A Transformer Based BiLSTM Model with Multihead Attention for Emotion Classification in Code-Mixed Hinglish Text

Brajesh Kumar Khare
Harcourt Butler Technical University, Kanpur, India

Dr. Imran Khan*
Assistant Professor,
Harcourt Butler Technical University, Kanpur, India

*Corresponding Author

Abstract- In today's digital world, where most communication occurs on platforms like Twitter, Facebook, and WhatsApp, understanding emotions from user-generated content has become essential for businesses, researchers, and decision-makers. Detecting emotions in code mixed presents unique challenges due to its informal structure, frequent language switching, and lack of standardized grammar. This paper proposes a novel deep learning framework for emotion detection in real-world Hinglish social media texts, combining XLM-RoBERTa embeddings with a BiLSTM-multi-head attention architecture. Unlike conventional methods that utilize only token-level features from the final layer of transformer models, our approach dynamically learns optimal combinations of hidden states from XLM-R, refining feature representation to better align with emotional cues. The BiLSTM network captures sequential context, while multi-head attention highlights emotionally significant tokens across mixed-language expressions. Layer normalization and dropout mechanisms improve generalization, and hyperparameter tuning ensures robust performance. Experimental results on benchmark Hinglish emotion datasets demonstrate the effectiveness of the proposed approach, achieving an F1 score of 87.50%, outperforming standard transformer-based baselines. We also present ablation studies validating the contribution of each architectural component. The proposed model offers a domain-aware solution to emotion classification in low-resource, multilingual NLP settings.

Keywords- Emotion, Sentiment Analysis, Machine Learning, Word Embeddings, Code Mixed, Hinglish.

Abbreviation-

AI	Artificial Intelligence
NLP	Natural Language Processing
SA	Sentiment Analysis
XLM-R	Cross-lingual Language Model - RoBERTa
BiLSTM	Bidirectional Long Short-Term Memory
HLSTM	Hierarchical Long Short-Term Memory
ML	Machine Learning
XGBoost	eXtreme Gradient Boosting
SVC	Support Vector Classifier
SVM	Support Vector Machine
RBF	Radial Basis Function
MLP	Multi-Layer Perceptron
TF-IDF	Term Frequency-Inverse Document Frequency
mBERT	Multilingual Bidirectional Encoder Representations from Transformers
GPT-4	Generative Pre-trained Transformer 4
MURIL	Multilingual Representations for Indian Languages
ADASYN	Adaptive Synthetic Sampling
mT5	Multilingual Text-to-Text Transfer Transformer
HEP	Hinglish Emoji Prediction
HAN	Hierarchical Attention Network
RLHF	Reinforcement Learning from Human Feedback
LoRA	Low-Rank Adaptation

1 Introduction

Artificial Intelligence is a field in computer science focused on creating systems that can perform tasks typically requiring human thinking, such as learning, reasoning, and making decisions. A key part of AI is Natural Language Processing, which enables computers to understand and use human language effectively. NLP works on two main goals: understanding human communication and generating appropriate responses in human language. Understanding involves figuring out the meaning and intent behind words, while generating involves producing clear and relevant language. Understanding human language is more challenging due to its unclear nature. NLP is used in various applications like Speech Recognition [1], Document Summarization [2], Question Answering [3], Speech Synthesis [4], Machine Translation [5], Recommendation Systems [6] and more. Within NLP, sentiment analysis and text emotion detection are crucial for understanding human emotions in written content. Sentiment analysis [7] determines the overall sentiment (positive, negative, or neutral) of a text, while text emotion detection identifies specific emotions like happiness, sadness, anger, fear, disgust, surprise, or sometimes no emotion at all. Social media has become a powerful platform for expressing and sharing emotions, connecting people across the globe through text, images, and videos. These days, people prefer using Twitter and Facebook for news and updates instead of watching TV or reading newspapers. On Twitter, users share their views, feedback, and complaints about products. They also tweet about their favorite celebrities, athletes, and politicians. With the rise of online interactions, people often mix languages called code-mixing, where they switch between two or more languages in the same conversation or sentence. For example, "Mujhe coffee pasand hai, but only with sugar," where Hindi and English are mixed.

The combination of Hindi and English, commonly referred to as 'Hinglish,' is widely used on digital platforms, creating a need for systems that can effectively handle this unique style of communication [8]. While this linguistic flexibility facilitates seamless communication, it presents unique challenges for NLP tasks, especially emotion classification. One of the main difficulties in handling code-mixed Hinglish text is its complex linguistic nature. Hinglish often has irregular sentence structures, with Hindi words written in Roman script mixed with English words, leading to non-standard representation. For example, the Hindi word "mujhe" can appear in various Romanized spellings like "mujhe," "muje," or "mujhey," all of which mean "me." Such spelling variations add to the ambiguity and complicate the processes of tokenization and preprocessing. Another challenge is the seamless integration of grammar rules and idiomatic expressions from both languages. Words and phrases often carry contextual and cultural nuances that conventional NLP models struggle to interpret. Additionally, Hinglish text frequently incorporates informal language, slang, abbreviations, and emojis, all of which add significant complexity to semantic analysis.

Models like XLMR [9]and mBERT [10] which are based on transformers, have proven effective for handling multilingual and code-mixed text. They learn from large datasets and understand the complexities of different languages and word relationships, making them suitable for multilingual tasks. Additionally, they can handle the ambiguity and inconsistencies found in code-mixed texts, providing better generalization compared to traditional models. While transformer-based models excel at capturing semantic meanings, they might not fully capture sequential dependencies in the text, which is crucial for tasks like emotion classification. BiLSTM networks are adept at understanding long-term dependencies and context by analyzing the text in both directions, forward and backward [11]. Additionally, multi-head attention enables the model to concentrate on various sections of the input, making sure it highlights important features and relationships within the text, which is crucial for accurate emotion detection [12]. Previous models often show low accuracy because they do not effectively address the complexities introduced by code-mixed languages, which involve the blending of multiple languages. These models also struggle to capture the nuanced emotional expressions embedded in the text. Moreover, the diverse range of expressions and cultural contexts in code-mixed data further complicates emotion classification. The use of informal language, slang, and region-specific idioms adds additional layers of difficulty, making it challenging for traditional models to accurately identify emotions, which shows the need for more advanced models capable of handling the unique characteristics of code-mixed language data. In this paper, we propose a Weighted XLM-R Embeddings layer to compute a weighted sum of hidden layers from XLM-R for improved feature extraction. The BiLSTM-Attention model utilizes these embeddings, integrating a BiLSTM network with multi-head attention to enhance model performance. By employing this novel approach, we observe substantial enhancements in performance metrics.

Motivation- While our approach builds on established transformer and sequence modeling frameworks, we introduce several domain-aware enhancements specifically designed for code-mixed Hinglish text. These include:

- A layer-weighted aggregation mechanism that dynamically identifies and combines the most informative hidden states from XLM-RoBERTa;
- A multi-head attention layer over BiLSTM outputs, enabling the model to focus on emotionally significant tokens across language boundaries;
- A custom preprocessing pipeline targeting transliterated Hindi, emojis, and informal expressions common in real-world social media text.

Emotion detection from code-mixed Hinglish text has many applications, and several researchers are working in this domain. This paper is organized to explore research in the field of emotion detection from code-mixed Hinglish text. The paper is structured as: Section 2 provides a literature review, discussing emotion detection in code-mixed data using machine learning and deep learning approaches. Section 3 introduces recent models developed for this task, while Section 4 presents the proposed model. Section 5 details the experimental setup, followed by Section 6, which discusses the results. Finally, Section 7 concludes the paper with a summary of findings and directions for future work.

2 Literature Review

Emotion detection in code-mixed languages like Hinglish has gained attention as social media users increasingly mix languages. The challenge lies in the linguistic diversity and informal style of online text. Researchers have proposed various approaches using machine learning and deep learning to improve accuracy in detecting emotions. This review examines various studies on emotion detection in code-mixed Hinglish text, employing diverse models.

2.1 Emotion Detection Based on Machine Learning Methods

Machine learning models, such as support vector machine, naive bayes, random forest, and decision tree, are important tools for detecting emotions in text, including code-mixed Hinglish data. These models work well by finding patterns in the text using features like n-grams, TF-IDF, sentiment lexicons, and part-of-speech tagging. They have been used for tasks like sentiment and emotion analysis and are favored for their efficiency and good performance with smaller datasets. ML models require manual feature engineering, where researchers turn language insights into input features to help the models understand and classify emotions accurately. Even with newer techniques available, machine learning models are still relevant, especially when computer resources are limited or when simpler, easier-to-understand models are needed. Additionally, combining machine learning with other techniques can further improve the accuracy of emotion detection in multilingual and code-mixed text, highlighting the continued importance of machine learning in this area.

Here, we provide an overview of the research conducted by various researchers on code-mixed text using ML techniques.

