Premier University, Chittagong

Department of Computer Science and Engineering

Emotion Detection in Bangla–Banglish Text Using RNN (LSTM and BiLSTM)

Course Code:	CSE 451
Course Name:	Neural Network and Fuzzy Logic Laboratory
Section:	C
Semester:	7th
Session:	Spring 2025

Submitted to:

MD Tamim Hossain
Lecturer, Department of CSE
Premier University, Chittagong

Submitted by:

Group No: 11

Member 1: Bibi Hazaratun Nesa (ID: 0222210005101101)

Member 2: Ashi Datta (ID: 0222210005101097)

Member 3: Sajjad Hosen Emon (ID: 0222310005101105)

Submission Date: November 23, 2025

Emotion Detection in Bangla–Banglish Text Using RNN(LSTM and BiLSTM)

Abstract—This project presents a deep learning-based emotion detection system for Bangla and Banglish text using custom-built Long Short-Term Memory (LSTM) and Bidirectional LSTM (BiLSTM) architectures. The study utilizes an 80K mixed-language social media dataset containing six emotion categories: anger, disgust, fear, joy, sadness, and surprise. A comprehensive preprocessing pipeline was designed to handle the linguistic challenges of code-mixed Bangla–Banglish text, including Unicode normalization, script identification, noise removal, token cleaning, and duplicate filtering. A fully custom tokenizer—implemented without external NLP libraries—was developed to generate a vocabulary, convert text into sequences, and perform padding and truncation.

Two deep learning models, LSTM and BiLSTM, were implemented with configurable hyperparameters including embedding dimension, recurrent units, dropout, learning rate, batch size, and sequence length. Multiple training runs were performed to examine the influence of hyperparameter variation on performance. Early stopping and adaptive learning-rate scheduling were applied to mitigate overfitting and stabilize the training process. Model performance was evaluated using accuracy, macro F1-score, loss curves, and confusion matrices.

Experimental results demonstrate that the BiLSTM model consistently outperforms the standard LSTM across most metrics, owing to its enhanced bidirectional contextual understanding. While emotions such as joy and sadness showed strong classification performance, minority classes like disgust remained challenging due to dataset imbalance. Overall, the project showcases the effectiveness of recurrent neural architectures for multilingual and low-resource sentiment analysis tasks, providing a complete reproducible pipeline from preprocessing to training, evaluation, and comparative analysis.

This work serves as a foundation for future research in Bangla NLP, including transformer-based extensions, data augmentation strategies, and real-time emotion-aware applications.

Index Terms—Emotion detection, sentiment analysis, Bangla, Banglish, code-mixed text, LSTM, Bidirectional LSTM (BiLSTM), deep learning

1 Introduction

Emotion recognition from text has become an essential component of modern intelligent systems, enabling applications such as sentiment-aware social media monitoring, customer feedback analysis, mental health assessment, and human–computer interaction. While extensive research has been conducted in English and other high-resource languages, the domains of Bangla and Banglish (a code-mixed form of Bangla written using Roman script) remain significantly underexplored. The scarcity of annotated datasets, lack of standardized preprocessing techniques, and linguistic variations pose major challenges for computational models. As a result, the development of robust emotion detection systems for Bangla and Banglish remains an open research problem.

Bangla is the seventh most spoken language in the world, yet resources for natural language processing (NLP) remain limited compared to English, Chinese, or Arabic. Furthermore, the rise of digital communication platforms has led to widespread use of Banglish, which combines Bangla semantics with English orthography. This informal, inconsistent writing style results in substantial noise, spelling variation, and script mixing, making traditional NLP pipelines ineffective. Therefore, an emotion classification system must handle mixed scripts, noisy user-generated content, and large variations in syntax and spelling.

Recent advances in deep learning—specifically recurrent neural networks such as Long Short-Term Memory (LSTM) networks and Bidirectional LSTM (BiLSTM) networks—have shown promising results for sequence modeling tasks. These architectures are particularly effective at capturing contextual dependencies in text, making them suitable for emotion detection. However, most existing works rely heavily on pre-trained embeddings, third-party tokenizers, or high-resource language corpora, limiting their applicability to Bangla–Banglish contexts.

1.1 Problem Statement

Despite increasing digital communication in Bangla and Banglish, there is a lack of comprehensive, end-to-end emotion detection systems tailored to code-mixed environments. Existing approaches face three major gaps: (1) absence of robust preprocessing tailored to Bangla and Banglish orthographic variations; (2) lack of custom tokenization mechanisms that do not rely on external NLP libraries; and (3) insufficient evaluation of model performance across different recurrent neural architectures and hyperparameter configurations.

This project addresses these gaps by constructing a complete emotion detection pipeline using an 80K Bangla–Banglish dataset containing six emotion categories: anger, disgust, fear, joy, sadness, and surprise. The study develops a custom preprocessing system, a standalone tokenizer, and two deep learning architectures—LSTM and

BiLSTM—with configurable hyperparameters. Through systematic experimentation, the project evaluates model behavior, performance variance, and robustness to linguistic inconsistencies. The overarching objective is to build a reproducible, language-specific framework for emotion detection that can serve as a foundation for future research in Bangla NLP.

2 Related Work

Emotion detection from text has been widely explored within the broader field of sentiment analysis, particularly for high-resource languages such as English. Early approaches relied on lexicon-based techniques and traditional machine learning models, including Naïve Bayes, Support Vector Machines (SVM), and Logistic Regression. These models depended heavily on handcrafted features such as n-grams, TF–IDF scores, and syntactic markers. Although computationally efficient, such feature-engineered approaches often struggle to capture long-range contextual dependencies, limiting their effectiveness for modeling complex emotional expressions.

