:** `DistilBertTokenizerFast` with max sequence length 128
- **Loss Function:** `BCEWithLogitsLoss`
- **Optimizer:** AdamW
- **Learning Rate:** 2×10^{-5}
- **Batch Size:** 64
- **Epochs:** 3
- **Scheduler:** Linear schedule with no warm-up
- **Reproducibility:** Manual seeding and deterministic training setup
- **Hardware:** If available, multi-GPU training was enabled via `DataParallel`

Validation Performance: The model was trained for 3 epochs using only lowercasing as preprocessing. Validation performance per epoch is summarized below:

- **Epoch 1:** Accuracy: 0.8404, Precision: 0.8659, Recall: 0.8057, F1-score: 0.8347, AUC: 0.9223
- **Epoch 2: Accuracy: 0.8473**, Precision: 0.8534, Recall: 0.8388, F1-score: 0.8460, AUC: 0.9262
- **Epoch 3:** Accuracy: 0.8468, Precision: 0.8485, Recall: 0.8442, F1-score: 0.8464, AUC: 0.9255

The best validation accuracy was observed at **Epoch 2** with a score of **84.73%**. Although Epoch 3 slightly improved the F1-score, the marginal drop in accuracy and AUC led us to select Epoch 2 as the optimal checkpoint.

Baseline Diagnostics. We present diagnostic plots for the initial DistilBERT baseline model trained with only lowercasing as preprocessing. This model achieved peak validation performance at epoch 2, reaching accuracy **0.8473**, precision **0.8530**, recall **0.8398**, F1-score **0.8463**, and AUC **0.9264**.

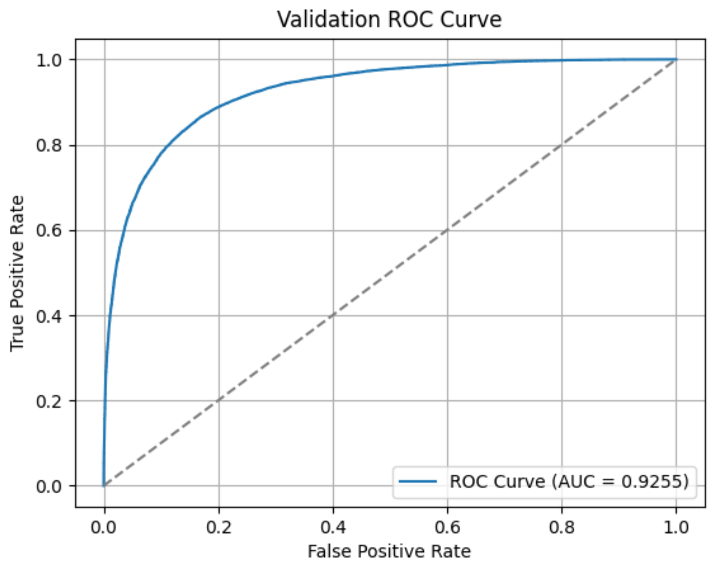

Figure 33: Validation ROC Curve (AUC = 0.9255)

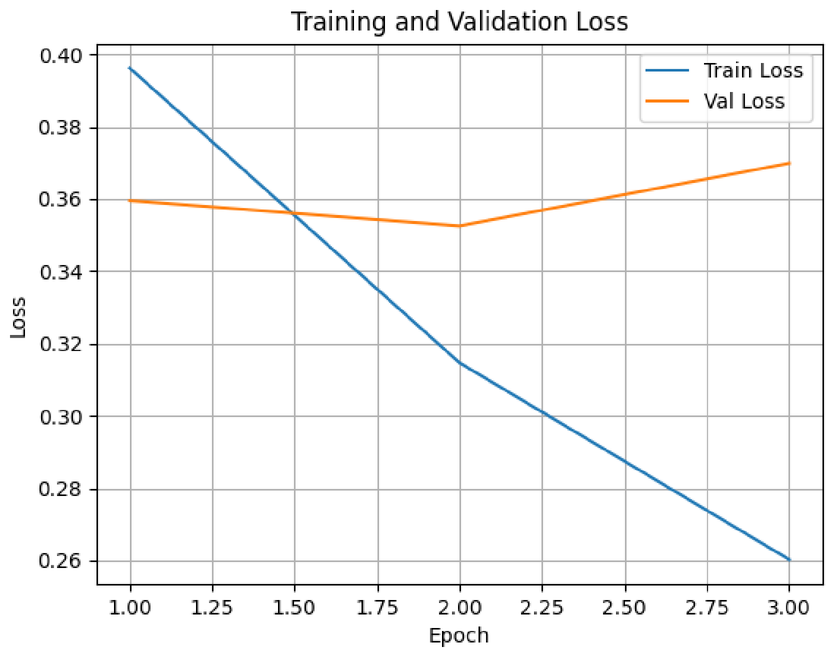

Figure 34: Training and Validation Loss

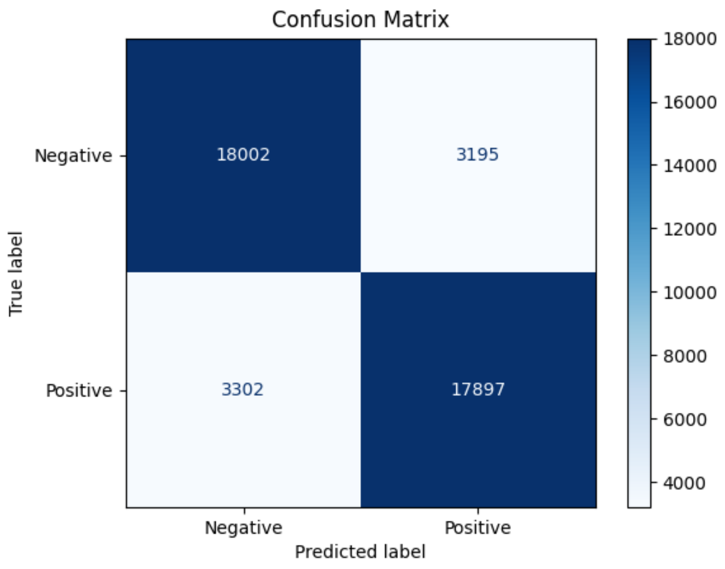

Figure 35: Confusion Matrix on Validation Set

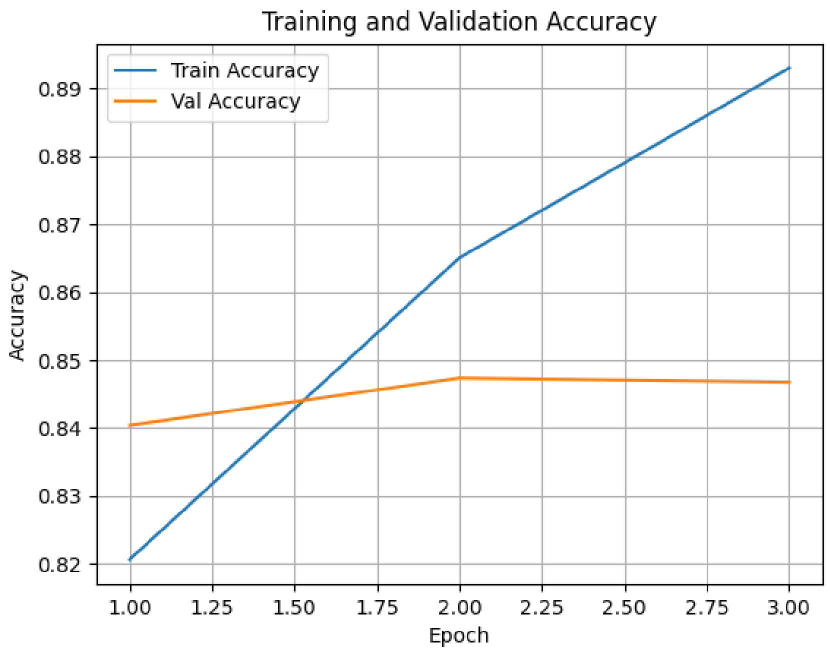

Figure 36: Training and Validation Accuracy

The ROC curve indicates strong class separability. Validation accuracy peaks early at epoch 2 before plateauing, suggesting convergence. The loss curves show steadily decreasing training loss, while validation loss flattens out and begins to rise in the third epoch—hinting at potential overfitting. The confusion matrix confirms generally balanced predictions, though there is a slight bias toward false positives.

4.3 Experiments and Improvements

4.3.1 Preprocessing Experiments

We conducted a series of preprocessing ablation experiments using a greedy forward selection strategy. Starting from the baseline setup with only lowercasing, we evaluated each candidate transformation by measuring its impact on validation accuracy. If a step improved accuracy, it was retained; otherwise, it was discarded. All experiments were performed using 2 epochs of fine-tuning under the same training configuration for consistency.

+ Mojibake Fixing. Correcting encoding artifacts using `ftfy` led to a slight but consistent improvement in robustness. Accuracy improved from 0.8473 to 0.8475, so the step was kept.

+ URLs Replacement. Replacing URLs with the token `url` did not improve performance. Accuracy got at 0.8468, slightly below the previous step, so it was discarded.

+ Mentions Replacement. Replacing Twitter user handles with the token `username` improved accuracy to 0.8480. This indicates that abstracting user-specific content is beneficial for generalization.

+ Repeated Letter Reduction. Reducing repeated characters (e.g., `soooo` → `soo`) before emoticon handling led to a significant performance gain when combined with emoticon replacement. On its own, however, it slightly decreased performance to 0.8479.

+ Slang Replacement. Mapping slang abbreviations (e.g., `gr8` → `great`) showed no improvement and was discarded.

+ Emoticons Replacement. Replacing emoticons like `:)`, `<3` with sentiment-indicative words such as `smile`, `love` provided the final improvement, raising validation accuracy to **0.8481**. This step was kept.