Sultana et al. [13] focuses on SA in code-mixed Bangla-English content, addressing the challenge of limited annotated data for low-resource languages like Bangla. The study aims to collect code-mixed Bangla-English data and expand it using data augmentation techniques. The researchers compare four ML algorithms: support vector machine, decision tree, stochastic gradient descent, and random forest, using TF-IDF for feature extraction. Among these, random forest with TF-IDF achieved the highest accuracy of 83%, outperforming the other methods. This work contributes to improving SA capabilities for code-mixed content in low-resource language scenarios, particularly for Bangla-English text found on social media and online platforms. Rajalakshmi et al. [14] investigated SA in code-mixed Hinglish tweets, addressing the unique linguistic challenges that arise when multiple languages blend at sentence and word levels. Their approach involved various ML algorithms, such as decision tree, linear SVC, logistic regression, naive bayes, and XGBoost. To tackle specific issues like phonetic typing and multilingual words, they developed an ensemble-based classifier. Extensive experimentation revealed XGBoost as the top performing method, achieving an F1-score of 83.10% and outperforming previous approaches on the Hinglish dataset. This work represents a notable contribution to SA in code-mixed contexts, with potential to enhance text understanding and classification across multilingual settings. Singh et al. [15] explores the application of unsupervised cross-lingual embeddings to understand code-mixed social media text, focusing on SA of Hinglish Tweets (a combination of English and transliterated Hindi). The study compares baseline models using monolingual embeddings with two cross-lingual embedding approaches: a supervised classifier and a transfer learning method.

The findings reveal that cross-lingual embeddings outperform monolingual baselines, achieving an F1-score of 0.635 compared to 0.616, without requiring parallel data. Moreover, the cross-lingual approach proves effective in distant supervision scenarios, with transfer learning experiments yielding an F1-score of 0.556, nearly matching supervised settings. These results highlight the robustness and potential of cross-lingual embeddings in addressing code-mixed text understanding challenges. Hossain et al. [16] focuses on SA of code-mixed Bangla-English social media comments that incorporate emojis, reflecting the complexity of modern online communication. The study preprocessed a dataset of 2055 Facebook comments, extracted features using TF-IDF vectorizer and CountVectorizer, and applied nine ML algorithms for analysis. The support vector classifier emerged as the most effective model, achieving 85.7% accuracy and an 85.0% F1 score. These results underscore the importance of including emoji-based features in SA of code-mixed data. The preprocessing phase involved cleaning the data and converting emojis to unicode short names, prepare ng the diverse dataset for comprehensive analysis.

Some other papers related to code-mixed data, where machine learning approaches were used, are discussed in Table 1

Table 1: Machine learning approaches for code-mixed data.

Reference	Task	Dataset	Feature selection	Classifier	Accuracy
Srinivasan et al. [17]	Sentiment analysis	15,744 Tamil-English Youtube comments	Tf-Idf	RF, LR, XGBoost, SVM and Naïve Bayes	Achieved 0.81 F1 score using RF
Swami et al. [18]	Sentiment Analysis of twitter data	31962 code-mixed tweets	TF-IDF	LR, DT, RF, Naïve bayes, SVM	RF achieved maximum accuracy 96.57%
Khandelwal et al. [19]	Gender Prediction	4015 tweets	Reference Tokens, Top Hashtags, Bag-of-words	SVM with RBF, Random Forest, and Naive Bayes classifier.	SVM model with RBF achieved an accuracy of 89.5%.
Bohra et al. [20]	Hate speech detection	4575 code-mixed tweets	Character N-Grams, Word N-Grams.	SVM and Random Forest	SVM achieved highest accuracy as 71.7%
Utsav J. et.al. [21]	Stance Detection	3545 Demonetization tweets and 4219 article 370 tweets	Word N-grams, stance indicative tokens, character N-grams.	RF, SVM, XGBoost, RBF	Achieved highest accuracy 69.1 % using XGBoost
Vijay et al. [22]	Corpus Creation and Emotion Prediction	2698 emotional code-mixed tweets	Character N-Grams and Word N-Grams (n varies from 1 to 3)	SVM classifier	Achieved the best accuracy of 58.2% using SVM.
Rahman et al. [23]	Cyberbullying Detection	8400 annotated comments	uni-grams, n-grams	NB, SVM, XGBoost, random forest, ensemble model	Ensemble model achieves highest accuracy 60.09%
Mohapatra et al. [24]	Hate Speech Detection	27,162 posts	word unigram, bigram, TF-IDF, word2vec	SVM, NB, RF	SVM with word2vec achieves highest accuracy with a 73% F1-score
Mishra et al. [25]	Code-Mixed SA	18461 hinglish sentences	TF-IDF and GloVe	SVM, Voting Classifier, MLP	Achieved an F1 score of 0.569 using voting classifier

2.2 Emotion Detection Based on Deep Learning Methods

Deep learning models have become essential in text emotion detection, particularly for code-mixed languages like Hinglish, which combines Hindi and English. These models, including RNNs, CNNs, and transformers, automatically learn patterns from vast amounts of data, eliminating the need for extensive feature engineering required by traditional machine learning methods. This ability allows DL models to capture complex relationships and contextual nuances in the text, significantly enhancing the accuracy of emotion detection in informal social media language. Models like LSTM networks and transformers such as BERT have performed well in identifying emotions in these texts. Even with these advancements, researchers continue to explore hybrid methods that blend DL and traditional ML techniques to address the challenges posed by linguistic diversity and informal language on

social media. As online communication continues to evolve, deep learning's role in accurately detecting emotions in code-mixed Hinglish remains an important area of research. Here, we present an overview of research on code-mixed text using deep learning techniques-

Thara et al. [26] examines the challenges of processing code-mixed text in social media, focusing on offensive language identification and SA in Malayalam-English mixed content. The study explores three key aspects: the impact of word embedding methods, the performance of various DL algorithms (including unidirectional, bidirectional, hybrid, and transformer models), and the effectiveness of selective translation, transliteration, and hyperparameter optimization. The proposed framework achieved impressive F1-scores of 0.76 and 0.99 for the FIRE 2020 and EACL 2021 datasets, respectively. A thorough error analysis provides valuable insights, and the approach outperforms existing benchmarks for Malayalam-English code-mixed messages, contributing to societal benefit through improved understanding of multilingual social media content. Ghosh et al. [27] addresses the growing need for sentiment and emotion analysis in code-mixed content, particularly focusing on Hindi-English (Hinglish) texts. The authors create an emotion-annotated Hinglish dataset by enhancing the existing SentiMix dataset. The authors introduce an innovative approach that combines sentiment detection and emotion recognition into a single multitask framework. This model is built on transformer architecture and leverages the pre-trained XLMR model. By fine-tuning XLMR with task-specific data, they leverage transfer learning to enhance overall performance. The multitask approach outperforms existing single-task and multitask baselines significantly, demonstrating that emotion recognition as an auxiliary task improves sentiment detection in a multitask setting. Notably, these results were achieved without ensemble techniques, suggesting a practical and efficient approach for real-world NLP applications. Wadhawan et al. [28] concentrate on identifying emotions in social media content written in Hinglish, a mix of Hindi and English commonly used in tweets. They introduce a new dataset of Hinglish tweets labeled for emotion detection and investigate several DL methods to tackle this challenge. Their approach incorporates bilingual word embeddings, along with transformer models. The research evaluates and contrasts the effectiveness of various DL architectures, including CNNs, LSTMs, bidirectional LSTMs (with and without attention), and advanced transformer models such as BERT, RoBERTa, and ALBERT. Among these, the BERT-based transformer model demonstrates superior performance, achieving the highest accuracy of 71.43%. This work contributes to the growing field of emotion detection in multilingual social media contexts, which has wide-ranging applications in consumer understanding, psychology, human-computer interaction, and smart system design. Das et al. [29] explore SA in the context of social media, specifically Hinglish tweets. Their research employs a labeled Hinglish dataset for emotion detection and applies various DL techniques. The study utilizes multilingual word embeddings from FastText methods and transformer-based models to analyze emotions in these code-mixed tweets. The researchers experiment with different DL architectures, including CNN, LSTM, and Bi-LSTM models. CNN model outperforms the others, achieving an accuracy of 75.25%. This research contributes valuable insights to the expanding field of multilingual SA in social media, with potential applications across diverse domains such as human-computer interaction, consumer behavior analysis, psychological studies, and smart system development. Sasidhar et al. [30] focuses on emotion analysis in Hinglish text. They developed a dataset comprising 12,000 code-mixed texts gathered from various sources, annotating them with three emotion categories: happy, anger, and sad. Their approach utilizes a pretrained bilingual model to generate feature vectors, which are then used in deep neural network classifiers. Among the various models tested, the CNN-BiLSTM architecture demonstrated superior performance, achieving a classification accuracy of 83.21%. This work contributes to the growing field of emotion analysis in multilingual contexts, particularly for code-mixed language data. Mursalin et al. [31] tackled the growing challenge of emotion detection in Bengali-English code-mixed text, driven by the increasing use of social media among Bengali speakers. Their study aimed to classify emotions into various categories. To address the scarcity of resources, they developed a corpus of 10,221 Bengali-English code-mixed sentences. The researchers experimented with various word embedding techniques, including Word2Vec, FastText, and Keras Embedding Layer, and applied different ML and DL algorithms. Through comparative analysis, they found that their proposed method combining Word2Vec and BiLSTM outperformed other models, achieving the highest accuracy of 76.1%. This research advances emotion recognition in low-resource languages and code-mixed data, with potential applications across diverse fields such as e-commerce, healthcare, suicide prevention, and crime detection. Abdullah et al. [32] focuses on emotion detection in Roman Urdu (RU) and English (EN) code-mixed text, an area that has received little attention despite its prevalence on social media. The study addresses limitations in existing research by creating a new corpus of 20,000 purely code-mixed sentences from 400,000 social media posts. The researchers developed comprehensive annotation guidelines and created the RU-EN-Emotion corpus, annotating sentences as Neutral or Emotion, with further emotion classification for the latter. They conducted 102 experiments comparing six classical ML and six DL techniques. Results showed that CNN with GloVe embeddings performed best, and their two-level classification

