ECE 684 Final Project Draft

Efficient Sentiment Classification for Twitter

Bohao (Lexi) Yang (by95)
Bingkun Wang (bw276)
Yuzhao (Clair) Tan (yt221)

November 2025

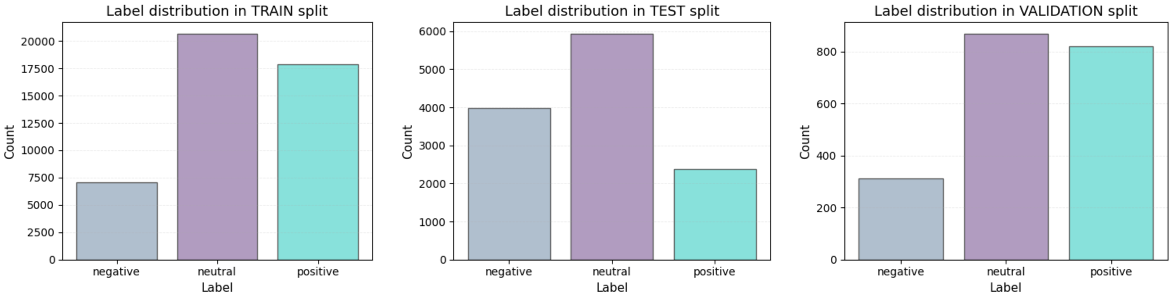
1 Introduction

Social media platforms generate massive amounts of unstructured text data every day. Analyzing the sentiment of this data is challenging due to the informal nature of internet language, which often includes slang, emojis, and irregular grammar. This project focuses on three-class sentiment analysis using the TweetEval "sentiment" dataset, a well-established benchmark for Twitter-based classification. The dataset labels tweets into three categories: negative, neutral, and positive, providing a realistic testbed for processing noisy social media text.
Our approach is based on fine-tuning Transformer-based language models. Specifically, we compare general-domain models, such as BERT-base and RoBERTa-base, against a domain-specific model, BERTweet, to assess the importance of pre-training on Twitter data. Additionally, we explore parameter-efficient fine-tuning (PEFT) using Low-Rank Adaptation (LoRA) to determine if we can reduce computational costs without sacrificing performance.
Finally, we address the challenge of class imbalance (where negative tweets are underrepresented) by implementing class weighting and label smoothing strategies. The goal of this project is to identify the most effective and efficient approach for tweet-level sentiment classification, balancing high accuracy with robustness and training speed.

2 Dataset and Exploratary Data Analysis

Dataset Discription Our project uses the TweetEval Sentiment dataset, a widely adopted benchmark for evaluating sentiment classification models on Twitter data. The dataset was compiled by unifying several existing Twitter corpora and standardizing their labels into three sentiment categories: negative (0), neutral (1), and positive (2). Because the tweets come from multiple sources, the dataset covers a broad range of topics, writing styles, punctuation habits, slang, hashtags, emojis, and user mentions, which makes it suitable for evaluating models under real Twitter-like noise. TweetEval provides pre-defined train, validation, and test splits, which allows consistent comparison across different models. In our project, we use the official splits without modification. The dataset contains 45,615 tweets for training, 2,000 tweets for validation, and 12,284 tweets for testing, following the same structure shown on the official HuggingFace dataset page. The original source of our dataset is: `https://huggingface.co/datasets/cardiffnlp/tweet_eval/viewer/sentiment?views%5B%5D=sentiment_train`

Label Distribution We first examined the label distribution across the three splits to understand the potential class imbalance. In the training set, neutral tweets form the largest class (about 21,000 samples), followed by positive tweets (around 18,000 samples), while negative tweets are the smallest group with roughly 7,000 samples (Figure 1a). The validation set shows a similar pattern, with neutral and positive tweets at comparable sizes and negative tweets being considerably fewer (Figure 1c). The test set also follows the same trend, with neutral tweets the most frequent and positive tweets slightly less represented (Figure 1b).
This imbalance indicates that the dataset is not perfectly balanced, particularly due to the smaller number of negative samples. As a result, models may naturally bias toward predicting the more common classes. To account for this, we use macro-F1 during evaluation so that all three labels contribute equally to the final metric.



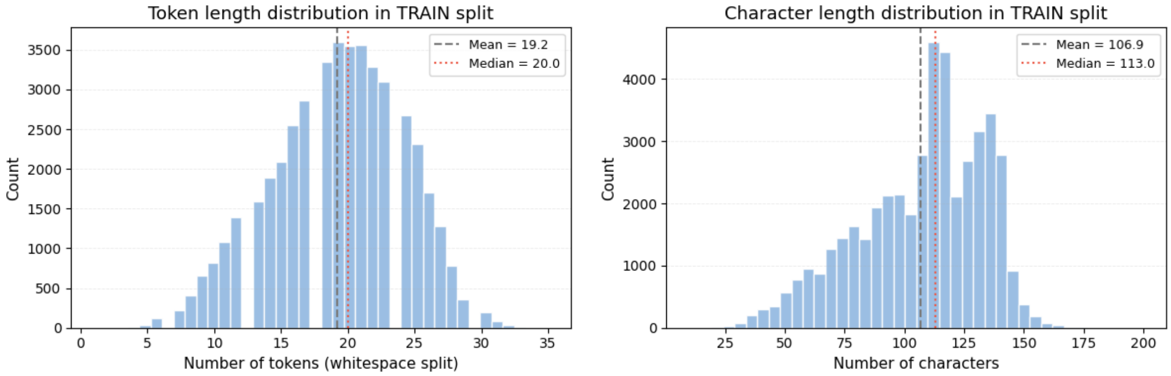
(a) Training Set

(b) Testing Set

(c) Validation Set

Figure 1: Label Distribution

Tweet Length Analysis We computed token-level lengths for the training set by splitting tweets on whitespace, which was shown in Figure 2a. The histogram shows that most tweets fall between 15 and 25 tokens, with the distribution centered around 19–20 tokens. The shortest tweets contain only one token, and the longest ones contain 35 tokens. These lengths reflect real Twitter usage, especially considering the previous 140-character limit during much of the dataset's collection period.



