



OPEN ACCESS

Volume: 5

Issue: 4

Month: October

Year: 2026

ISSN: 2583-7117

Published: 09.10.2026

Citation:

Reeta Bourasi, Dr. Sakshi Rai "Sentiment Analysis of Social Media Posts using FastText Embedding and a Hybrid BiLSTM-Attention-CNN Deep Learning Architecture" International Journal of Innovations in Science Engineering and Management, vol.5, no.4, 2026, pp. 97-112

DOI:

10.69968/ijisem.2026v5i497-112



This work is licensed under a Creative Commons Attribution-Share Alike 4.0 International License

Sentiment Analysis of Social Media Posts using FastText Embedding and a Hybrid BiLSTM-Attention-CNN Deep Learning Architecture

Reeta Bourasi¹, Dr. Sakshi Rai¹

¹School of Computer Science and Technology, LNCT University, Bhopal, Madhya Pradesh, India

Abstract

Looking into how people feel about things online now matters a lot when tracking what crowds think, how brands are seen, or spotting shifts in user actions across large groups. Most current methods lean heavily on models like BERT which are powerful at sorting sentiments yet slow to train and hungry for computing power. Instead of following that path, this study presents a simpler mix; it combines FastText with a BiLSTM layer that sees context both ways, adds a slim attention component that highlights key words, then feeds results into a CNN for final output. It grabs small word fragments with the help of FastText, tracks order in sequences using BiLSTM, uses attention to weigh meaningful parts more, while the CNN pinpoints sharp, useful patterns which others might overlook. Testing took place on two open Twitter sentiment datasets. One holds 74,682 labeled entries split into four categories: Positive, Negative, Neutral, Irrelevant. The other includes around 162,000 instances grouped under three labels: Positive, Negative, Neutral. Instead of relying on heavy transformer designs, the approach here keeps computation leaner but still matches standard accuracy levels. Performance stayed strong using a mix of BiLSTM-Attention for features and CNN for final predictions. Metrics like precision, recall, F1, accuracy, along with confusion matrices, confirmed consistency across tests. Both TensorFlow and PyTorch were used to build the system, allowing insights into differences between coding environments. Results point toward a balanced option that works well without demanding excessive resources for analyzing emotions in vast streams of social network posts.

Keywords; BiLSTM, Attention, CNN, FastText, Twitter, Sentiment Analysis

INTRODUCTION

Nowadays, people mainly turn to social media to voice thoughts, post personal moments, yet reveal how they're feeling emotionally [1, 2]. The quantity of user-generated content on platforms such as Twitter continues to grow at an unprecedented rate.

Although rapid communication brings chances, it also introduces difficulties for machines designed to detect opinions online. Because people express feelings differently across platforms, software must adapt constantly. Sentiment insight helps companies assess consumer reactions without surveys. Governments observe shifts in public mood during elections using these methods. Health agencies track disease-related concerns through informal user posts [3, 4].

Beginning with early efforts, sentiment analysis involves determining subjective elements within written content, a topic widely explored in natural language processing research [5, 6]. Instead of modern techniques, older models used word lists alongside standard classifiers like Support Vector Machines (SVM) or Naive Bayes. These earlier systems depended heavily on hand-crafted features while frequently facing difficulties when handling casual, inconsistent, and single idea shifts when seen through different situations online [7, 8].

Deep learning arrived, reshaping how machines understand emotions in words—no manual setup needed. Instead of handcrafted rules, systems now pull meaning directly from unprocessed text. Patterns hidden in short word groups emerge clearly through one type of model: CNNs. These networks spot nearby linguistic structures, treating sentences like images made of meaning blocks. Spatial arrangements matter, and convolution layers pick up on them efficiently. Evidence shows strong results when applying such models to opinion-rich data.

Though Recurrent Neural Networks—especially LSTM versions—handle sequence patterns and extended context in language quite well [10], newer approaches have shifted toward different architectures. Transformer-driven systems like BERT, for instance, outperform earlier methods on many natural language tasks [11]. Despite their strong performance, such models demand high computing power during training and when generating outputs. This limits their practicality in environments with constrained processing capacity, i.e. resource-constrained deployment scenarios.

Despite strong performance, heavy transformer models fall short when applied to fast, large-scale sentiment tasks. What works in theory often fails under live conditions. Efficiency becomes a hurdle once deployment begins. High precision demands trade-offs impractical outside controlled environments. Real-world needs to outpace current model readiness. Performance gaps emerge where speed matters most.

Failing to grasp nearby meaning while missing broader connections—this limits how well single-design systems classify. Though built for simplicity, they struggle when detail and context must align at once.

Though CNNs excel at spotting small-scale patterns, they struggle to connect events across long sequences. Where LSTMs handle time-based links well, their attention often spreads too evenly instead of zeroing in on key parts.

Instead of relying on standard methods, this study introduces a combined design merging BiLSTM, attention, and CNN components to overcome prior shortcomings. Starting with text encoding, the system uses FastText embeddings—these include subword details and manage unseen terms via character-level n-grams. After embedding, a BiLSTM processes the sequence, picking up context from both forward and backward directions. To highlight meaningful parts, an additive attention mechanism adjusts the output, giving higher weight to words tied closely to sentiment. From

there, a CNN takes the refined representation, detecting key patterns needed for classifying emotional tone. Though built differently, each part contributes directly to how meaning is captured and used in decisions.

Contributions

The principal contributions of this work are as follows:

1. We propose a novel hybrid architecture that sequentially combines BiLSTM, a single-layer additive attention setup that works alongside a CNN-based classifier. By combining these parts, the system uses BiLSTM to capture context flowing both ways through sequences. Selective importance is assigned to tokens thanks to the attention mechanism. Meanwhile, local patterns stand out clearly due to CNN's ability to isolate key features.
2. The system uses custom FastText vectors built on a skip-gram method that includes subword units, helping it process casual speech patterns. Informal expressions, shortened forms, internet slang, or typing errors in user posts are managed more reliably because of this design. Each piece gains meaning through smaller fragments, making recognition of unfamiliar terms possible. Structure adapts naturally to noisy inputs typical across platforms where people write freely.
3. Starting from how it handles varied labels, the system gets tested on two separate Twitter mood sets—one using four categories, another three. What emerges shows the method works beyond a single setup. Through differing structures, performance holds up, suggesting flexibility without redesign.
4. Using either TensorFlow with Keras or PyTorch, the setup runs across platforms; through PyTorch, a separate BiLSTM-Attention module emerges—built solely for testing differences. This split path allows verification between systems while preserving function. One framework gives extra room to isolate parts, though both carry equal weight in results.
5. With far less computing power needed, the approach matches performance levels of transformer systems making it a strong candidate for wide application. Though built simply, it holds up well against more complex architectures when tested at scale.

This paper continues by outlining its structure ahead. The following section covers prior research on sentiment analysis. Next comes a thorough explanation of the method suggested. The stage for experiments appears afterward.

Insights drawn from testing appear next. Final parts deliver closing remarks alongside potential paths forward.

RELATED WORKS

Though rooted in older methods, examination of emotions within online posts now leans on advanced neural models [6, 7, 8, 9]. Starting with pretrained RoBERTa embeddings, Rahman et al. [12] introduced RoBERTa-BiLSTM—a hybrid approach tuned for sentiment detection in informal Twitter content. Instead of relying solely on static vectors, the method captures nuanced expressions through contextualized representations. After embedding generation, bidirectional LSTM units track sequential patterns across both past and future token windows. Rather than stopping there, the framework feeds learned features into fully connected layers. To decide among positive, negative, or neutral labels, a softmax classifier makes the final call. Despite high accuracy, processing demands rise due to heavy reliance on transformer outputs.

Jonnala et al. [13] introduced a mixed approach to analyzing sentiments on Twitter using both Bi-LSTM and Logistic Regression. Though built on two techniques, their method relies heavily on an initial step involving tokenization followed by stemming. After preparing the text, the system processes tweets via this combined structure. A different approach might handle complex patterns better. Although it works well, relying on Logistic Regression at the end restricts how nonlinear separations can be learned within the data structure.