approach using the new corpus proved more effective than previous methods. Jyoti et al. [33] focused on SA of Dravidian language social media posts, specifically those combining Kannada, Malayalam, or Tamil with English. They developed a model using a dense neural network combined with character-level TF-IDF features to categorize posts into five sentiment classes. This approach yielded promising results, achieving weighted F1-scores of 0.61, 0.72, and 0.60 for Kannada-English, Malayalam-English, and Tamil-English posts respectively. Their research addresses a significant gap in the field, as most existing work on social media SA has centered on English-language content. By tackling the understudied area of Dravidian code-mixed languages, this study makes a valuable contribution to expanding the scope of SA in multilingual contexts. Mishra et al. [34] present a comprehensive report on the SemEval-2020 Task 9, which focused on SA of code-mixed tweets, also known as SentiMix 2020. This task addressed the growing importance of analyzing sentiment in multilingual social media content, particularly in code-mixed languages. The researchers developed and introduced two significant new corpora for this task: a Hinglish dataset comprising 20,000 tweets, and a Spanglish (Spanish-English) dataset with 19,000 tweets. These datasets were meticulously annotated, featuring word-level language identification to distinguish between the mixed languages, as well as sentence-level sentiment labels categorized as Positive, Negative, or Neutral. The competition garnered substantial interest from the research community, attracting a total of 89 submissions. The Hinglish contest saw participation from 61 teams, while the Spanglish contest involved 28 teams, indicating a strong interest in both language pairs. The results were promising, with the top-performing systems achieving impressive F1 scores: 75.0% for the Hinglish task and 80.6% for the Spanglish task.

Some other papers related to code-mixed data, where deep learning approaches were used, are discussed in Table 2

Table 2: Deep learning approaches for code-mixed data.

Reference	Task	Dataset	Classifier	Accuracy
Joshi et al. [35]	SA of Hindi-English Text	3879 sentences	Subword-LSTM and Char-LSTM	Subword-LSTM achieved highest accuracy 69.7%.
Shashi Shekhar et al. [36]	Hatred and trolling detection system	Total 5905 social media sentences	HLSTM, SVM, RF	HLSTM achieved best 97.49% accuracy.
Singh et al. [37]	Predicting multi-label emojis, sentiment and emotions.	20,000 tweets	XLMR, mBERT, CM-RFT	Best model CM-RFT achieved 75.81% accuracy for emoji, 82.35% for sentiment and 63.73% for emotion detection.
Pillai et al. [38]	Sentiment and offensive text detection.	50K Tweets	Feature fusion + HAN	Achieved highest accuracy of 95.6%
Younas et al. [39]	SA	Dataset contains 20,735 tweets	mBERT and XLM-RoBERTa	XLM-R achieved better accuracy with F1 score of 71%.
Sane et al. [40]	Humor Detection	Around 200k tweets	CNN and BiLSTM (with and without Attention)	Attention based BiLSTM model achieved an accuracy of 73.6%
Jadon et al. [41]	SA of Hinglish text	18000 tweets	Hybrid LSTM-GRU model	Achieved 96.76% accuracy and 98.49% Precision.
Himabindu et al. [42]	Emoji Prediction in code-mixed Hinglish language.	HEP dataset contains 86,072 tweets.	BiLSTM with self-attention and random forest	Achieved Accuracy: 61.14%, Precision: 0.66, Recall: 0.59, F1 Score: 0.59
Mursalin et al. [43]	Classifying emotions from Bengali-English	10,221 sentences	BiLSTM with Word2Vec embedding	Achieved 76.1% accuracy.
Yann et al. [44]	SA of Malay-English mixed language	7,907 samples of Malay-English comments	BiLSTM, LSTM, Naïve Bayes, Logistic Regression	biLSTM with tuned hyper-parameters achieved the highest accuracy of 76.6% and a macro F1-score of 69.6%

3 Recent Models for Hinglish Text Emotion Detection - Emotion detection in code-mixed Hinglish presents unique challenges due to linguistic complexity, informal structure, and the presence of transliterated words. Various advance models have been explored to address these challenges effectively, which are shown in Table 3.

Table 3: Recent models for text emotion detection

Model Type	Examples	Characteristics	Advantages	Limitations
Transformer-Based Models [45]	XLM-R, mBERT, mT5, MuRIL, HingBERT	- Context-aware embeddings - pre-trained on multilingual corpora	- High accuracy in code-mixed text - Handles Hinglish variations well	- Requires significant computational power - Needs fine-tuning on Hinglish data
Multimodal Learning [46]	CLIP, MMBERT	- Processes text and images/videos - Enhances emotion detection via multimodal cues	- Useful for social media content - Captures richer context	- Requires labeled multimodal datasets - Computationally expensive
Contrastive Learning [47]	SimCSE, Sentence-BERT, mT5 + Contrastive Loss	- Learns better text representations - Effective for emotion similarity	- Improves embedding quality - Works well with low-resource languages	- Needs large positive-negative sample pairs - Longer training time
Meta-Learning [48]	BiLSTM + Random Forest,	- Combines transformer embeddings with traditional ML classifiers	- Enhances model generalization - Improves explainability	- Complex pipeline - Requires fine-tuning
Prompt-Based Learning [49]	GPT-4, Llama 3, Bloom	- Uses prompts to extract emotions - Few-shot learning capabilities	- Works without fine-tuning - Handles Hinglish variations naturally	- High inference cost - Requires prompt engineering
Low-Resource Adaptation [50]	LoRA, Adapter Layers	- Reduces number of trainable parameters	- Faster training - Requires less data	- May still need fine-tuning on Hinglish datasets
Graph Neural Networks (GNNs) [51]	GraphSAGE, GAT	- Captures relational context in conversations - Effective for emotion flow detection	- Works well with social media conversations - Learns semantic relationships	- Requires graph structure - Computationally complex
Reinforcement Learning for NLP [52]	RLHF	- Model fine-tuned using human feedback	- More robust emotion detection	-Computationally expensive

4 Proposed Framework

The proposed framework presents an emotion detection pipeline with three key components. The first component is data preprocessing, which begins with ADASYN oversampling to balance the dataset. This is followed by text cleaning, including tasks such as removing URLs and usernames, converting text to lowercase, eliminating stopwords, and applying stemming. The next component, the XLM-R embedding layer, processes the cleaned text through its hidden layers to produce weighted embeddings. The architecture then splits into two parts: a BiLSTM with multi-head attention and a processing block featuring normalization and ReLU activation. The final component, the output block, consists of dropout and dense layers with softmax activation, which predicts the probability scores for seven emotion classes. The proposed framework is presented in Figure 1.

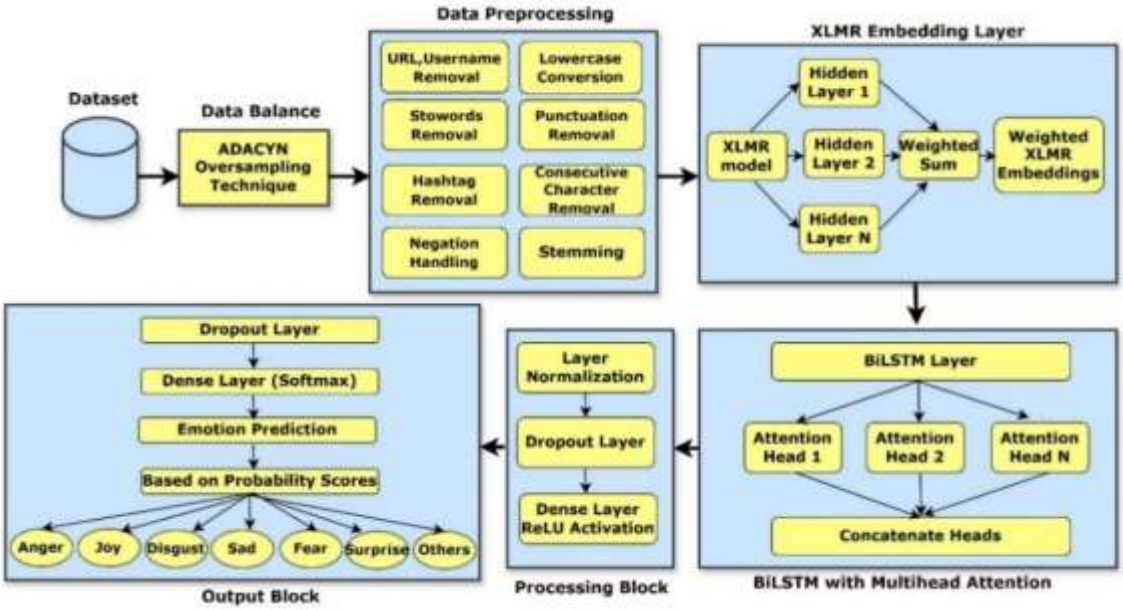

Figure 1: Proposed framework

4.1 Dataset

We used Task 9 of the SemEval 2020 shared task dataset, which is separated into three distinct subsets: training, testing, and validation, consisting of 14,000, 3,000, and 3,000 tweets, respectively [53]. On average, the sentence length is 134.9 characters. In the training set, the average sentence length is 136.9 characters, with a vocabulary size of 60,115. The validation set has an average sentence length of 127.7 characters and a vocabulary size of 19,499. Meanwhile, the test set's average sentence length is 129.9 characters, with a vocabulary size of 19,331 words.
The classification of various emotions in the dataset is presented in Table 4.