Table 7: Cumulative Effect of Preprocessing Steps on DistilBERT Validation Performance (Greedy Selection)

Cumulative Setup	**Accuracy**	**Precision**	**Recall**	**F1**	**AUC**
Lowercasing only	0.8473	0.8534	0.8388	0.8460	0.9262
+ Mojibake Fixing	0.8475	0.8530	0.8398	0.8463	0.9264
+ Mentions Replacement	0.8480	0.8546	0.8387	0.8466	0.9265
+ Emoticons	0.8481	0.8550	0.8382	0.8466	0.9264
+ Repeated Letters & Emoticons	**0.8486**	**0.8595**	**0.8334**	**0.8463**	**0.9265**

Baseline Model Diagnostics After Preprocessing. We present diagnostics for the DistilBERT model trained with the best-performing preprocessing pipeline: `[lowercase, mojibake, mentions, repeated_letters, emoticons]`. The model achieved its highest validation performance at **epoch 3**, with the following metrics:

- **Accuracy:** 0.8491
- **Precision:** 0.8539
- **Recall:** 0.8424
- **F1-score:** 0.8481
- **AUC:** 0.9258

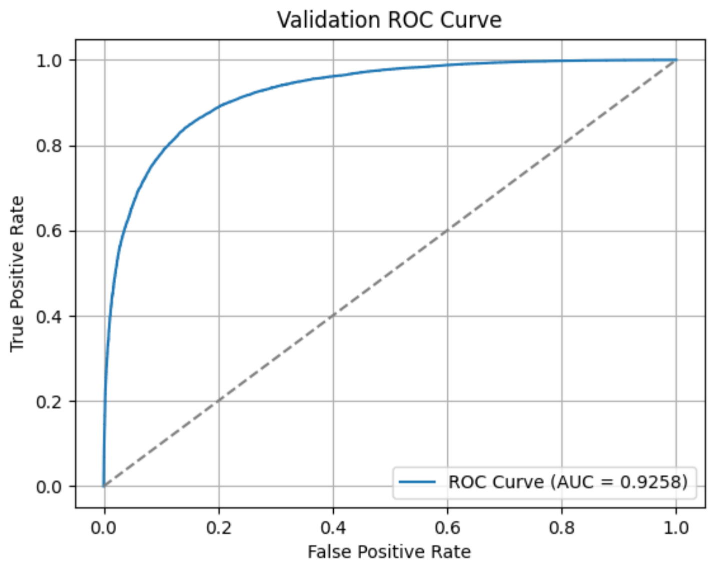

Figure 37: Validation ROC Curve (AUC = 0.9258)

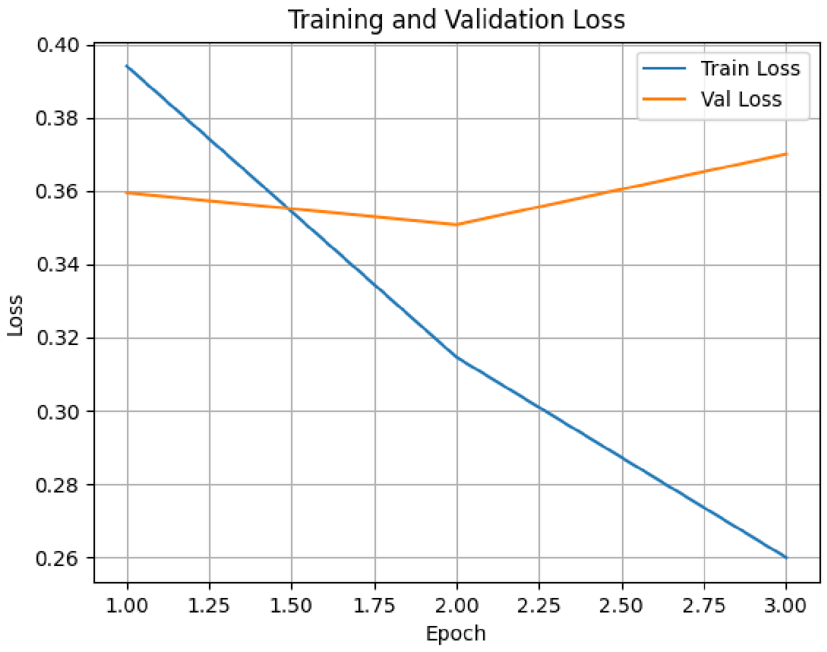

Figure 38: Training and Validation Loss

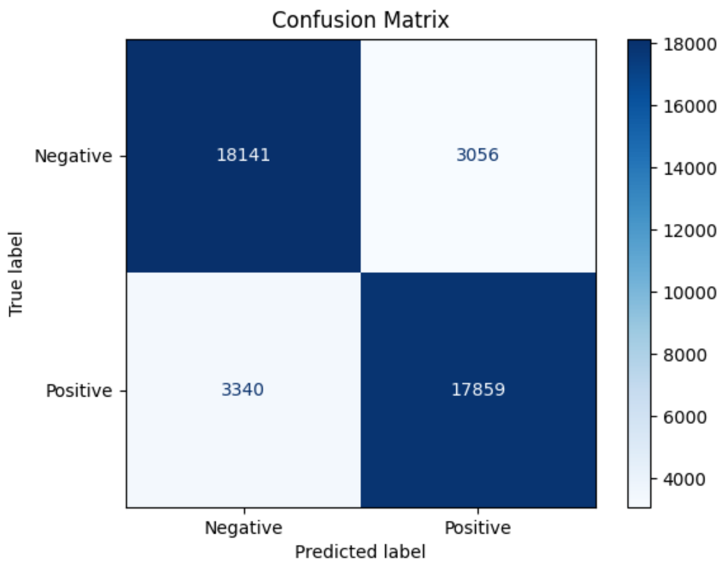

Figure 39: Confusion Matrix on Validation Set

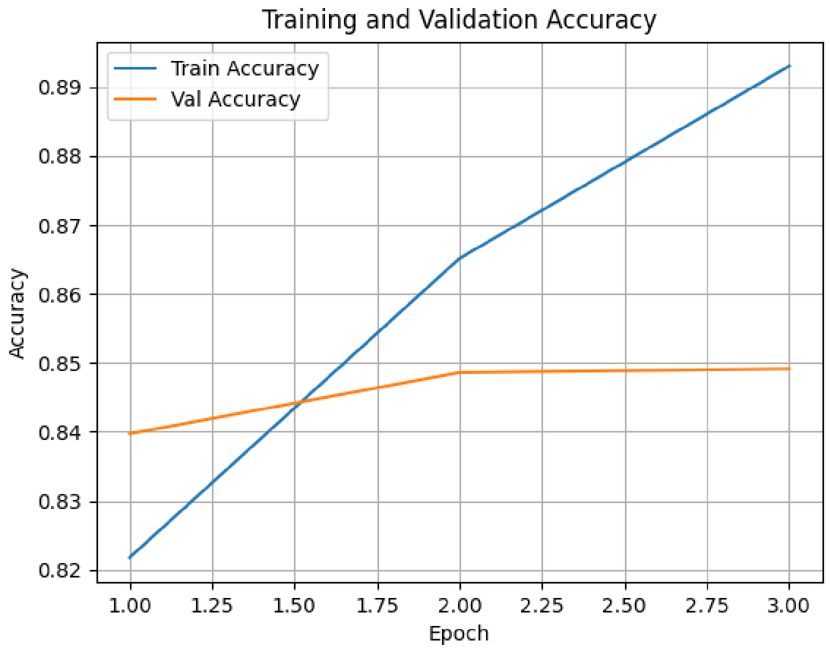

Figure 40: Training and Validation Accuracy

The ROC curve indicates strong discriminative ability. Validation accuracy plateaus near epoch 2–3, while validation loss begins to increase after epoch 2, suggesting mild overfitting. The confusion matrix shows a balanced trade-off between false positives and false negatives.

4.3.2 Increasing Maximum Sequence Length

The default configuration used a maximum sequence length of 128 tokens. To assess whether important context was being truncated, we examined the maximum tweet lengths (after tokenization) in each split:

- **Train Set:** 195 tokens
- **Validation Set:** 197 tokens
- **Test Set:** 155 tokens

Since tweets often exceeded 128 tokens, we increased `max_len` to 150. This allowed the model to retain more of the tweet content during training and evaluation.

Training and Results. The model was trained for 3 epochs using the same hyperparameters as the full-preprocessing baseline. The results are summarized below:

- **Epoch 1:** Accuracy: 0.8404, Precision: 0.8662, Recall: 0.8052, F1: 0.8346, AUC: 0.9230
- **Epoch 2:** Accuracy: 0.8483, Precision: 0.8572, Recall: 0.8357, F1: 0.8463, AUC: 0.9267
- **Epoch 3:** Accuracy: 0.8483, Precision: 0.8519, Recall: 0.8432, F1: 0.8476, AUC: 0.9260

Table 8: max_len = 128 vs. max_len = 150 (Epoch 3 Results)

Metric	**Baseline (128)**	**max_len = 150**
Accuracy	**0.8491**	0.8483
Precision	**0.8539**	0.8519
Recall	**0.8424**	0.8432
F1-score	**0.8481**	0.8476
AUC	**0.9258**	0.9260

Comparison with Full Preprocessing Baseline (Epoch 3). While the performance between the two settings was comparable, the original configuration with `max_len = 128` had a slight edge in accuracy, precision, and F1-score at epoch 3. The extended length model had marginally better recall and AUC, but the overall results suggest that the 128-token cap already captured most of the necessary context for sentiment classification.

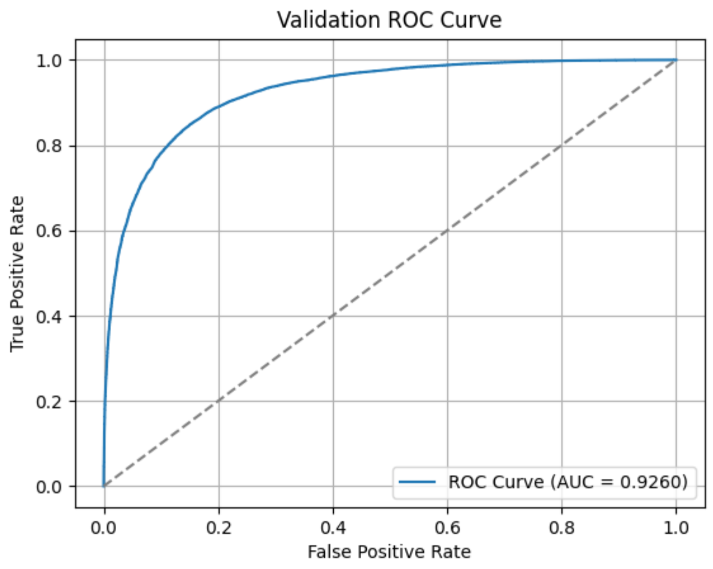

Figure 41: ROC Curve (AUC = 0.9260) with max_len = 150

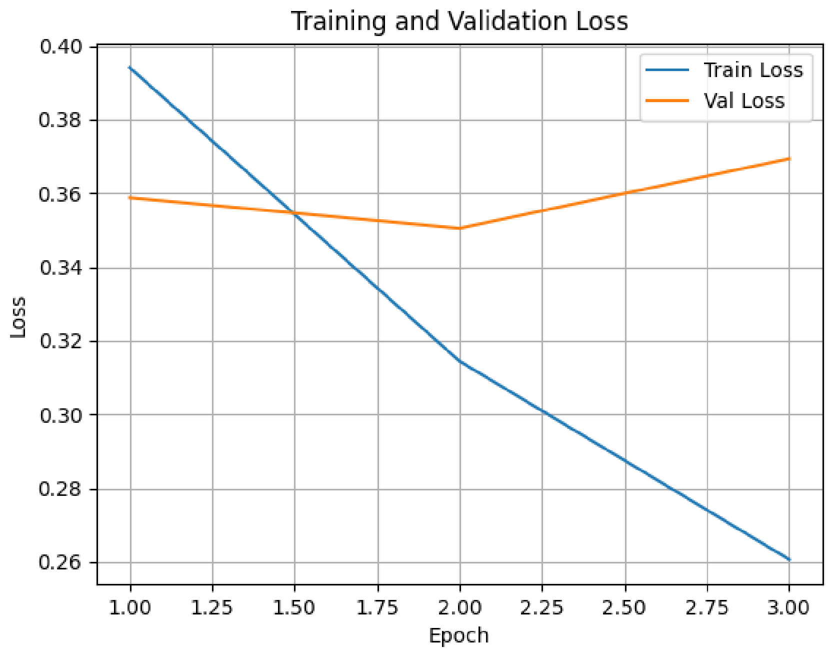

Figure 42: Training and Validation Loss (max_len = 150)