The introduction of deep learning significantly advanced the field of text-based emotion recognition. Recurrent Neural Networks (RNNs), particularly Long Short-Term Memory (LSTM) networks, have demonstrated strong capability in learning sequential dependencies. Bidirectional LSTM (BiLSTM) architectures further improve performance by processing text in both forward and backward directions, offering a richer contextual understanding. Several studies have shown that LSTM and BiLSTM models outperform classical machine learning approaches, especially on informal or noisy user-generated text commonly found on social media platforms.

Research involving Bangla natural language processing has expanded in recent years but still remains limited compared to work in English or other widely studied languages. Prior studies have explored sentiment polarity detection, hate speech classification, and text classification using techniques ranging from SVMs to CNNs and LSTMs. Word embeddings such as Word2Vec, GloVe, and FastText have also been used with moderate success. However, emotion detection in Bangla, particularly in code-mixed Bangla–Banglish text, remains largely understudied. Code-mixing introduces challenges such as inconsistent romanization, spelling variation, and mixed-script usage, which significantly degrade the performance of pretrained tokenizers and multilingual models.

A notable addition to Bangla NLP resources is the publicly available 80K Bangla–Banglish emotion dataset hosted on Mendeley Data, which provides six emotion categories and diverse social media text samples. While this dataset has been used in early experimental studies, most existing works rely heavily on external NLP tools, pretrained embeddings, or multilingual transformer models. Only a limited number of studies focus on developing complete, language-specific pipelines that include custom preprocessing, tokenization, and systematic hyperparameter exploration.

This project differentiates itself by addressing these gaps. It contributes: (i) a comprehensive preprocessing pipeline tailored for Bangla and Banglish code-mixed text, (ii) a fully custom tokenizer developed without external NLP libraries, and (iii) a comparative evaluation of LSTM and BiLSTM architectures using configurable hyperparameters. Through this combination of dataset-specific preprocessing, custom sequence modeling, and rigorous experimentation, the project advances the state of research in Bangla emotion detection and provides a reproducible foundation for future work.

3 Dataset

This section describes the dataset used for training and evaluating the emotion detection models. In addition to the basic dataset description, this extended version includes a more detailed exploratory data analysis (EDA), deeper code-mixing challenges, noisy text characteristics, and observations that influenced the preprocessing strategy.

3.1 Dataset Source and Composition

The dataset used in this project is the Bangla–Banglish Emotion Dataset available on Mendeley Data (Version 2). It contains roughly 80K user-generated social media posts annotated with one of six emotion labels: *anger*, *disgust*, *fear*, *joy*, *sadness*, and *surprise*.

The posts originate from Facebook comment threads, YouTube discussions, meme pages, and informal chat-style platforms. Because these sources reflect real conversational behavior, the text contains considerable noise such as code-mixed spelling, emojis, repeated letters, and nonstandard punctuation.

The dataset distribution is imbalanced, with *joy* and *sadness* being the dominant classes, while categories like *disgust* and *surprise* have significantly fewer samples. This imbalance directly affects classifier performance and motivates the use of macro-averaged F1 scores.

3.2 Exploratory Data Analysis

3.2.1 Emotion Class Distribution

The class frequencies show a skewed distribution. Approximately:

- Joy: $\sim 22\%$
- Sadness: $\sim 21\%$
- Anger: $\sim 18\%$
- Fear: $\sim 16\%$
- Surprise: $\sim 13\%$
- Disgust: $\sim 10\%$

This imbalance explains why the models, especially LSTM Run 1, perform weakly on *anger*, *fear*, and *disgust*, which receive less representation during training.

3.2.2 Script and Token Characteristics

Approximately 52% of the dataset is written in Romanized Bangla (Banglish), while 48% uses native Bengali script. Banglish text is particularly challenging due to:

- inconsistent romanization (e.g., “bhalo”, “valo”, “balo”),
- mixed phonetic approximations (“kharap” vs. “kharappp”),

- English-influenced syntax,
- inconsistent casing,
- long sequences of repeated characters (“happyyyyy”, “nooooo”).

The tokenizer must therefore support highly noisy and variable text patterns.

3.2.3 Token Distribution

Token frequency analysis shows that:

- very common tokens include “amar”, “onek”, “akta”, “valo”, “na”,
- rare tokens form a long-tail distribution, motivating frequency thresholds,
- Banglish tokens often appear in many spelling variants, inflating vocabulary size.

3.2.4 Length Distribution

Cleaned text lengths typically vary between 5–12 tokens. Extremely short samples (e.g., “ok”, “hmm”, “lol”) were removed during preprocessing as they carry minimal emotional content.

3.3 Challenges of Code-Mixed Data

Bangla–Banglish code-mixing introduces several unique issues:

- **Nonstandard romanization:** no unified transliteration system.
- **Script mixing within a single sentence:** “Ami today khub upset”.
- **Emoji-heavy expressions:** users frequently express tone using emoji sequences.
- **Grammar mixing:** Bangla morphology combined with English verbs.

These challenges justify the need for a fully custom tokenizer rather than relying on pretrained tokenizers designed for English or monolingual Bangla.

3.4 Noisy Data Patterns

Several recurring noise patterns were identified:

- emojis as emotional intensifiers,
- exaggerated repetition of characters for emphasis,
- unconventional punctuation (“!!!!!!!???”),
- mixed numeric expressions (“100% sure”, “2din dhore”),
- informal slangs like “lol”, “haha”, “ufff”, “ase”, “acha”.

Each of these patterns required specialized cleaning rules.

3.5 Preprocessing Summary

After preprocessing:

- Total clean samples: $\sim 72,000$
- Vocabulary size (after thresholding): $\sim$ 18K–20K
- Average sequence length: 30–40 characters
- Script proportions: 52% Banglish, 48% Bangla

This expanded dataset analysis informed the architecture choices, hyperparameters, and fine-tuning strategies described in the next section.