Table 4: Emotion distribution

Emotion	Train	Test	Validation
Anger	2095(14.96%)	680(22.67%)	415(13.83%)
Disgust	1048(7.49%)	105(3.5%)	148(4.93%)
Fear	56(0.4%)	13(0.43%)	4(0.13%)
Joy	3893(27.81%)	1008(33.6%)	973(32.43%)
Sadness	856(6.11%)	122(4.07%)	307(10.23%)
Surprise	51(0.36%)	7(0.23%)	6(0.2%)
Others	6001(42.86%)	1065(35.5%)	1048(34.93%)
Total	14000	3000	3000

4.2 Data Balancing

The training dataset contains 14,000 instances and includes seven classes: Anger, Joy, Disgust, Sadness, Fear, Surprise and Others. The resulting dataset was highly imbalanced, with 6,001 tweets in the 'Others' class and only 56 and 51 tweets in the 'Fear' and 'Surprise' classes, respectively. To address this imbalance, we employed ADACYN oversampling technique [54].

The total number of synthetic samples required for a minority class c is:

$$G_c = \alpha(N_{max} - N_c) \quad (25)$$

Where G_c is the total number of synthetic samples for class c, α is a user-defined sampling intensity factor, N_{max} is the number of instances in the majority class and N_c is the number of instances in the minority class.

For each minority instance X_i, a synthetic sample is generated by selecting a random minority neighbor X_j-

$$X_{new} = X_i + \mu.(X_j - X_i) \quad (26)$$

Where $\mu \sim U(0,1)$ is a random number drawn from a uniform distribution. The balance dataset is shown in table 5.

Table 5: Emotion distribution after balancing train dataset

Class	Anger	Joy	Disgust	Sadness	Fear	Surprise	Others	Total
Number of instances	5452	6114	5941	5985	5977	6021	6001	41491

After balancing the training dataset, the resulting balanced dataset is shown in Figure 2.

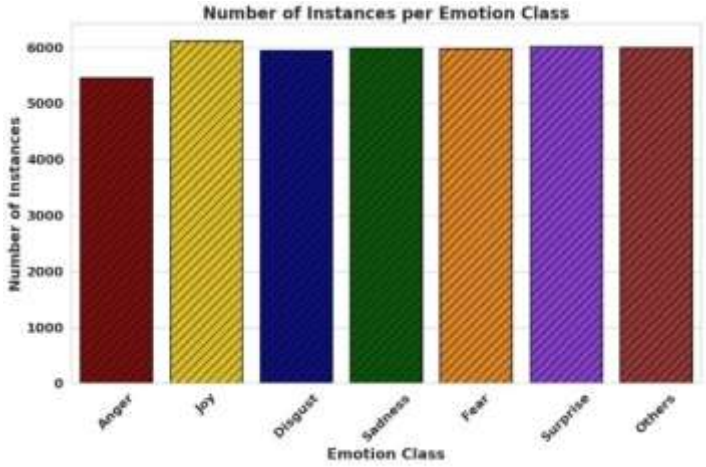

Figure 2: Balanced dataset

4.3 Data Preprocessing

Our dataset is almost evenly distributed. Next, we move on to the data preprocessing phase. During this stage, we implement several techniques, including the removal of URLs, usernames, and the "#" symbol from tweets, converting text to lowercase, reducing consecutive characters, eliminating Hinglish Stopwords, addressing negations, removing all punctuation marks, and applying stemming[55].

The raw input text $\mathcal{D}$ is split into a sequence of tokens using a tokenization function T

$$\mathbb{T} = T(\mathcal{D}) = w_1, w_2, \ldots . w_N \quad (27)$$

Where $\mathcal{D}$ is the input document, T is the tokenization function and $\mathbb{T}$ is the tokenized sequence consisting of N tokens.

A filtering function $\mathbb{F}$ removes unwanted symbols, numbers, and links:

$$\mathbb{T}' = \mathbb{F}(\mathbb{T}) = \{w_i | w_i \notin \Omega, w_i \notin \mathbb{R}^+, w_i \notin \mathbb{U}, w_i \notin \mathbb{M}\} \quad (28)$$

Where Ω is the set of punctuation and special characters, $\mathbb{R}^+$ represents numerical tokens, $\mathbb{U}$ represents URLs, $\mathbb{M}$ represents mentions and $\mathbb{T}'$ is the cleaned token sequence.

To reduce noise, stopwords $\mathbb{T}'' = \{w_i | w_i \notin \mathbb{S}\}$ (29)

Where $\mathbb{S}$ is the predefined Stopword set, $\mathbb{T}''$ is the token sequence after Stopword removal.

Then, a transformation function £ converts all tokens to lowercase: $\mathbb{T}''' = £(\mathbb{T}'')$ (30)

To normalize tokens into their root forms, lemmatization ℓ is applied- $\mathbb{T}^* = \ell(\mathbb{T}''')$ (31)

Where ℓ applies lemmatization and $\mathbb{T}^*$ is the final preprocessed token sequence.

The final preprocessed text is converted into a numerical vector representation X

$$X = \phi(\mathbb{T}^*) \in \mathbb{R}^d \quad (32)$$

Where ϕ is the feature extraction function and X is the numerical representation of the text.
The complete text preprocessing pipeline is mathematically represented as:

$$X = \phi(\ell\left(£\left(₹(T(\mathcal{D}))\right)\right)) \quad (33)$$

4.4 XLMR Embedding Layer

XLM-R is a multilingual transformer-based model designed to handle multiple languages and trained on a large amount of multilingual data. It is particularly suitable for code-mixed data (like Hinglish) due to its ability to capture linguistic patterns across languages. The model generates embeddings by processing input text through multiple hidden layers (transformer layers) that capture complex semantic and syntactic information from the text. Each hidden layer in XLM-R captures different levels of linguistic information. Early layers focus on basic linguistic structures, while deeper layers capture more complex and abstract information. Instead of using just one hidden layer's output, a weighted sum technique is applied across all layers. This means each hidden layer's output is assigned a specific weight, and these weighted outputs are summed together to form a final embedding. Weighting layers allows the model to dynamically emphasize layers that provide the most relevant features for emotion detection in Hinglish text. The result of the weighted sum operation is a single embedding vector, referred to as Weighted XLM-R Embeddings. This vector is contextually rich and tailored to the input text's structure and semantics, containing information relevant to both languages (English and Hindi). These embeddings are then passed to subsequent layers (BiLSTM and attention layers) for further processing and classification into specific emotions. The XLM-R embedding layer block generates contextual embeddings for code-mixed Hinglish tweets by using a weighted sum of hidden layers from the pretrained XLM-R model. These weighted layers are then summed to create a robust composite embedding.
These equations mathematically represent how the XLM-R embedding layer generates robust contextual embeddings using a weighted sum technique.
Let the input text $T = \{t_1, t_2, \ldots\ldots, t_n\}$ where n is the number of tokens after tokenization using the XLM-R tokenizer. The model processes T through L transformer layers, and the hidden state at layer l is represented as:

$$H^{(l)} = \{h_1^{(l)}, h_2^{(l)}, \ldots\ldots, h_n^{(l)}\},\ l\ \in \{1,2, \ldots\ldots, L\} \quad (34)$$

Where $H^{(l)}\ \in\ \mathbb{R}^{nXd}$ and d is the embedding dimension of the hidden states.
To emphasize the most relevant layers for the task, we assign a learnable scalar weight $\alpha^{(l)}$ to each hidden layer l. These weights are normalized using the softmax function:

$$\alpha^{(l)} = \frac{\exp(\beta^{(l)})}{\sum_{j=1}^{L}\exp(\beta^{(j)})},\ l\ \in \{1,2, \ldots\ldots, L\} \quad (35)$$

Where $\beta^{(l)}$ are the learnable parameters associated with each layer.
The weighted sum of hidden states across all L layers produce the final contextual embedding vector for the input sequence:

$$E_{XLMR} = \sum_{l=1}^{L}\ \alpha^{(l)}.H^{(l)} \quad (36)$$

Where $E_{XLMR}\ \in \mathbb{R}^{nXd}$
To represent the entire input sequence as a single embedding vector e_{XLMR}, we apply a pooling operation over the token embeddings:

$$e_{XLMR} = Pooling\ (E_{XLMR}) \quad (37)$$

Where $e_{XLMR}\ \in \mathbb{R}^d$ is the final sequence-level embedding.
The final embedding e_{XLMR} is rich in contextual information, as it combines linguistic structures from multiple layers, emphasizing task-specific relevance through the learned weights $\alpha^{(l)}$. This embedding is further processed by downstream layers (BiLSTM with attention) for classification into emotion categories.
Figure 3 shows XLM-R architecture processing flow, starting with Hinglish text input, passing through tokenization, embedding, and position encoding. The tokens are processed through layers for syntactic, semantic, and contextual features, incorporating Self Attention, Feed Forward, and Normalize operations. The weighted XLMR embeddings are then input into a BiLSTM with an Attention Layer for final processing.