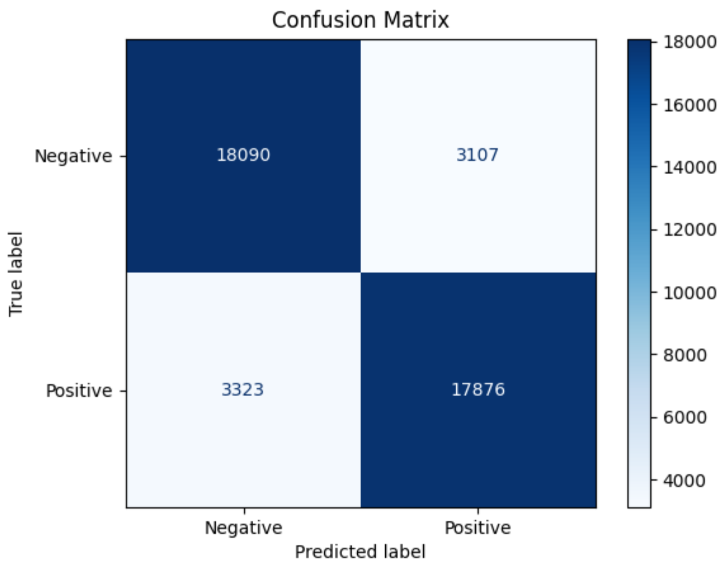

Figure 43: Confusion Matrix at Epoch 3 (max_len = 150)

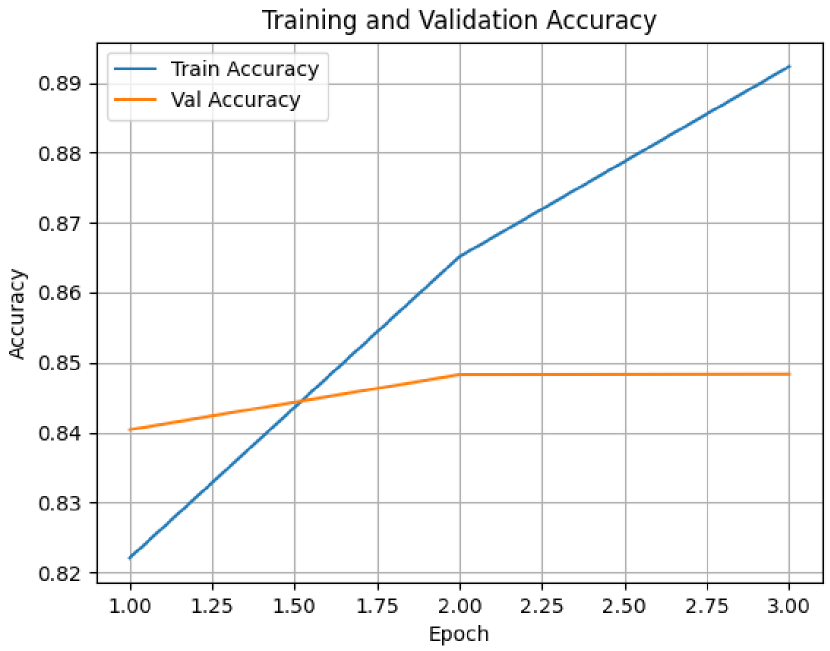

Figure 44: Training and Validation Accuracy (max_len = 150)

Diagnostics for Increased Max Sequence Length.

4.3.3 Decreasing Maximum Sequence Length

To explore whether shorter input sequences could retain sufficient sentiment information, we reduced the maximum sequence length from 128 to 100 tokens.

Token Length Analysis. After tokenization, the average and maximum tweet lengths were:

- **Train Set:** Avg = 43.3 tokens, Max = 195 tokens
- **Validation Set:** Avg = 43.0 tokens, Max = 197 tokens
- **Test Set:** Avg = 42.4 tokens, Max = 155 tokens

Although the average tweet was well below 100 tokens, a significant portion exceeded this threshold. This experiment tests whether truncating longer tweets improves training efficiency without harming accuracy.

Training and Results. The model was trained for 3 epochs with `max_len = 100`, using the same preprocessing pipeline, optimizer, batch size, and scheduler as the full-preprocessing baseline. Results:

- **Epoch 1:** Accuracy: 0.8402, Precision: 0.8649, Recall: 0.8063, F1: 0.8346, AUC: 0.9225
- **Epoch 2:** Accuracy: 0.8485, Precision: 0.8517, Recall: 0.8441, F1: 0.8479, AUC: 0.9263
- **Epoch 3:** Accuracy: **0.8488**, Precision: 0.8519, Recall: **0.8443**, F1: **0.8481**, AUC: 0.9256

Table 9: max_len = 128 vs. max_len = 100 (Epoch 3 Results)

Metric	**Baseline (128)**	**max_len = 100**
Accuracy	**0.8491**	0.8488
Precision	**0.8539**	0.8519
Recall	0.8424	**0.8443**
F1-score	**0.8481**	**0.8481**
AUC	**0.9258**	0.9256

Comparison with Full Preprocessing Baseline (Epoch 3). Despite truncating tweets above 100 tokens, the model maintained nearly identical performance to the baseline. This suggests that sentiment-relevant information is often located in the first part of tweets. Using `max_len = 100` can therefore reduce memory consumption and training time without sacrificing predictive accuracy—making it a practical trade-off for faster deployment scenarios.

4.3.4 Batch Size

We evaluated the impact of different batch sizes (**32**, **64**, **128**) on validation performance using the same training configuration and full preprocessing pipeline. All models were trained for 3 epochs and evaluated using the best epoch (Epoch 3 in all cases).

Table 10: Comparison of Validation Metrics at Epoch 3 for Different Batch Sizes

Metric	**Batch Size 32**	**Batch Size 64**	**Batch Size 128**
Train Accuracy	0.9057	0.8930	0.8781
Val Accuracy	0.8472	**0.8491**	0.8464
Precision	0.8487	**0.8539**	0.8499
Recall	0.8450	**0.8424**	0.8414
F1-score	0.8468	**0.8481**	0.8456
AUC	0.9253	**0.9258**	0.9251

Results.

- Batch size **64** yielded the highest F1-score (0.8481) and AUC (0.9258), indicating the best balance between generalization and stability.
- Batch size **32** slightly outperformed in recall but showed more overfitting (train accuracy = 0.9057).
- Batch size **128** resulted in faster training but slightly lower performance, suggesting underfitting at that scale.

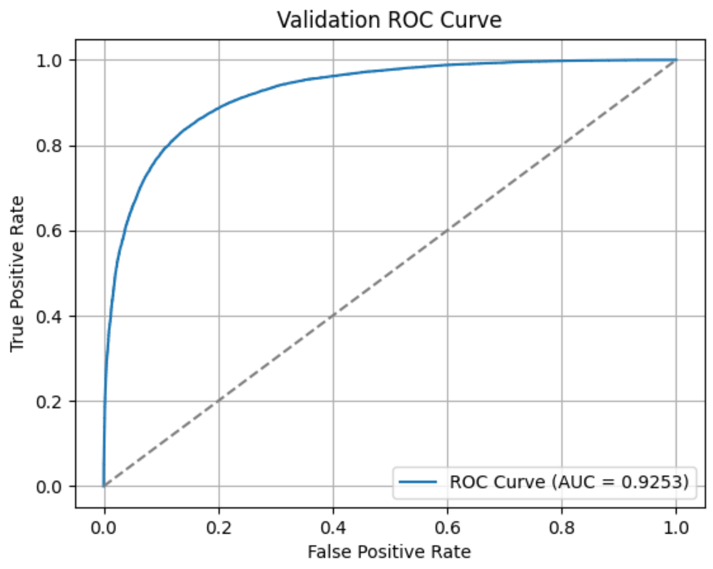

Figure 45: ROC — Batch 32 (AUC=0.9253)

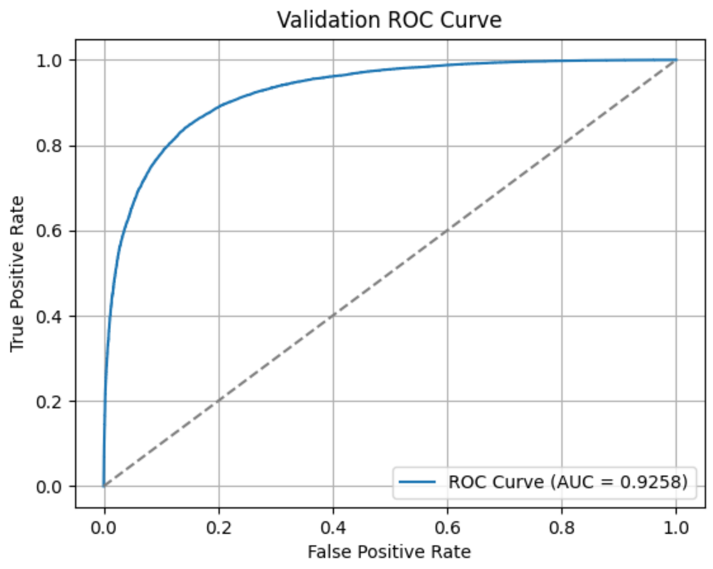

Figure 46: ROC — Batch 64 (AUC=0.9258)