(a) Token Length Distribution

(b) Character Length Distribution

Figure 2: Length Distribution of the Training Set

We also examined character-level length (Figure 2b). The majority of tweets fall between 80 and 140 characters, and the distribution peaks around 110–120 characters. The longest tweets reach up to 200 characters. This confirms that the dataset contains many tweets close to the former character limit, which aligns well with the nature of Twitter text.
Understanding tweet length is important because it helps determine appropriate model settings, such as tokenizer truncation length. Since most tweets are relatively short, we set the maximum token length to 128 during preprocessing, ensuring that no meaningful information is lost.

Random Sample Examination To better understand the content of each sentiment class, we inspected random samples from the training set. The negative tweets often contain complaints, frustrations, or negative emotional expressions. The neutral tweets typically present statements of fact, news, or conversations without emotional content. The positive tweets commonly express excitement, praise, or enthusiasm. These examples provide qualitative evidence that the dataset labels reflect natural patterns of sentiment expression on Twitter.

Basic Text Cleaning Because social media text often includes usernames, emojis, hashtags, and informal spellings, we applied minimal cleaning. Following the project requirement to preserve Twitter's stylistic features, we intentionally kept @user mentions and emojis, since they can contribute to model performance in sentiment analysis. Our cleaning process only removed repeated whitespace and performed light trimming. This ensures that tweets retain their original emotional tone and Twitter-specific structure while still being standardized enough for tokenization. We also displayed before-and-after examples for several tweets, confirming that the cleaning step does not alter the meaning or stylistic markers of the text. This approach allows the model to learn from authentic Twitter signals while maintaining consistency.

3 Methods

In this project, we study tweet sentiment classification using several Transformer-based models. Our goal is to compare their performance on the TweetEval sentiment dataset and to understand whether techniques such as LoRA fine-tuning or class-balanced training can improve the results. In this section, we describe our overall workflow, the model architectures we use, and the training strategies applied.

3.1 Overall Pipeline

Our pipeline has four major steps. First, we preprocess the dataset by removing empty samples, normalizing whitespace, and tokenizing the tweets using the tokenizer corresponding to each model. Second, we load the data into PyTorch dataloaders with fixed train, validation, and test splits. Third, we fine-tune each model for several epochs using cross-entropy loss. Finally, we evaluate each model using accuracy, macro-F1, confusion matrices, and per-class precision, recall, and F1-scores.

3.2 Model Descriptions

BERT-base BERT-base is a bidirectional Transformer encoder with 12 layers and roughly 110M parameters. It is pre-trained on large general-domain corpora (BooksCorpus and English Wikipedia) and is not specifically adapted to Twitter-style text. We use the standard `bert-base-uncased` checkpoint from HuggingFace and fine-tune all model parameters on the TweetEval sentiment dataset.

RoBERTa-base RoBERTa-base is an improved variant of BERT that removes the next-sentence prediction objective and uses more training data, larger batches, and dynamic masking during pre-training. It also has 12 layers and a similar parameter count to BERT-base, but typically achieves better performance due to its more robust pre-training recipe. In our experiments, we use the `roberta-base` checkpoint and fine-tune all parameters as a stronger general-domain baseline.

BERTweet-base BERTweet-base is a RoBERTa-style model pre-trained specifically on 850M English tweets. Its tokenizer and vocabulary are adapted to user mentions, hashtags, URLs, and informal language commonly found on Twitter. Although it has a similar architecture and parameter size to BERT-base, the domain-matched pre-training often leads to improved performance on tweet classification tasks. We use the `vinai/bertweet-base` checkpoint from HuggingFace and fine-tune it end-to-end on TweetEval.

LoRA-BERT LoRA (Low-Rank Adaptation) is a parameter-efficient fine-tuning method that keeps the original model weights frozen and inserts small trainable low-rank matrices into selected layers (e.g., the query and value projections in self-attention). In our LoRA-BERT setting, we wrap `bert-base-uncased` with LoRA adapters and only train these additional parameters, while the base BERT weights remain fixed. This greatly reduces the number of trainable parameters and memory usage, while still allowing the model to adapt to the TweetEval sentiment task.

LoRA-RoBERTa We also applied the same low-rank adaptation idea to `roberta-base` model. We add LoRA modules to the attention layers of RoBERTa and fine-tune only these adapters on the training set. Compared to full RoBERTa fine-tuning, this approach significantly lowers the trainable parameter count while retaining most of the performance benefits of RoBERTa on sentiment classification. This setting allows us to directly compare full fine-tuning versus parameter-efficient fine-tuning on a strong general-domain encoder.

LoRA-BERTweet LoRA-BERTweet combines domain-specific pre-training with parameter-efficient adaptation. Starting from `vinai/bertweet-base`, we freeze the base model and inject LoRA adapters into its attention modules. Only the LoRA parameters are updated during training. This setup tests whether we can exploit BERTweet's strong prior on Twitter text while keeping the computational cost similar to other LoRA models, providing a fair comparison between domain-specific and general-domain encoders under the same parameter-efficient fine-tuning scheme.

BERTweet with class weights and label smoothing To further improve robustness on the TweetEval sentiment classes, we also experiment on BERTweet with class weights and label smoothing strategies. In this setting, we still fine-tune `vinai/bertweet-base` end-to-end, but modify the loss function. First, we compute class weights from the training label distribution and apply them to the cross-entropy loss to slightly up-weight under-represented classes. Second, we apply label smoothing (e.g., $\epsilon = 0.05$), which replaces one-hot targets with a softened label distribution. This combination aims to reduce overconfidence and mitigate the impact of mild class imbalance, potentially improving macro-F1 across the three sentiment categories.

3.3 Training Setup

All models use the same input preprocessing. Tweets are tokenized with the corresponding tokenizer (BERT, RoBERTa, or BERTweet), truncated to a maximum sequence length of 128, and dynamically padded in each batch. We use the AdamW optimizer with a linear learning rate schedule and a batch size of 32 for training (and 64 for evaluation). All experiments were trained using Colab A100 GPU.

