COMP 647 Final Project
Sentiment Analysis of Twitter Data

Group 6: Connor Smith, Nicholas Andrade, Nina Luong, Arati Pati

I. ABSTRACT

Social media platforms, especially Twitter, serve as significant venues for individuals to express their opinions on emerging technologies such as self-driving cars. Analyzing these sentiments provides invaluable information for companies, researchers, and policymakers, enabling them to gauge public perception and inform decision-making processes. This study compares various machine learning and deep learning models for sentiment analysis of tweets related to self-driving cars, categorizing them on a five-point scale from highly negative to highly positive. The research evaluates traditional approaches alongside transformer-based and transfer-learning methods, incorporating techniques to address imbalanced data for improved model performance. By systematically analyzing the performance of these models, the findings improve the understanding of the dynamics of public opinion with respect to autonomous vehicle technologies. Furthermore, this study provides a framework for future sentiment analysis efforts in similar domains, highlighting the strengths and limitations of various analytical techniques in interpreting complex emotional tones within large-scale social media data.

II. INTRODUCTION

Over the years, the methodologies used in sentiment analysis have evolved significantly, reflecting broader advancements in machine learning and deep learning. Initially, traditional machine learning techniques such as Logistic Regression, Naive Bayes, Support Vector Machines (SVM), and XGBoost served as the backbone of sentiment classification tasks. These models relied heavily on feature engineering, where specific attributes of the text, like word frequencies and n-grams, were manually extracted and utilized to train classifiers. Although effective to a certain extent, these approaches often struggled to capture the underlying and contextual meanings inherent in human language.

The advent of deep learning introduced more sophisticated models capable of automatically learning representations from raw data. Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks [1] became prominent in sentiment analysis due to their ability to capture spatial and temporal dependencies in text, respectively. CNNs excelled at identifying local patterns and phrases that contribute to sentiment, while LSTMs were adept at understanding the sequential nature of language, allowing for better interpretation of context over longer text spans.

In recent years, transformer-based models like BERT (Bidirectional Encoder Representations from Transformers) [2] and RoBERTa (A Robustly Optimized BERT Pretraining Approach) [3] have revolutionized the field of sentiment analysis. These models utilize self-attention mechanisms to capture intricate relationships between words in a sentence, enabling a deeper and more comprehensive understanding of context and semantics. Their pretraining on vast corpora allows them to generalize effectively across various tasks with minimal fine-tuning, significantly enhancing performance compared to their predecessors.

Building upon the strengths of transformer-based models, transfer learning approaches that integrate models such as RoBERTa with LSTM layers have gained traction. This hybrid architecture combines the contextual strengths of transformers with the sequential processing capabilities of LSTMs, aiming to exploit the best of both worlds for more accurate sentiment classification.

The rapid growth and continuous improvement of these methodologies underscore the dynamic nature of sentiment analysis. As social media platforms grow and generate vast amounts of textual data, the demand for more accurate and efficient sentiment analysis tools intensifies. This study aims to explore and compare a spectrum of machine learning and deep learning models—from traditional classifiers and neural networks to advanced transformer-based and transfer learning approaches—for sentiment analysis of tweets related to self-driving cars. By evaluating these models on a five-point sentiment scale, the research seeks to identify the most effective techniques to capture public perceptions and to contribute to a broader understanding of sentiment dynamics in the context of autonomous vehicle technologies.

III. PREVIOUS WORK

Sentiment analysis has progressed from traditional machine learning techniques to deep learning and transformer-based models. Early work by Pang et al. (2002) [4] compared methods like Naive Bayes and SVM for sentiment classification on movie reviews. As deep learning emerged, Kalchbrenner et al. (2014) [5] introduced CNNs for text classification, improving performance on sentiment tasks. The Transformer model, presented by Vaswani et al. (2017) [6], revolutionized NLP by using attention mechanisms, paving the way for BERT and RoBERTa, which further advanced sentiment analysis by utilizing large pre-trained models. Recent studies, such as Tan et al. (2022) [7], have explored hybrid models that combine transformers with LSTMs to enhance performance on sentiment classification tasks, specifically for the IMDb and Twitter US Airline sentiment datasets.

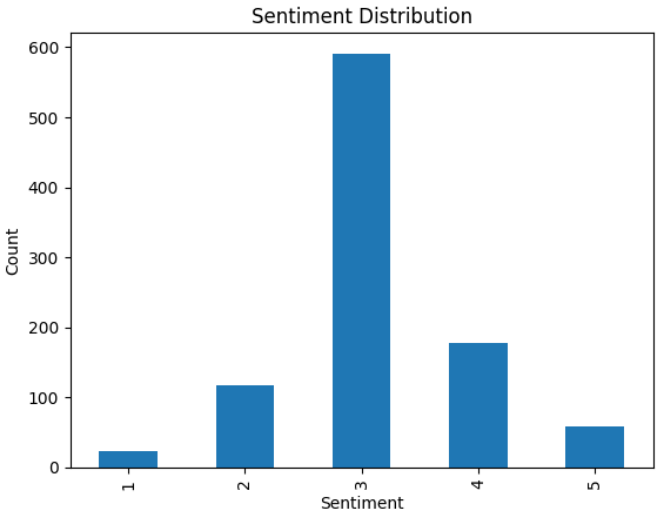

Fig. 1. Data Distribution

Several studies have analyzed public sentiment toward self-driving cars using sentiment analysis of social media data. Sadiq and Khan (2018) conducted sentiment analysis on Twitter data to gauge user perceptions of autonomous vehicles, uncovering both excitement and concerns about safety and reliability [8]. Dutta and Das (2020) applied a hierarchical attention-based LSTM network to classify sentiments in tweets about self-driving cars, achieving an accuracy of 85% [9]. Kim (2024) further assessed consumer expectations and concerns regarding autonomous vehicles, focusing on the need for improved user experience [10]. These studies highlight the importance of addressing safety concerns and enhancing user experience to promote greater public acceptance of self-driving cars.

IV. DATA SOURCE

The project uses the Twitter Dataset from Kaggle [11], containing 981 tweets from 2017. Each tweet is labeled with sentiment 1-5, with a tweet concerning self-driving cars. The data is of a platykurtic distribution making it a skinnier version of the normal distribution as seen in Figure 1.

Figure 2 illustrates a word cloud generated from the cleaned Twitter dataset, highlighting the most frequently mentioned terms in discussions about self-driving cars. Prominent words such as ”driverless,” ”car,” ”Google,” and ”future” emphasize the focus on autonomous vehicle technology and its implications. Other terms like ”testing,” ”road,” and ”technology” reflect the ongoing development and deployment of driverless cars, while mentions of ”California” and ”law” hint at regional and regulatory aspects. This visualization provides an overview of the dataset’s thematic content and underscores key areas of public interest and discussion.

V. METHODOLOGY

We have included a flowchart, shown in Figure 3, to outline the step-by-step process of data preparation, model training, and evaluation. This visualization clarifies the workflow and highlights key decision points in the modeling pipeline.

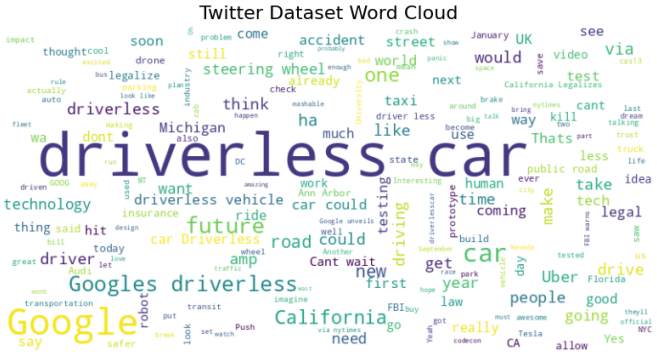

Fig. 2. Word Cloud of Twitter Dataset

A. Data Preprocessing

Tweets were cleaned by removing URLs, mentions, special characters, and other non-text elements. Tokenization, stemming, and lemmatization were applied to standardize the text, ensuring consistency across the dataset.

To address the imbalanced nature of the dataset, strategies such as data augmentation and class weight computation were employed, enhancing the model’s ability to learn from all sentiment classes effectively.

B. Feature Extraction

Three primary methods of feature extraction were employed in this study:

1) TF-IDF (Term Frequency-Inverse Document Frequency): TF-IDF [12] was utilized as a traditional feature extraction technique to convert textual data into numerical representations. By calculating the importance of each term in a document relative to the entire corpus, this method provided a baseline feature set for traditional machine learning models.