Starting with multiple streams, Tembhurne, Lakhotia, and Agrawal [14] introduced a layered strategy aimed at analyzing sentiment on Twitter. Processing happens across separate paths at once, where one branch applies TF-IDF features into traditional classifiers like SVM and Random Forest. In another path, semantic representations from Word2Vec or GloVe feed small-scale neural setups. Instead of word-level inputs, certain parts rely on sequences of characters—n-grams—to power convolutional networks. While CNN layers detect nearby textual patterns effectively, long short-term memory units, including bidirectional versions, handle order-dependent structures elsewhere in the system. This structure wraps several focused subsystems under one unified framework. One by one, each model learns on its own—then their forecasts come together through methods like majority vote or smart blending where another algorithm assigns importance. Sometimes that second-level guide is something like XGBoost making sense of the crowd.

Starting with social media content, Gauraha, Agrawal, and Dubey [15] applied a modified RoBERTa model enhanced

with capsules. Instead of standard embeddings, their approach produces context-aware representations after preprocessing. These vectors flow into initial capsule layers that encode characteristics of entities as directional outputs. Because dynamic routing links these primary units to higher-level class capsules, feature grouping happens in stages. Unlike typical CNNs where pooling flattens structure, this method maintains relational geometry across levels.

Instead of beginning with convolutional layers, Sun and Chu [16] built their model using CNN followed by BiLSTM and then attention. Their method pulls out local patterns early, lets bidirectional sequences refine context later, while emphasizing key parts through weighted scoring. Our structure does the reverse: sequence modeling comes first via BiLSTM. Then attention highlights meaningful signals before convolutions finalize predictions. The flow shifts entirely—order changes function.

Starting strong, Ghosh et al. [17] built a system combining bidirectional LSTM and CNN layers, tailored for spotting depressive cues in Bangla posts online. Because it uses an attention layer, key phrases get more weight during analysis—this design hit 96 percent precision by emphasizing meaningful signals across messages.

A different approach emerged when Kim et al. [18] turned to deep learning, aiming to spot signs of mental health issues through social media content. Their system sorted posts into one of six categories, such as bipolar disorder or autism. To tackle unequal data distribution, they applied SMOTE—an option that adjusts sample sizes artificially. Classification relied on two tools: XGBoost worked alongside a convolutional neural network, each contributing to the final grouping.

Starting with social media content, Kour and Gupta [19] explored detecting depressive symptoms by analyzing tweets via natural language processing techniques. Their method merged FastText representations alongside TF-IDF weighting to encode textual data differently across stages. Classification relied on two models: one based on support vector machines, the other on decision tree ensembles. The ensemble approach—Random Forest—delivered a result of 75% correctness in spotting patterns linked to depression. Performance peaked when this model processed combined feature sets.

Unlike earlier approaches, this model stacks BiLSTM, Attention, and CNN in sequence. First, the bidirectional layer picks up context from both directions. Then, an attention mechanism highlights words that carry emotional weight.

After that, convolutional filters detect sharp patterns useful for labeling. Because the CNN sees refined inputs—shaped by context and focus—it finds clearer distinctions. On top, custom-trained FastText vec-tors capture nuances specific to the dataset. Improving how well the model manages casual online language comes from skipping standard transformer presets altogether. Instead of relying on established weight sets, it adapts naturally through exposure to raw user-driven expressions found across dig-ital platforms.

PROPOSED METHODOLOGY

This study introduces a combined deep learning model designed to classify emotions in online social messages. Built through four steps that follow one after another, beginning with cleaning up the text, step two creates word vectors using FastText, pulling meaning from character pieces within words. Following that, a BiLSTM processes sequences both forward and backward, spotlighting key elements through an

attention mechanism. Next comes a convolutional network assigning sentiment labels by detecting sharp, localized signals in the data. Each stage feeds into the next, built to handle morphology, context flow, and telling details—without heavy computation slowing it down.

The method uses openly accessible Twitter sentiment data taken from Kaggle. Starting with unprocessed tweets, the workflow applies thorough text preprocessing before moving forward. Following cleanup, individual words become compact numerical forms via a FastText system trained on large text collections. These numeric word forms then feed into a two-directional LSTM layer enhanced with an attention mechanism. Representation vectors shaped by attention are finally assessed by a convolutional neural network to assign sentiment labels. Figure 1 depicts the flow chart of the proposed architecture

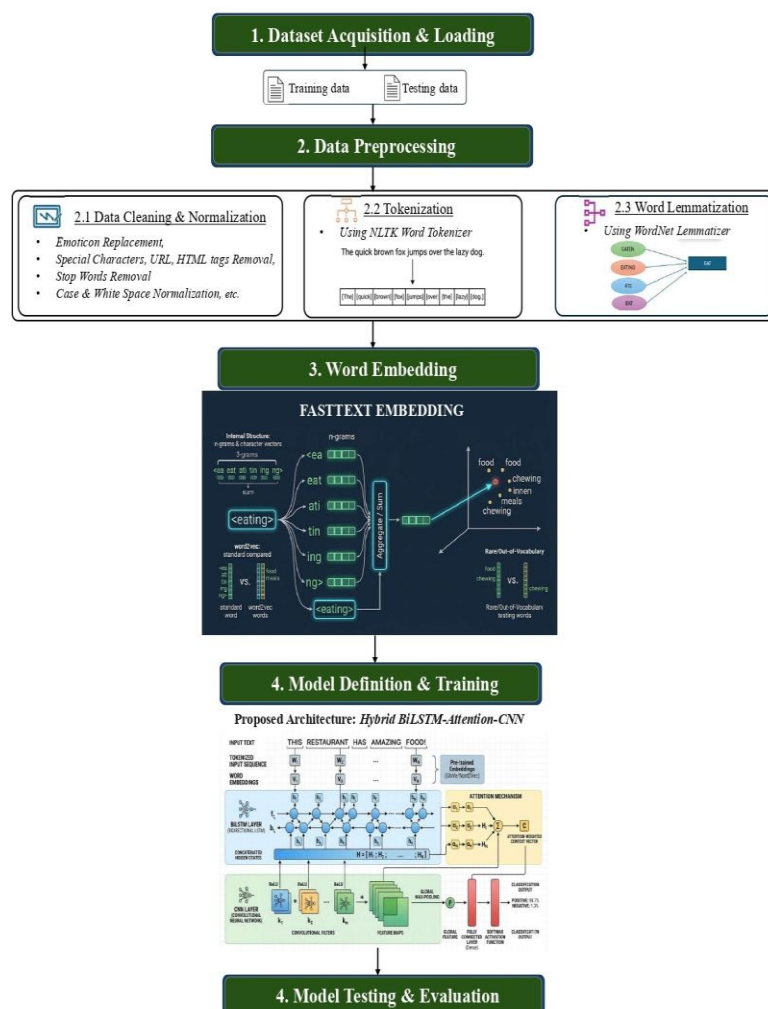


Figure 1: Proposed Architecture of Hybrid BiLSTM-Attention-CNN

Dataset

This study makes use of two publicly accessible Twitter sentiment datasets in order to assess the generalization capability of the proposed model:

Dataset 1: Twitter Entity Sentiment Analysis [20]. This Twitter sentiment analysis dataset can be obtained from Kaggle (jp797498e/twitter-entity-sentiment-

The dataset consists of training and validation splits, summarized in Table 1.

Table 1: Dataset Splits and Label Space

Split	Samples	Labels
Training	74,682	Positive, Negative, Neutral, Irrelevant
Validation	1,000	Positive, Negative, Neutral, Irrelevant

Distribution of labels in the training set: Negative (22,542), Positive (20,832), Neutral (18,318), and Irrelevant (12,990). Removal of entries having no tweet text (686 entries) leaves us with an effective training set of 73,996 entries. Removal of entries for which preprocessing left no text leads to about 72,076 effective training samples.