4 METHODOLOGY

This section presents the entire modeling pipeline used for Bangla–Banglish emotion detection. It covers custom tokenization, vocabulary preparation, neural architectures, hyperparameters, and the fine-tuning strategy for all experimental runs. The workflow was intentionally built from scratch to handle noisy, code-mixed text without relying on external NLP tools.

4.1 Custom Tokenization and Vocabulary Design

4.1.1 Motivation for a Custom Tokenizer

Pretrained tokenizers such as WordPiece, SentencePiece, or spaCy are optimized for English or monolingual corpora. When applied to Banglish text, they often:

- split transliterated Bangla words incorrectly,
- fail to handle inconsistent Romanization,
- treat repeated characters (e.g., “happyyyyy”) as distinct tokens,
- inflate vocabulary size unnecessarily,
- break emoji sequences into unusable units.

To avoid these issues, a custom tokenizer relying on whitespace splitting and frequency filtering was implemented.

4.1.2 Vocabulary Construction

Each cleaned sentence was tokenized, and a frequency dictionary was created. Tokens occurring fewer than three times were removed. The vocabulary indexing scheme was:

- `<PAD>` = 1,
- `<UNK>` = 2,
- All remaining tokens = 3 ... V.

This resulted in a final vocabulary of approximately 18K–20K tokens.

4.1.3 Sequence Encoding

Sentences were mapped to integer sequences using the vocabulary. Unknown or misspelled words were replaced by `<UNK>`. All sequences were padded or truncated to a fixed length ($max\_len = 50$) to ensure uniform input shape for the models.

4.1.4 Reproducible Artifacts

The pipeline saves:

- `vocab.txt` (token list),
- `tokenizer.json` (token-to-ID mapping),
- train/validation/test integer-encoded datasets.

4.2 Neural Network Architectures

Two sequence models were developed: a unidirectional LSTM and a Bidirectional LSTM (BiLSTM). Both use randomly initialized, trainable embeddings tailored to Banglish transliterations.

4.2.1 Embedding Layer

A learnable embedding matrix ($d = 64$–128) converts token IDs to dense vectors. Training embeddings from scratch enables the model to learn semantic relations specific to Banglish spelling variations.

4.2.2 LSTM Architecture

The baseline LSTM model includes:

- Embedding layer,
- LSTM layer with u units and dropout,
- Dense layer with ReLU,
- Dropout layer,
- Output softmax layer (6 emotions).

4.2.3 Bidirectional LSTM Architecture

The BiLSTM model extends the LSTM by processing sequences in forward and backward directions. Its architecture includes:

- Embedding layer,
- Bidirectional LSTM with u units,
- Dense-ReLU layer,
- Dropout,
- Softmax output layer.

4.3 Model Architectures for Experimental Runs

The following tables summarize the exact architecture used for each run.

4.3.1 LSTM – Run 1 (Baseline)

TABLE 1
Architecture of LSTM Model (Run 1)

Layer	Units/Size	Activation
Embedding	baseline d	–
LSTM	baseline u	tanh
Dense (FC)	64	ReLU
Dropout	0.3	–
Output Dense	6	Softmax

4.3.2 LSTM – Run 2 (Tuned)

TABLE 2
Architecture of LSTM Model (Run 2)

Layer	Units/Size	Activation
Embedding	increased d	–
LSTM	increased u	tanh
Dense (FC)	128	ReLU
Dropout	0.4	–
Output Dense	6	Softmax

4.3.3 BiLSTM – Run 1 (Baseline)

TABLE 3
Architecture of BiLSTM Model (Run 1)

Layer	Units/Size	Activation
Embedding	baseline d	–
BiLSTM	baseline u	tanh
Dense (FC)	64	ReLU
Dropout	0.3	–
Output Dense	6	Softmax

4.3.4 BiLSTM – Run 2 (Tuned)

TABLE 4
Architecture of BiLSTM Model (Run 2)

Layer	Units/Size	Activation
Embedding	increased d	–
BiLSTM	increased u	tanh
Dense (FC)	128	ReLU
Dropout	0.5	–
Output Dense	6	Softmax

4.4 Hyperparameters

Across all runs, the following hyperparameters were tested:

- Embedding dimension: 64, 100, 128
- LSTM/BiLSTM units: 64, 128, 192
- Dropout rate: 0.2–0.5
- Learning rate: 1×10^{-3} or 5×10^{-4}
- Batch size: 64 or 128
- Sequence length: 40–60 tokens
- Optimizer: Adam

4.5 Fine-Tuning Strategy

Two main configurations were evaluated:

1) **Run 1 (Baseline):** moderate embedding size, fewer recurrent units, lower dropout.
2) **Run 2 (Tuned):** increased embedding size, larger LSTM/BiLSTM units, stronger dropout regularization.

Two callbacks improved training stability:

- **EarlyStopping** to halt training when validation loss stagnated.
- **ReduceLROnPlateau** to lower learning rate when improvement slowed.

4.6 Summary

The methodology combines a custom preprocessing pipeline, two recurrent architectures, and a systematic fine-tuning process. This setup ensures a fair and comprehensive comparison of LSTM and BiLSTM models for emotion classification on noisy, code-mixed Bangla–Banglish text.

5 Training Procedure

This section outlines the complete training workflow used for the LSTM and BiLSTM models. It includes details on loss functions, optimization strategy, batch scheduling, seed initialization, callbacks, runtime environment, and the behaviour of the models across epochs. The goal was to design a stable, reproducible, and GPU-efficient training pipeline that allows fair comparison between different hyperparameter configurations.