Full Fine-tuning For the full fine-tuning experiments (BERT-base, RoBERTa-base, and BERTweet-base), we update all model parameters with a learning rate of 2×10^{-5} for three epochs. In addition, we test a variant of BERTweet with class weighting and label smoothing: class weights are computed from the training label distribution and applied to the cross-entropy loss, and we use a small label smoothing factor (e.g., $\varepsilon = 0.05$) to soften the target distribution. This setting is designed to mitigate mild class imbalance and reduce overconfident predictions.

LoRA Fine-tuning For the LoRA experiments (LoRA-BERT, LoRA-RoBERTa, and LoRA-BERTweet), we keep the base encoder weights frozen and only train the classification head together with the LoRA adapters. We follow a common configuration with rank $r = 8$, scaling factor $\alpha = 16$, and LoRA dropout of 0.1. Adapters are inserted into the self-attention modules, targeting the `query` and `value` projection layers. Because the number of trainable parameters is much smaller in this setting, we use a slightly higher learning rate of 1×10^{-4} while keeping the number of epochs (three) and batch size unchanged. This setup allows a direct comparison between full fine-tuning and parameter-efficient fine-tuning under similar training budgets.

Class weights and label smoothing In this variant, we keep the BERTweet architecture unchanged but modify the training objective. Let n_k denote the number of training examples in class k and $N = \sum_k n_k$ the total number of training examples. We first construct inverse-frequency class weights $w_k \propto \frac{N}{n_k}$ and then normalize them so that their mean equals 1. These weights are passed to a weighted cross-entropy loss, which slightly up-weights under-represented classes while keeping the overall loss scale stable. In addition, we apply label smoothing with factor $\varepsilon = 0.05$, replacing one-hot targets with a softened distribution that assigns $(1 - \varepsilon)$ to the true class and spreads ε uniformly over the remaining classes. The combined effect of class weighting and label smoothing was aimed to reduce sensitivity to mild class imbalance and overconfident predictions, while still using full-parameter fine-tuning of `vinai/bertweet-base`.

3.4 Evaluation Metrics

We evaluate each model using the following metrics:

- Overall accuracy on the test set, measuring the proportion of correctly classified tweets.
- Macro-F1 score, which computes the F1-score for each class and then averages them, giving equal weight to negative, neutral, and positive tweets.
- Confusion matrices to visualize how predictions are distributed across true and predicted classes.
- Precision, recall, and F1-score for each individual class, to examine where the model is more conservative or more error-prone.

Together, these metrics allow us to assess both overall performance and class-specific behavior. This is particularly important for TweetEval sentiment, where the class distribution is imbalanced and macro-F1 is more informative than accuracy alone.

4 Results

In this section, we present the performance of all models on the TweetEval sentiment dataset. For each configuration, we report the evolution of training and validation loss, the final macro-F1 and accuracy on the test set, the corresponding confusion matrix, and per-class precision, recall, and F1-scores. These results allow us to compare full fine-tuning and LoRA-based approaches, and to analyze how different architectures behave across the three sentiment classes.

4.1 Overall Performance

Our goal is to train and identify the best model for TweetEval sentiment classification. Table 1 and Table 2 summarize the validation and test performance of all candidates.

The general-domain BERT baseline performs the worst, with a test macro-F1 of 0.691. Introducing LoRA on top of BERT improves macro-F1 slightly to 0.698, but both variants remain clearly behind the RoBERTa and BERTweet families and are therefore not strong contenders for the final model. RoBERTa-based models form the next tier. RoBERTa-base and LoRA-RoBERTa obtain test macro-F1 scores of 0.698 and 0.707, respectively, confirming that RoBERTa is a stronger encoder than BERT and that LoRA can match or slightly improve full fine-tuning while training far fewer parameters. However, their performance is still dominated by the BERTweet models.Full BERTweet fine-tuning achieves the highest validation macro-F1 (0.745) and the best test results (0.723 accuracy and 0.721 macro-F1). LoRA-BERTweet and BERTweet with class weights and label smoothing are close behind (test macro-F1 of 0.714 and 0.715), but they do not surpass the standard BERTweet fine-tuning. Overall, the BERTweet-base with full fine-tuning was the best-performing model, providing the strongest overall performance on TweetEval sentiment.

Table 1: Validation performance on TweetEval sentiment.

Model	Val Accuracy	Val Macro-F1
BERT-base	0.752	0.736
LoRA-BERT	0.724	0.707
RoBERTa-base	0.751	0.734
LoRA-RoBERTa	0.743	0.730
BERTweet-base	0.759	**0.745**
LoRA-BERTweet	0.741	0.728
BERTweet + class weights + label smoothing	0.753	0.740

Table 2: Test performance on TweetEval sentiment.