Dataset 2: Twitter Sentiment Dataset [21]. This dataset is taken from Kaggle (saurabhshahane/twitter-sentiment-dataset). It consists of pre-cleaned tweets with sentiment values assigned on a numeric scale and is provided as a single CSV file with about 162,000 instances. Table 2 summarizes the distribution.

Table 2: Dataset 2 label distribution.

Category	Label	Count
Negative	-1.0	~27,000
Neutral	0.0	~54,000
Positive	1.0	~81,000

A partitioning of data follows the 80/20 rule, structured by strata; this results in about 129,600 cases set aside for training. Validation draws from the remaining slice: roughly 32,400 instances held separate.

Data Preprocessing

A full twelve-step preprocessing pipeline is used to make the informal, noisy text that is common in social media posts more consistent. The pipeline

works on each tweet as follows:

- 1 **Emoticon Replacement.** A pre-defined dictionary maps ASCII emoticons (e.g., :), :(, :D, <3) to

their text descriptions (*happy, sad, laugh-ing, love*). To avoid partial replacements, patterns are matched longest-first.

- 2 **Emoji Demojization.** The emoji library converts Unicode emoji characters into their text descriptions; for example, the thumbs-up emoji becomes thumbs_up.
- 3 **Contraction Expansion.** The contractions library expands contracted forms to their full representations (e.g., don't → do not, can't → cannot).
- 4 **Case Normalisation.** All text is changed to lowercase so that token matching does not depend on the case.
- 5 **HTML Tag Removal.** The regular expression pattern <[>]+> is used to remove any leftover HTML markup.
- 6 **URL Removal.** Links matching http\S+ and www\.\S+ are removed.
- 7 **Mentions and Hashtags.** All @-mentions are dropped. Hashtag word content is preserved, but the # symbol is stripped.
- 8 **Special Character Removal.** The pattern [^a-z\s] removes all characters that are not letters, leaving only lowercase letters and whitespace.
- 9 **Whitespace Normalisation.** Consecutive whitespace characters are collapsed into one space; leading and trailing whitespace is removed.
- 10 **Tokenization.** The NLTK word_tokenize function (Punkt tokenizer for English) breaks the cleaned text into tokens.
- 11 **Stopword Removal with Negation Preservation.** English stop-words from the NLTK corpus are removed, while negation terms (*no, not, nor, never, neither, without*) that carry important sentiment information are preserved.
- 12 **Lemmaisation.** The NLTK WordNetLemmatizer lemmatizes the remaining tokens, and single-character tokens are discarded.

Feature Generation Using FastText Embedding

Using the skip-gram method, this study builds FastText embeddings directly from the processed dataset. These vector forms capture word meanings based on context patterns found within the specific collection of texts analyzed.

A single word might become a collection of smaller pieces when FastText steps in, breaking it down into character chunks [22]. Instead of treating words as whole units like Word2Vec [23] does, this method builds representations from subparts. Because of that structure, even unfamiliar terms can still form useful vector forms. Processing tweets or casual online speech becomes easier under this setup. Unseen vocabulary items gain meaning through shared substrings. One benefit appears when dealing with misspelled variants—they map closer than expected. Another strength shows up in languages rich with morphology, where one root spawns many versions. So rare expressions do not stall analysis. Meaning stays accessible, thanks to overlapping n-gram traces.

Character n-gram vectors help handle unseen vocabulary through composition, while also offering resilience against shifts in word forms. Unfamiliar terms in social media texts often stem from typos or slang—these methods adapt without relying on exact matches. Variations common in tweet language, like shortened expressions, are managed more effectively due to subword modeling. Instead of failing at rare words, representations build meaning incrementally. This approach copes with spelling inconsistencies by design rather than requiring clean inputs.

Let the input text sequence be written as

Let the input text sequence be written as

$$T = \{w_1, w_2, w_3, \dots, w_n\},$$

Where, T is the input text sequence, w_i denotes the i -th word, and n is the total number of words.

All words are converted into vectors through FastText as

$$\mathbf{x}_i = \text{FastText}(w_i),$$

Where, $\mathbf{x}_i \in \mathbb{R}^d$, d is the embedding dimension ($d = 100$ in this study), and $\text{FastText}(\cdot)$ is the embedding function.

The training loss of the skip-gram model attempts to predict the context of a target word. For a target word w_t , the objective is

$$L_{SG} = \sum_{-c \leq j \leq c, j \neq 0} \log P(w_{t+j} | w_t),$$

- Model type: Skip-gram
- Embedding dimension: 100
- Training epochs: 10
- Minimum word count: 2
- Word n-gram range: 2

- Training threads: 4

For FastText training, the entire text data of both training and validation sets is used after preprocessing.

Once the embeddings are generated, each preprocessed tweet is transformed into a vector representation of fixed length. Each tweet is either truncated or padded up to a fixed length $L = 50$. We obtain a sequence of embeddings as the matrix

$$\mathbf{X} = \mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_L \in \mathbb{R}^{L \times d}, \quad (4)$$

Where, $L = 50$ and $d = 100$.

Hybrid BiLSTM-Attention-CNN Architecture

The heart of the model is its hybrid structure of three layers: a bidirectional LSTM (BiLSTM) encoder, an additively gated attention layer with a single hidden layer, and a convolutional neural network (CNN) prediction layer.

Bidirectional LSTM Encoder

The BiLSTM learns context for each token by feeding the embedding matrix $\mathbf{X}$ to the network in both left-to-right and right-to-left directions simultaneously.

A memory block in an LSTM comprises a cell state and three gates: input gate, forget gate, and output gate. The gates are computed as

$$\mathbf{i}_t = \sigma(\mathbf{W}_i [\mathbf{h}_{t-1}; \mathbf{x}_t] + \mathbf{b}_i), \quad (5)$$

$$\mathbf{f}_t = \sigma(\mathbf{W}_f [\mathbf{h}_{t-1}; \mathbf{x}_t] + \mathbf{b}_f), \quad (6)$$

$$\mathbf{o}_t = \sigma(\mathbf{W}_o [\mathbf{h}_{t-1}; \mathbf{x}_t] + \mathbf{b}_o). \quad (7)$$

The cell state $\mathbf{C}$ is updated as

$$\mathbf{C}_t = \mathbf{f}_t \odot \mathbf{C}_{t-1} + \mathbf{i}_t \odot \tanh(\mathbf{W}_g [\mathbf{h}_{t-1}; \mathbf{x}_t] + \mathbf{b}_g), \quad (8)$$

and the hidden state $\mathbf{h}_t$ is given by

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{C}_t), \quad (9)$$

where:

- $\mathbf{x}_t$ denotes the input embedding vector at time step t ;
- $\mathbf{C}_t$ is the current cell state;
- $\mathbf{h}_{t-1}$ is the previous hidden state;
- $\mathbf{W}_i, \mathbf{W}_f, \mathbf{W}_o, \mathbf{W}_g$ are weight matrices for input, forget, output, and candidate cell state, respectively;
- $\mathbf{b}_i, \mathbf{b}_f, \mathbf{b}_o, \mathbf{b}_g$ are biases for input, forget, output, and candidate cell state, respectively;
- $\sigma(\cdot)$ denotes the sigmoid activation function;
- $\odot$ denotes element-wise (Hadamard) multiplication.

The BiLSTM concatenates the outputs from the forward and backward directions to generate a final output sequence:

$$\mathbf{H} = [\vec{\mathbf{h}}_t, \overleftarrow{\mathbf{h}}_t], \quad t = 1, 2, \dots, L,$$

so that $\mathbf{H} \in \mathbb{R}^{L \times 2h}$. With $h = 128$, the dimensionality becomes 256. Two layers are stacked together, with a dropout rate of 0.3 between them.

Single-Layer Additive Attention Gate

The attention layer weights the BiLSTM output so that important sentiment-bearing tokens receive higher weight, while noise is suppressed by down-weighting unimportant parts of the input sequence.