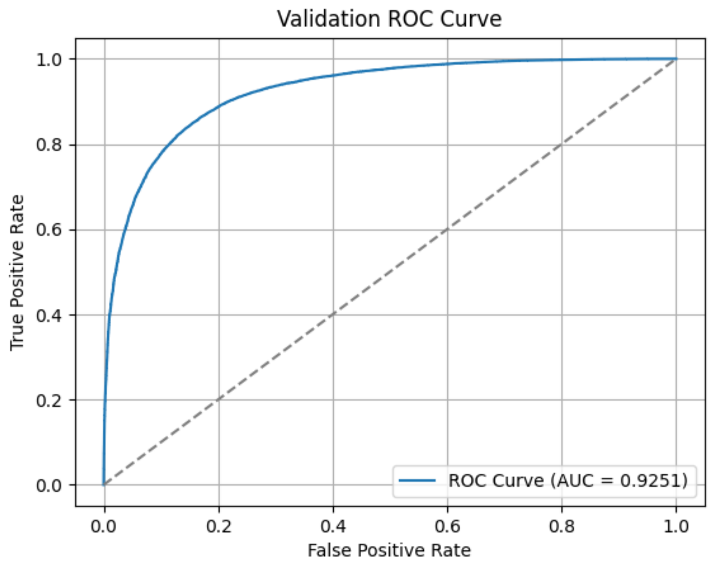

Figure 47: ROC — Batch 128 (AUC=0.9251)