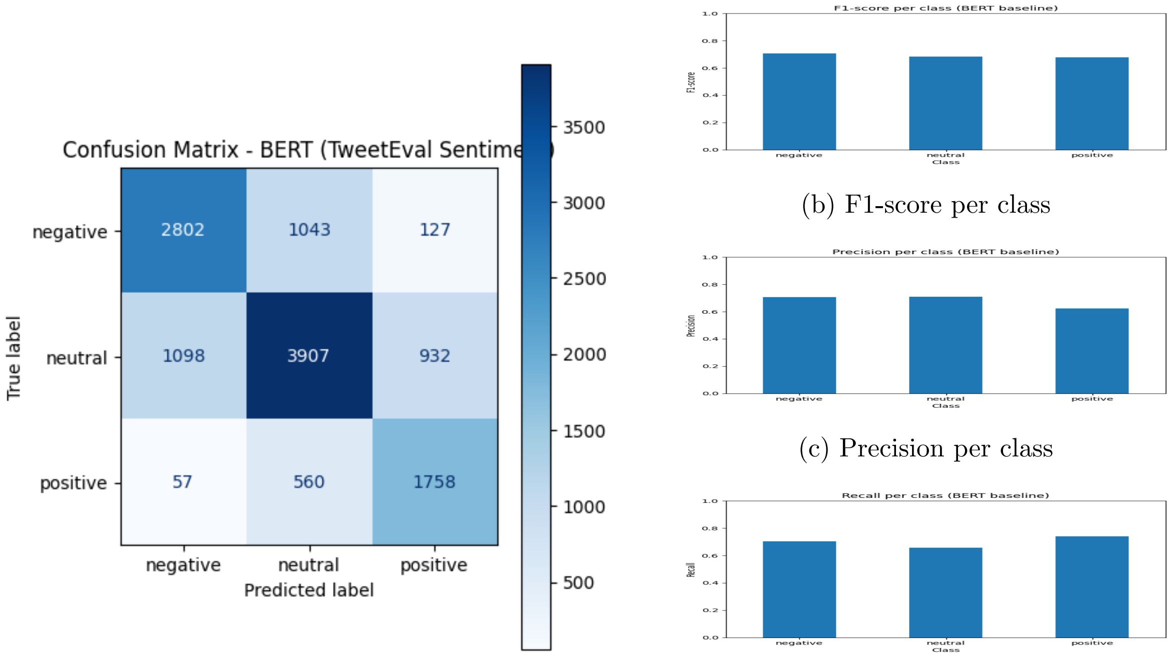
Model	Test Accuracy	Test Macro-F1
BERT-base	0.692	0.691
LoRA-BERT	0.700	0.698
RoBERTa-base	0.699	0.698
LoRA-RoBERTa	0.706	0.707
BERTweet-base	0.723	**0.721**
LoRA-BERTweet	0.714	0.714
BERTweet + class weights + label smoothing	0.714	0.715

4.2 BERT-base

During Training, the BERT-base baseline reaches an overall accuracy and macro-F1 of 0.69 on the TweetEval test set (Table 3). Performance is fairly balanced across classes, with F1-scores of 0.71 for negative, 0.68 for neutral, and 0.68 for positive tweets. BERT is slightly stronger on the negative class, where both precision and recall are around 0.71–0.72, but it struggles more on neutral tweets, where recall drops to 0.66. Overall, these numbers confirm that BERT-base provides a reasonable baseline, but leaves clear room for improvement by stronger or domain-specific models.

Table 3: Classification report on the TweetEval test set (BERT-base).

Class	Precision	Recall	F1-score	Support
Negative	0.71	0.72	0.71	3972
Neutral	0.71	0.66	0.68	5937
Positive	0.62	0.74	0.68	2375
Accuracy	—	—	0.69	12284
Macro avg	0.68	0.70	0.69	12284
Weighted avg	0.69	0.69	0.69	12284



(b) F1-score per class

(c) Precision per class

(a) Confusion matrix

(d) Recall per class

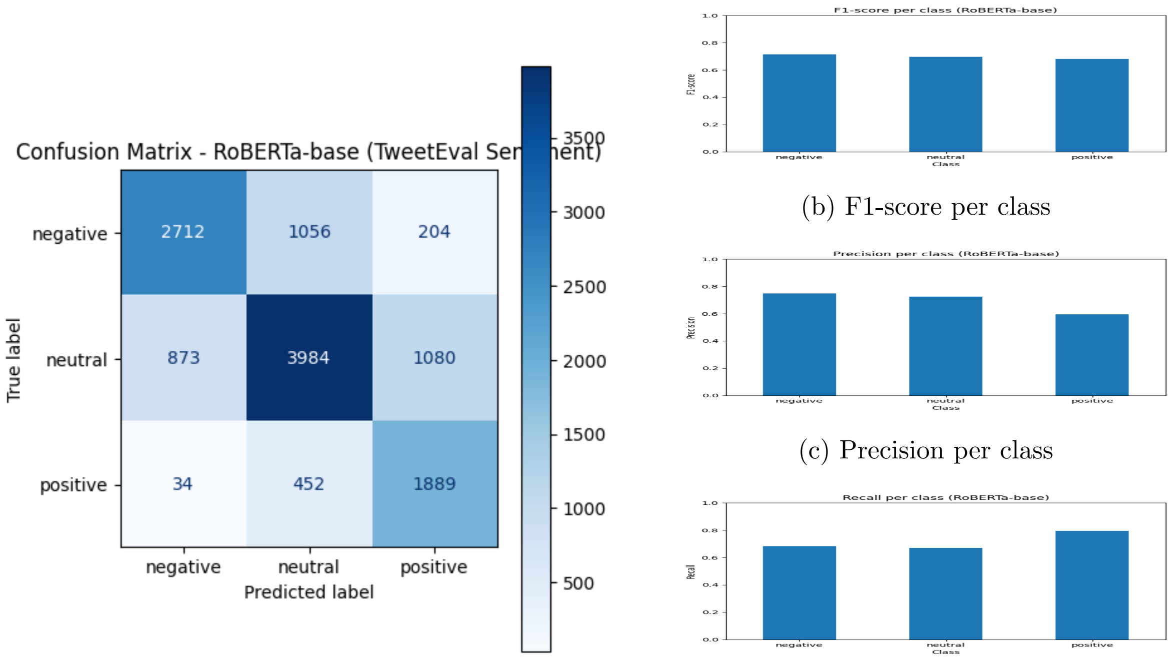
Figure 3: BERT-base performance on TweetEval sentiment: confusion matrix (left) and per-class metrics (right).

4.3 RoBERTa-base

Table 4 shows that RoBERTa-base improves slightly over the BERT baseline, reaching 0.70 accuracy and macro-F1 on the test set. Performance is relatively balanced across classes, with F1-scores of 0.71, 0.70, and 0.68 for negative, neutral, and positive tweets, respectively. RoBERTa is particularly strong on the positive class in terms of recall (0.80), but this comes at the cost of lower precision (0.60), indicating that it tends to over-predict positive sentiment.