2) GloVe (Global Vectors for Word Representation): GloVe embeddings [13] were used to generate dense, pre-trained word vectors that capture semantic relationships between words. These embeddings, trained on large text corpora, allowed the models to incorporate global co-occurrence statistics of words, enabling them to understand semantic similarity and context at the word level.

3) Contextualized Embeddings: Advanced word embedding techniques derived from transformer-based models, such as BERT and RoBERTa, were also used. These embeddings provided context-aware representations of words, allowing the models to capture both word meanings and sentence-level dependencies effectively.

C. Models

The models examined in this study are categorized into the following methods:

- Traditional Machine Learning: Logistic Regression, Naive Bayes, XGBoost, SVM

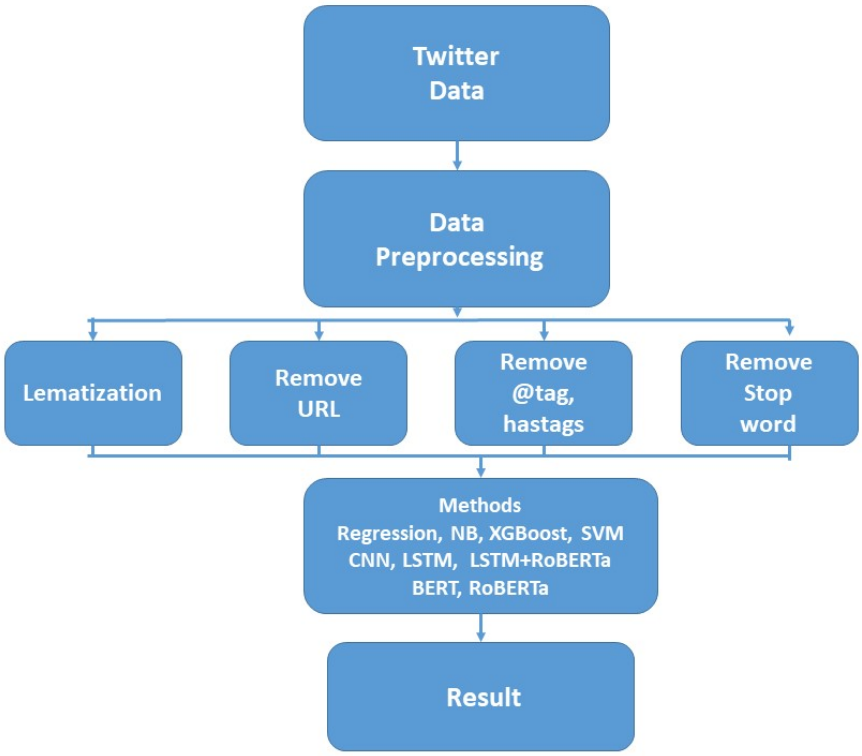

Fig. 3. Flowchart of Twitter Sentiment Analysis

- Traditional Deep Learning: CNN, LSTM
- Transformer-based methods: BERT, RoBERTa
- Hybrid approaches: RoBERTa + LSTM

D. Evaluation

Model performance is assessed using metrics such as accuracy, precision, recall, F1-score, and ROC curves.

E. Baseline and Benchmark

The baseline accuracy is set at 61%, which reflects the accuracy a model could achieve by always predicting the majority class (class 3) due to the imbalanced dataset. While this highlights the dataset's challenges, the objective is to develop a model that surpasses this baseline by effectively capturing the underlying sentiment across all classes. Our final model's performance will also be compared to state-of-the-art architectures on Kaggle's leaderboard, where top scores are reported as 96%, 75%, and 65%, respectively [14].

VI. TRADITIONAL MACHINE LEARNING METHODS

As a preliminary evaluation, the performance of various non-deep learning methods was assessed. Logistic Regression, Multinomial Naive Bayes, XGBoost, Support Vector Machine (SVM), and a simple Neural Network were applied.

The text analysis pipeline was built around the TF-IDF (Term Frequency-Inverse Document Frequency) [12] vectorization, which transforms raw tweet text into numerical features suitable for machine learning algorithms. TF-IDF assigns weights to words based on their frequency in individual tweets and rarity across the entire dataset, effectively capturing the importance of each term. The vectorizer was configured to select the 5000 most significant terms, thereby reducing dimensionality.

The data set was divided into training and testing subsets, with 80% of the data used for training and 20% for testing. Notably, the majority class (neutral sentiment) accounted for approximately 60% of the dataset, establishing a baseline accuracy against which model performance could be compared.

Among the non-deep learning models, the simple Neural Network emerged as the top performer, achieving an accuracy of 72.1%. It was closely followed by SVM at 70.0%, Logistic Regression at 68.0%, Naive Bayes at 67.0%, and XGBoost at 65.0%. This ranking indicates that the relationship between features and sentiments in the dataset is complex and may involve nonlinear interactions, which are better captured by more sophisticated models like Neural Networks and SVMs. Despite these results, the dataset's inherent class imbalance had a pronounced impact on model performance. Classes 1 and 5, representing the minority sentiments, were particularly challenging for all models, often resulting in precision and recall metrics of 0.00. Conversely, for class 3 (the majority sentiment), the models displayed high recall but lower precision, suggesting a bias toward predicting the dominant class.

These findings underscore the need for additional strategies to address the limitations posed by class imbalance and the inherent complexity of the task. Approaches such as oversam-

Model	Accuracy	Macro Avg F1	Weighted Avg F1
Logistic Regression	0.680	0.19	0.57
Naive Bayes	0.670	0.16	0.54
SVM (Linear)	0.701	0.23	0.60
XGBoost	0.650	0.30	0.61
Neural Network	0.721	0.41	0.69

TABLE I
COMPARISON OF MODEL PERFORMANCE ON SELF-DRIVING CAR SENTIMENT ANALYSIS

pling the minority classes, under-sampling the majority class, or applying class weighting during training could improve the models' ability to generalize across all sentiment categories. Furthermore, while the Neural Network outperformed the traditional machine learning models, its relatively simple architecture may still be insufficient to fully capture the nuances in the data, and this preliminary analysis indicates the need for deep learning models.

VII. TRADITIONAL DEEP LEARNING METHODS

This section describes the application of traditional deep learning methods, including Convolutional Neural Networks (CNNs) and Long Short-Term Memory Networks (LSTMs), to address the challenges posed by the given dataset. The dataset presented two major obstacles: extreme class imbalance and a small sample size. To mitigate these issues, we implemented a two-stage classification strategy using binary and multi-class classifiers, adapting both CNN and LSTM architectures to the task.

A. Dataset Challenges

The dataset underwent text pre-processing and tokenization mentioned in Methodology, and more details are explained in Transformer-based Models section.

a) Class Imbalance: Over 50% of the instances belong to a single dominant class (class 3, representing neutral opinions), while the remaining four classes (representing non-neutral opinions) are unevenly distributed, shown in Figure 1. Within the non-neutral classes, classes 4 and 5 are more dominant compared to classes 1 and 2. This severe imbalance significantly increases the likelihood of models being biased toward the majority class (neutral), reducing their effectiveness in identifying underrepresented non-neutral classes.

b) Small Dataset Size: The limited dataset size compounds the risk of over-fitting, restricting the model's ability to generalize effectively. This issue is further aggravated by the imbalanced class distribution, reducing the available information for learning meaningful patterns from the underrepresented non-neutral classes. As a result, the models are at risk of consistently predicting the dominant class (neutral) or favoring the more frequent non-neutral classes (classes 4 and 5), neglecting the others.

B. Proposed Strategy

1) Two-stage Framework: The two-stage framework includes a binary classification model and a multi-class classification model. This isolates the dominant class (neutral class - class 3) through binary classification, reducing its influence and simplifying the multi-class classification task for the remaining classes. This approach mitigates class imbalance by focusing separately on the dominant and non-dominant classes and improves generalization on a small dataset by enabling more focused learning in each stage.

a) Binary Classification:

- Purpose: Isolating the dominant class reduces its biasing effect, allowing the subsequent multi-class classification to better distinguish between non-dominant classes. In this case, neutral class 3 is labeled as 0, other classes are labeled as 1.
- Model Architecture: The binary classifier uses a CNN or LSTM-based model with embedding layers, hidden layers, and fully connected layers with dropout for regularization.

b) Multi-class Classification:

- CNN Multi-class Model: The binary classifier's outputs are concatenated with the original dataset features and used as additional input for training the CNN-based multi-class classifier.
- LSTM Multi-class Model: The LSTM-based model is trained exclusively on non-neutral data, filtering out samples based on class labels to focus on the remaining classes.