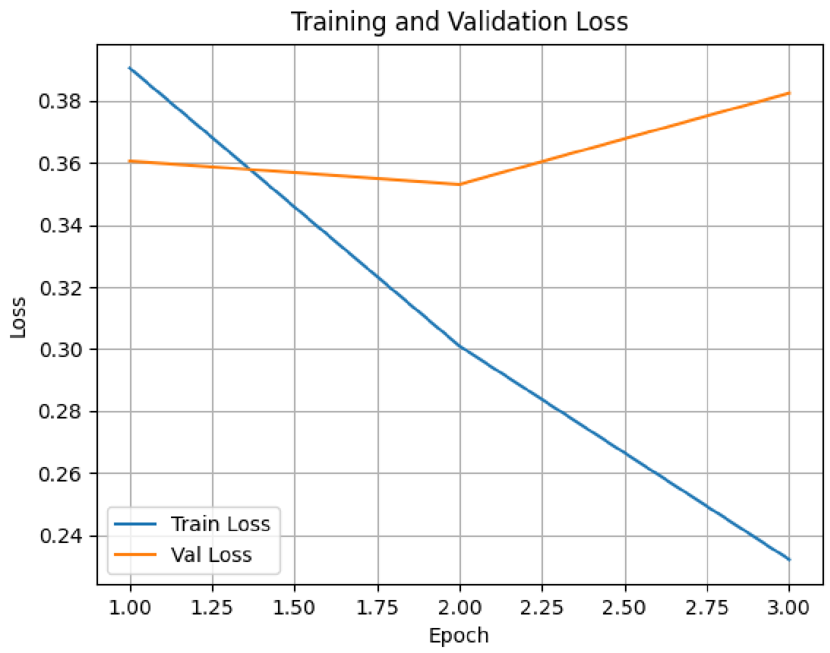

Figure 48: Loss — Batch 32

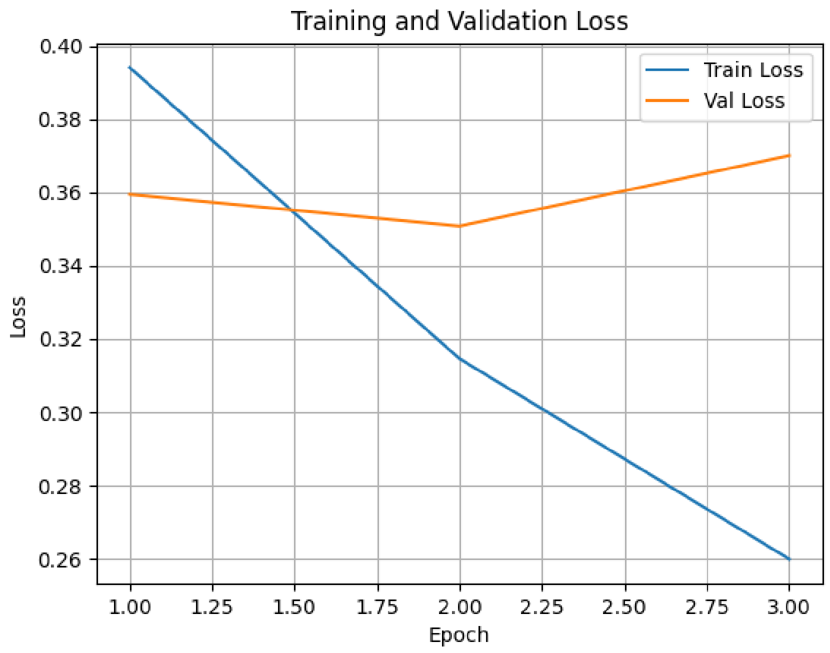

Figure 49: Loss — Batch 64

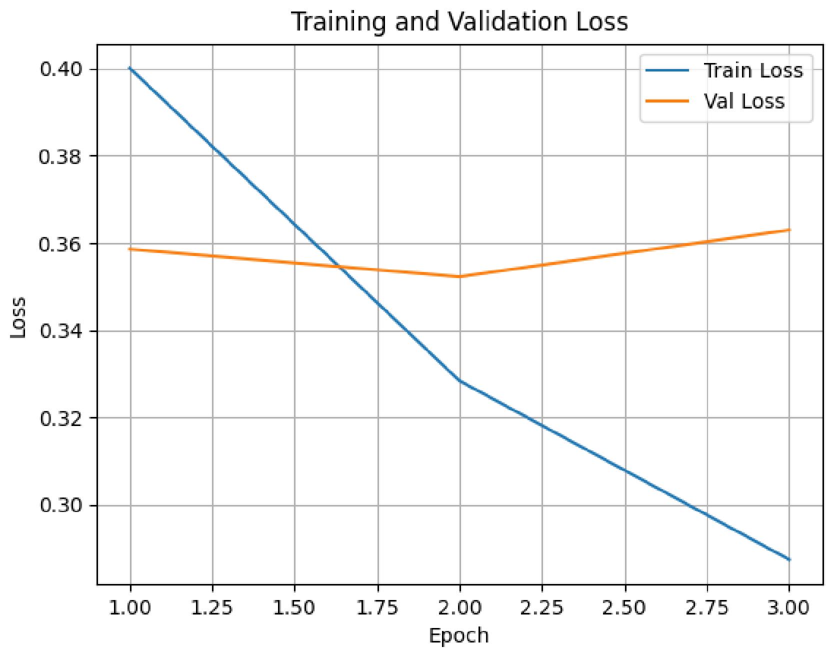

Figure 50: Loss — Batch 128

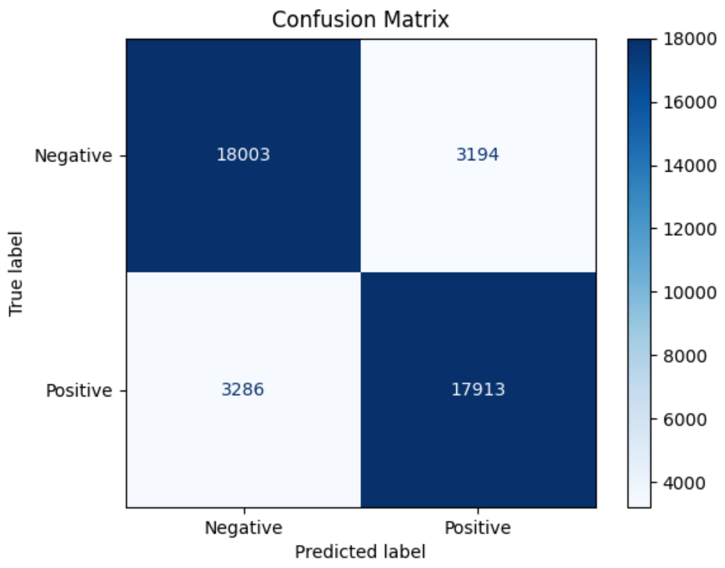

Figure 51: Confusion Matrix — Batch 32

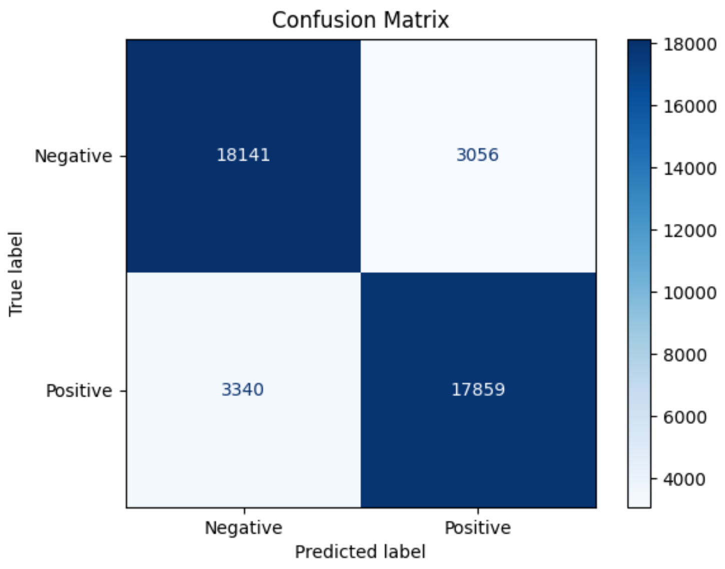

Figure 52: Confusion Matrix — Batch 64

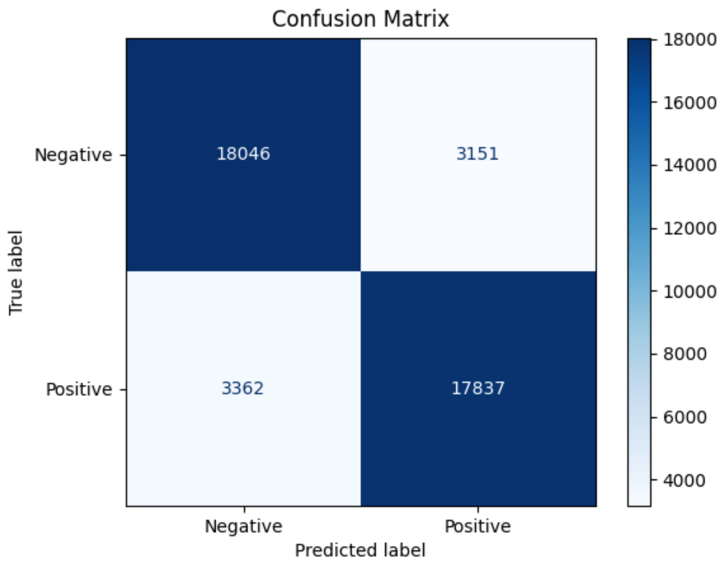

Figure 53: Confusion Matrix — Batch 128

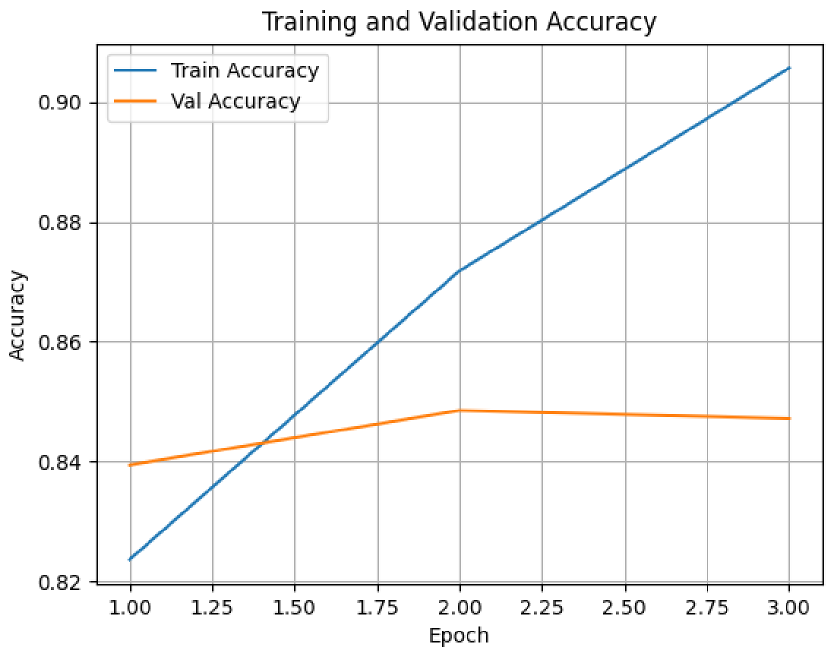

Figure 54: Accuracy — Batch 32

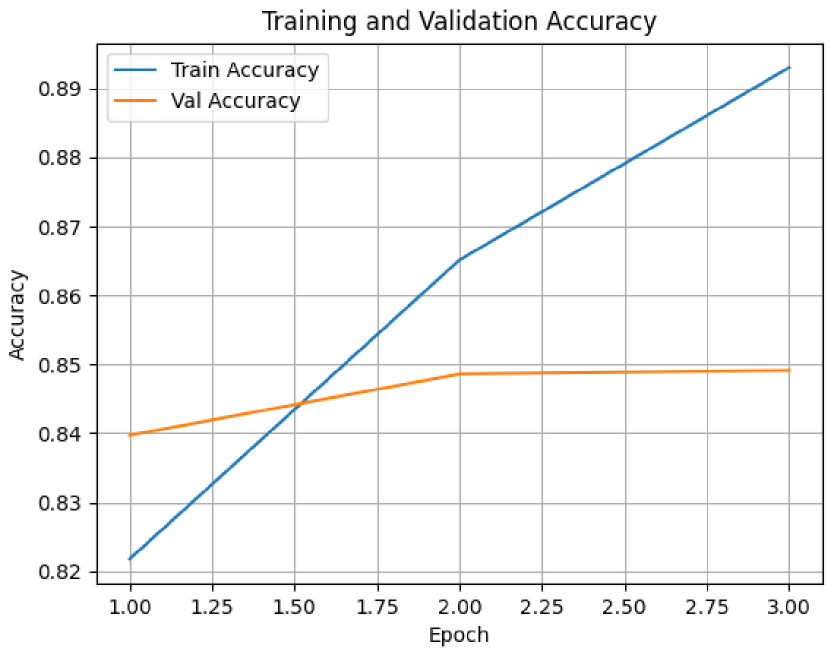

Figure 55: Accuracy — Batch 64

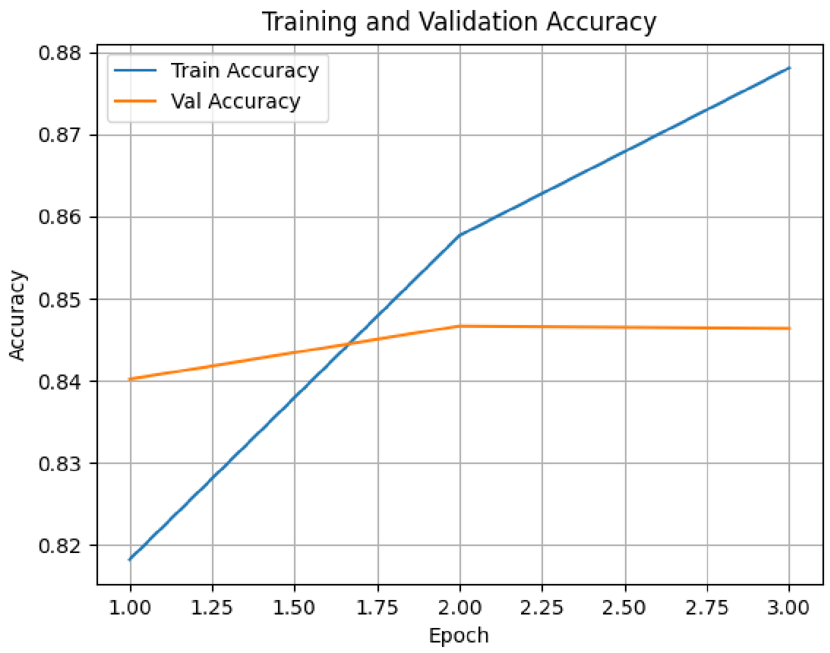

Figure 56: Accuracy — Batch 128

Diagnostics for Different Batch Sizes.

4.3.5 Warm-up Scheduler

To improve training stability and early-stage convergence, we introduced a warm-up phase in the learning rate schedule. Specifically, the learning rate was linearly increased during the initial 10% of training steps, followed by a linear decay. This strategy is commonly used in transformer fine-tuning to avoid large, unstable gradients at the start.

Training and Results. The model was trained for 3 epochs using the same optimizer, learning rate, and preprocessing pipeline as the full-preprocessing baseline. The following validation metrics were recorded:

- **Epoch 1:** Accuracy: 0.8365, Precision: 0.8671, Recall: 0.7949, F1: 0.8294, AUC: 0.9209
- **Epoch 2:** Accuracy: 0.8476, Precision: 0.8555, Recall: 0.8366, F1: 0.8459, AUC: 0.9263
- **Epoch 3:** Accuracy: 0.8487, Precision: 0.8521, Recall: 0.8439, F1: 0.8480, AUC: 0.9256

Table 11: Warm-up Scheduler vs. Full Preprocessing Baseline (Epoch 3)

Metric	Baseline	Warm-up Scheduler
Accuracy	0.8571	0.8487
Precision	0.8621	0.8521
Recall	0.8503	**0.8439**
F1-score	0.8562	0.8480
AUC	0.9331	0.9256

Comparison with Full Preprocessing Baseline. While the warm-up scheduler enhances recall compared to the baseline, it comes at a cost of slightly reduced precision and AUC. The overall F1-score remains competitive, but the baseline still outperforms marginally in most metrics.

Diagnostics with Warm-up Scheduler. We visualize the model's learning behavior through diagnostic plots including accuracy curves, training loss, ROC curve, and confusion matrix.

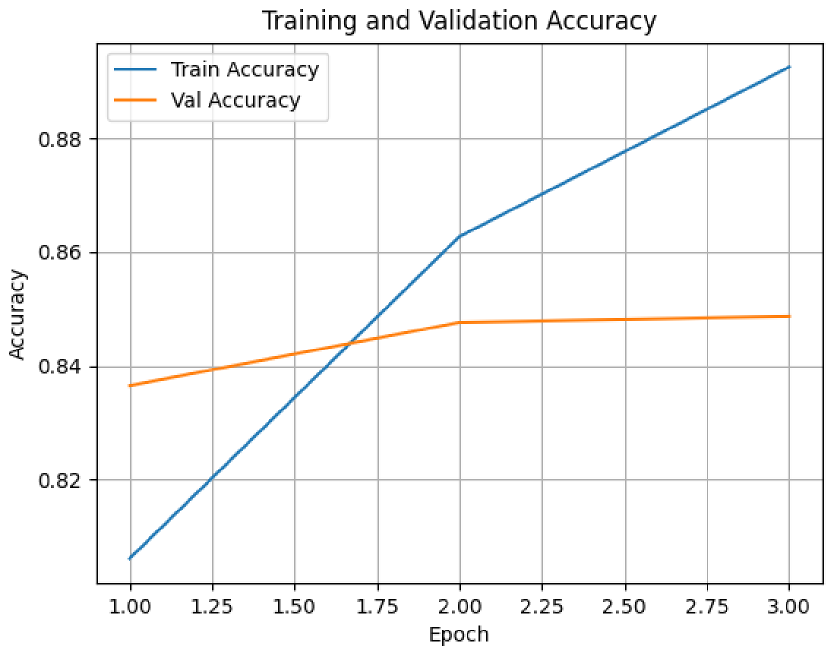

Figure 57: Validation Accuracy over Epochs (Warm-up Scheduler)

The validation accuracy rises consistently through epoch 2, then plateaus near 0.849. This indicates stable learning but suggests limited generalization gains from additional training.

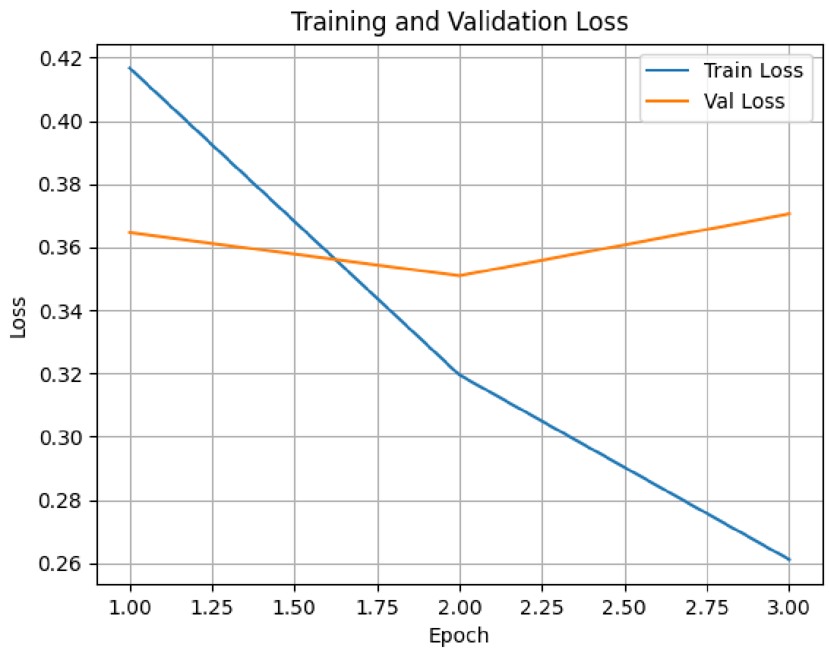

Figure 58: Training and Validation Loss (Warm-up Scheduler)

Training loss decreases steadily, while validation loss begins to rise after epoch 2. This may indicate slight overfitting beyond the second epoch.