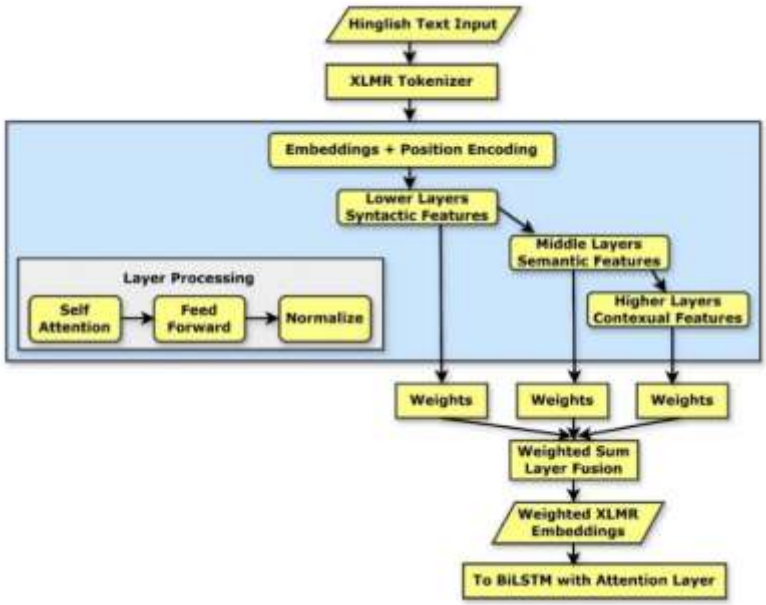

Figure 3: XLMR processing pipeline

4.5 BiLSTM with Multihead Attention

In the BiLSTM with Multihead Attention block, we refine the XLM-R embeddings by combining sequential modeling with attention mechanisms to better capture emotional cues in code-mixed Hinglish tweets. First, we pass the embeddings through a BiLSTM layer, which processes them in both forward and backward directions, enabling the model to learn context from the entire sequence and create richer representations of each token. To improve stability and training efficiency, we apply Layer Normalization to the BiLSTM outputs. Next, we use a Multihead Attention layer with several attention heads, allowing the model to focus on multiple relationships between tokens across the sequence and capture both local and global dependencies. We add a dropout layer to reduce overfitting, and finally, a fully connected layer maps the processed information to the emotion classes, enabling the model to make accurate predictions.

The input to the BiLSTM model is the embedding matrix obtained from XLM-R, denoted as:

$$E = \{e_1, e_2, e_3, \dots \dots . e_T\} \quad (38)$$

Where T is the sequence length and $e_t \in \mathbb{R}^d$, $t = \{1,2,3, \dots . , T\}$

The BiLSTM processes the input embeddings in both forward and backward directions:

$$\overrightarrow{h_t} = LSTM_{fwd}(e_t, \overrightarrow{h_{t-1}}) \quad (39)$$

$$\overleftarrow{h_t} = LSTM_{bwd}(e_t, \overleftarrow{h_{t+1}}) \quad (40)$$

The concatenated output of the BiLSTM is:

$$H = \{h_1, h_2, h_3, \dots \dots \dots . h_T\}, h_t = [\overrightarrow{h_t}, \overleftarrow{h_t}] \quad (41)$$

To stabilize training, layer normalization is applied to the BiLSTM outputs:

$$H_{norm} = LayerNorm(H) \quad (42)$$

The Multihead Attention mechanism computes attention scores for h heads. Each head computes scaled dot-product attention:

$$Attention(Q, K, V) = Softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)\mathrm{V} \quad (43)$$

Where $Q, K \; and \; V$ are the query, key, and value matrices derived from H_{norm} and d_k is the dimensionality of the keys.

The outputs of all heads are concatenated and passed through a dense layer:

$$A = concat(head_1, head_2, \dots \dots \dots , head_h)W_o \quad (44)$$

Where W_o is a learnable weight matrix.

Dropout is applied to the Multihead Attention output:

$$A_{drop} = Dropout(A) \quad (45)$$

Finally, a fully connected layer maps the processed sequence representation to the emotion classes:

$$y = softmax\ (A_{drop}W + b) \quad (46)$$

Where W and b are the weights and biases of the fully connected layer, and y is the output probability distribution over the emotion classes.

4.6 Processing Block

The Processing Block is a crucial component in our architecture designed to manage and transform the input data into a format suitable for model training and inference. This block performs several preprocessing steps that prepare raw textual data, including tokenization, padding, and batch formation, before feeding it into the embedding and subsequent model layers. The first step in the Processing Block is tokenization, which involves converting the code-mixed Hinglish tweets into a sequence of tokens that the XLM-R model can interpret. Here, we use the XLM-R tokenizer, which breaks down each tweet into sub-word tokens and assigns each a unique identifier. Additionally, special tokens like [CLS] (indicating the start of the sentence) and [SEP] (denoting the end) are added to each tweet to mark sentence boundaries. Tweets in the dataset can vary significantly in length. To ensure that each tweet is represented as a fixed-length sequence, the Processing Block applies padding and truncation. Alongside tokenization, attention masks are generated to indicate which tokens should be attended to during processing. Tokens representing actual words have a mask value of 1, while padding tokens are marked as 0. This helps the model to focus only on meaningful tokens and ignore padded elements during attention-based computations. After tokenization, padding, and mask creation, the data is split into batches for efficient model training. Batching allows multiple tweets to be processed simultaneously, accelerating training and optimizing GPU utilization. For each batch, the tokenized inputs, attention masks, and corresponding labels (emotion classes) are grouped together. The tokenized and formatted data is then passed to a DataLoader for efficient handling during training and evaluation. The DataLoader shuffles the data, creating batches with uniform dimensions, and ensures smooth data flow to the model.

Each tweet $T = \{T_1, T_2, \dots\dots\dots, T_N)$ is tokenized using the XLM-R tokenizer into a sequence of sub-word tokens:

$$T_t = \{[CLS], e_1 e_2, \dots\dots\dots e_{n_t}, [SEP]\}, t = 1,2, \dots\dots, N \quad (47)$$

Where N is the total number of tweets, n_t is the number of tokens in tweet T_t and $[CLS]$, $[SEP]$ are special tokens marking the beginning and end of the sequence, respectively.

To standardize input dimensions, all tokenized sequences are either padded or truncated to a fixed length L:

$$T_t^{padded} = \begin{cases} \{[CLS], e_1 e_2, \dots\dots\dots e_{n_t}, [SEP], 0,0,0, \dots\dots 0\} & n_t < L \\ \{[CLS], e_1 e_2, \dots\dots\dots e_{L-2}, [SEP]\} & n_t \geq L \end{cases} \quad (48)$$

Where 0 represents the padding token.

Attention masks M_t are generated for each padded sequence to indicate which tokens should be attended to during computations:

$$M_t = \{m_1 m_2, \dots\dots\dots m_L\}, m_i = \begin{cases} 1, if\ token\ e_i \neq 0 \\ 0, if\ token\ e_i = 0 \end{cases} \quad (49)$$

The tokenized, padded sequences, attention masks, and emotion labels y_t are grouped into batches for efficient processing:

$$Batch_k = \{(T_t^{padded}, M_t, y_t) \mid t \in \delta_k\}\ k = 1,2, \dots\dots\dots, B \quad (50)$$

Where δ_k represents the indices of tweets in the k batch and B is the total number of batches.

The DataLoader organizes batches into the dataset for training and inference:

$$\lambda = \{Batch_1 Batch_2 Batch_3, \dots\dots Batch_6\} \quad (51)$$

4.7 Output Block

The output block is the final part of our emotion detection system that determines the emotional class of the input. Starting with a dropout layer that prevents overfitting by dropping random neurons, the data then passes through a dense layer with softmax activation. This layer transforms the input into probability scores between 0 and 1 for each emotion category, with all scores summing to 1. For example, given a text input, the model computes probability scores like P(joy) = 0.75, P(surprise) = 0.15, P(anger) = 0.05, and so on. The emotion with the highest probability score becomes the final prediction - so in this case, the model would classify the text as expressing "joy" since it has the highest probability of 0.75. These probability scores not only provide the final classification but also indicate the model's confidence in its prediction. The model covers seven emotion categories: anger, joy, disgust, sadness, fear, surprise, and others, where "others" captures emotional expressions that don't clearly fit into the main categories.

Figure 4 shows the complete data preprocessing and model prediction pipeline for code-mixed Hinglish tweets. The process begins with input tweets passing through a Processing Block, where the XLM-R tokenizer processes the text, adds special tokens, and converts them into token IDs. The next step is the Padding & Truncation block, which adjusts text length by padding shorter texts and truncating longer ones based on a defined maximum length. An Attention Mask is then created, marking actual tokens as 1 and padding tokens as 0. During the Batch Formation stage, inputs, masks, and labels are grouped together into a data loader. Finally, the Output Block processes the data through dropout and dense layers, applying softmax activation to generate emotion probabilities.

Figure 4: Model prediction pipeline

Finally, the Probability Distribution $P(y = c|x) = \frac{\exp(o_c)}{\sum_{j=1}^{k} \exp(o_j)}, c \in \{1,2, \dots, k\}$ (52)

Where $P(y = c|x)$ represents the probability of the input tweet x belonging to class c.

Final prediction- $c^* = argmax\ (c \in 1,2,, \dots . k)\ P(y = c|x)$ (53)

Where c^* is the predicted emotion class.

Algorithm 1 summarizes the key steps of our proposed framework.