2) Embedding Layer: Both CNN and LSTM models utilized Twitter 27B, 200-dimension pre-trained embeddings [13] for rich word representations. These embeddings capture contextual relationships between words, reducing the reliance on extensive labeled data and improving the models' ability to generalize across all classes, including underrepresented ones.

3) Data Augmentation: To address the limited dataset size and enhance the model's ability to generalize, synthetic data augmentation was employed. Additionally, the class weights technique was used to further handle underrepresented classes.

a) CNN-based Models: For each row in the dataset, a second version was generated by replacing words in the sentence with their synonyms. This process effectively doubled the dataset size and introduced variability in the data, improving the model's robustness and reducing the risk of over-fitting. By expanding the dataset with realistic variations, the models could learn richer patterns and better handle underrepresented classes.

b) LSTM-based Models: A targeted augmentation strategy was applied for multi-class model on non-neutral classes. Based on class distribution, more augmentations are generated for underrepresented classes and none for well-represented ones.

In addition to data augmentation, class weights were applied during the training process to further address class imbalance. The class weight for each class i was calculated using the inverse frequency method:

$$w_i = \frac{N}{n_i}$$

where N is the total number of samples in the dataset and n_i is the number of samples in class i.

According to a study [15], the class weighting technique modifies the loss function to assign higher penalties to misclassifications of minority classes and lower penalties to majority classes. This adjustment ensures that the model places greater emphasis on correctly predicting underrepresented classes during training, thereby mitigating the inherent bias towards more frequent classes. By integrating these weights into the loss function, the LSTM model was encouraged to focus more on minority classes, reducing the likelihood of overfitting to majority classes and improving overall model fairness.

This combination of synthetic data augmentation and class weighting significantly enhanced the ability of the LSTM model to generalize across all classes, leading to improved performance metrics, especially for less frequent categories.

C. Convolutional Neural Networks (CNNs)

1) Model Architectures: CNNs were utilized for both binary and multi-class classification tasks due to their ability to automatically extract and learn hierarchical features from textual data. The CNN architecture consists of multiple convolutional layers with varying filter sizes to capture different n-gram patterns, followed by pooling layers that reduce dimensionality and enhance feature robustness. These layers are integrated with fully connected layers that perform the final classification using activation functions such as ReLU and softmax for multi-class tasks or sigmoid for binary classification. Additionally, incorporating pre-trained word embeddings [13] enriched the input representations, allowing the CNN to effectively leverage contextual information and achieve high performance across both classification scenarios. Below are the best-performing architectures for the binary and multi-class classifiers

a) Binary Classifier Architecture: The binary classifier was designed to effectively isolate the dominant class. Its architecture includes:

- Embedding Layer: Transforms input tokens into dense vectors using pre-trained embeddings. This layer leverages the Twitter 27B, 200-dimensional embeddings to capture semantic relationships between words, providing a robust foundation for feature extraction.
- Convolutional Layers
  - 3 CNN layers with 16 filters each.
  - Kernel size = 3, padding = 1.
- Fully Connected Layers
  - 3 linear layers with 32 neurons each.
  - Activation Function: Leaky ReLU activation is applied after each layer to introduce non-linearity.
  - Regularization: A dropout rate of 0.2 is implemented to prevent overfitting by randomly deactivating neurons during training.

The outputs from the binary classifier are concatenated with the original training and validation datasets, enhancing the subsequent stage of the model by providing enriched feature representations.

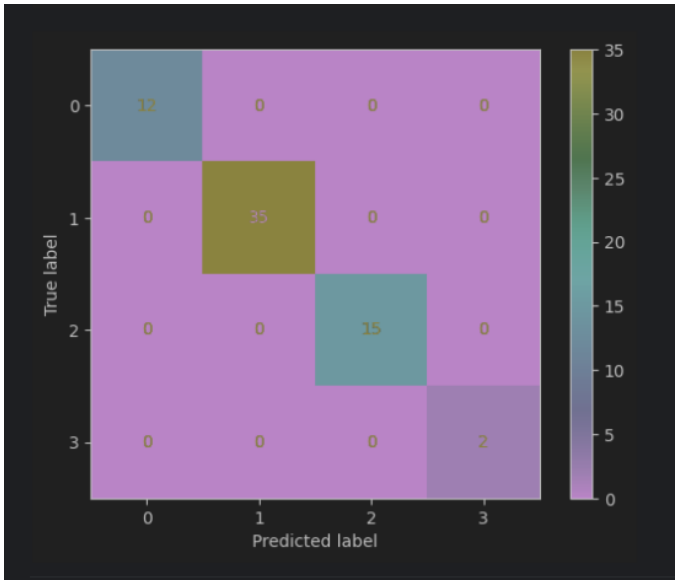

Fig. 4. Confusion Matrix for Non-neutral Classes - CNN

b) Multi-class Classifier Architecture: The multi-class classifier is structured to handle non-neutral classes (classes 1, 2, 4, 5) effectively. Its architecture comprises:

- Embedding Layer: Transforms input tokens into dense vectors using pre-trained embeddings
- Convolutional Layers
  - 3 CNN layers with 64 filters each.
  - Kernel size = 3, padding = 1.
- Fully Connected Layers
  - 4 linear layers with 32 neurons each.
  - Activation Function: Leaky ReLU activation is applied after each layer to introduce non-linearity and enable the network to learn more complex relationships.
  - Regularization: A dropout rate of 0.2 is implemented/
  - Batch Normalization: Applied to improve convergence speed and model stability by normalizing the inputs of each layer.

This combination of embedding, convolutional, and fully connected layers ensures that the multi-class classifier can effectively learn and generalize from the input data, providing robust performance across multiple classification categories. The results are illustrated in the confusion matrix shown in Figure 4. The figure demonstrates perfect classifications, indicating the model's high accuracy and reliability in distinguishing between the various classes.

2) Evaluation: Figure 5 illustrates the training and validation loss and accuracy over the training epochs for the CNN models.

The proposed CNN-based multi-class classification model achieved excellent performance, with perfect accuracy, precision, recall, and F1 scores across all five classes. The training and validation loss steadily decreased, while the accuracy

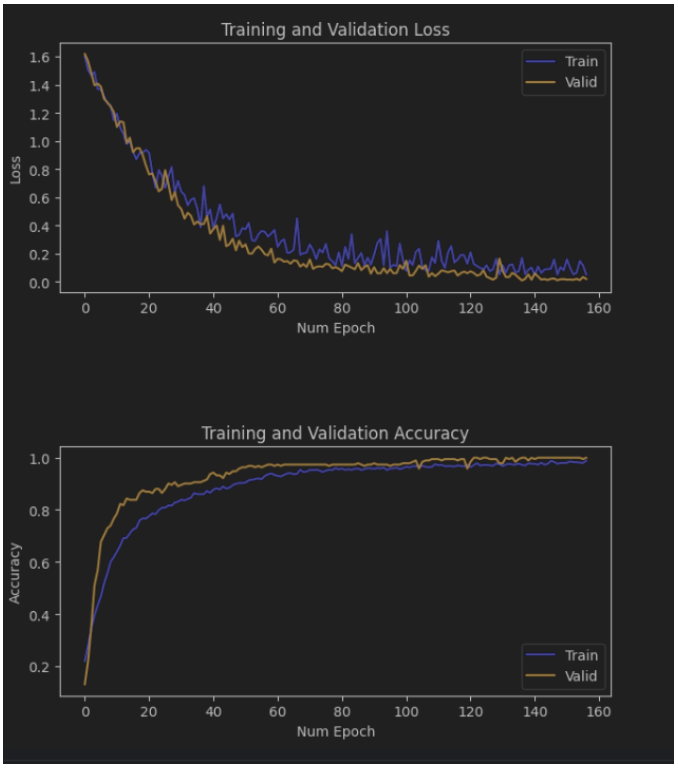

Fig. 5. Training and Validation Loss and Accuracy Plots - CNN Models

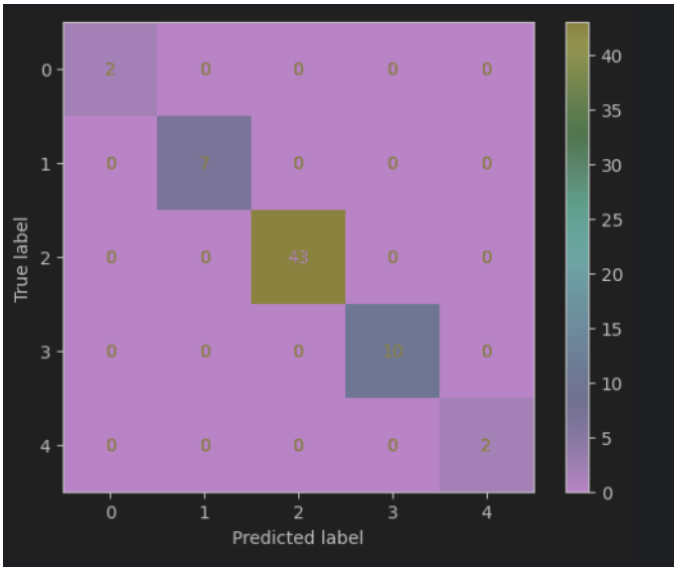

Fig. 6. Confusion Matrix for All 5 Classes - CNN

consistently improved over the epochs, as shown in the training and validation curves, indicating effective learning without over-fitting in the figure 5 .