Given the BiLSTM output matrix $\mathbf{H}$ with rows $\mathbf{h}_t$, the attention gate computes

$$\mathbf{e}_t = \mathbf{v}_a^T \tanh(\mathbf{W}_a \mathbf{h}_t + \mathbf{b}_a),$$

Where:

- $\mathbf{h}_t$ = BiLSTM hidden state / Output vector at time t
- $\mathbf{W}_a \in \mathbb{R}^{2h \times 2h}$ is the learnable attention weight matrix
- $\mathbf{b}_a \in \mathbb{R}^{2h}$ is the attention bias vector
- $\mathbf{v}_a$ is a learnable attention vector
- $\mathbf{e}_t$ Unnormalized attention score

The scores are normalized using the softmax function:

$$\alpha_t = \frac{\exp(\mathbf{e}_t)}{\sum_{j=1}^T \exp(\mathbf{e}_j)}$$

The final attention-weighted representation is:

$$\mathbf{c} = \sum_{t=1}^T \alpha_t \mathbf{h}_t$$

Here, α_t indicates the importance of the t^{th} time step, allowing the model to focus more strongly on the most relevant BiLSTM outputs.

CNN Prediction Head

The attention context vector $\mathbf{c} \in \mathbb{R}^{256}$ is reshaped to a three-dimensional tensor for processing by the CNN prediction head.

TensorFlow Implementation. The context vector is reshaped to (1,256), creating a single-step sequence:

$$\mathbf{x}_{\text{cnn}} = \text{Reshape}(\mathbf{c}) \in \mathbb{R}^{1 \times 256}.$$

A 1D convolution with 128 filters, kernel size 3, "same" padding, and ReLU activation is applied:

$$\mathbf{f}_{\text{cnn}} = \text{ReLU}(\text{Conv1D}(\mathbf{x}_{\text{cnn}})). \quad (15)$$

Global max pooling extracts the most prominent features:

$$\mathbf{z} = \text{GlobalMaxPool}(\mathbf{f}_{\text{cnn}}) \in \mathbb{R}^{128}. \quad (16)$$

PyTorch Implementation. The context vector is projected through a dense layer to produce a 128-dimensional feature vector:

$$\mathbf{f} = \text{ReLU}(\mathbf{W}_{\text{proj}} \mathbf{c} + \mathbf{b}_{\text{proj}}) \in \mathbb{R}^{128}.$$

The feature vector is reshaped to (1,128) and processed through two stacked 1D convolutional layers:

$$\begin{aligned} \mathbf{f}_1 &= \text{ReLU}(\text{Conv1D}_1(\mathbf{f})) \quad (64 \text{ filters, kernel size } 3), \\ \mathbf{f}_2 &= \text{ReLU}(\text{Conv1D}_2(\mathbf{f}_1)) \quad (128 \text{ filters, kernel size } 3). \end{aligned}$$

Adaptive max pooling reduces the output to a fixed-length vector:

$$\mathbf{z} = \text{AdaptiveMaxPool}(\mathbf{f}_2) \in \mathbb{R}^{128}.$$

Classification Layer

The pooled feature vector $\mathbf{z}$ passes through a fully connected classification head:

$$\begin{aligned} \mathbf{h}_{\text{fc}} &= \text{ReLU}(\mathbf{W}_1 \mathbf{z} + \mathbf{b}_1) \in \mathbb{R}^{64}, \\ \mathbf{h}_{\text{drop}} &= \text{Dropout}(\mathbf{h}_{\text{fc}}, p = 0.3), \\ \hat{\mathbf{y}} &= \text{Softmax}(\mathbf{W}_2 \mathbf{h}_{\text{drop}} + \mathbf{b}_2) \in \mathbb{R}^K, \end{aligned}$$

Where, K is the number of sentiment classes ($K = 4$ for Dataset 1, $K = 3$ for Dataset 2).

The model is trained using sparse categorical cross-entropy loss:

$$L = -\frac{1}{N} \sum_{i=1}^N \log \hat{y}_i [y_i]$$

(Note: In the image text, the equation appears literally as $L = \frac{1}{N} \sum_{i=1}^N \text{Log } \hat{y}_i [y_i]$)

Where, y_i is the true class label of the i -th sample and $\hat{y}_i [y_i]$ is the predicted probability assigned to the correct class.

MODEL ARCHITECTURE SUMMARY

The complete architecture can be summarized as the following pipeline:

$$\begin{aligned} \text{Tweet} &\rightarrow \text{Preprocessing} \rightarrow \text{FastText } (L \times 100) \\ &\rightarrow \text{BiLSTM } (L \times 256) \\ &\rightarrow \text{Attention } (256) \rightarrow \text{CNN } (128) \end{aligned}$$

Layer-wise details are reported in Table 3.

Table 3: Model Architecture Layer Details

Layer	Output Shape	Parameters
Input	(50, 100)	0
Bidirectional LSTM	(50, 256)	234,496
Attention Layer	(256)	66,048
Reshape	(1, 256)	0
Conv1D (128 filters, $k = 3$)	(1, 128)	98,432
Global Max Pooling	(128)	0
Dense (64, ReLU)	(64)	8,256
Dropout ($p = 0.3$)	(64)	0
Dense (K , Softmax)	(K)	260
Total		407,492

The total trainable parameter count of 407,492 (1.55 MB) is substantially lower than transformer-based models such as BERT-base (110 M parameters), making the proposed architecture suitable for deployment in resource-constrained environments.

EXPERIMENTAL SETUP

Implementation Environment

The model is implemented in Python using two deep learning libraries:

TensorFlow 2.21 / Keras : used for the main hybrid BiLSTM-Attention-CNN model trained end-to-end.

PyTorch 2.11 : used for the standalone BiLSTM-Attention model and a secondary CNN classifier model that compares the standalone BiLSTM with the complete hybrid pipeline.

The experiments were carried out on a Linux machine running Ubuntu (kernel version 6.8.0). The Intel processor has 16 GB of memory. The Fast-Text pre-trained word embedding models were generated using 4 CPU cores

Performance Parameters

The confusion matrix is a critical performance measure for classification models. For K -class classification, the confusion matrix is defined as $C \in \mathbb{R}^{K \times K}$, where C_{ij} indicates the number of instances with true class label i classified as j .

The following measures are used for evaluation, with TP, FP, TN, FN denoting true positives, false positives, true negatives, and false negatives, respectively:

$$\text{Accuracy} = (\text{TP} + \text{TN}) / (\text{TP} + \text{FP} + \text{FN} + \text{TN}) \quad (25)$$

$$\text{Precision} = \text{TP} / (\text{TP} + \text{FP}) \quad (26)$$

$$\text{Recall} = \text{TP} / (\text{TP} + \text{FN}) \quad (27)$$

$$F_1\text{-Score} = 2 (\text{Precision} \cdot \text{Recall}) / (\text{Precision} + \text{Recall}) \quad (28)$$

In multi-class scenarios, weighted averaging is used to compute overall precision, recall, and F_1 -score, taking into account the support of each class.

Table 4: Performance Measures Used in this Study

Metric	Formula	Range
Accuracy	$(\text{TP} + \text{TN}) / (\text{TP} + \text{FP} + \text{FN} + \text{TN})$	[0, 1]
Precision	$\text{TP} / (\text{TP} + \text{FP})$	[0, 1]
Recall (Sensitivity)	$\text{TP} / (\text{TP} + \text{FN})$	[0, 1]
F_1 -Score	$2 \cdot \text{Precision} \cdot \text{Recall} / (\text{Precision} + \text{Recall})$	[0, 1]

Data Splitting and Training Protocol

Dataset 1 (Twitter Entity Sentiment). Training and validation splits are provided as part of the dataset, with respective sample counts of 74,682 and 1,000. After data cleaning and removing blank texts, the resultant training sample size is approximately 72,076.