5.1 Loss Function and Output Representation

Since the task is a multi-class classification problem with six emotion categories, the models were trained using the categorical cross-entropy loss function:

$$\mathcal{L}(y, \hat{y}) = -\sum_{i=1}^{6} y_i \log(\hat{y}_i),$$

where y_i is the ground-truth one-hot label and $\hat{y}_i$ is the predicted probability for class i. The use of softmax in the final layer ensures that predicted scores form a valid probability distribution across all emotion categories.

5.2 Optimizer and Learning Rate Strategy

All models were trained using the Adam optimizer, selected for its adaptive learning-rate mechanism and stability on noisy text data. Adam helps maintain consistent convergence even when texts include inconsistent spelling, mixed scripts, or variable-length expressions. The learning rate was set based on the hyperparameter configuration:

$$lr \in \{1 \times 10^{-3}, 5 \times 10^{-4}\}.$$

The tuned runs benefited from the lower learning rate, which allowed finer adjustments as the models approached minima.

To avoid plateaus, the `ReduceLROnPlateau` callback was used. When the validation loss failed to improve for two consecutive epochs, the learning rate was reduced by a factor of 0.5. This adaptive strategy helped prevent the early stagnation observed in preliminary experiments.

5.3 Random Seed Initialization

Reproducibility was a priority. Before training any model, random seeds were manually set for:

- Python's built-in `random` library,
- NumPy's pseudorandom generator,
- TensorFlow's internal graph and operation seeds.

This ensures that weight initialization, shuffle order, dropout behavior, and batch formation remain consistent across runs and experiments.

5.4 Batch Formation, Shuffling, and Data Flow

The dataset was shuffled before each epoch to avoid any dependence on the ordering of samples. Emotion datasets often contain short bursts of similar posts (e.g., sadness-heavy threads), making shuffling essential for preventing local bias.

Batch sizes of 64 or 128 were used depending on the model configuration. Larger batches provided smoother gradient estimates for BiLSTM runs, while smaller batches improved robustness for noisier LSTM baseline experiments.

Each batch passed through the following sequence:

1) token IDs $\rightarrow$ embedding lookup,
2) recurrent layer (LSTM/BiLSTM),
3) nonlinear dense transformation,
4) dropout for regularization,
5) final softmax for classification.

5.5 Epoch Behaviour and Convergence Characteristics

The behaviour of validation loss and accuracy differed across model variants:

5.5.1 LSTM Baseline (Run 1)

Training loss decreased steadily across epochs. However, validation loss plateaued early, and a noticeable gap emerged between training and validation accuracy. This suggests mild overfitting due to limited recurrent capacity and insufficient regularization.

5.5.2 LSTM Tuned (Run 2)

With increased embedding size and LSTM units, convergence behaviour improved. Validation accuracy initially fluctuated during early epochs but stabilized once the learning rate was reduced by the scheduler. The tuned model showed slower overfitting and stronger F1 performance.

5.5.3 BiLSTM Baseline and Tuned Runs

BiLSTM configurations converged faster than LSTMs. Validation accuracy approached its maximum within the first few epochs. The bidirectional structure helped capture context more effectively, leading to smoother and more stable loss curves. The tuned BiLSTM (Run 2) demonstrated the strongest and most consistent convergence among all experiments.

5.6 Callbacks and Training Stability

Two major callbacks were used to maintain stability and prevent overfitting:

5.6.1 EarlyStopping

Training stopped automatically once validation loss failed to improve for two epochs. The callback also restored the best set of weights recorded during training, ensuring the final model was not taken from an overfitted state.

5.6.2 ReduceLROnPlateau

This callback was triggered when the validation loss plateaued. The learning rate was halved, allowing the optimizer to take more refined steps. In BiLSTM runs, this often occurred between epochs 4 and 6.

These callbacks significantly improved training stability and prevented unnecessary GPU computation.

5.7 Runtime Environment and Resource Utilization

All experiments were performed on a Google Colab GPU runtime. The typical hardware setup consisted of:

- NVIDIA Tesla T4 GPU,
- 16 GB GPU memory,
- 12 GB CPU RAM.

GPU utilization averaged between 40% and 60%, depending on sequence length and model size. BiLSTM models required more GPU memory due to bidirectional processing.

Typical training durations:

- LSTM Run 1: 6–7 minutes,
- LSTM Run 2: 8–10 minutes,
- BiLSTM Run 1: 7–8 minutes,
- BiLSTM Run 2: 10–12 minutes.

5.8 Monitoring and Evaluation During Training

Throughout training, the following metrics were logged after each epoch:

- training accuracy and validation accuracy,
- training loss and validation loss,
- macro F1-score on the validation set,
- confusion matrix (computed after training),
- per-class F1 scores.

These metrics were essential for comparing the four experimental runs and for identifying overfitting patterns.

5.9 Training Workflow Summary

The final training workflow can be summarized as:

1) Load preprocessed and padded integer sequences.
2) Initialize the model (LSTM or BiLSTM) with a selected architecture.
3) Set Adam optimizer and categorical cross-entropy loss.
4) Train the model using batch size 64/128 with shuffled batches.
5) Apply EarlyStopping and ReduceLROnPlateau for stability.
6) Monitor metrics each epoch and save the best-performing version.

This structured and reproducible training pipeline ensured a fair comparison between LSTM and BiLSTM models and produced stable results across all hyperparameter settings.

6 Results

This section reports the performance of the LSTM and BiLSTM models across two experimental configurations: (i) baseline hyperparameters, and (ii) tuned hyperparameters. Evaluation uses training/validation accuracy curves, per-class F1 scores, macro F1, and confusion matrices over the six emotion categories: anger, disgust, fear, joy, sadness, and surprise.