Figure 6 presents the confusion matrix for the multi-class classifier on all labels.

The accuracy, precision, recall, and F1 scores for both the CNN-based multi-class classification model and the combined model are all perfect, as reflected in Figure 6. Notably,

Model: "sequential_3"

Layer (type)	Output Shape	Param #
text_vectorization (TextVectorization)	(None, 200)	0
embedding_3 (Embedding)	(None, 200, 200)	200,200
bidirectional_3 (Bidirectional)	(None, 128)	135,680
dropout_6 (Dropout)	(None, 128)	0
dense_6 (Dense)	(None, 64)	8,256
dropout_7 (Dropout)	(None, 64)	0
dense_7 (Dense)	(None, 1)	65

Fig. 7. Binary Classifier Architecture - LSTM

the diagonal entries in the confusion matrix indicate correct predictions for each class, confirming the model's ability to handle class imbalance and represent all categories effectively. The absence of off-diagonal entries further highlights the model's precision in minimizing misclassifications.

These results emphasize the impact of using pre-trained embeddings, synthetic data augmentation, and the two-stage framework in addressing the dataset's challenges. The combination of these techniques enabled the CNN model to achieve flawless classification performance, demonstrating its robustness and effectiveness in handling both balanced and imbalanced data scenarios.

The performance improvements observed validate the model's architecture and training strategies, making it a strong solution for this classification task.

D. Long Short-Term Memory Networks (LSTMs)

1) Model Architectures: Long Short-Term Memory Networks (LSTMs) networks were employed for both binary and multi-class classification tasks due to their superior ability to capture sequential dependencies and contextual information in textual data. The LSTM architectures utilized pre-trained Twitter 27B, 200-dimensional embeddings to provide rich word representations, enhancing the models' understanding of semantic relationships within the text. Below are the best-performing architectures for the binary and multi-class classifiers on this dataset.

a) Binary Classifier Architecture: The binary classifier's architecture comprises several key layers that work in unison to process and classify the input data. Class weights were integrated into the training process to address class imbalance, ensuring that minority classes (non-neutral classes) received higher importance during model training. Model summary is shown in Figure 7

The table II shows the best hyperparameters combination for binary model to classify neutral class 3 with other non-neutral classes.

The figure 8 shows the training performance of the best binary classifier. The training loss decreases steadily, and accuracy improves consistently, indicating effective learning.

Hyperparameter	Binary Model's Value	Multi-class Model's Value
LSTM Units	64	64
Dropout Rate	0.3	0.3
Learning Rate	0.0005	0.001
Batch Size	64	32

TABLE II
BEST HYPERPARAMETERS FOR LSTM MODELS

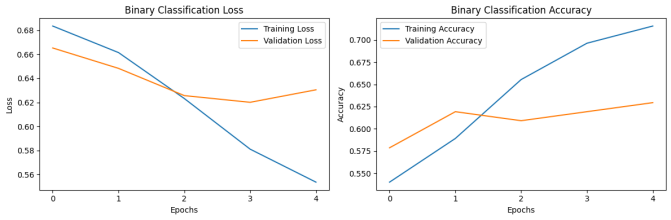

Fig. 8. Training and Validation Loss and Accuracy Plots for Binary Classifier - LSTM

However, validation loss initially decreases but begins to rise after the third epoch, while validation accuracy plateaus. This gap between training and validation suggests overfitting, which could be caused by the class imbalance in the dataset. Further regularization techniques and class balancing strategies may be necessary to improve generalization.

b) Multi-class Classifier Architecture: Figure 9 displays the architecture of a sequential model designed for non-neutral tweets classification with four output classes (1, 2, 4, and 5). The model begins with a TextVectorization layer to preprocess textual input, followed by an embedding layer to represent words as dense vectors. A Bidirectional LSTM layer is used processes the sequential data with 64 units each direction (forward and backward). To prevent over-fitting, dropout layers are applied, and two dense layers—one with 64 units for feature extraction and the other with 4 units for classification—finalize the model. The architecture is shown in figure 9.

Similar to the binary classifier, class weights were incorporated into the training process to address class imbalance, ensuring that minority classes (class 1 and 5) were appropriately represented during model training.

Model: "sequential_2"

Layer (type)	Output Shape	Param #
text_vectorization (TextVectorization)	(None, 200)	0
embedding_2 (Embedding)	(None, 200, 200)	200,200
bidirectional_2 (Bidirectional)	(None, 128)	135,680
dropout_4 (Dropout)	(None, 128)	0
dense_4 (Dense)	(None, 64)	8,256
dropout_5 (Dropout)	(None, 64)	0
dense_5 (Dense)	(None, 4)	260

Fig. 9. Multi-class Classifier Architecture - LSTM

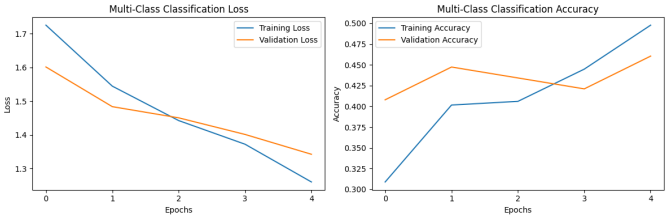

Fig. 10. Training and Validation Loss and Accuracy Plots for Multi-class Classifier - LSTM

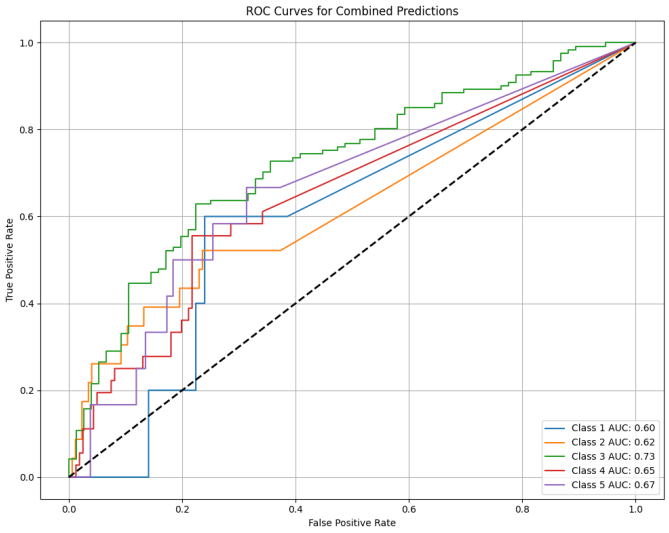

Fig. 11. ROC Curves for Validation Data: LSTM

The table II shows the best hyperparameters combination for multi-class model to classify four non-neutral classes.

The figure 10 illustrates the performance of the multi-class classification model. The training loss decreases steadily, while the validation loss follows a similar downward trend, indicating effective learning and improved generalization. Both training and validation accuracy improve consistently over epochs, with validation accuracy initially surpassing training accuracy, likely due to regularization and the small dataset size. By the fourth epoch, the accuracies align, showing balanced performance and minimal over-fitting. This suggests the model's configuration effectively handles the multi-class task.

2) Evaluation: The ROC curve in Figure 11 demonstrates the multi-class classification performance of the combined binary and multi-class LSTM model. Each class's Area Under the Curve (AUC) is reported, with the highest AUC achieved by Class 3 (0.73), indicating better separability for this dominant class. Other classes, particularly Classes 1 and 2, show lower AUC values (0.60 and 0.62), reflecting the challenges in predicting underrepresented categories.