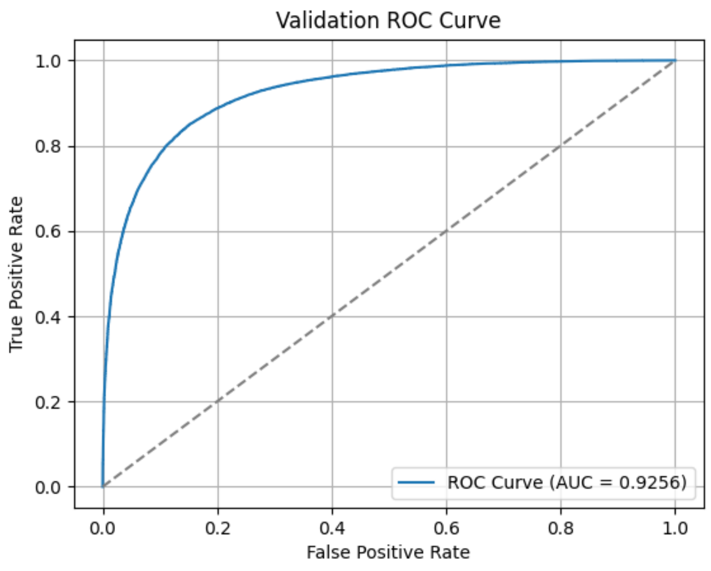

Figure 59: Validation ROC Curve (Warm-up Scheduler)

With an AUC of 0.9256, the classifier maintains strong separability between classes, although marginally below the full-preprocessing baseline.

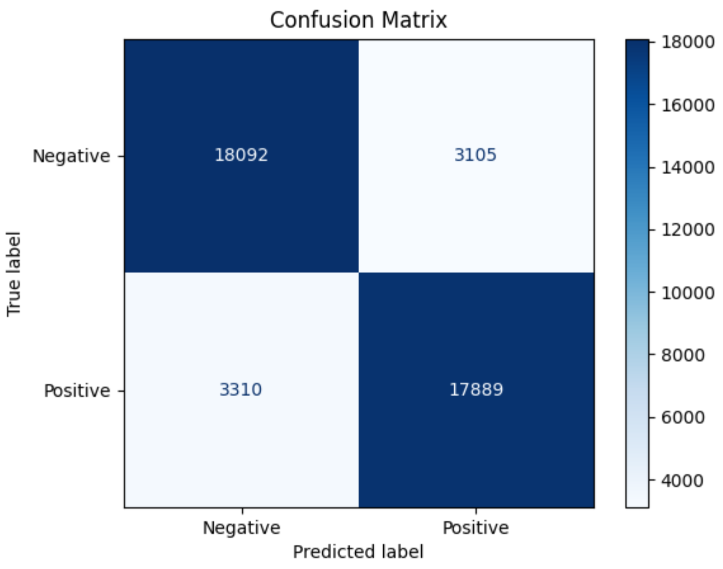

Figure 60: Confusion Matrix on Validation Set (Warm-up Scheduler)

Compared to the baseline, we observe slightly more false positives (3105 vs. 2883) and fewer false negatives (3310 vs. 3174), reinforcing the shift toward higher recall and lower precision with warm-up scheduling.

4.3.6 Adding Special Tokens for Mentions

To investigate whether explicitly marking user mentions helps sentiment classification, we extended the tokenizer with a special token `username` and replaced mentions accordingly in the dataset.

Training and Results. The model was trained for 3 epochs using the same setup as the full preprocessing baseline. Validation performance peaked at Epoch 3:

- **Epoch 1:** Accuracy: 0.8389, F1: 0.8318, AUC: 0.9224
- **Epoch 2:** Accuracy: 0.8473, F1: 0.8452, AUC: 0.9263
- **Epoch 3: Accuracy: 0.8477, F1: 0.8469, AUC: 0.9256**

Table 12: Effect of Mention Special Token on Validation Performance

Metric	**Baseline (No Token)**	**With Mention Token**
Accuracy	0.8491	0.8477
Precision	0.8539	0.8519
Recall	0.8424	0.8419
F1-score	0.8481	0.8469
AUC	0.9258	0.9256

Comparison with Full Preprocessing Baseline. While both models are closely matched, the baseline slightly outperforms the version using a mention token. The added semantic marker does not significantly enhance performance, possibly due to effective preprocessing already capturing the role of mentions.

Diagnostics with Special Tokens. We include visual diagnostics from the best-performing epoch (Epoch 3) when using a special token for mentions. These help verify consistency across performance metrics.

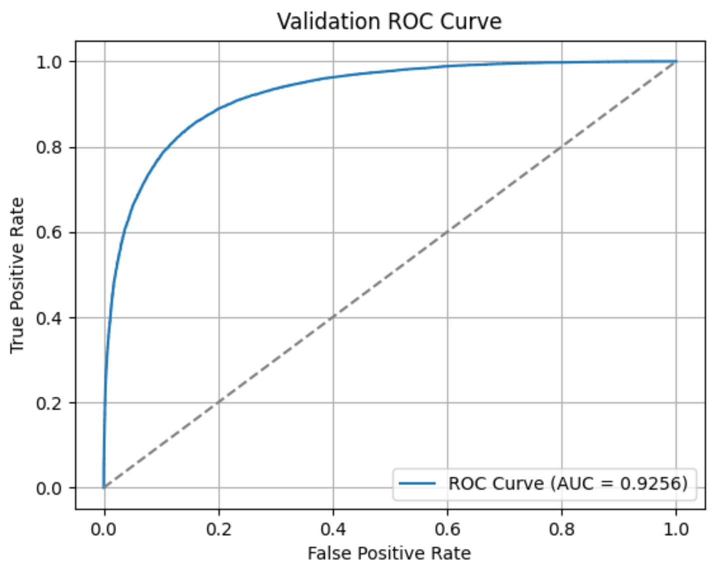

Figure 61: Validation ROC Curve (AUC = 0.9256)

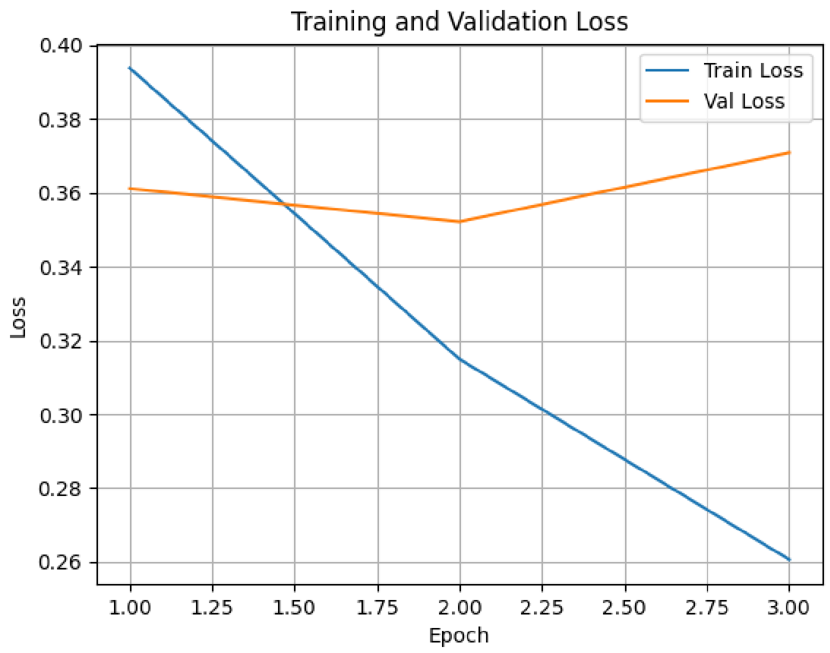

Figure 62: Training and Validation Loss across Epochs

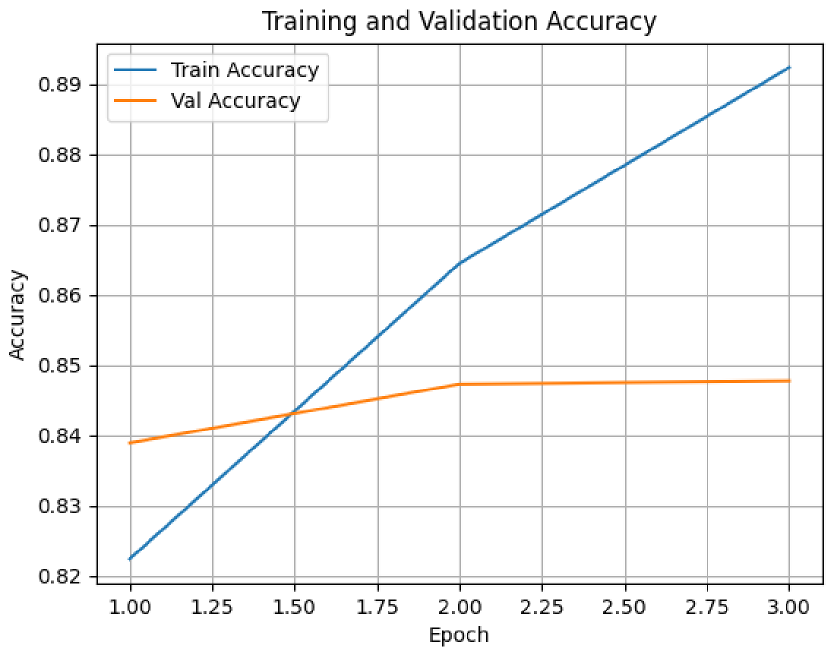

Figure 63: Training and Validation Accuracy across Epochs

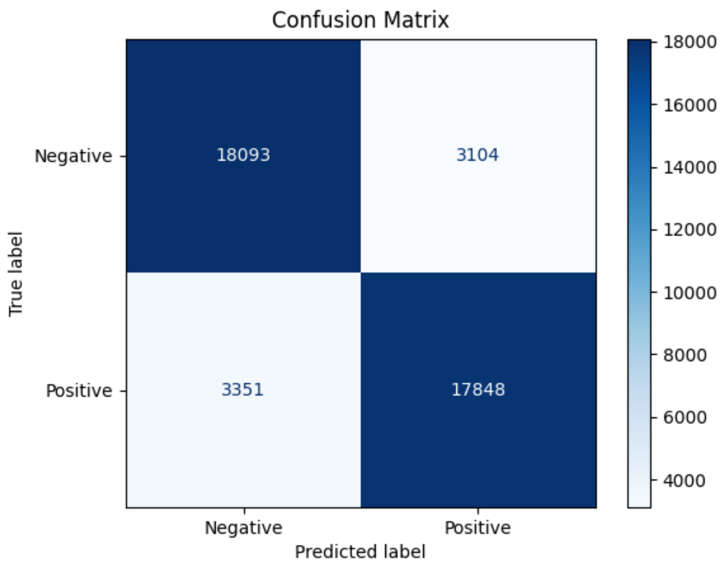

Figure 64: Confusion Matrix at Epoch 3

4.3.7 Hyperparameter Tuning with Optuna

To improve validation accuracy, we performed hyperparameter tuning on DistilBERT using the Optuna optimization framework. The objective was to maximize validation accuracy over three training epochs. We tuned the learning rate and weight decay using the AdamW optimizer, while keeping all other configurations constant.