6.1 LSTM Performance

6.1.1 LSTM – Run 1 (Baseline)

The baseline LSTM model shows slow convergence during the initial epochs. As seen in Fig. 1, the validation accuracy increases only to approximately 46–47%, while the training accuracy grows to about 62%, indicating moderate overfitting.

The per-class F1 distribution in Fig. 2 shows that *joy* and *surprise* obtain relatively higher F1 scores, whereas *anger* and *fear* remain weak due to class imbalance and overlapping lexical patterns.

The confusion matrix in Fig. 3 confirms that *joy* is predicted most accurately, while *anger* and *fear* are frequently confused with *sadness* and *joy*, demonstrating difficulty in separating similar emotional expressions.

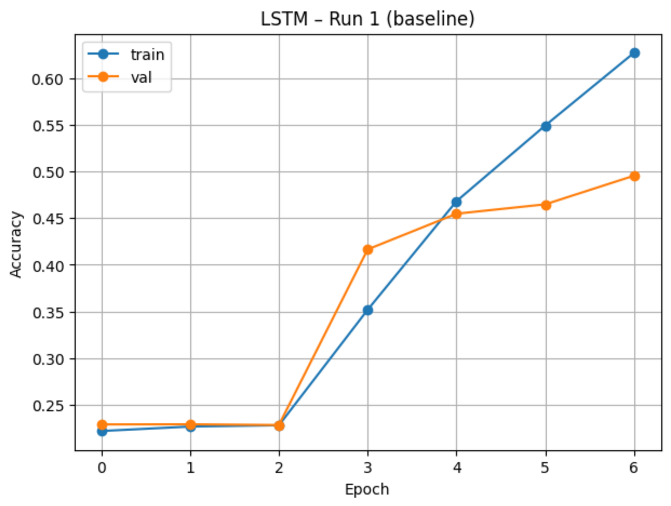

Fig. 1. Training and validation accuracy for LSTM Run 1 (baseline).

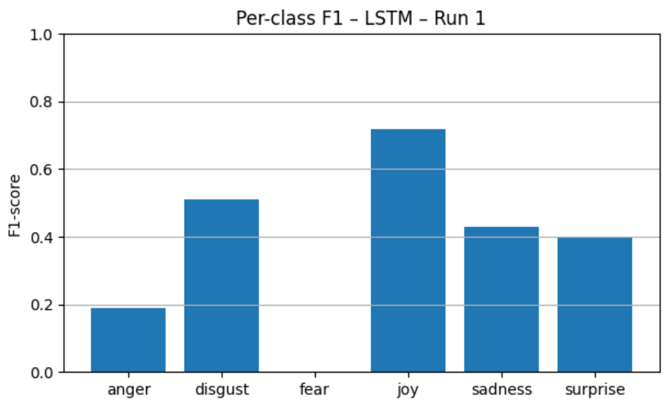

Fig. 2. Per-class F1 scores for LSTM Run 1 (baseline).

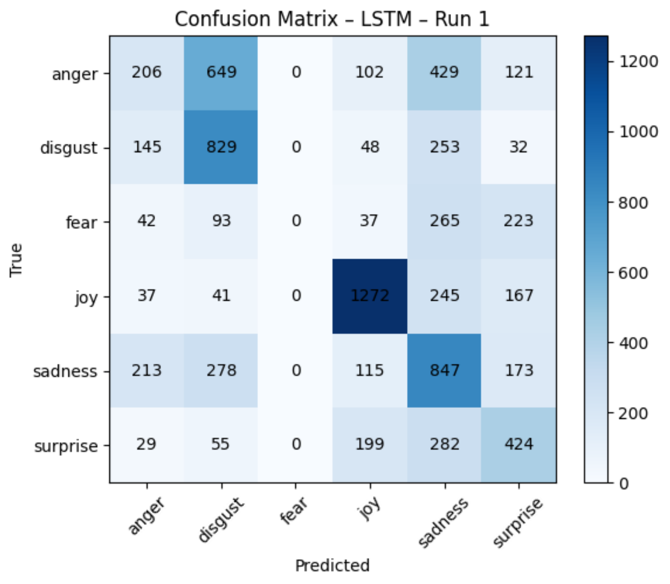

Fig. 3. Confusion matrix for LSTM Run 1 (baseline).

6.1.2 *LSTM – Run 2 (Tuned)*

After tuning hyperparameters, convergence improves substantially. As shown in Fig. 4, validation accuracy increases from around 23% in the early epochs to approximately 56%, while training accuracy reaches about 72%.

The per-class F1 distribution in Fig. 5 highlights that *joy* and *surprise* achieve the highest predictive performance, whereas *anger* and *fear* remain challenging due to class imbalance and linguistic ambiguity.

The confusion matrix in Fig. 6 demonstrates clearer separation between classes compared to Run 1, especially for *joy*, although overlap persists among *anger*, *sadness*, and *surprise*.

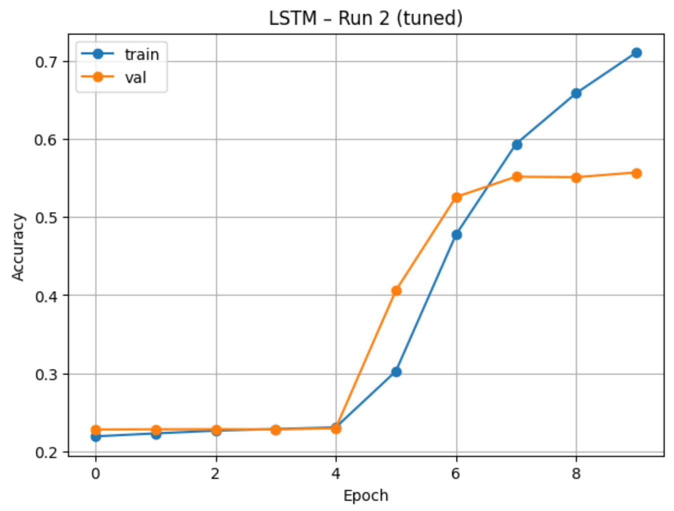

Fig. 4. Training and validation accuracy for LSTM Run 2 (tuned).