The straight lines in the ROC curves for non-neutral classes likely result from the combined prediction approach. The binary model's filtering of neutral vs. non-neutral classes may lack nuance, making it challenging for the multi-class

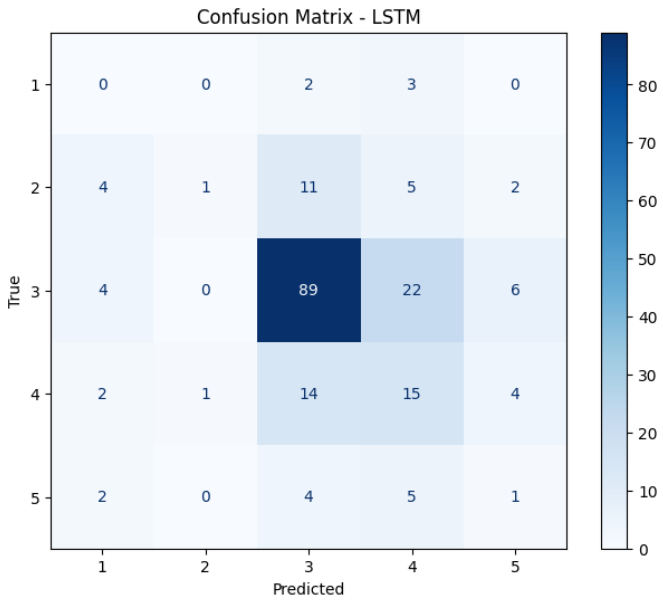

Fig. 12. Confusion Matrix for All 5 Classes - LSTM

model to distinguish effectively between non-neutral classes. This issue is further amplified by the imbalanced data, which limits the model's ability to learn robust representations for underrepresented classes.

Figure 12 presents the confusion matrix for the combined model. Class 3 (neutral) is the most accurately predicted, with 89 correct classifications, aligning with its dominance in the dataset. However, significant misclassifications are observed across other classes. For instance, Class 4 is often confused with Class 3, while predictions for Classes 1 and 5 remain sparse.

These results indicate that despite targeted augmentation and class weighting strategies, imbalanced data remains a challenge, affecting the model's ability to generalize for minority classes effectively. Further efforts in balancing data or refining the model architecture may enhance these outcomes.

3) Prediction on Test Set: The final predictions were generated using a two-step process: the binary model first classified inputs as neutral (label 3) or non-neutral. For non-neutral inputs, the multiclass model assigned specific sentiment labels (1, 2, 4, or 5). Neutral predictions were directly retained, and the non-neutral outputs were combined to produce the final predictions.

The LSTM model, when evaluated on the test dataset, achieved a Kaggle accuracy of 0.55214. This performance fell short of the baseline accuracy of 61%, highlighting the challenges faced by sequential-only architectures in addressing class imbalance and capturing complex contextual relationships in sentiment analysis tasks.

VIII. TRANSFORMER-BASED MODELS

Sentiment analysis is a crucial application in natural language processing (NLP) that aims to determine the sentiment or emotional tone expressed in textual data. In the context of self-driving cars, sentiment analysis can provide valuable insights into public opinion, user feedback, and market sentiment, making it a powerful tool for assessing the technology's acceptance and perception.

For this task, we utilized two state-of-the-art transformer-based models: BERT and RoBERTa. Both models are pre-trained on extensive text corpora and fine-tuned for specific tasks, including sentiment classification. Their ability to capture contextual relationships and meanings in text makes them highly effective for analyzing sentiment in complex datasets. Both require significant computational resources due to their large-scale architecture. BERT is trained on a dataset of 16 GB of uncompressed text from BookCorpus and English Wikipedia, while RoBERTa extends this by training on 160 GB of uncompressed text, including data from BookCorpus, English Wikipedia, CC-News, OpenWebText, and Stories datasets. These extensive datasets enable the models to learn rich contextual representations of language, making them ideal for tasks like sentiment analysis.

A. BERT

Developed by Google, BERT [2]is a groundbreaking model that introduced bidirectional pre-training, enabling it to understand the context of a word based on both its preceding and succeeding words. This capability is particularly beneficial for sentiment analysis, where small details in the context often determine the sentiment. In our implementation, we employed the `bert-base-uncased` model from Hugging Face's [16]Transformers library [17], fine-tuning it on our dataset of tweets related to self-driving cars using PyTorch.

B. RoBERTa

RoBERTa [3], developed by Facebook, builds upon BERT by optimizing its training methodology. It eliminates the Next Sentence Prediction (NSP) task used in BERT, utilizes larger batches, and trains on more extensive data for longer periods. These improvements enhance RoBERTa's performance and make it particularly robust for tasks like sentiment analysis. In this study, we employed the `roberta-base` model from Hugging Face's Transformers library, fine-tuning it in a manner similar to BERT to classify sentiments in the dataset related to self-driving cars.

C. Data

The analysis was conducted using the given three datasets: `train.csv`, `test.csv`, and `sample.csv`. These datasets are structured as follows:

- **`train.csv`:** Contains 981 entries comprising text data and their corresponding sentiment labels. The sentiment labels in this dataset range from 1 to 5, indicating varying levels of sentiment.
- **`test.csv`:** Consists of 979 entries with text data and unique IDs but does not include sentiment labels.
- **`sample.csv`:** Provides a small subset of the `test.csv` data, containing 50 entries with IDs and sentiment labels.

D. Data Cleaning

Effective sentiment analysis on social media content requires robust preprocessing to address the inherent noise within text data. Twitter data, in particular, presents unique challenges, as it often includes hashtags, mentions, slang, and URLs, which can contribute to noise and hinder analysis.

1) Challenges in Text Data: Text data from Twitter typically includes:

- **Hashtags:** These highlight trending topics but often comprise non-standard words or slang, which add little value to sentiment analysis.
- **Mentions:** Mentions (e.g., `@username`) are used to tag other users but do not carry sentiment-relevant information.
- **Stopwords:** Frequently occurring words (e.g., "and," "the") that are syntactically necessary but semantically redundant for analysis.
- **URLs:** Links to external resources, which are irrelevant for the textual sentiment analysis task but are prevalent in social media content.

2) Text Cleaning Process: To prepare the data for analysis, the following preprocessing steps were undertaken:

- **Mention Removal:** All words starting with `@` (e.g., `@TeslaMotors`) were removed, as these serve as user tags without contributing meaningful information to sentiment analysis.
- **URL Removal:** URLs were detected and eliminated from the text, as they do not provide value for sentiment classification.
- **Hashtag Removal:** The `#` symbol was stripped while preserving the subsequent words, ensuring meaningful terms are retained. For instance, `#driverless` was transformed into `driverless`.
- **Stopword Removal:** Common stopwords were filtered out to minimize noise and emphasize significant terms. This was achieved using the Natural Language Toolkit (NLTK) stopword list, which supports multiple languages. For example:

  "Uber is going to buy Driverless Cars" was reduced to "Uber buy Driverless Cars" by removing stopwords like "is," "going," and "to."
- **Lemmatization:** Words were normalized to their base forms (e.g., "drives," "driving," and "drove" became "drive"). Lemmatization, as opposed to stemming, ensures grammatically correct and semantically meaningful base forms.

3) The `clean_text_nltk` Function: The `clean_text_nltk` function integrates these preprocessing steps to transform raw text into a clean and analyzable format:

- **Removes:** URLs, hashtags, mentions, non-alphanumeric characters, and repeated characters.
- **Tokenizes:** Splits text into individual words for further processing.
- **Lemmatizes:** Converts words into their base forms.
- **Filters:** Excludes stopwords.

Additionally, the function normalizes Unicode characters to remove accents, eliminates non-ASCII characters, and converts text to lowercase to prevent case-sensitive mismatches. This comprehensive preprocessing ensures the text data is standardized and optimized for downstream sentiment analysis.

E. Data Splits

The dataset preparation involved strategic splitting to facilitate training, validation, and testing of the model:

- **Training and Validation Split:** The `train.csv` dataset, consisting of labeled entries, was divided into 80% training and 20% validation subsets. This split was used to train and fine-tune the model effectively.
- **Testing on Labeled Data:** The `sample.csv` dataset, which includes sentiment labels for 50 entries, was utilized to evaluate the model's performance on labeled test data. This provided direct insight into the model's predictive accuracy.
- **Additional Splits for Analysis:** The `test.csv` dataset, which lacks sentiment labels, was divided into two equal parts:
  - **Validation Test (50%):** Used as a proxy test set for intermediate evaluation and to analyze the model's consistency during development.
  - **Final Test (50%):** Reserved for robustness testing and final evaluation of the model.
- **Dummy Labels:** Since the `test.csv` dataset lacks ground truth labels, dummy labels were applied to both the Validation Test and Final Test subsets. This enabled analysis of the model's predictions and confidence scores in the absence of true labels.