Table 4: Classification report on the TweetEval test set (RoBERTa-base).

Class	Precision	Recall	F1-score	Support
Negative	0.75	0.68	0.71	3972
Neutral	0.73	0.67	0.70	5937
Positive	0.60	0.80	0.68	2375
Accuracy	—	—	0.70	12284
Macro avg	0.69	0.72	0.70	12284
Weighted avg	0.71	0.70	0.70	12284



(b) F1-score per class

(c) Precision per class

(a) Confusion matrix

(d) Recall per class

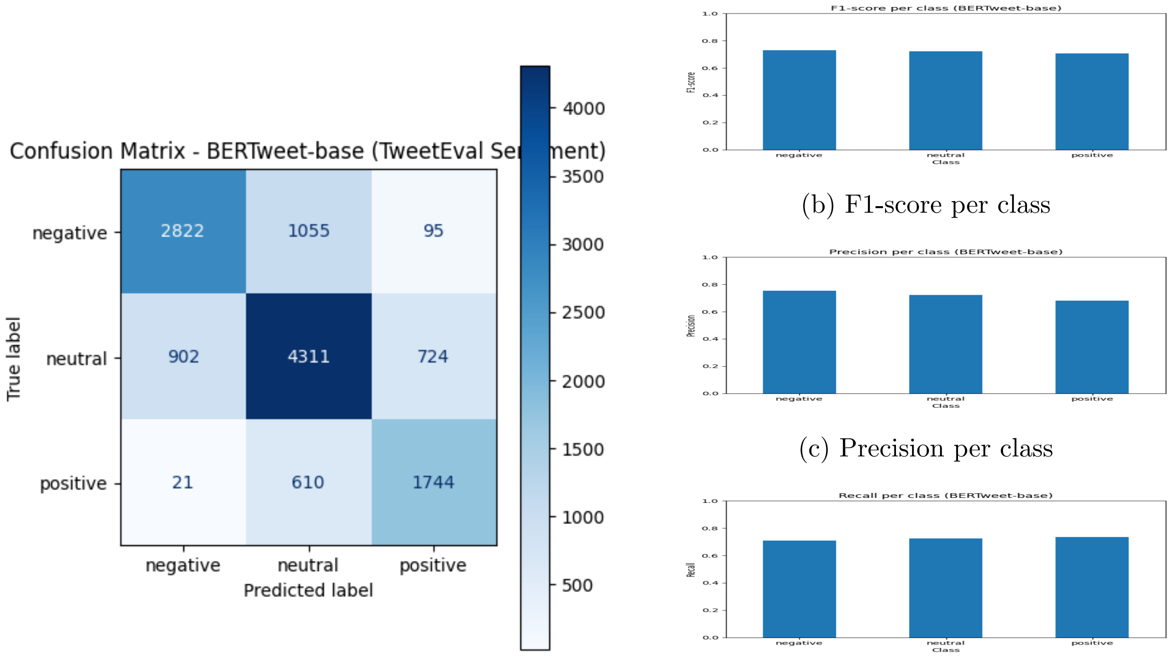
Figure 4: RoBERTa-base performance on TweetEval sentiment: confusion matrix (left) and per-class metrics (right).

4.4 BERTweet-base

As shown in Table 5, BERTweet-base achieves the best overall test performance among all models, with 0.72 accuracy and macro-F1. The three classes are handled very evenly, with F1-scores between 0.71 and 0.73. Compared to RoBERTa, BERTweet delivers higher precision and more balanced recall across classes, reflecting the benefit of pre-training on Twitter data for this task.

Table 5: Classification report on the TweetEval test set (BERTweet-base).

Class	Precision	Recall	F1-score	Support
Negative	0.75	0.71	0.73	3972
Neutral	0.72	0.73	0.72	5937
Positive	0.68	0.73	0.71	2375
Accuracy	—	—	0.72	12284
Macro avg	0.72	0.72	0.72	12284
Weighted avg	0.72	0.72	0.72	12284



(a) Confusion matrix

(b) F1-score per class

(c) Precision per class

(d) Recall per class

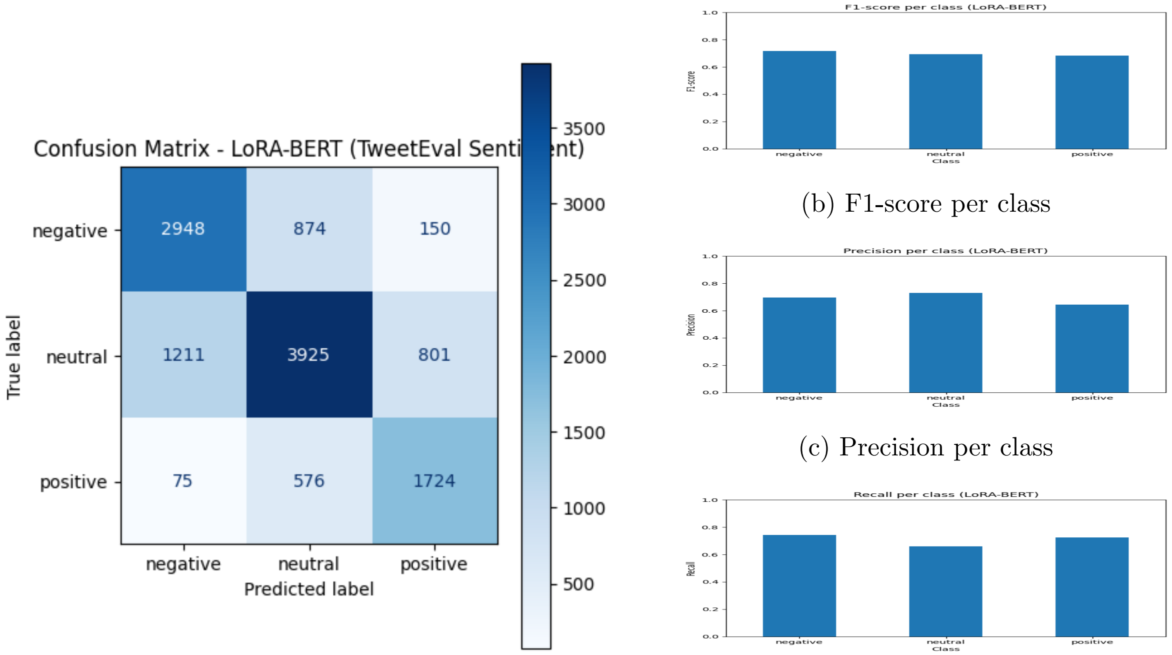
Figure 5: BERTweet performance on TweetEval sentiment: confusion matrix (left) and per-class metrics (right).