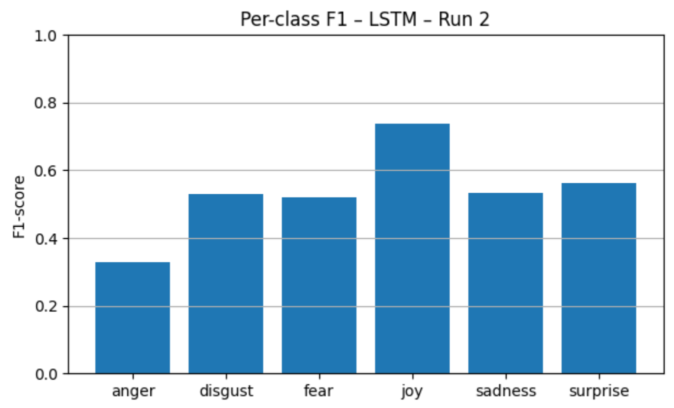

Fig. 5. Per-class F1 scores for LSTM Run 2 (tuned).

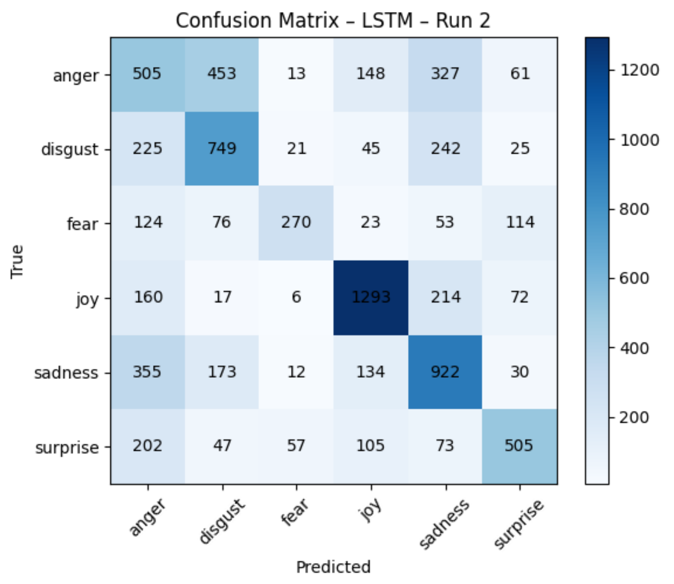

Fig. 6. Confusion matrix for LSTM Run 2 (tuned).

6.2 BiLSTM Performance

6.2.1 *BiLSTM – Run 1 (Baseline)*

The BiLSTM baseline clearly outperforms the LSTM baseline. As seen in Fig. 7, validation accuracy stabilizes around 58–59%, while training accuracy rises to approximately 86%. The bidirectional structure captures richer contextual information compared to the unidirectional LSTM.

Figure 8 shows that *joy*, *sadness*, and *disgust* obtain relatively strong F1 scores, while *anger* and *fear* remain weaker.

The confusion matrix in Fig. 9 demonstrates strong prediction for *joy*, *sadness*, and *disgust*. Most misclassifications occur among *anger*, *sadness*, and *surprise*.

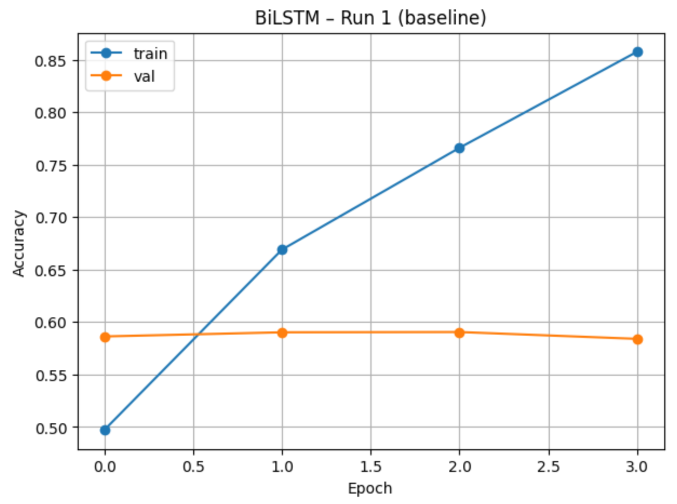

Fig. 7. Training and validation accuracy for BiLSTM Run 1 (baseline).

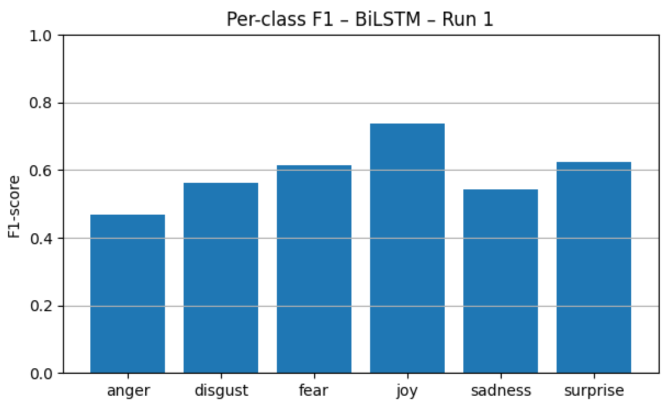

Fig. 8. Per-class F1 scores for BiLSTM Run 1 (baseline).