Algorithm 1: Emotion Detection (XLM-R + BiLSTM-Attention)

1. Dataset $D = \{(T_1, y_1), (T_2, y_2) \dots . (T_n, y_n)\}$ where T_i is the tweet and y_i is the corresponding emotion label.
2. Initialize: Pre-trained XLM-R model M_{XLMR} with parameters θ_{XLMR}
3. BiLSTM parameters $\{W_{weights}, W_{biases}\}$ attention heads h and hyperparameters, $\{\mu, \upsilon, \lambda_{epochs}\}$ (μ is the learning rate, υ is the batch size and λ_{epochs} is the maximum number of epochs)
4. Define emotion classes $c = \{anger, joy, disgust, sad, fear, surprise, others\}$
5. Training Loop: For each epoch $e = 1\ to\ \lambda_{epochs}$
6. For each batch $B_j \subseteq D$:
   - Preprocess tweet $T_i \in B_j$ using function $f(T_i) = preprocess\ (clean(T_i))$
   - Extract XLM-R embeddings: $E_j = M_{XLMR}(B_j)$
   - Compute weighted feature vectors: $W_j = \sum_{i=1}^{\upsilon} W_i \,.\, E_i$
   - Apply BiLSTM to weighted features: $H_j = BILSTM(W_j)$
   - Apply multi-head attention: $A_j = multiheadattention(H_j, h)$
   - Normalize the output: $N_j = LayerNorm(A_j)$
   - Apply dropout regularization: $D_j = Dropout(N_j)$
   - Compute logits: $L_j = Dense(D_j, softmax)$
   - Calculate cross-entropy loss: $L = crossentropyloss(L_j, y_j)$
   - Update parameters $\theta_{XLMR}, W_{weights}, W_{biases}$ using gradient descent: $\theta_{XLMR}, W_{weights}, W_{biases} \leftarrow \theta_{XLMR}, W_{weights}, W_{biases} - \mu . \nabla \mathrm{L}$
7. **Evaluate** the model on the validation set D_{val} after each epoch.
8. **Return:** Trained model M_{final}
9. Evaluate M_{final} on the test set D_{test}
10. Output predictions $\widehat{y_i}$, accuracy θ_{acc}, precision α_{pre}, recall β_{rec} F1 score φ_{F1}

5 Experimental Section

The experimental setup operates on a Windows 10 system with an Intel Core i7-8700K CPU and an NVIDIA GeForce GTX 1080 GPU. The software environment comprises Anaconda3 and Python 1.7.0. Model training is optimized using the Adam optimizer with cross-entropy as the loss function. Table 6 provides detailed parameter settings.

Table 6: Model Parameters

Parameters	Description	Values (During Grid Search)	Best Value (After Grid Search)
hidden_dim	The size of the hidden state in the LSTM layer.	[128, 256, 512, 1024]	256
lstm_layers	Number of layers in the LSTM.	[1, 2, 3]	2
attention_heads	The count of attention heads in the multi-head attention layer	[2, 4, 8]	4
dropout_rate	Dropout rate for regularization.	[0.1, 0.2, 0.3, 0.4]	0.2
batch_size	Batch size used in the DataLoader for training and evaluation.	[16, 32, 64, 128]	32
learning_rate	Learning rate for the optimizer.	[1e-5, 1e-4, 1e-3]	1e-4
num_epochs	Number of epochs for training the model.		10

6 Result and Discussion

We compared our proposed model with mainstream text classification models, including XLMR [9], mBERT [10], IndicBERT [56], Mistral 7B [57], MURIL [58], and mT5 [59]. Four metrics are used to evaluate the classification performance: accuracy, precision, recall, and F1 score [60]. Table 7 presents the comparison results across various models.

Table 7: Model performance comparison

Model	Accuracy	Precision	Recall	F1 Score
XLMR[9]	66.40%	68.71%	64.40%	67.50%
mBERT[10]	60.03%	62.06%	60.03%	61.92%
IndicBERT[61]	64.83%	66.35%	65.83%	65.40%
Mistral 7B[57]	77.45%	74.56%	72.12%	78.36%
MURIL[58]	63.37%	65.43%	64.37%	65.22%
mT5 [59]	26.03%	49.22%	26.03%	32.69%
Proposed Model	**87.50%**	**88.28%**	**87.19%**	**87.77%**

The table shows that the proposed model achieved the highest F1 score of 83.51%, while Mistral 7B achieved 78.36%. Confusion matrix[62] of proposed model is shown in figure 5.

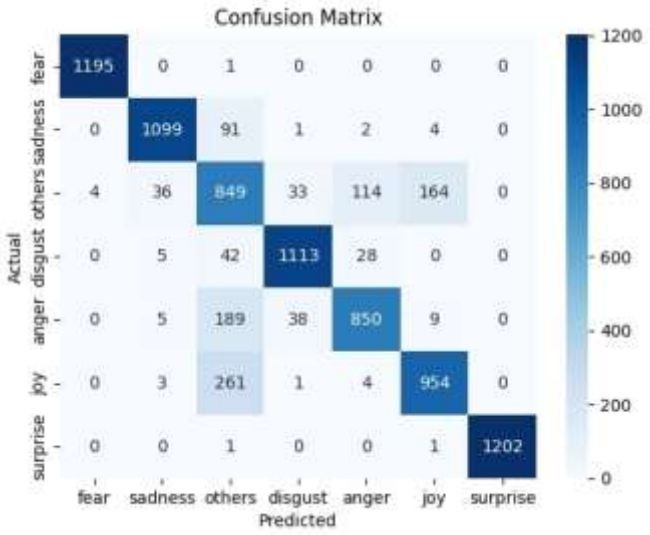

Figure 5: Confusion matrix

Accuracy and Loss graph for training and test data is shown in figure 6.

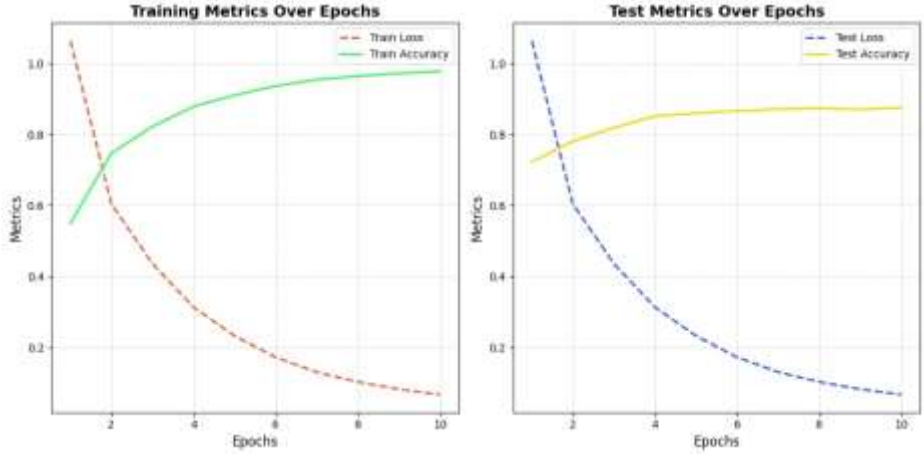

Figure 6: Accuracy and loss graph

Figure 7 shows a detailed comparison graph showcasing the performance of the different models. This graph shows how each model performed, helping to understand their effectiveness and suitability for the task.

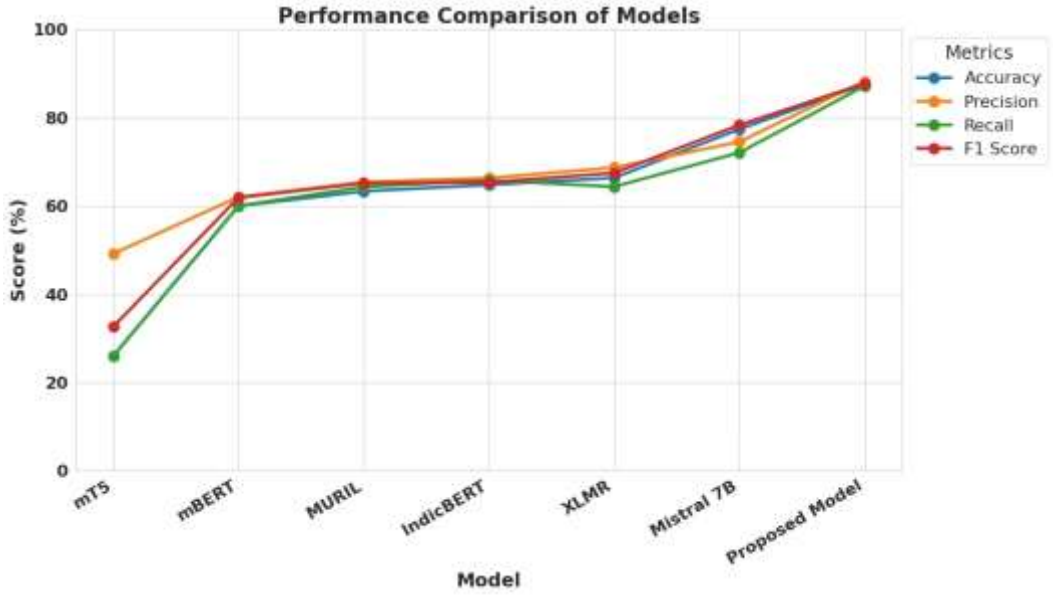

Figure 7: Comparison graph

6.1 Ablation study- To evaluate the contribution of each architectural component in our proposed model, we conducted an ablation study by systematically removing or modifying individual parts of the model while keeping other settings constant. The goal was to assess how each design choice affects the final performance on emotion detection in code-mixed Hinglish text.