4.5 LoRA-BERT

Table 6 reports the results for BERT with LoRA adapters. This configuration attains 0.70 accuracy and macro-F1, slightly outperforming full BERT fine-tuning. The gains are modest but consistent across classes, with F1-scores of 0.72, 0.69, and 0.68 for negative, neutral, and positive tweets. In other words, LoRA recovers most of BERT's performance while substantially reducing the number of trainable parameters.

Table 6: Classification report on the TweetEval test set (BERT + LoRA).

Class	Precision	Recall	F1-score	Support
Negative	0.70	0.74	0.72	3972
Neutral	0.73	0.66	0.69	5937
Positive	0.64	0.73	0.68	2375
Accuracy	—	—	0.70	12284
Macro avg	0.69	0.71	0.70	12284
Weighted avg	0.70	0.70	0.70	12284



(b) F1-score per class

(c) Precision per class

(a) Confusion matrix

(d) Recall per class

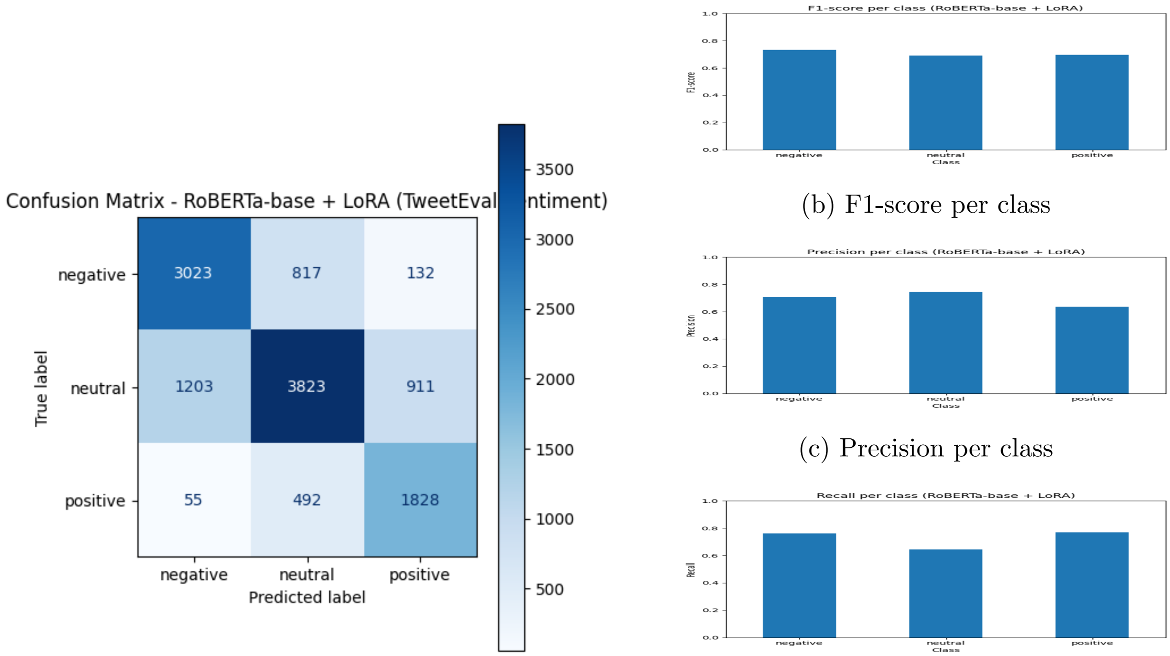
Figure 6: BERT + LoRA performance on TweetEval sentiment: confusion matrix (left) and per-class metrics (right).

4.6 LoRA-RoBERTa

The LoRA-RoBERTa model (Table 7) achieves 0.71 accuracy and macro-F1 on the test set, slightly improving over full RoBERTa-base. F1-scores for the three classes (0.73, 0.69, 0.70) are again quite balanced, with particularly strong recall for the negative and positive classes (0.76 and 0.77). This indicates that LoRA can match or even exceed full RoBERTa fine-tuning while being more parameter-efficient.

Table 7: Classification report on the TweetEval test set (RoBERTa-base + LoRA).

Class	Precision	Recall	F1-score	Support
Negative	0.71	0.76	0.73	3972
Neutral	0.74	0.64	0.69	5937
Positive	0.64	0.77	0.70	2375
Accuracy	—	—	0.71	12284
Macro avg	0.70	0.72	0.71	12284
Weighted avg	0.71	0.71	0.71	12284