F. Tokenization

For this analysis, we employed the pre-trained `bert-base-uncased` and `roberta-base` tokenizers from Hugging Face's library. These tokenizers play a crucial role in converting text into numerical representations that the models can effectively process.

1) Features of Tokenization:

- **Special Token Handling:**
  - `[CLS]`: Added at the beginning of each sentence to represent the overall sentence embedding.
  - `[SEP]`: Inserted to separate individual sentences within a sequence.
- **Uniform Input Length Management:**
  - **Padding:** Shorter texts are padded with `[PAD]` tokens to ensure consistent sequence lengths across the dataset.
  - **Truncation:** Longer texts are truncated to a specified `max_length` (set to 128 tokens in this study) to comply with model input size constraints.

This preprocessing step ensures compatibility with the BERT and RoBERTa models while preserving critical textual information. It facilitates effective learning by standardizing

input sequences and maintaining uniformity in text representation.

G. Model Training

The training process employed `BertForSequenceClassification` and `RobertaForSequenceClassification`, configured with 5 output labels to represent the sentiment classes. To ensure compatibility with the model, the sentiment labels in the dataset, originally indexed from 1 to 5, were adjusted to start from 0, resulting in the label set [0, 1, 2, 3, 4].

1) Training Parameters: The training parameters were carefully selected to optimize the fine-tuning of the pre-trained BERT model:

- **Learning Rate:** A small learning rate of 2×10^{-5} was used to ensure gradual updates during optimization and prevent drastic weight changes.
- **Early Stopping:** Early stopping was implemented to prevent overfitting.

2) Performance Metrics: Table III summarizes the performance metrics of the BERT and RoBERTa models on the sample dataset. The evaluation includes accuracy, weighted-average precision, recall, and F1-score, providing a comprehensive assessment of the models' ability to classify sentiment across all classes.

Method	Accuracy	Precision	Recall	F1-Score
BERT	0.66	0.52	0.66	0.56
RoBERTa	0.64	0.41	0.64	0.50

TABLE III
PERFORMANCE METRICS: BERT & ROBERTA

Class 2 demonstrated the best performance with the BERT model, achieving Precision: 67%, Recall: 97%, and F1-score: 79%. These results reflect the model's strong ability to identify and classify the dominant sentiment class with high confidence. Similarly, RoBERTa showed comparable trends, with Class 2 achieving Precision: 64%, Recall: 100%, and F1-score: 78%. However, both models exhibited lower metrics for other classes, likely due to class imbalance and insufficient support.

3) Training Visualizations: The training and validation loss curves for the BERT model are shown in Figure 13. These curves illustrate the learning process, where the gradual reduction in both training and validation losses signifies effective optimization.

H. Model Testing

In Figure 14, we present the model's performance on the sample dataset by comparing the true sentiment labels with the predicted labels. This figure displays 10 representative examples, showcasing the text of each data point, its corresponding true label, and the predicted label.

The true labels are derived from the ground-truth sentiment classifications in the dataset, while the predicted labels are generated by the BERT model. The model demonstrates its ability to capture dominant sentiment patterns, correctly classifying many entries. While certain entries show discrepancies

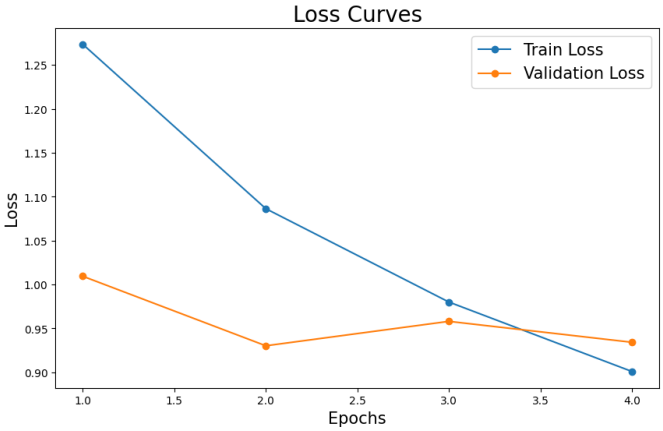

Fig. 13. Training and Validation Loss Curves for BERT Model

between the true and predicted labels, these cases provide valuable opportunities to further refine the model.

1) ROC Curve Analysis: Figure 15 illustrates the Receiver Operating Characteristic (ROC) curves for all sentiment classes in the sample dataset. The ROC curve plots the True Positive Rate (TPR) against the False Positive Rate (FPR) at various classification thresholds, providing a visual representation of the model's performance.

Similar trends are observed for the RoBERTa model, highlighting the effectiveness of transformer-based architectures in sentiment analysis. Both models show potential for improvement, particularly in addressing challenges like class imbalance.

I. Confidence Analysis

For datasets lacking true labels or ground truth, confidence distribution results are primarily illustrative. Figure 16 presents the confidence distribution across all classes for the validation and final test datasets. The x-axis represents the predicted confidence scores (ranging from 0 to 1), while the y-axis indicates the frequency of predictions. Each bar graph (histogram) is color-coded to represent a specific class.

In both datasets, Class 2 consistently stands out as the most confidently predicted class, with a strong peak in the 0.7–0.8 confidence score range. Conversely, low-confidence predictions are prevalent across Class 0, Class 1, Class 3, and Class 4, highlighting the model's uncertainty for these classes.

The confidence scores reflect the model's probabilistic output for each class, providing insight into how it distributes probabilities across the classes. Table IV summarizes the mean and standard deviation of confidence scores for BERT and RoBERTa models. The statistics reveal a significant imbalance, with Class 2 (corresponding to label 3 in the original data) achieving a mean confidence score of 67%. This highlights the model's strong preference and higher certainty for Class 2, suggesting either an imbalance in the dataset or more distinguishable features for this class.

Understanding and analyzing confidence scores is crucial, particularly when ground-truth labels are unavailable. They

Text	True Label	Predicted Label
two place id invest money could printing selfdriving car	2	2
awesome google driverless car help blind travel often	2	2
autonomous vehicle could reduce traffic fatality im	2	2
really good presentation jan becker bosch automated vehicle research autoauto check	1	2
ford revealed automated ford fusion hybrid vehicle pretty amazing fordtrends ford testiaa	2	2
yeah throwing would totally beta test autonomous car	3	2
teslamotors musk reluctant partner apple google android controlled autonomous smartcar would awesome	2	2
finished sfgtla drive itiaas lagtoc rush hour meeting caniaat wait autonomous google car	2	2
google autonomous car paid visit nvidia hq pretty cool technology person whoiaa	2	2
finally realistic timeline full autonomous car capability hat morganstanley autoforum	3	2

Fig. 14. True and Predicted Labels for Sample Dataset: BERT Model

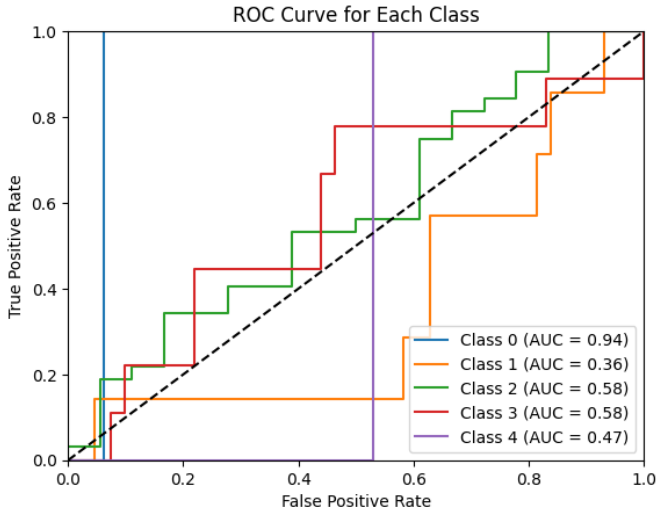

Fig. 15. ROC Curves for Sample Data: BERT Model

Method	Class	Mean Confidence	Std Deviation
BERT	0	0.025	0.013
	1	0.090	0.038
	2	0.674	0.151
	3	0.159	0.074
	4	0.051	0.029
RoBERTa	0	0.018	0.011
	1	0.080	0.053
	2	0.672	0.135
	3	0.177	0.051
	4	0.053	0.023