We started with a baseline model using only the final layer of XLM-RoBERTa for classification and gradually added components such as BiLSTM, multi-head attention, weighted hidden layer aggregation, and hyperparameter tuning.

Table 8 presents an ablation study evaluating the contribution of each architectural component to overall performance. Starting with a baseline model using only the final layer of XLM-R, we gradually add proposed components and observe consistent improvements in both accuracy and F1 score. The most significant gains come from incorporating multi-head attention and weighted hidden layer aggregation, highlighting their effectiveness in capturing emotion cues in code-mixed text. Finally, hyperparameter tuning further boosts performance to our best result of 83.51% F1 score, validating the value of each enhancement.

Table 8: Ablation study

Model variant	Description	Accuracy	Precision	Recall	F1 score
Baseline (XLM-R Only)	Use only the last layer output from XLM-RoBERTa	80.40%	78%	79%	78%
+ Weighted sum of Hidden Layers	Learn importance of intermediate XLM-R layers	82.5%	80.43%	81.55%	80.34%
+ BiLSTM	Add Bidirectional LSTM over XLM-R outputs	84.59%	82.23%	83.67%	82.22%
+ Multi-head Attention	Apply attention to BiLSTM outputs	85.75%	85.15%	84%	84.90%
+ Hyperparameter Tuning	Optimized learning rate, dropout, batch size	87.51%	88.28%	87.19%	87.77%

From the results, it's evident that each component contributes positively to the final performance:
Weighted hidden layer aggregation provides the most notable gain, increasing accuracy to 82.5%, demonstrating the effectiveness of dynamically learning which layers contribute most to emotion detection.

Adding BiLSTM improves contextual modeling and boosts accuracy score to 84.59%, indicating the benefit of sequential modeling.

Multi-head attention further enhances performance to 85.75%, highlighting its ability to focus on emotionally significant tokens in code-mixed text.
Finally, hyperparameter tuning leads to the best result of 87.51%, validating the overall value of each enhancement.
These findings confirm that the proposed design choices are not arbitrary but play a meaningful role in improving emotion classification accuracy in complex, multilingual Hinglish texts.

6.2 Error Analysis- To gain deeper insights into the model's behavior, we conducted a qualitative error analysis, focusing on misclassified samples from the test set. Our goal was to understand:

- Why certain tweets were misclassified
- What linguistic characteristics made them challenging
- How our model could be improved based on these findings

We selected a randomly chosen misclassified instances and analyzed them manually. The following patterns emerged as common sources of confusion:

a) Ambiguous Emotion Labels- A few cases lacked clear emotional cues, leading to ambiguous labeling even for humans, which affected model performance.

b) Class-wise Confusion Insights- From the confusion matrix, we observed the following key misclassification trends:

Predicted vs True	Anger	Joy	Disgust	Sadness	Fear	Surprise	Others
Anger	-	↑	↔	↑	↔	↔	↑
Joy	↑	-	↑	↑↑	↔	↑	↑
Disgust	↓	↑	-	↓	↔	↔	↑
Sadness	↑	↑	↑	-	↑	↑	↑
Fear	↔	↑	↔	↑	-	↑	↑
Surprise	↔	↑	↑	↑	↑	-	↑
Others	↓	↓	↓	↓	↓	↓	-

Where ↑↑= high confusion, ↑=Moderate confusion, ↔=Mild/occasional confusion, ↓=Often misclassified as this class and -=True class (diagonal)

c) Linguistic Characteristics of Difficult Samples- We categorized misclassified samples by linguistic features, identifying several contributing factors:

Feature	Description	Impact on model
Transliteration Errors	boleto→bola tha	Reduced accuracy due to inconsistent tokenization
Informal Spelling	Accha →Achha	Affected embedding quality
Switching Between Hindi and English Verbs	She helped me but main confuse tha (English verbs helped, karne, Hindi verb tha)	Confused syntactic parsing
Informal Spelling Variants	Non-standard spellings and phonetic transliterations (e.g., "msg" for message, "aap kesi ho?" instead of "aap kaise ho?").	Increases ambiguity and reduces model accuracy due to inconsistent input representations.
Lack of Standardization	No consistent spelling, grammar, or structure	Makes preprocessing and modeling highly challenging
Politeness Markers	Use of expressions like "plz", "aap", "ji"	Nuanced cues may be ignored by models unaware of politeness levels

7 Conclusion and Future Work

This study proposed a deep learning framework for emotion classification in code-mixed Hinglish text, leveraging XLM-R embeddings and a BiLSTM model enhanced with a multi-head attention mechanism. The inclusion of a Weighted XLM-R Embeddings layer allowed for optimized feature extraction, while the BiLSTM-Attention architecture, equipped with layer normalization and dropout, contributed to improved generalization and performance. Extensive experimentation, including hyperparameter tuning through grid search, demonstrated the robustness of our approach, achieving a notable accuracy of 87.50%. The results validate the proposed methodology's effectiveness in addressing the challenges of emotion detection in code-mixed Hinglish text, offering a promising solution for multilingual text.

While the proposed model demonstrates strong performance, there is room for further improvement. Future research could explore the integration of other transformer-based embeddings such as mT5 or Mistral to enhance feature representation further. Incorporating transfer learning with larger and more diverse datasets could improve the model's ability to adapt to real-world scenarios. Additionally, expanding the study to include multimodal data, such as images or audio alongside text, could offer deeper insights into emotions. Advanced optimization techniques, such as Hyperband or Bayesian optimization, can be applied for more efficient hyperparameter tuning. Finally, deploying the model in real-world applications like social media monitoring or sentiment analysis pipelines will allow for practical evaluation and further refinements based on user feedback and performance in diverse settings.

Declarations

Conflict of Interest- The authors declare no conflicts of interest. All authors have reviewed and approved the manuscript's content, with no financial interests to disclose. This submission is confirmed as original work and is not under consideration for publication elsewhere.

Funding- No funding.

Data Availability Statement (DAS)- In this study, we employed Task 9 from the SemEval 2020 shared task dataset, which is partitioned into three distinct subsets: a training set comprising 14,000 tweets, a testing set of 3,000 tweets, and a validation set containing 3,000 tweets [53].

Reference

[1] Win, Khin Myat Nwe, Zar Hnin, and Y. M. K. K. Thaw. "Review and perspectives of natural language processing for speech recognition." *Int. J. All Res. Writ* 1.10 (2020): 112-115.

[2] Pandiaraja, P., et al. "An analysis of document summarization for educational data classification using NLP with machine learning techniques." *International Conference on Computing in Engineering & Technology*. Singapore: Springer Nature Singapore, 2022.

[3] Rogers, Anna, Matt Gardner, and Isabelle Augenstein. "Qa dataset explosion: A taxonomy of nlp resources for question answering and reading comprehension." *ACM Computing Surveys* 55.10 (2023): 1-45.

[4] Adam, Edriss Eisa Babikir. "Deep learning-based NLP techniques in text to speech synthesis for communication recognition." *Journal of Soft Computing Paradigm (JSCP)* 2.04 (2020): 209-215.

[5] Jiang, Kai, and Xi Lu. "Natural language processing and its applications in machine translation: a diachronic review." *2020 IEEE 3rd International Conference of Safe Production and Informatization (IICSPI)*. IEEE, 2020.

[6] Sharma, Akhilesh Kumar, et al. "An efficient approach of product recommendation system using NLP technique." *Materials Today: Proceedings* 80 (2023): 3730-3743.

[7] Jim, Jamin Rahman, et al. "Recent advancements and challenges of NLP-based sentiment analysis: A state-of-the-art review." *Natural Language Processing Journal* (2024): 100059.

[8] Mangla, Ankur, Rakesh Kumar Bansal, and Savina Bansal. "Code-mixing and code-switching on social media text: A brief survey." *2023 IEEE International Conference on Computer Vision and Machine Intelligence (CVMI)*. IEEE, 2023.

[9] Goyal, Naman, et al. "Larger-scale transformers for multilingual masked language modeling." *arXiv preprint arXiv:2105.00572* (2021).

[10] Blazhuk, V., O. Mazurets, and O. Zalutska. "An Approach to Using the mBERT Deep Learning Neural Network Model for Identifying Emotional Components and Communication Intentions." (2024).

[11] Liu, Gang, and Jiabao Guo. "Bidirectional LSTM with attention mechanism and convolutional layer for text classification." *Neurocomputing* 337 (2019): 325-338.

[12] Li, Jian, et al. "On the diversity of multi-head attention." *Neurocomputing* 454 (2021): 14-24.

[13] Sultana, Babe, and Khondaker A. Mamun. "Enhancing bangla-english code-mixed sentiment analysis with cross-lingual word replacement and data augmentation." *2024 6th International Conference on Electrical Engineering and Information & Communication Technology (ICEEICT)*. IEEE, 2024.

[14] Rajalakshmi, Ratnavel, et al. "Sentimental analysis of code-mixed Hindi language." *Congress on Intelligent Systems: Proceedings of CIS 2021, Volume 2*. Singapore: Springer Nature Singapore, 2022.

[15] Singh, Pranaydeep, and Els Lefever. "Sentiment analysis for hinglish code-mixed tweets by means of cross-lingual word embeddings." *Proceedings of the 4th Workshop on Computational Approaches to Code Switching*. 2020.

[16] HOSSAIN, FAHIMA, NUSRAT JAHAN, and JOYREYA ABADIN. "SENTIMENT ANALYSIS OF BANGLA-ENGLISH CODE-MIXED AND TRANSLITERATED SOCIAL MEDIA COMMENTS USING MACHINE LEARNING."