Search Space.

- `learning rate` $\in [2 \times 10^{-5}, 4 \times 10^{-5}]$ (log-uniform)
- `weight decay` $\in [1 \times 10^{-6}, 1 \times 10^{-5}]$ (log-uniform)

Training Details. The model was trained with:

- `max_len = 128`, `batch_size = 64`
- Binary cross-entropy loss (BCEWithLogitsLoss)
- 3 training epochs
- Linear learning rate scheduler without warm-up
- Full preprocessing pipeline

Table 13: Top 5 Trials from First Optuna Study (DistilBERT, Accuracy Optimized)

Trial	Accuracy	Learning Rate	Weight Decay
1	**0.8493**	3.47×10^{-5}	2.00×10^{-6}
5	0.8491	2.11×10^{-5}	6.19×10^{-6}
10	0.8490	3.64×10^{-5}	3.53×10^{-6}
3	0.8490	2.40×10^{-5}	3.16×10^{-6}
0	0.8487	2.97×10^{-5}	1.08×10^{-6}

Top Trials.

Observations. All top-performing trials achieved validation accuracy between 0.8487 and 0.8493. The best result (Trial 1) used a moderately high learning rate (3.47×10^{-5}) and low weight decay (2.00×10^{-6}). This configuration was selected for subsequent experiments.

4.3.8 Refined Hyperparameter Tuning with Optuna

Following the initial hyperparameter search, a refined Optuna study was conducted to further improve validation accuracy. The search space was narrowed around the most promising regions found previously.

Refined Search Space.

- `learning rate` $\in [2.5 \times 10^{-5}, 4.5 \times 10^{-5}]$
- `weight decay` $\in [1 \times 10^{-6}, 1 \times 10^{-5}]$

Training Details.

- `max_len = 128`, `batch_size = 64`, `epochs = 3`
- Optimizer: AdamW with linear learning rate scheduler (no warm-up)
- Loss: BCEWithLogitsLoss
- Evaluation metric: Validation accuracy

Optuna Study.

- Storage: `sqlite:///study_refined.db`
- Study name: `distilbert_refined`
- Trials: 10
- Objective: Maximize validation accuracy

Table 14: Best Hyperparameters from Refined Optuna Study

Metric / Parameter	**Value**
Validation Accuracy	0.8494
Learning Rate	3.401×10^{-5}
Weight Decay	1.713×10^{-6}

Best Trial Results. The best trial achieved a validation accuracy of **0.8494**, slightly surpassing the best trial of the initial Optuna run (**0.8493**). This validates the effectiveness of focusing the search range, especially for high-performing models like DistilBERT. The improvement, though modest, confirms the value of refining hyperparameter boundaries after an exploratory study.

4.3.9 Final Model Performance After Tuning

After completing hyperparameter tuning with Optuna, the best DistilBERT model was selected from the refined study. It used the following parameters:

- **Learning Rate:** 3.40×10^{-5}
- **Weight Decay:** 1.71×10^{-6}
- **Epochs:** 3
- **Batch Size:** 64
- **Max Sequence Length:** 128
- **Loss:** BCEWithLogitsLoss
- **Optimizer:** AdamW with linear scheduler (no warm-up)

Epoch-wise Training Results. The final model was trained for 3 epochs using the best hyperparameters found via the refined Optuna study. Key metrics per epoch are presented below:

- **Epoch 1:**
  - Accuracy: 0.8414, Precision: 0.8688, Recall: 0.8041, F1: 0.8352, AUC: 0.9235
- **Epoch 2:**
  - Accuracy: **0.8494**, Precision: **0.8602**, Recall: 0.8345, F1: 0.8472, AUC: **0.9274**
- **Epoch 3:**
  - Accuracy: 0.8483, Precision: 0.8515, Recall: **0.8437**, F1: **0.8476**, AUC: 0.9248

Although validation accuracy peaked at Epoch 2, the highest F1-score was achieved in Epoch 3 (0.8476), indicating slightly improved balance between precision and recall. The model maintained consistent performance across all epochs, demonstrating stable convergence.

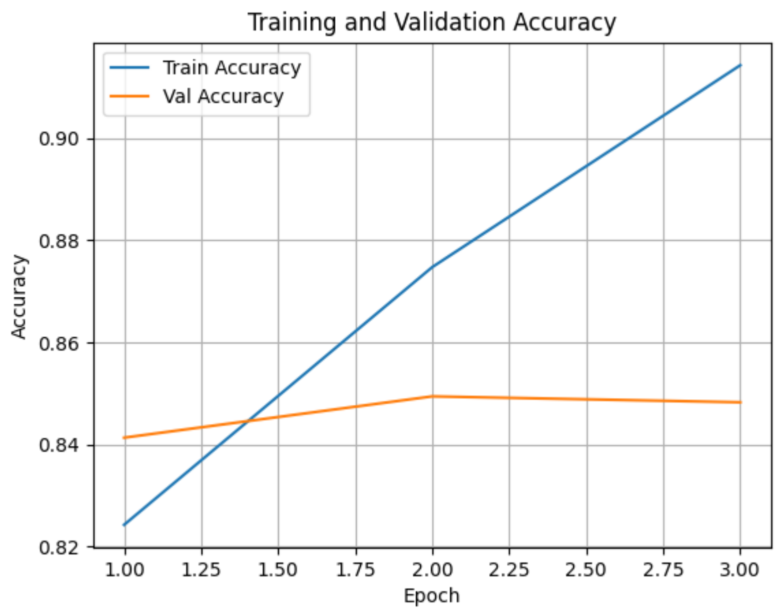

Figure 65: Validation Accuracy over Epochs (Final Model)

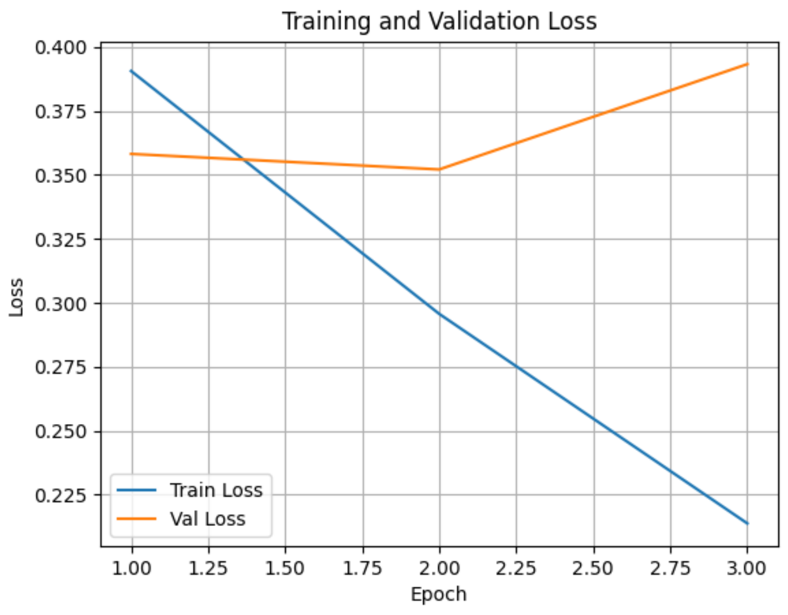

Figure 66: Training and Validation Loss (Final Model)

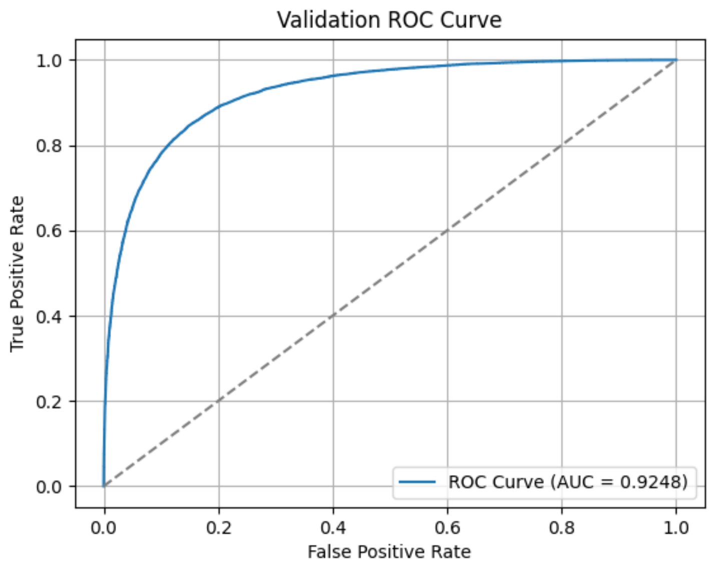

Figure 67: Validation ROC Curve (Final Model)

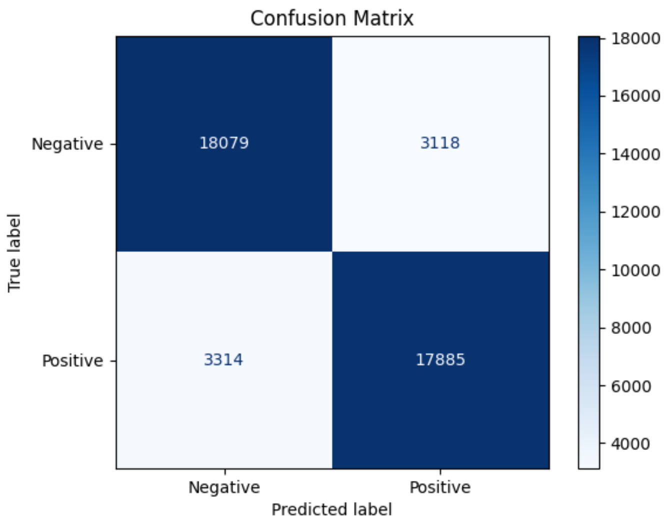

Figure 68: Confusion Matrix on Validation Set (Final Model)

Diagnostic Plots:

4.3.10 Final Model with Gradient Clipping

To enhance training stability, we applied gradient clipping with a maximum norm of 1.0 using `torch.nn.utils.clip_grad_norm_`. This prevents exploding gradients during backpropagation, especially important when training deep transformer models.