(b) F1-score per class

(c) Precision per class

(a) Confusion matrix

(d) Recall per class

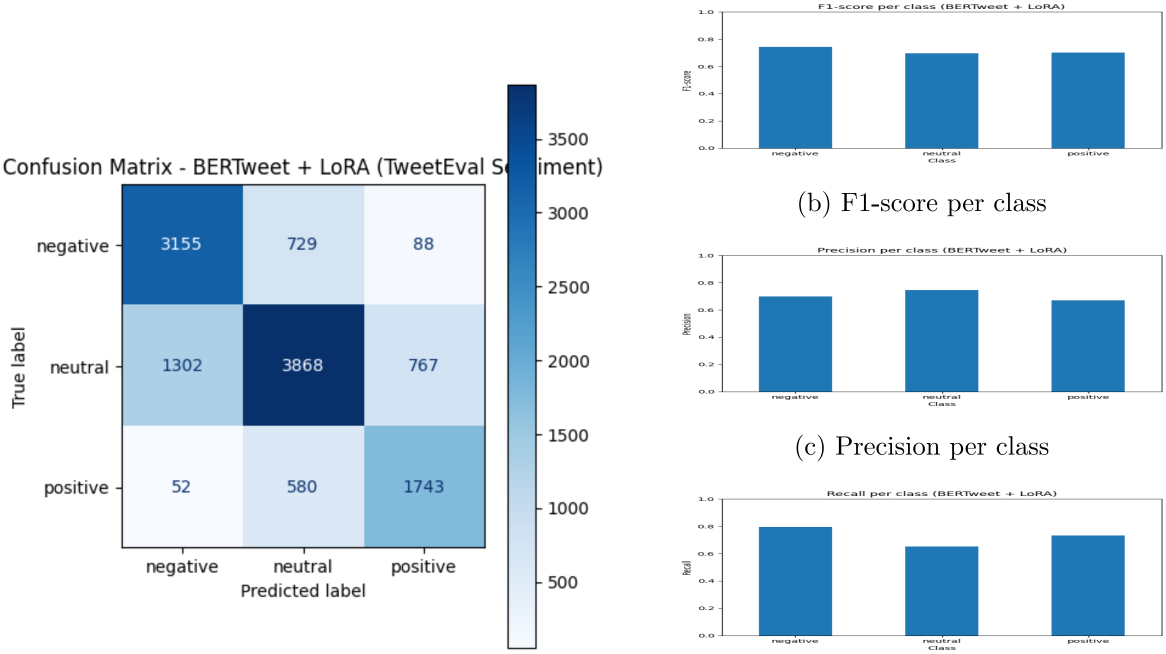
Figure 7: RoBERTa-base + LoRA performance on TweetEval sentiment: confusion matrix (left) and per-class metrics (right).

4.7 LoRA-BERTweet

Table 8 shows that LoRA-BERTweet attains 0.71 accuracy and macro-F1, only slightly below the fully fine-tuned BERTweet model. The F1-scores are 0.74, 0.70, and 0.70 for negative, neutral, and positive tweets, respectively, with high recall on the negative class (0.79). These results suggest that LoRA preserves most of BERTweet's predictive power while offering clear advantages in training efficiency.

Table 8: Classification report on the TweetEval test set (BERTweet-base + LoRA).

Class	Precision	Recall	F1-score	Support
Negative	0.70	0.79	0.74	3972
Neutral	0.75	0.65	0.70	5937
Positive	0.67	0.73	0.70	2375
Accuracy	—	—	0.71	12284
Macro avg	0.71	0.73	0.71	12284
Weighted avg	0.72	0.71	0.71	12284



(b) F1-score per class

(c) Precision per class

(a) Confusion matrix

(d) Recall per class

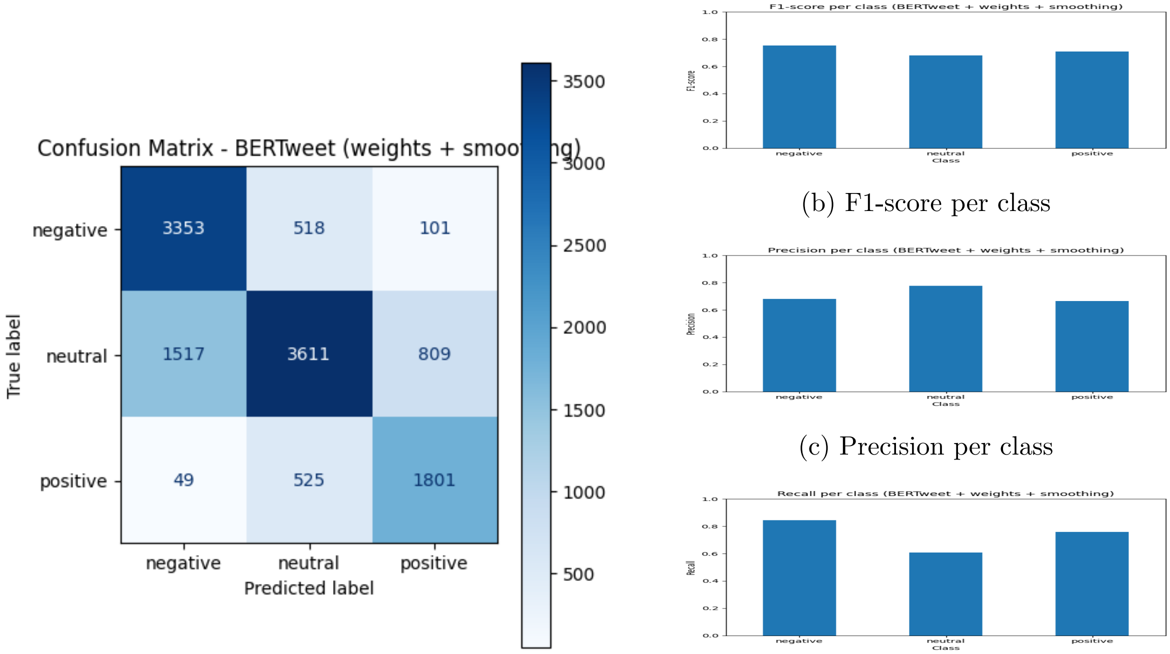
Figure 8: BERTweet + LoRA performance on TweetEval sentiment: confusion matrix (left) and per-class metrics (right).

4.8 BERTweet with Class Weights and Label Smoothing

Table 9 presents the results for BERTweet with class weighting and label smoothing. This variant reaches 0.71 accuracy and macro-F1, comparable to LoRA-BERTweet but still slightly below standard BERTweet. It achieves the highest F1 on the negative class (0.75), driven by very high recall (0.84), while performance on neutral tweets is somewhat weaker (F1 = 0.68). Overall, the modified loss improves robustness for the minority classes but does not yield a clear gain over the best full fine-tuning configuration.