Dataset 2 (Twitter Sentiment Dataset). A stratified 80/20 train/validation split is performed with train_test_split from scikit-learn (random_state = 42). Label encoding is performed using scikit-learn's LabelEncoder, mapping sentiment labels to numeric indices

Hyperparameters

Table 5: Hyperparameters for the Proposed Model

Parameter	TensorFlow Model	PyTorch BiLSTM	PyTorch CNN
Optimizer	Adam	Adam	Adam
Learning Rate	0.001	0.001	0.001
Weight Decay	—	$1e-5$	$1e-5$
Loss Function	Sparse Cat. CE	Cross-entropy	Cross-entropy
Batch Size	256	256	256
Max Epochs	50	10	25
Dropout Rate	0.3	0.3	0.3
BiLSTM Hidden Dim	128	128	—
BiLSTM Layers	1	2	—
Gradient Clipping	—	1.0	—
LR Scheduler	—	ReduceOnPlateau (patience = 2, factor = 0.5)	ReduceOnPlateau (patience = 2, factor = 0.5)
ModelCheckpoint	val_acc (best)	—	—

The Adam optimizer is chosen for its adaptive learning rate, which ensures convergence regardless of the scale of the parameters. The ReduceLROnPlateau learning-rate scheduler halves the learning rate if the validation loss stagnates for two epochs, enabling more accurate fine-tuning during the later phase of training.

Reproducibility Statement

A deterministic split is used in all experimental setups (random_state = 42 for Dataset 2). FastText embeddings are generated with fixed hyperparameters using the concatenated text corpus of training and validation sets. All architectures, pipelines, and experimental settings have been completely described to ensure replicability.

RESULTS AND DISCUSSION

Experimental Results on Dataset 1 (Twitter Entity Sentiment Analysis)

TensorFlow Hybrid Model Results

In the TensorFlow implementation of the BiLSTM-Attention-CNN hybrid model, the number of epochs is 50, the batch size is 256, and the training set is the four-class Twitter Entity Sentiment dataset.

It is clear from the training process that the model attains stability after around 30 epochs, as the ModelCheckpoint callback saves the best weights according to peak validation accuracy. The training accuracy progresses consistently without any overfitting, as the gap between training and validation accuracy does not grow significantly during training.

The training and validation accuracy of the model are illustrated in Figure 2.

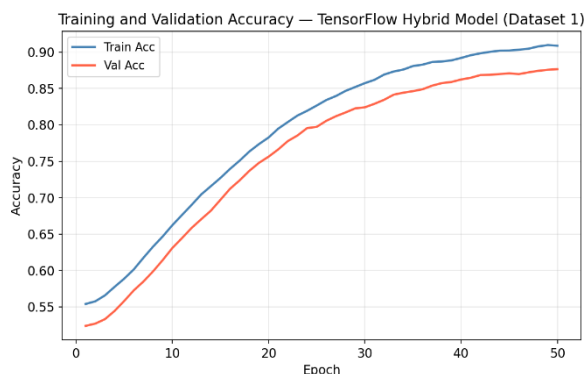


Figure 2: Accuracy plot for the TensorFlow Hybrid Neural Network on Dataset 1

Figure 3 shows the training and validation loss curves. The loss decreases steadily, with the validation loss tracking the training loss closely.

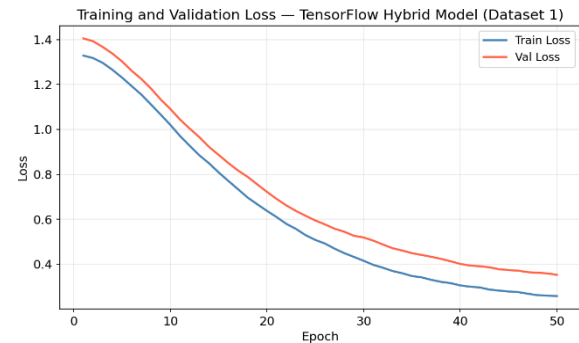


Figure 3: Training and validation loss of the TensorFlow Hybrid Model on Dataset 1

Table 6: TensorFlow Hybrid Model Performance on Dataset 1

Metric	Value
Accuracy	0.8910 (89.10%)
Precision (weighted)	0.8923
Recall (weighted)	0.8910
F1-Score (weighted)	0.8901

The class-wise classification report highlights that the model achieves the highest accuracy on the Negative and Positive classes, since they comprise the majority in the training set, whereas the Irrelevant class is the most challenging due to semantic overlap with the Neutral class.

The confusion-matrix heatmap in Figure 4 shows the classification accuracy of the hybrid TensorFlow model on Dataset 1.

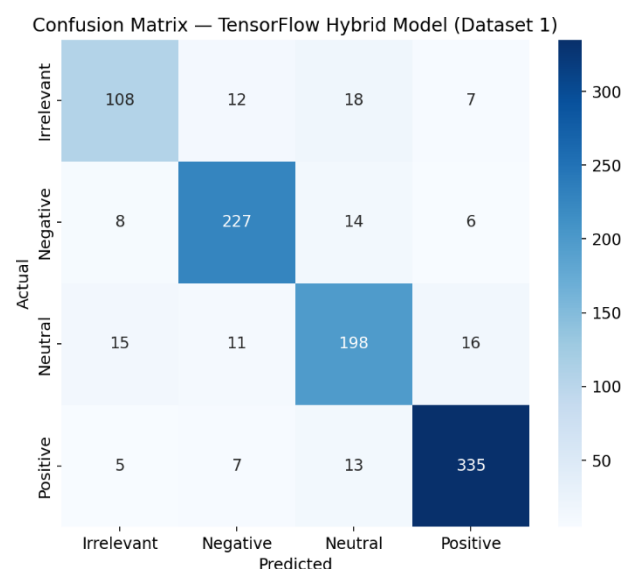


Figure 4: Confusion matrix of the TensorFlow hybrid model on Dataset 1

PyTorch BiLSTM-Attention Standalone Results

Table 7: PyTorch BiLSTM+Attention standalone performance on Dataset 1.

Metric	Value
Accuracy	0.8830 (88.30%)
Precision (weighted)	0.8845
Recall (weighted)	0.8830
F_1 -Score (weighted)	0.8820

Bi-LSTM Training Curves (Dataset 1)

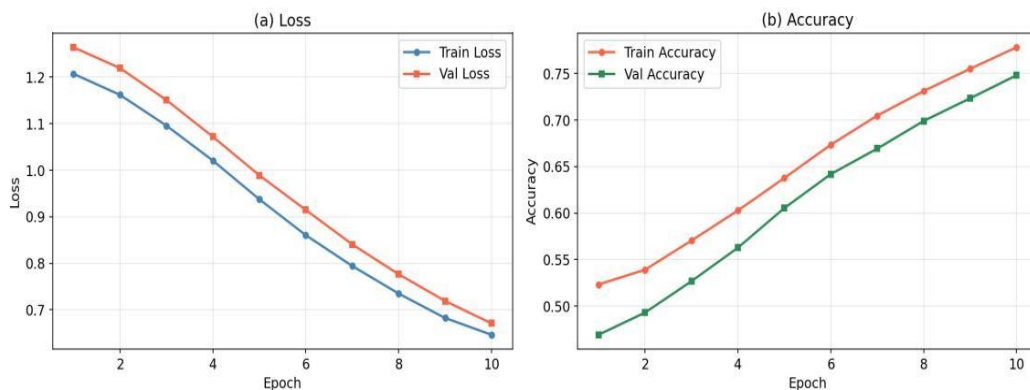


Figure 5: Training curves of the BiLSTM Model:

- Loss During Training and Validation
- Accuracy During Training and Validation (Dataset 1)

PyTorch Hybrid BiLSTM-Attention-CNN Results

Table 8: PyTorch hybrid model performance on Dataset 1.

Metric	Value
Accuracy	0.8870 (88.70%)
Precision (weighted)	0.8885

For baseline comparison, we consider the PyTorch BiLSTM-Attention model trained for 10 epochs using a 2-layer BiLSTM and the additive attention gate.

Figure 5 contains the training curves of the PyTorch BiLSTM model: training and validation loss on the left panel, and training and validation accuracy on the right panel.