TABLE IV
CONFIDENCE SUMMARY: BERT & ROBERTA

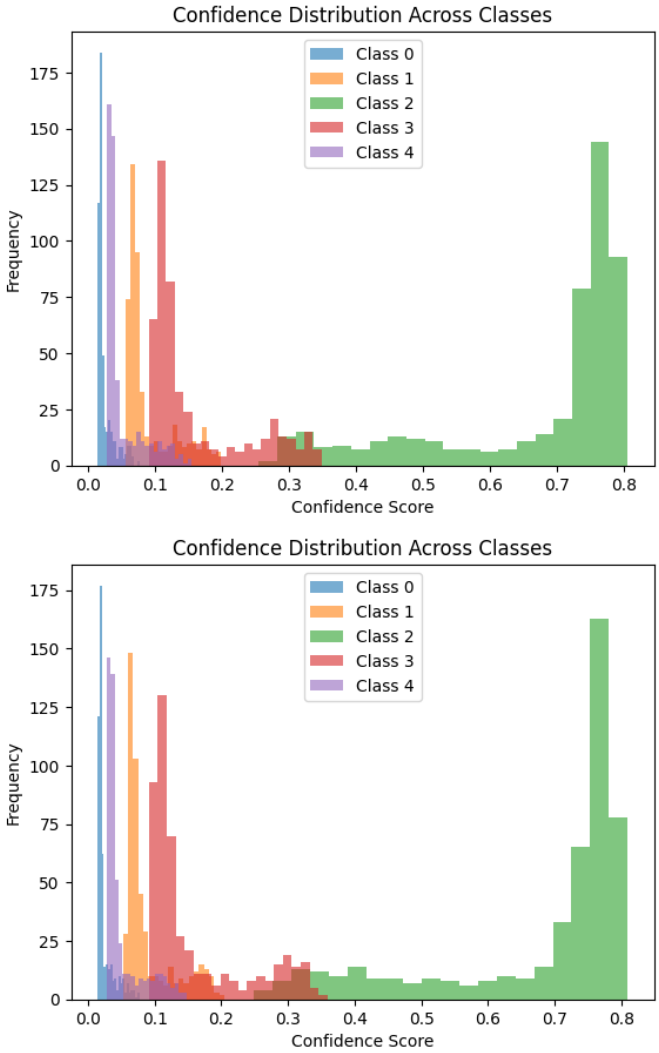

Fig. 16. Confidence Scores: Validation Test (Top) and Final Test (Bottom): BERT Model

help evaluate the model's reliability, interpretability, and trustworthiness.

IX. TRANSFER LEARNING

This section examines the RoBERTa-LSTM hybrid model, a state-of-the-art approach for sentiment analysis that combines the strengths of Transformer-based and sequence-based architectures.

Inspired by the article "RoBERTa-LSTM: A Hybrid Model for Sentiment Analysis With Transformer and Recurrent Neural Network" [18], the model integrates RoBERTa for gener-

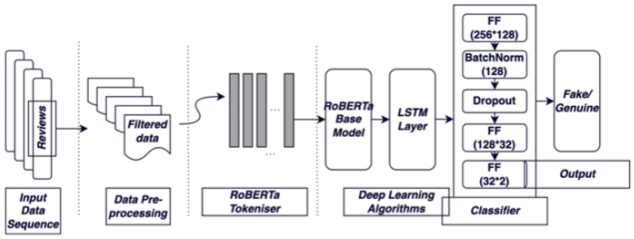

Fig. 17. Model Architecture: RoBERTa with LSTM [18]

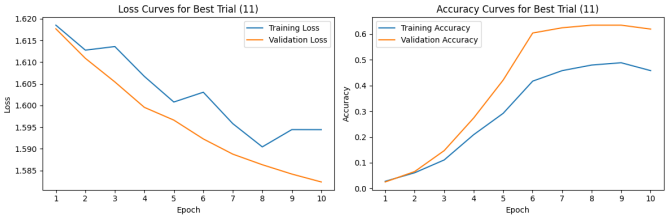

Fig. 18. Training and Validation Loss and Accuracy Curves for RoBERTa-LSTM Model

ating contextualized word embeddings with LSTM to capture long-distance dependencies in text. By leveraging the compact representation power of RoBERTa and the sequential learning capabilities of LSTM, this architecture addresses challenges such as lexical diversity, imbalanced datasets, and contextual dependencies. The use of class weights technique further enhances the model's performance, achieving state-of-the-art results across diverse sentiment analysis tasks.

A. Data Preparation and Handling Imbalance

The dataset underwent the text pre-processing and tokenization as done in Transformer-based Models section.

A thorough data cleaning process was applied, and the dataset was split into training and validation subsets in an 80:20 ratio using stratified sampling to maintain label distribution. Class imbalance was addressed by computing class weights which were incorporated into the $CrossEntropyLoss()$function, ensuring fair contributions from all classes during training [15].

B. Model Architecture

Figure 17 illustrates the RoBERTa-LSTM model proposed in the article [18].

The model begins with the RoBERTa tokenizer, which converts input text into token IDs and attention masks. These tokenized inputs are then processed by the RoBERTa base model combined with an LSTM layer, enabling the capture of both contextual and sequential information from the data.

The resulting features are passed through a classifier comprising fully connected layers, dropout, and activation functions, which ultimately produce logits for multi-class classification. In output layer, raw logits are converted into probability distributions via the softmax function during evaluation, enabling the calculation of performance metrics such as ROC/AUC.

C. Training

Hyper-parameter Optimization: Optuna [19] was used to optimize hyper-parameters, including LSTM units, dropout rate, learning rate, and batch size. The objective was to maximize validation accuracy efficiently. The study consisted of 20 trials, each running for a maximum of 10 epochs.

Early Stopping: To prevent over-fitting, early stopping was applied with a patience of 3 epochs, ensuring that training stopped if validation performance did not improve within the specified limit.

Optimization: The model was trained using the Adam optimizer, and a learning rate scheduler was used to adjust the learning rate dynamically based on validation performance. Training and validation loss and accuracy were tracked across epochs for each trial, with the best-performing model saved for further evaluation.

After training, trial 11 achieved the highest validation accuracy of 0.6345 with the following hyperparameters: LSTM units: 128, dropout rate: 0.3686, learning rate: 1.96×10^{-5}, and batch size: 16.

The plots in Figure 18 show the RoBERTa-LSTM model's training and validation performance during the best trial. Both losses decrease steadily, with validation loss dropping faster, indicating effective learning. Validation accuracy surpasses training accuracy by the 6th epoch and plateaus, highlighting the model's strong generalization. The combination of RoBERTa's embeddings and LSTM's sequential processing demonstrates effective sentiment classification with minimal over-fitting.

D. Evaluation on Validation Set

The performance of the RoBERTa-LSTM model on the validation set is summarized through a confusion matrix and ROC curves.

The confusion matrix, shown in Figure 19 highlights the model's predictive strengths and weaknesses across five sentiment labels. It performed well in identifying neutral sentiments, with 113 correctly predicted samples. However, there was notable misclassification in other classes, such as label 4 (positive sentiment) and label 5 (very positive), which were frequently misidentified as neutral. This indicates a bias toward the majority sentiment, likely caused by imbalanced data.