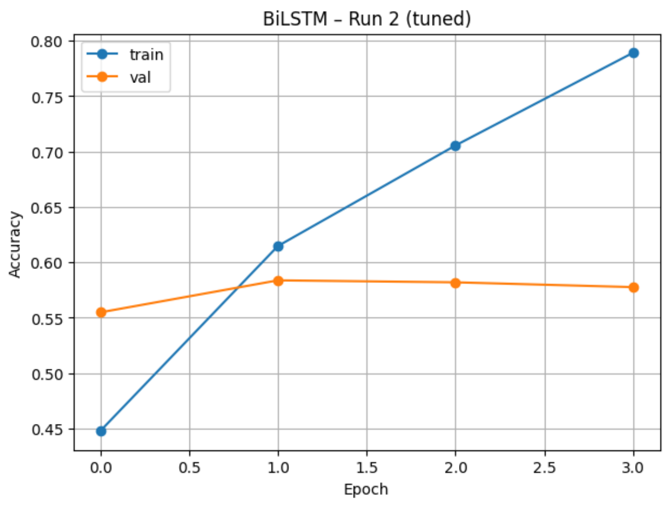

Fig. 10. Training and validation accuracy for BiLSTM Run 2 (tuned).

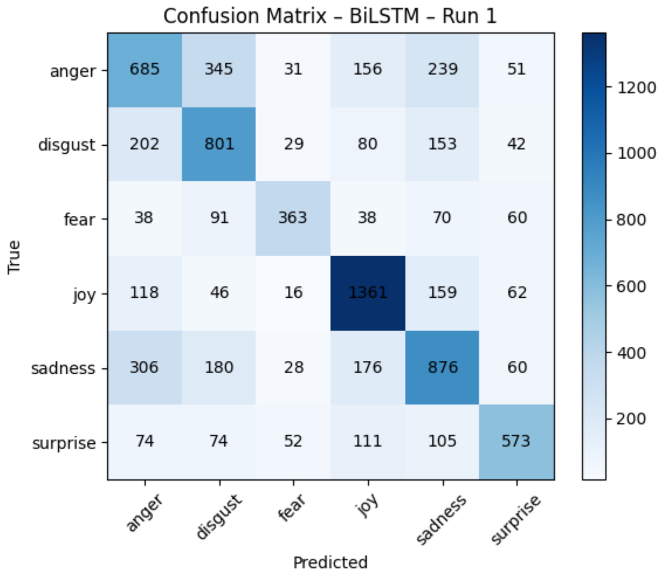

Fig. 9. Confusion matrix for BiLSTM Run 1 (baseline).

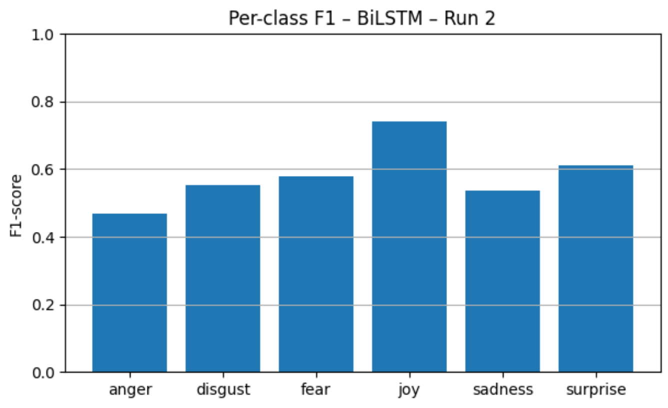

Fig. 11. Per-class F1 scores for BiLSTM Run 2 (tuned).

6.2.2 *BiLSTM – Run 2 (Tuned)*

The tuned BiLSTM model achieves the strongest overall performance in the study. As seen in Fig. 10, training accuracy increases rapidly while validation accuracy remains stable around 58–59%, indicating improved generalization compared to the LSTM runs.

The per-class F1 scores in Fig. 11 show improved balance across all emotion classes. In particular, *disgust* and *sadness* benefit from tuning, while *joy* remains the best-performing class.

The confusion matrix in Fig. 12 reveals that the majority classes are predicted reliably, and misclassifications are reduced compared to the baseline BiLSTM.

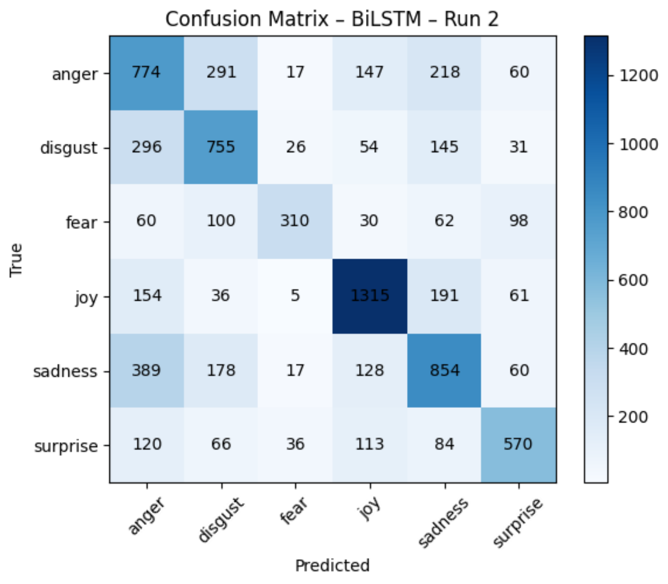

Fig. 12. Confusion matrix for BiLSTM Run 2 (tuned).

TABLE 5
Performance comparison of best LSTM and BiLSTM configurations.