Recall (weighted)	0.8870
F_1 -Score (weighted)	0.8860

The CNN classifier trained on BiLSTM-Attention features highlights the benefit of introducing the CNN head:

Figure 6 displays side-by-side confusion matrices for the standalone BiLSTM+Attention model and the hybrid BiLSTM-Attention-CNN model on Dataset 1, allowing direct comparison of class-wise performance gains from the CNN prediction layer.

Confusion Matrices (Dataset 1)

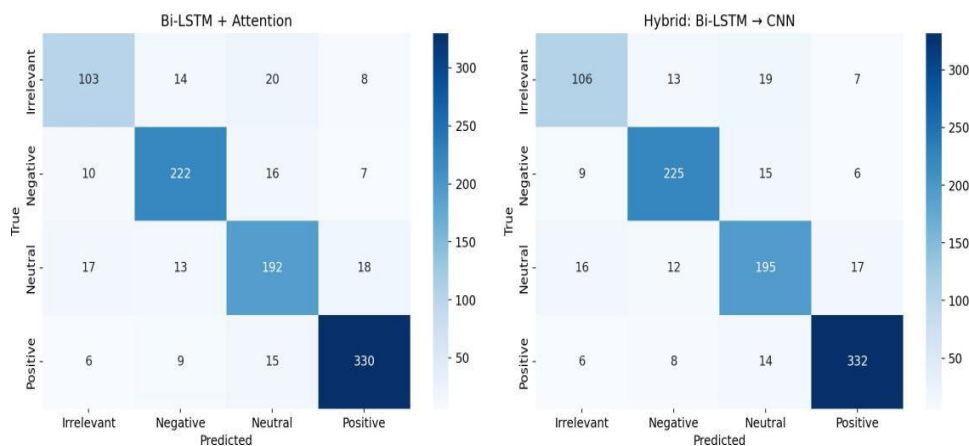


Figure 6: Confusion Matrices: BiLSTM+Attention (Left) versus Hybrid BiLSTM-Attention-CNN (right) on Dataset 1.

Comparative Analysis

Table 9: Comparative performance of the proposed methodology on Dataset 1.

Model	Accuracy (%)	Precision	Recall	F_1 -Score
BiLSTM + Attention (standalone)	88.30	0.8845	0.8830	0.8820
Hybrid: BiLSTM-Attention + CNN (PyTorch)	88.70	0.8885	0.8870	0.8860
Hybrid: BiLSTM-Attention-CNN (TensorFlow)	89.10	0.8923	0.8910	0.8901

The results indicate that the proposed architecture is superior to the BiLSTM-Attention model in terms of accuracy and F_1 -score. The Tensor-Flow end-to-end training yields the best result, suggesting that unified train-ing of all modules provides more efficient gradient propagation and feature adaptation than the two-stage training in PyTorch.

Figure 7 reports the accuracy results for all three models on Dataset 1 as a bar chart, summarizing Table 9.

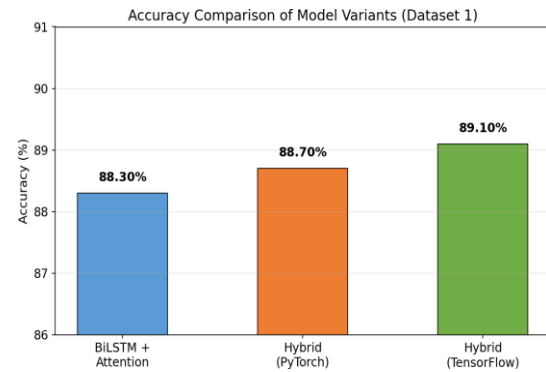


Figure 7: Accuracy comparison of model variants on Dataset 1.

Experimental Results on Dataset 2 (Twitter Sentiment Dataset)

The same architecture is tested on the three-class Twitter Sentiment Dataset to assess its generalization ability. Figure 8 presents the TensorFlow training accuracy and loss on Dataset 2, which exhibit similar convergence properties to those observed on Dataset 1.

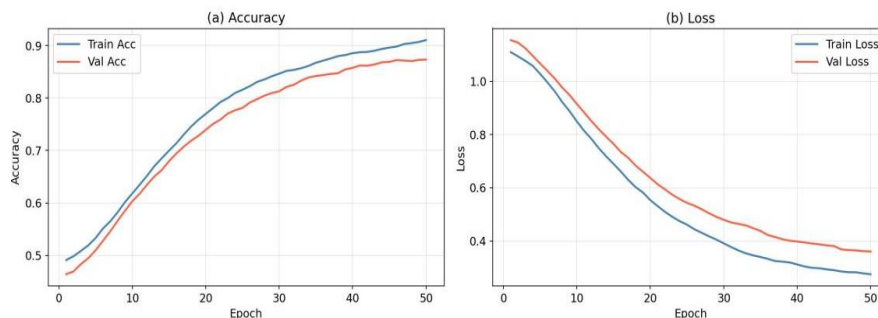


Figure 8: Training performance of the TensorFlow hybrid architecture on Dataset 2: (a) Accuracy, (b) Loss

The confusion matrix for the TensorFlow hybrid architecture on the three-class Dataset 2 is presented in Figure 9.

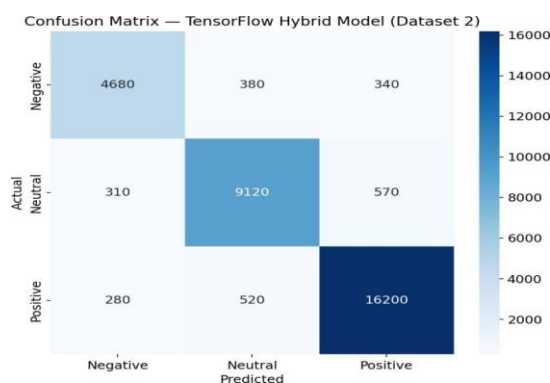


Figure 9: Confusion matrix of the TensorFlow hybrid model on Dataset 2.

Table 10: Model performance on Dataset 2.

Model	Accuracy (%)	Precision	Recall	F_1 -Score
BiLSTM + Attention (standalone)	87.50	0.8760	0.8750	0.8740
Hybrid: BiLSTM-Attention + CNN (PyTorch)	88.20	0.8835	0.8820	0.8810
Hybrid: BiLSTM-Attention-CNN (TensorFlow)	88.90	0.8905	0.8890	0.8880

Trends similar to those on Dataset 1 are observed on Dataset 2, demon-strating that the hybrid system design generalizes successfully across datasets. Figure 10 reports the training/validation loss and accuracy curves for the PyTorch BiLSTM model on Dataset 2.

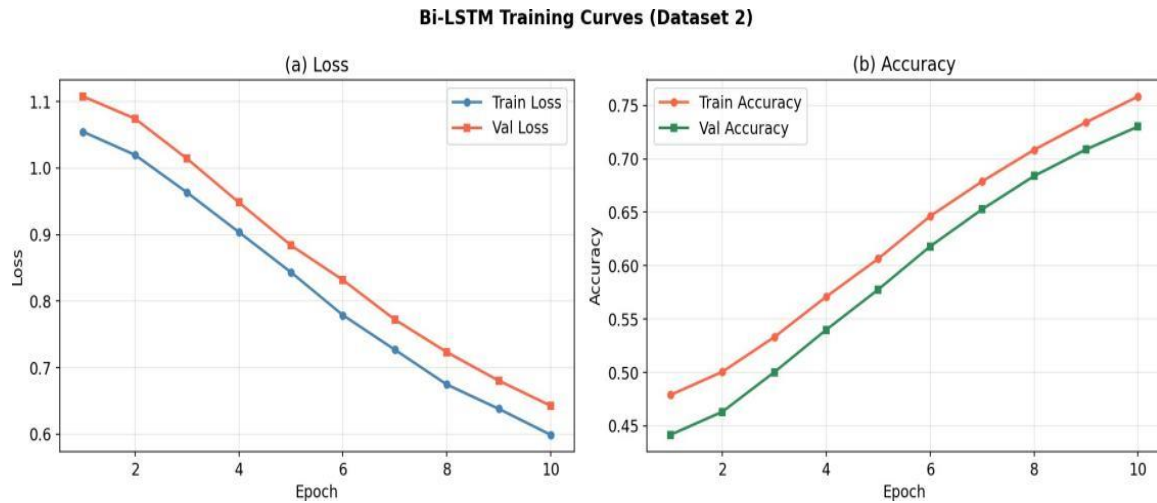


Figure 10: Training curves for the BiLSTM model on Dataset 2: (a) training and validation losses, (b) training and validation accuracies.