The ROC curves, shown in Figure 20, illustrate the model's ability to differentiate among sentiments. Class 0, corresponding to label 1, achieved the highest AUC (0.84), indicating strong separability despite having no correct classifications in the confusion matrix. This highlights the impact of heavily imbalanced data in the validation set, where small improvements in ranking probabilities can lead to a high AUC. Conversely, class 3, corresponding to label 4, had the lowest AUC (0.58), reflecting challenges in distinguishing this sentiment from others, even though it is dominant among non-neutral classes. The relatively lower AUC values across several classes further

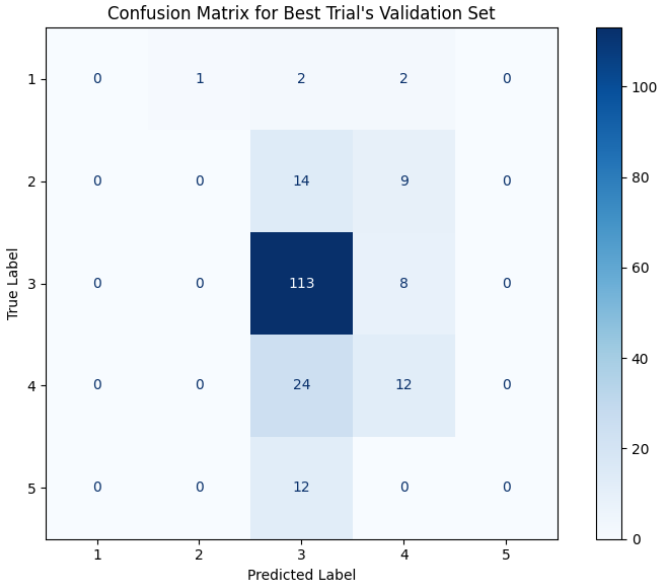

Fig. 19. Confusion Matrix for All 5 Classes - RoBERTa-LSTM

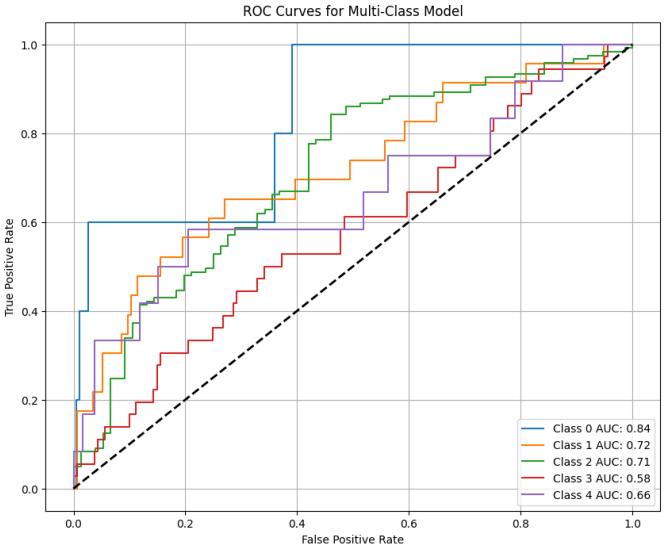

Fig. 20. ROC Curves for Validation Data: RoBERTa-LSTM Model

suggest that the model struggles with imbalanced data and overlapping feature distributions.

Overall, while the model demonstrates reasonable performance, further improvements could involve refining class weights or applying targeted augmentation for underrepresented labels to improve separability and reduce misclassifications.

E. Prediction on Test Set

The best-performing RoBERTa-LSTM model was evaluated on the test dataset to generate predictions for sentiment analysis. The trained model, loaded from the best trial's checkpoint, predicted sentiment labels by selecting the class with the highest probability. The predicted labels were mapped back to their original sentiment categories and combined with test IDs to create the submission file for the Kaggle competition.

The model achieved an accuracy of 0.65848, outperforming standalone LSTM models and the baseline accuracy of 61%. This improvement highlights the strength of transformer-based approaches like RoBERTa, which enhance the hybrid model's performance while requiring less manual feature engineering compared to traditional deep learning methods.

X. FUTURE WORK

In our paper, while we achieved promising results in addressing sentiment prediction challenges, there remain areas for improvement and further exploration. One key limitation lies in handling imbalanced datasets, where certain sentiment classes were underrepresented, potentially impacting the robustness of our model. Future work could focus on employing more advanced techniques to mitigate this imbalance, such as data augmentation (grouping sentiments), synthetic data generation, or ensemble learning approaches.

Additionally, our reliance on labeled data highlights the need to explore unsupervised and semi-supervised methods to predict sentiments without ground truth. Techniques like clustering and weak supervision could further enhance the model's ability to generalize to real-world scenarios where labeled data is scarce, complementing our exploration of pre-trained models such as BERT. While BERT and RoBERTa provided a strong baseline for sentiment prediction, future work could investigate alternative or supplementary methods, such as unsupervised clustering to identify sentiment groups or applying weakly supervised techniques using heuristics or external knowledge sources. These approaches could build on the strengths of pre-trained models and address gaps in handling unlabeled data, thereby extending the impact of this research and advancing sentiment analysis methodologies.

XI. CONCLUSION

In our evaluation using the Kaggle test set, traditional Machine Learning methods outperformed Deep Learning and hybrid approaches. SVM achieved the highest accuracy (0.701), followed by Logistic Regression (0.680) and Naive Bayes (0.670), with XGBoost lagging at 0.650. Deep Learning with LSTM performed poorly at 0.55, likely due to dataset limitations. A hybrid Roberta-LSTM model improved to 0.658 but fell short of SVM. The Deep Learning method using CNN achieved a perfect score on the validation set, indicating strong learning of the training patterns. However, its performance on the test set was ambiguous, suggesting potential overfitting to the validation data. The transformer-based model BERT achieved an accuracy of 0.66, while RoBERTa achieved 0.64 on the sample dataset. Both models demonstrated strong confidence in predicting Class 2 (label 3), with high mean confidence scores, indicating their ability to reliably classify this sentiment. These results suggest that while Deep Learning and hybrid models hold promise, simpler Machine Learning methods remain effective for this task, especially when resources are limited.

REFERENCES

[1] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, pp. 1735–80, 12 1997.

[2] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," *CoRR*, vol. abs/1810.04805, 2018. [Online]. Available: http://arxiv.org/abs/1810.04805

[3] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized BERT pretraining approach," *CoRR*, vol. abs/1907.11692, 2019. [Online]. Available: http://arxiv.org/abs/1907.11692

[4] B. Pang, L. Lee, and S. Vaithyanathan, "Thumbs up? sentiment classification using machine learning techniques," in *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP 2002)*. Association for Computational Linguistics, Jul. 2002, pp. 79–86. [Online]. Available: https://aclanthology.org/W02-1011

[5] N. Kalchbrenner, E. Grefenstette, and P. Blunsom, "A convolutional neural network for modelling sentences," in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, K. Toutanova and H. Wu, Eds. Baltimore, Maryland: Association for Computational Linguistics, Jun. 2014, pp. 655–665. [Online]. Available: https://aclanthology.org/P14-1062

[6] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[7] K. L. Tan, C. P. Lee, K. S. Muthu Anbananthen, and K. M. Lim, "Roberta-lstm: A hybrid model for sentiment analysis with transformers and recurrent neural network," *IEEE Access*, vol. 10, pp. 21 517–21 525, 2022.

[8] R. Sadiq and M. Khan, "Analyzing self-driving cars on twitter," *CoRR*, vol. abs/1804.04058, 2018. [Online]. Available: http://arxiv.org/abs/1804.04058

[9] A. Dutta and S. Das, *Tweets About Self-Driving Cars: Deep Sentiment Analysis Using Long Short-Term Memory Network (LSTM)*, 06 2020.

[10] D. Kim, "Sentiment analysis of self-driving cars using text mining," in *Kansei Engineering*, ser. AHFE (2024) International Conference, M. Nagamachi and S. Ishihara, Eds., vol. 145. AHFE International, USA, 2024. [Online]. Available: http://doi.org/10.54941/ahfe1005136

[11] F. Z, "Twitter sentiment analysis: Self-driving cars," https://kaggle.com/competitions/twitter-sentiment-analysis-self-driving-cars, 2017, kaggle.

[12] scikit-learn developers, "sklearn.feature_extraction.text.tfidfvectorizer," 2024, version 1.5, Accessed: 2024-01-11. [Online]. Available: https://scikit-learn.org/1.5/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html

[13] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543. [Online]. Available: http://www.aclweb.org/anthology/D14-1162

[14] F. Z, "Twitter sentiment analysis: Self-driving cars," https://www.kaggle.com/competitions/twitter-sentiment-analysis-self-driving-cars/leaderboard, 2017, kaggle.

[15] R. Prakash and K. Kumar, "Class weight technique for handling class imbalance," 07 2022.

[16] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. L. Scao, S. Gugger, M. Drame, Q. Lhoest, and A. Rush, "Transformers: State-of-the-art natural language processing," https://github.com/huggingface/transformers, pp. 38–45, 2020, accessed: 2023-12-10.

[17] H. Face, "Transformers: State-of-the-art machine learning for NLP, CV, and Speech," https://huggingface.co/transformers, 2023, accessed: 2023-12-10.

[18] K. L. Tan, C. P. Lee, K. S. M. Anbananthen, and K. M. Lim, "Roberta-lstm: A hybrid model for sentiment analysis with transformer and recurrent neural network," *IEEE Access*, vol. 10, pp. 21 517–21 525, 2022.

[19] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *The 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 2623–2631.