[17] Srinivasan, R., and C. N. Subalalitha. "Sentimental analysis from imbalanced code-mixed data using machine learning approaches." *Distributed and Parallel Databases* 41.1 (2023): 37-52.

[18] Swamy, Sowmya, Jyoti Kundale, and Dipti Jadhav. "Sentiment analysis of multilingual mixed-code, twitter data using machine learning approach." *International Conference on Innovative Computing and Communications: Proceedings of ICICC 2021, Volume 2*. Springer Singapore, 2022.

[19] Khandelwal, Ankush, et al. "Gender prediction in english-hindi code-mixed social media content: Corpus and baseline system." *Computación y Sistemas* 22.4 (2018): 1241-1247.

[20] Bohra, Aditya, et al. "A dataset of Hindi-English code-mixed social media text for hate speech detection." *Proceedings of the second workshop on computational modeling of people's opinions, personality, and emotions in social media*. 2018.

[21] Utsav, Jethva, et al. "Stance detection in hindi-english code-mixed data." *Proceedings of the 7th ACM IKDD CoDS and 25th COMAD*. 2020. 359-360.

[22] Vijay, Deepanshu, et al. "Corpus creation and emotion prediction for Hindi-English code-mixed social media text." *Proceedings of the 2018 conference of the North American chapter of the Association for Computational Linguistics: student research workshop*. 2018.

[23] Rahman-Laskar, Sahinur, et al. "Cyberbullying Detection in a Multi-classification Codemixed Dataset." *Computación y Sistemas* 28.3 (2024): 1091-1113.

[24] Mohapatra, Sudhir Kumar, et al. "Automatic hate speech detection in English-Odia code mixed social media data using machine learning techniques." *Applied Sciences* 11.18 (2021): 8575.

[25] Mishra, Pruthwik, Prathyusha Danda, and Pranav Dhakras. "Code-mixed sentiment analysis using machine learning and neural network approaches." *arXiv preprint arXiv:1808.03299* (2018).

[26] Thara, S., and Prabaharan Poornachandran. "Social media text analytics of Malayalam–English code-mixed using deep learning." *Journal of big Data* 9.1 (2022): 45.

[27] Ghosh, Soumitra, et al. "Multitasking of sentiment detection and emotion recognition in code-mixed Hinglish data." *Knowledge-Based Systems* 260 (2023): 110182.

[28] Wadhawan, Anshul, and Akshita Aggarwal. "Towards emotion recognition in Hindi-English code-mixed data: A transformer-based approach." *arXiv preprint arXiv:2102.09943* (2021).

[29] Das, Shubham, and Tanya Singh. "Sentiment recognition of hinglish code-mixed data using deep learning models-based approach." *2023 13th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*. IEEE, 2023.

[30] Sasidhar, T. Tulasi, B. Premjith, and K. P. Soman. "Emotion detection in hinglish (hindi+ english) code-mixed social media text." *Procedia Computer Science* 171 (2020): 1346-1352.

[31] Mursalin, Golam Sarwar Md, Suborno Deb Bappon, and Muhammad Ibrahim Khan. "A deep learning approach for recognizing textual emotion from bengali-english code-mixed data." *2022 25th International Conference on Computer and Information Technology (ICCIT)*. IEEE, 2022.

[32] Ilyas, Abdullah, Khurram Shahzad, and Muhammad Kamran Malik. "Emotion detection in code-mixed roman urdu-english text." *ACM Transactions on Asian and Low-Resource Language Information Processing* 22.2 (2023): 1-28.

[33] Kumari, Jyoti, and Abhinav Kumar. "A deep neural network-based model for the sentiment analysis of dravidian code-mixed social media posts." *management* 5 (2021): 6.

[34] Mishra, Shreyash, et al. "Overview of memotion 3: Sentiment and emotion analysis of codemixed hinglish memes." *arXiv preprint arXiv:2309.06517* (2023).

[35] Joshi, Aditya, et al. "Towards sub-word level compositions for sentiment analysis of hindi-english code-mixed text." *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. 2016.

[36] Shekhar, Shashi, et al. "Hatred and trolling detection transliteration framework using hierarchical LSTM in code-mixed social media text." *Complex & Intelligent Systems* 9.3 (2023): 2813-2826.

[37] Singh, Gopendra Vikram, et al. "Predicting multi-label emojis, emotions, and sentiments in code-mixed texts using an emojifying sentiments framework." *Scientific Reports* 14.1 (2024): 12204.

[38] Pillai, Aditya R., and Biri Arun. "A feature fusion and detection approach using deep learning for sentimental analysis and offensive text detection from code-mix Malayalam language." *Biomedical Signal Processing and Control* 89 (2024): 105763.

[39] Younas, Aqsa, et al. "Sentiment analysis of code-mixed Roman Urdu-English social media text using deep learning approaches." *2020 IEEE 23rd international conference on computational science and engineering (CSE)*. IEEE, 2020.

[40] Sane, Sushmitha Reddy, et al. "Deep learning techniques for humor detection in Hindi-English code-mixed tweets." *Proceedings of the Tenth Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*. 2019.

[41] Jadon, Adarsh Singh, Mahesh Parmar, and Rohit Agrawal. "Hinglish Sentiment Analysis: Deep Learning Models for Nuanced Sentiment Classification in Multilingual Digital Communication." *2024 2nd International Conference on Device Intelligence, Computing and Communication Technologies (DICCT)*. IEEE, 2024.

[42] Himabindu, Gadde Satya Sai Naga, Rajat Rao, and Divyashikha Sethia. "A self-attention hybrid emoji prediction model for code-mixed language:(Hinglish)." *Social Network Analysis and Mining* 12.1 (2022): 137.

[43] Mursalin, Golam Sarwar Md, Suborno Deb Bappon, and Muhammad Ibrahim Khan. "A deep learning approach for recognizing textual emotion from bengali-english code-mixed data." *2022 25th International Conference on Computer and Information Technology (ICCIT)*. IEEE, 2022.

[44] Yann, Lim May, et al. "Sentiment analysis on Malay-English mixed language text using artificial neural network." *AIP Conference Proceedings*. Vol. 2898. No. 1. AIP Publishing, 2024.

[45] Gillioz, Anthony, et al. "Overview of the Transformer-based Models for NLP Tasks." *2020 15th Conference on computer science and information systems (FedCSIS)*. IEEE, 2020.

[46] Xu, Peng, Xiatian Zhu, and David A. Clifton. "Multimodal learning with transformers: A survey." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45.10 (2023): 12113-12132.

[47] Zhang, Rui, et al. "Contrastive data and learning for natural language processing." *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Tutorial Abstracts*. 2022.

[48] Luo, Shengyao. "Applications and Challenges of Meta-Learning in Natural Language Processing." *Applied and Computational Engineering* 109 (2024): 1-8.

[49] Kamath, Uday, et al. "Prompt-based Learning." *Large Language Models: A Deep Dive: Bridging Theory and Practice*. Cham: Springer Nature Switzerland, 2024. 83-133.

[50] Nooralahzadeh, Farhad. "Low-Resource Adaptation of Neural NLP Models." *arXiv preprint arXiv:2011.04372* (2020).

[51] Wu, Lingfei, et al. "Graph neural networks for natural language processing: A survey." *Foundations and Trends® in Machine Learning* 16.2 (2023): 119-328.

[52] Wang, William Yang, Jiwei Li, and Xiaodong He. "Deep reinforcement learning for NLP." *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics: Tutorial Abstracts*. 2018.

[53] Patwa, Parth, et al. "Semeval-2020 task 9: Overview of sentiment analysis of code-mixed tweets." *arXiv preprint arXiv:2008.04277* (2020).

[54] He, Haibo, et al. "ADASYN: Adaptive synthetic sampling approach for imbalanced learning." *2008 IEEE international joint conference on neural networks (IEEE world congress on computational intelligence)*. Ieee, 2008.

[55] Chai, Christine P. "Comparison of text preprocessing methods." *Natural Language Engineering* 29.3 (2023): 509-553.

[56] Kannan, R. Ramesh, Ratnavel Rajalakshmi, and Lokesh Kumar. "IndicBERT Based Approach for Sentiment Analysis on Code-Mixed Tamil Tweets." *FIRE (Working Notes)*. 2021.

[57] Jiang, Fengqing. *Identifying and mitigating vulnerabilities in llm-integrated applications*. MS thesis. University of Washington, 2024.

[58] Khanuja, Simran, et al. "Muril: Multilingual representations for indian languages." *arXiv preprint arXiv:2103.10730* (2021).

[59] Fuadi, Mukhlish, Adhi Dharma Wibawa, and Surya Sumpeno. "Adaptation of multilingual t5 transformer for indonesian language." *2023 IEEE 9th Information Technology International Seminar (ITIS)*. IEEE, 2023.

[60] Sokolova, Marina, Nathalie Japkowicz, and Stan Szpakowicz. "Beyond accuracy, F-score and ROC: a family of discriminant measures for performance evaluation." *Australasian joint conference on artificial intelligence*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006.

[61] Jain, Lakshit, Moolchand Sharma, and Zainab R. Abdulsada. "Offensive Tweets Detection in Hinglish Using HingBERT." *International Conference on Data Analytics & Management*. Singapore: Springer Nature Singapore, 2023.

[62] Liang, Jingsai. "Confusion matrix: Machine learning." *POGIL Activity Clearinghouse* 3.4 (2022).