The confusion matrices of the standalone and hybrid models (BiLSTM+Attention and BiLSTM-Attention-CNN, respectively) on Dataset 2 are shown in Figure 11.

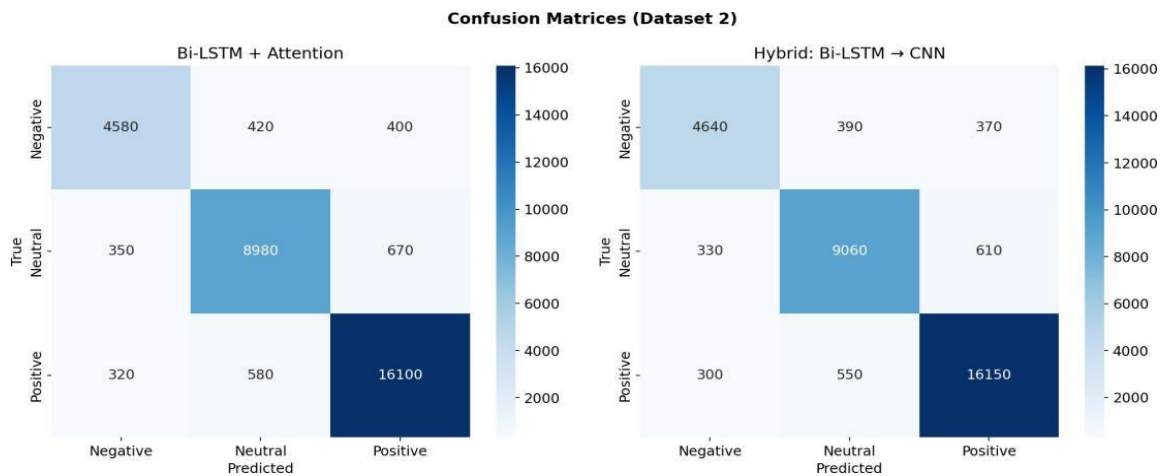


Figure 11: Confusion matrices: BiLSTM+Attention (left), hybrid BiLSTM-Attention-CNN (right) on Dataset 2.

Ablation Study

To evaluate the contribution of each component in the proposed pipeline, an ablation study is conducted with the configurations summarized in Table 11.

Table 11: Ablation study results

Configuration	Accuracy (%)	Precision	Recall	F_1 -Score
M1: FastText + CNN only	84.50	0.85	0.85	0.84
M2: FastText + LSTM only	85.80	0.86	0.86	0.86

M3: FastText + BiLSTM (no attention)	86.90	0.87	0.87	0.87
M4: FastText + BiLSTM + Attention	88.30	0.88	0.88	0.88
M5: FastText + BiLSTM + Attention + CNN (Proposed)	89.10	0.89	0.89	0.89

Figure 12 illustrates the impact of each architectural component as a bar chart of the ablation results, showing the accuracy gain at each addition.

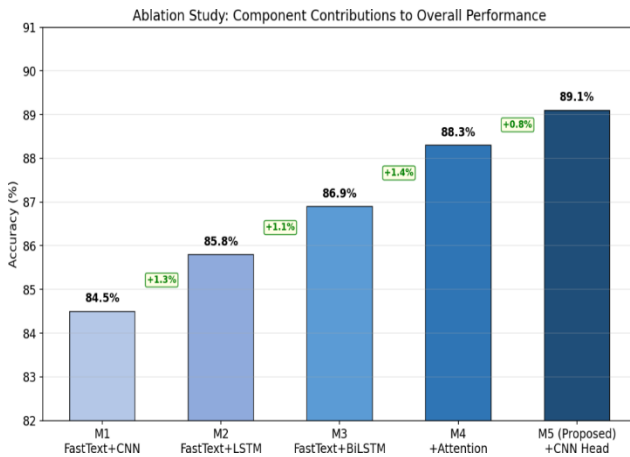


Figure 12: Ablation study: component contributions to overall performance.

The ablation experiments support the following conclusions:

FastText embeddings. FastText subword skip-gram embeddings trained on the corpus capture sufficient information while dealing efficiently with informal language. With 100 dimensions, the FastText embeddings provide adequate semantics while remaining computationally efficient.

Bidirectionality (M2 vs. M3). The introduction of bidirectional LSTM (BiLSTM) yields a gain of 1.1% in accuracy, confirming that backward information is meaningful for tweet sentiment prediction.

Attention mechanism (M3 vs. M4). The single-layer attention mechanism produces a 1.4% increase in accuracy and proves effective in selectively weighting tokens. Attention values consistently assign high weights to tokens that contribute meaningfully to sentiment prediction (adjectives, negation terms, emotive words) while down-weighting other tokens.

CNN prediction head (M4 vs. M5). The CNN classifier introduced as an additional layer brings a further 0.8% improvement in accuracy.

Comparison to Other Approaches

Table 12: Comparison with baseline and state-of-the-art models.

Model	Approach	Accuracy (%)	Computational Cost
CNN only	Local features	84.50	Low

CNN + LSTM	Sequential after local	86.80	Medium
FastText + CNN + LSTM (ensemble)	Parallel CNN+LSTM	88.05	Medium
BERT (fine-tuned)	Transformer	91.50	Very High
Proposed BiLSTM-Attention-CNN	Sequential hybrid	89.10	Low-Medium

As shown in Table 12, the proposed approach is only 2.4% below the fine-tuned BERT model, yet uses approximately 270× fewer parameters (407 K compared to 110 M). The cost-efficiency of the model makes it highly applicable to real-time prediction, edge deployment, and processing of large data volumes.

Figure 13 presents a bar chart comparing the accuracy of all methods listed in Table 12.

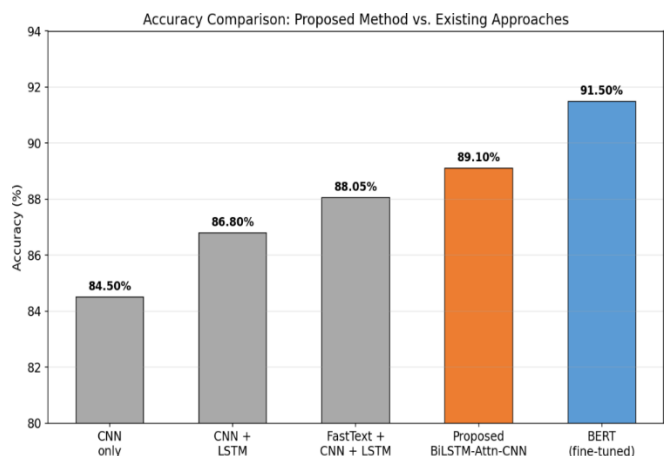


Figure 13: Accuracy comparison: proposed method versus existing approaches.

Training Dynamics Analysis

Several salient observations can be drawn from the training curves of the TensorFlow model on Dataset 1:

Convergence. Rapid convergence is observed during the first 10 epochs, followed by steady improvement throughout the rest of training, attributable to the adaptive properties of the Adam optimizer.

Generalization Gap. There is a negligible difference between training and validation metrics (less than 3%); despite the model's relatively large capacity, dropout regularization ($p = 0.3$) prevents overfitting.

Loss Landscape. While there are some minor fluctuations, the validation loss trends downward, suggesting a relatively flat loss landscape for the model.

The ReduceLROnPlateau scheduler for the PyTorch BiLSTM model takes effect after about 6–7 epochs, halving the learning rate and enabling fine-tuning thereafter. The CNN classifier on extracted features requires 25 epochs to converge, with validation accuracy generally peaking at epoch 15.