Training and Results. The model was trained for 3 epochs with the best hyperparameters found during Optuna tuning:

- `learning rate` = 3.401×10^{-5}, `weight decay` = 1.713×10^{-6}
- Optimizer: AdamW
- Scheduler: Linear decay (no warm-up)
- Loss: BCEWithLogitsLoss
- Preprocessing: Full pipeline
- `batch_size = 64`, `max_len = 128`

- **Epoch 1:** Accuracy: 0.8397, Precision: 0.8633, Recall: 0.8073, F1: 0.8344, AUC: 0.9225
- **Epoch 2:** Accuracy: 0.8497, Precision: 0.8540, Recall: 0.8436, F1: 0.8488, AUC: 0.9270
- **Epoch 3:** Accuracy: 0.8477, Precision: 0.8488, Recall: 0.8461, F1: 0.8475, AUC: 0.9245

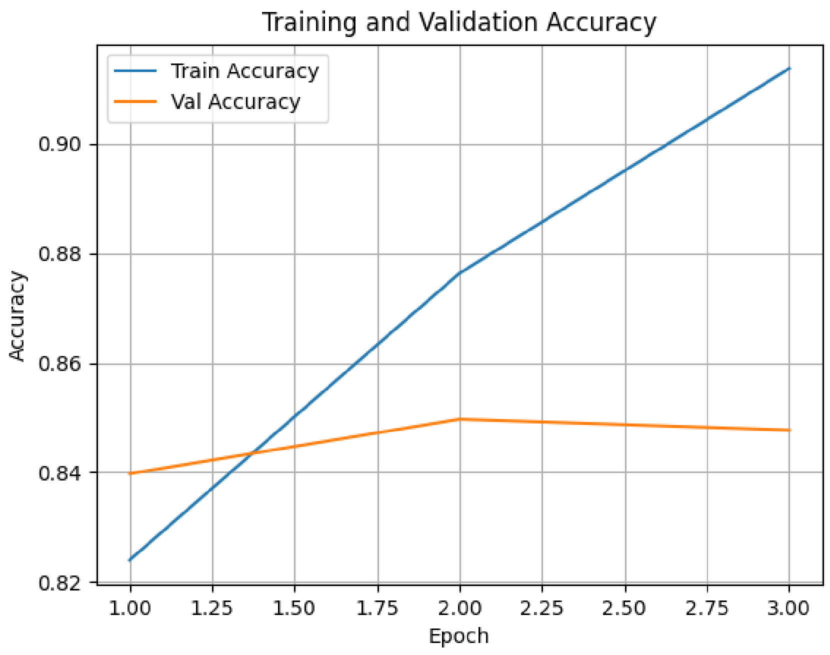

Figure 69: Validation Accuracy over Epochs (Final Model with Gradient Clipping)

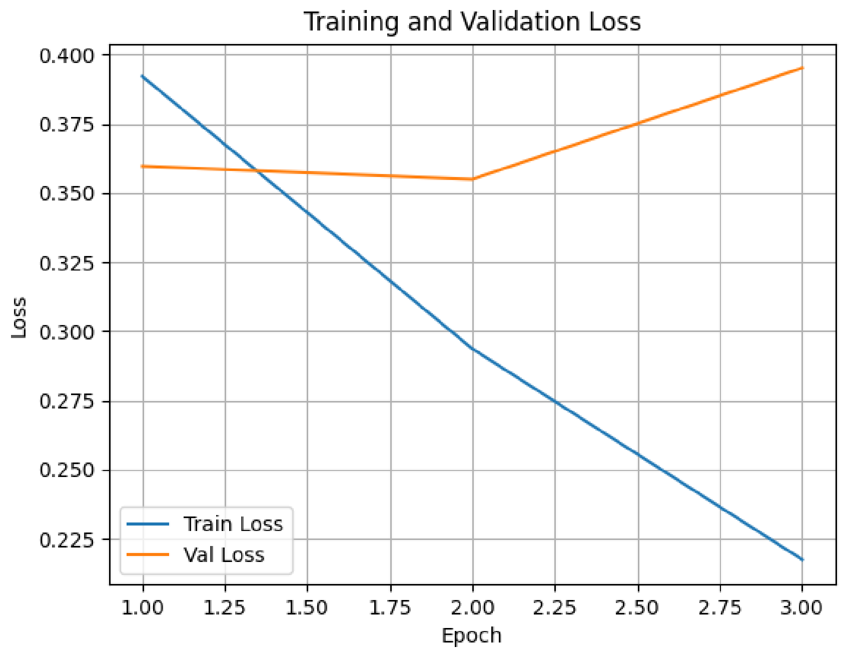

Figure 70: Training and Validation Loss (Final Model with Gradient Clipping)

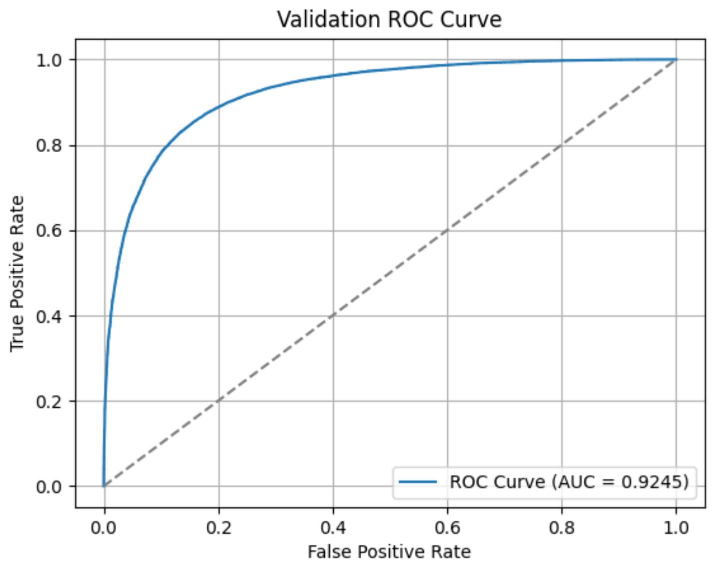

Figure 71: Validation ROC Curve (Final Model with Gradient Clipping)

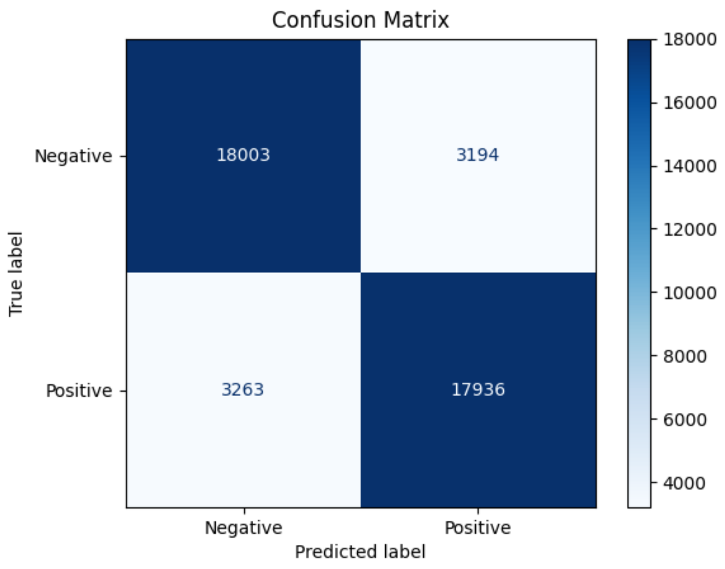

Figure 72: Confusion Matrix on Validation Set (Final Model with Gradient Clipping)

Diagnostics with Final Model.

Summary. The addition of gradient clipping yielded a slight stability boost, leading to a peak F1-score of **0.8488** at epoch 2. The model maintains strong precision-recall balance and consistent performance across all validation metrics.

Test Set Accuracy: 0.84900%

These results confirm that with proper tuning and preprocessing, the more lightweight DistilBERT model can achieve nearly comparable performance to full BERT while offering improved training efficiency.

4.4 Conclusion

DistilBERT proved to be an effective model for tweet sentiment classification. Starting from a lowercased baseline, we improved performance through selective preprocessing and hyperparameter tuning. The final model achieved a validation F1-score of **0.8488** and a test accuracy of **84.90%**, demonstrating that DistilBERT can deliver strong results with lower computational cost.

5 Comparison Between BERT and DistilBERT

We compare the performance of **BERT** and **DistilBERT** on the tweet sentiment classification task using the same preprocessing pipeline, training setup, and evaluation criteria. The results on the validation and test sets are summarized in Tables 15 and 16.

Table 15: Validation set performance comparison.

Model	Accuracy	Precision	Recall	F1 Score	AUC
BERT	**0.8577**	**0.8610**	**0.8532**	**0.8571**	**0.9326**
DistilBERT	0.8497	0.8540	0.8436	0.8488	0.9270

Table 16: Test set accuracy.

Model	Test Accuracy
BERT	**0.8551**
DistilBERT	0.8490

6 Conclusion.

BERT consistently outperforms **DistilBERT** across all evaluation metrics on the validation set and also achieves slightly higher accuracy on the test set. However, this improved performance comes at a cost: BERT is notably slower to train and infer due to its larger architecture. In contrast, DistilBERT offers a competitive alternative with significantly reduced computational overhead, making it more suitable for resource-constrained environments.

Reproducibility. The full implementations of both the BERT and DistilBERT experiments—including preprocessing, training configuration, and evaluation—are available in the following Kaggle notebooks:

- BERT: `https://www.kaggle.com/code/missarte/notebook7310d888fd`
- DistilBERT: `https://www.kaggle.com/code/missarte/notebook8cc2fe995a`