Table 9: Classification report on the TweetEval test set (BERTweet + class weights + label smoothing).

Class	Precision	Recall	F1-score	Support
Negative	0.68	0.84	0.75	3972
Neutral	0.78	0.61	0.68	5937
Positive	0.66	0.76	0.71	2375
Accuracy	—	—	0.71	12284
Macro avg	0.71	0.74	0.71	12284
Weighted avg	0.72	0.71	0.71	12284



(b) F1-score per class

(c) Precision per class

(a) Confusion matrix

(d) Recall per class

Figure 9: BERTweet with class-weighted and label smoothing training performance on Tweet-Eval sentiment: confusion matrix (left) and per-class metrics (right).

4.9 Efficiency Comparison

Beyond predictive performance, we also compare the computational cost of different models, since a "best" model in practice should balance accuracy and efficiency. For each configuration, Table 10 reports the number of epochs (fixed to three) and the approximate wall-clock training time on the same Colab A100 GPU.

Overall, full fine-tuning of BERT, RoBERTa, and BERTweet requires about 9–10 minutes for three epochs, whereas the corresponding LoRA variants finish in roughly 5–7 minutes. For example, LoRA-RoBERTa reduces training time from 09:22 to 05:52, and LoRA-BERTweet reduces training time from 09:15 to 05:43, a speed-up of almost 40% while training only a small fraction of the parameters.

Therefore, If we only optimize for accuracy, full BERTweet fine-tuning is the best model, achieving the highest test macro-F1 (0.721). However, LoRA-BERTweet achieves a very similar macro-F1 (0.714) at substantially lower computational cost. In settings where GPU time or memory is constrained, LoRA-BERTweet therefore offers an attractive alternative: it is nearly as accurate as the best model while being significantly more efficient to train.

Table 10: Approximate wall-clock training time on TweetEval sentiment (3 epochs on A100 GPU)

Model	Epochs	Train time (mm:ss)
BERT-base	3	09:30
LoRA-BERT	3	06:41
RoBERTa-base	3	09:22
LoRA-RoBERTa	3	05:52
BERTweet-base	3	09:15
LoRA-BERTweet	3	**05:43**
BERTweet + class weights + label smoothing	3	09:16

5 Discussion and Limitations

In this section, we analyze the trade-offs between model performance and computational cost, discuss the impact of domain-specific pre-training, and evaluate the robustness of our class-imbalanced strategies.

5.1 Impact of Domain Adaptation

Our results strongly suggest that domain-specific pre-training is the most critical factor for success in sentiment analysis on Twitter. While RoBERTa-base (Test Macro-F1: 0.698) slightly outperformed BERT-base (Test Macro-F1: 0.691), both were surpassed by BERTweet-base (Test Macro-F1: 0.721). Though RoBERTa is generally considered a more robust model than BERT, it was trained on general text (books, Wikipedia). In contrast, BERTweet was pre-trained on 850M English tweets. This implies that for social media tasks, the model's ability to understand informal grammar, hashtags, and emojis is more important than the architectural optimizations found in RoBERTa.

5.2 Efficiency vs. Performance (LoRA)

One of the main goals of this project was to determine if Parameter-Efficient Fine-Tuning (LoRA) could match full fine-tuning results while lowering computational costs. We compared the wall-clock training time for all three architectures and observed a consistent speedup across the board.

- **BERT-base:** Training time dropped from 09:30 (full) to 06:41 (LoRA), a reduction of approximately 30%.
- **RoBERTa-base:** Training time improved from 09:22 (full) to 05:52 (LoRA), reducing costs by nearly 37%.
- **BERTweet-base:** We observed similar gains, with time dropping from 09:15 (full) to 05:43 (LoRA), reducing costs by 38%.

Crucially, this efficiency did not come at the cost of accuracy. While LoRA-BERTweet had a marginal drop in Macro-F1 (0.721 to 0.714), the LoRA versions of BERT and RoBERTa actually achieved slightly higher scores than their fully fine-tuned counterparts. This suggests that LoRA is not only a time-saving technique but also a robust method that can effectively adapt various Transformer architectures to the sentiment analysis task with significantly fewer trainable parameters.

5.3 Robustness and Class Imbalance

The TweetEval dataset is imbalanced, with negative tweets being the minority class (roughly 7,000 samples compared to 21,000 neutral). Standard models often prioritize the majority class to maximize accuracy. Our experiment with Class Weights and Label Smoothing addressed this issue effectively. While the overall Test Macro-F1 (0.715) was slightly lower than the standard BERTweet model, the robustness on the minority class improved drastically:

- **Standard BERTweet:** Recall for Negative class = 0.71.
- **Weighted BERTweet:** Recall for Negative class = 0.84.

This increase in recall indicates that the weighted model is much "safer" for applications where identifying negative sentiment (e.g., hate speech or customer complaints) is critical, even if it results in slightly more false positives.

5.4 Limitations

There are two main limitations to our approach. First, we truncated all tweets to a maximum length of 128 tokens. While our EDA showed most tweets are short, the longest tweets in the dataset extend to 200 characters. Truncation might have resulted in the loss of context for these longer samples. Second, we only evaluated our models on the specific TweetEval splits. Social media language evolves rapidly, and a model trained on this static dataset might struggle with current slang or trending topics in 2025 that were not included during the dataset's collection.

6 Conclusion

In this project, we developed and evaluated a sentiment classification pipeline for Twitter data using BERT, RoBERTa, and BERTweet. We established that domain-specific pre-training is superior to general-domain models for this task, with BERTweet-base achieving the highest Test Macro-F1 of 0.721.

Furthermore, we demonstrated that LoRA is an efficient alternative to full fine-tuning, offering comparable accuracy while reducing training time by over 30%. Finally, we showed that applying class weights and label smoothing is a viable strategy to boost robustness, significantly improving the model's ability to detect negative tweets in an imbalanced dataset. Future work could involve testing these models on cross-domain datasets to see if the BERTweet advantage holds on other social media platforms like Reddit or Instagram.