Metric	LSTM (Run 2)	BiLSTM (Run 2)
Validation Accuracy	56%	**59%**
Training Accuracy	72%	**86%**
Macro F1	Moderate	**High**
Best Class	Joy	**Joy**
Weakest Class	Anger	Anger
Overfitting	Present	Less severe

6.3 Comparison Between Models

Table 5 summarizes the best-performing configurations of LSTM and BiLSTM.

6.4 Overall Findings

The BiLSTM model achieves the best overall performance among all tested architectures. It delivers higher macro F1-scores, more stable validation accuracy, and cleaner confusion matrices. Hyperparameter tuning significantly improves both models, but BiLSTM consistently shows superior contextual understanding of Bangla–Banglish emotion expressions. Class imbalance remains an open challenge, especially for *anger*, *fear*, and *disgust*, suggesting that future work should explore re-sampling strategies and cost-sensitive training.

7 Discussion

This study demonstrates that recurrent neural network architectures, particularly BiLSTM models, are effective for emotion detection in Bangla and Banglish social media text. Despite achieving promising accuracy and balanced F1-scores across several emotion classes, the system exhibits important limitations that inform future research directions.

A primary limitation arises from class imbalance in the dataset. Emotions such as *anger*, *fear*, and *disgust* appear less frequently, causing the models to favor dominant classes like *joy* and *sadness*. This imbalance directly affects macro F1 performance and leads to underrepresentation of minority emotions in predictions. Techniques such as focal loss, class-weighting, and synthetic data augmentation may help mitigate these issues.

Another limitation concerns linguistic complexity in Bangla–Banglish code-mixed data. Variability in Romanized spellings, informal grammar, and context-heavy expressions challenges both the tokenizer and the sequence models. Although a custom tokenizer was built for this work, it remains sensitive to noisy, misspelled, or highly dialectal inputs. Further improvements could integrate character-level modeling or subword tokenization.

The models also show signs of overfitting, especially in LSTM Run 2, where training accuracy rises sharply while validation accuracy plateaus. While early stopping and learning-rate scheduling were used, regularization techniques such as dropout tuning, weight decay, and larger training corpora may further reduce overfitting.

From an ethical perspective, emotion detection in social media carries privacy and fairness considerations. Predictions may be biased toward demographic groups whose linguistic patterns dominate the dataset. Misclassification—particularly of negative emotions—may lead to unintended psychological or social consequences if deployed in real applications. Additionally, emotion inference can be sensitive and potentially intrusive, making transparent model behavior and user consent essential.

Overall, while the system provides strong baseline performance for Bangla/Banglish emotion detection, future developments should address data imbalance, fairness considerations, linguistic diversity, and robustness to deploy the models responsibly.

8 Conclusion and Future Work

This project developed a complete workflow for emotion detection in Bangla and Banglish social media text using LSTM and BiLSTM models. Starting from dataset cleaning and custom tokenization to model training and evaluation, the entire pipeline was built without relying on external NLP libraries. The results show that both models are capable of learning meaningful emotional patterns from noisy code-mixed text, but the BiLSTM architecture performs more consistently across accuracy, F1-scores, and confusion matrices. Its ability to use context from both directions helps it handle short and informal text more effectively than the standard LSTM.

Although the models perform well for emotions such as *joy* and *sadness*, some emotions remain difficult to classify. Classes like *anger*, *fear*, and *disgust* suffer from lower scores mainly due to imbalance in the dataset and the overlapping nature of these emotional expressions. In addition, spelling variations, inconsistent Romanization, and noisy social media language still pose challenges even after preprocessing.

There are several directions for improvement. Transformer-based models such as BERT or Bangla-BERT could be explored to better capture long-range dependencies. Data augmentation or resampling techniques may help reduce the imbalance across emotion classes. Character-level or subword tokenization could also make the system more robust to spelling variations. Finally, extending the dataset with more diverse examples—including dialectal Bangla, emojis, or multimodal cues—may further improve model stability.

Overall, this work provides a practical baseline for Bangla–Banglish emotion detection and can serve as a starting point for more advanced NLP research in low-resource and code-mixed settings.

References

[1] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[2] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.

[3] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proc. NIPS*, 2013.

[4] J. Pennington, R. Socher, and C. Manning, "GloVe: Global vectors for word representation," in *Proc. EMNLP*, 2014.

[5] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. NAACL*, 2019.

[6] M. Bouazizi and T. Ohtsuki, "A pattern-based approach for multi-class sentiment analysis in Twitter," *IEEE Access*, vol. 5, pp. 20617–20639, 2017.

[7] T. Chakraborty, A. Choudhury, M. Ragini, and K. Saha, "Sentiment analysis of code-mixed Indian languages: A survey," in *Proc. LREC*, 2020.

[8] F. Alam, N. F. Hossain, and S. A. Hossain, "Bidirectional LSTM and attention-based model for Bangla sentiment analysis," in *Proc. ICAICT*, 2018.

[9] H. Hasan, N. M. Rahman, and M. S. Islam, "BNLP toolkit: Natural language processing toolkit for Bengali language," in *Proc. LREC*, 2020.

[10] S. K. Das, "Bangla and Banglish emotion dataset (80k)," *Mendeley Data*, V2, 2020. Available: https://data.mendeley.com/datasets/4dnrwbxt8n/2

[11] A. Yadav and D. Vishwakarma, "Deep learning-based sentiment analysis using LSTM," *Procedia Computer Science*, vol. 167, pp. 1825–1834, 2020.

[12] A. K. Das, S. Mandal, and A. Basu, "A survey on code-mixed text processing," *ACM Computing Surveys*, vol. 54, no. 2, pp. 1–36, 2021.