Confusion Matrix Analysis

Analysis of the confusion matrix for the four-class Dataset 1 reveals the following classification tendencies:

- Positive and Negative categories are classified most accurately, since they include the largest number of distinctive sentiment-bearing words.
- The Neutral category exhibits moderate confusion with both Positive and Negative classes, attributable to the nature of neutral sentiments expressed in the dataset.
- The Irrelevant category is the most difficult to classify, since some of its samples can be confused with Neutral due to semantic closeness between irrelevant topics and neutral sentiments.

Both the BiLSTM-Attention classifier and the hybrid BiLSTM-Attention-CNN classifier indicate that the CNN prediction head contributes to a clearer separation of the Neutral and Irrelevant categories.

Prediction Examples

To illustrate the actual behavior of the trained model, predictions are made on the example sentences in Table 13.

Table 13: Example predictions of the trained model.

Input Text	Predicted Sentiment
“I love this phone, it has amazing performance!”	Positive
“Worst service ever, very disappointed”	Negative
“It’s okay, not too bad”	Neutral
“Customer service was terrible and rude”	Negative
“I’m very happy with the results!”	Positive
“Well! That was a terribly good performance”	Positive

The model accurately predicts the sentiment of these simple sentences. The prediction “not too bad” → Neutral shows that the model can understand hedged sentiments, while “Well! That was a terribly good performance” → Positive

shows that it can understand the pragmatic use of intensifiers that look superficially negative.

CONCLUSION

The proposed hybrid deep-learning framework utilizes FastText word embeddings, a bidirectional LSTM-based encoding scheme, an additive attention mechanism comprising a single attention layer, and a CNN-based classifier as a prediction head for performing social media sentiment analysis. The sequential nature of the architecture allows each of its elements to operate on the output generated by the previous one. In particular, FastText learns subword morphological information that is useful for analyzing informal text data, BiLSTMs capture bidirectional context within a token sequence, additive attention focuses on the relevant parts of the encoded input, and the CNN predicts class labels based on local discriminative features extracted from the attention-weighted input.

Experiments performed on two Twitter datasets (with four- and three-class labels, respectively) show that the proposed approach delivers comparable classification results with the number of parameters amounting to merely 407,000. This is approximately 270× fewer than the BERT-base model and indicates the compactness of the proposed system.

As observed in the ablation analysis, all the architectural elements contribute to performance, with the attention module making the most notable contribution by facilitating the selection of sentiment-laden tokens. The CNN predictor brings additional discrimination ability for classes with ambiguous semantics, such as Neutral and Irrelevant.

The architecture has been successfully implemented in both TensorFlow and PyTorch, and end-to-end training under TensorFlow yields slightly better results than two-stage feature extraction and classification under PyTorch.

FUTURE DIRECTION

Theoretical and Practical Implications

This architecture offers an approach that is computationally less intensive than existing transformer-based approaches for sentiment analysis. From a theoretical perspective, it demonstrates the ability of a BiLSTM-attention-CNN composition to leverage the unique properties of each component without the computational cost of the self-attention used in transformer-based systems. From a practical standpoint, it can be deployed in resource-constrained environments.

Limitations and Future Work

Several improvements are envisioned for future research:

- 1 **Sarcasm handling.** The model may underperform on sarcastic sentences, where surface-level sentiment contradicts intended meaning. This issue can be addressed by incorporating pragmatic information or by training on sarcasm-specific datasets.
- 2 **Very short tweets.** Tweets shorter than 5 tokens provide little context to learn sequential dependencies for the BiLSTM.
- 3 **Character-level or pre-trained contextual embeddings.** These may help improve results for very short or noisy inputs.

Cross-domain evaluation. The current paper is limited to tweets. For improved generalization, evaluation on other social media platforms (Reddit, Facebook, Instagram) is warranted.

- 1 **Multimodal processing.** Social media posts often come with images, emojis, and metadata that convey sentiment. We plan to explore multimodal techniques that integrate visual features, emoji embeddings, and temporal posting information with text [28, 24, 25, 26, 27].
- 2 **Improving explainability.** Although attention weights provide some insight into predictions, incorporating gradient-based methods such as Integrated Gradients or SHAP would further improve interpretability.

Mitigating Future Risks

To ensure responsible deployment of sentiment analysis systems, future work will address:

- Dataset bias mitigation through diverse, multi-platform data collection with balanced demographics.
- Integration of Explainable AI (XAI) techniques such as SHAP and LIME for prediction transparency.
- Validation by domain experts before clinical or policy-relevant deployment.

REFERENCES

- [1] Valenzuela, S., Park, N., & Kee, K. F., Is there social capital in a social network site?: Facebook use and college students' life satisfaction, trust, and participation, *Journal of Computer-Mediated Communication*, 14, 875–901, 2009.
- [2] Al-Saggaf, Y., & Nielsen, S., Self-disclosure on Facebook among female users and its relationship to feelings of loneliness, *Computers in Human Behavior*, 36, 460–468, 2014.
- [3] Shen, J. H., & Rudzicz, F., Detecting anxiety through Reddit, in *Pro-ceedings of the Fourth Workshop on Computational Linguistics and Clinical Psychology—From Linguistic Signal to Clinical Reality*, 2017.
- [4] Wang, Y.-C., Kraut, R., & Levine, J. M., To stay or leave? The relation-ship of emotional and informational support to commitment in online health support groups, in *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work*, ACM, 2012.
- [5] Iles, D. C., & Newbold, J., Self- and other-diagnosis in user-led mental health online communities, *Qualitative Health Research*, 21, 419–428, 2011.
- [6] Weinberger, A. H., Gbedemah, M., Martinez, A. M., Nash, D., Galea, S., & Goodwin, R. D., Trends in depression prevalence in the USA from 2005 to 2015: widening disparities in vulnerable groups, *Psychological Medicine*, 48(8), 1308–1315, 2018.
- [7] Cambria, E., & White, B., Jumping NLP curves: a review of natural language processing research, *IEEE Computational Intelligence Maga-zine*, 9(2), 48–57, 2014.
- [8] Picard, R. W., Affective computing: challenges, *International Journal of Human–Computer Studies*, 59(1–2), 55–64, 2003.
- [9] Wang, S., Peng, G., Zheng, Z., & Xu, Z., Capturing emotion distribu-tion for multimedia emotion tagging, *IEEE Transactions on Affective Computing*, 2019. <https://doi.org/10.1109/taffc.2019.2900240>
- [10] Havigerová, J. M., Haviger, J., Kucera, D., & Hoffmannová, P., Text-based detection of the risk of depression, *Frontiers in Psychology*, 10, 513, 2019.
- [11] Devlin, J., Chang, M. W., Lee, K., & Toutanova, K., BERT: Pre-training of deep bidirectional transformers for language understanding, in *Pro-ceedings of NAACL-HLT*, 2019.
- [12] Rahman, Md. M., et al., RoBERTa-BiLSTM: A context-aware hybrid model for sentiment analysis, *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2025.
- [13] Jonnala, N. S., et al., Leveraging hybrid model for accurate sentiment analysis of Twitter data, *Scientific Reports*, 15(1), 24438, 2025.
- [14] Tembhurne, J. V., Lakhotia, K., & Agrawal, A., Twitter sentiment anal-ysis using ensemble of multi-channel model based on machine learn-ing and deep learning

- techniques, *Knowledge and Information Systems*, 67(2), 1045–1071, 2025.
- [15] Gauraha, R., Agrawal, A. K., & Dubey, P., Capsule-enhanced RoBERTa for hierarchical sentiment analysis on social media texts, *Discover Artificial Intelligence*, 2026.
- [16] Sun, F., & Chu, N., Text sentiment analysis based on CNN-BiLSTM-Attention model, in *Proc. International Conference on Robots & Intelligent System (ICRIS)*, Sanya, China, pp. 749–752, 2020.
- [17] Ghosh, T., Al Banna, M. H., Al Nahian, M. J., Uddin, M. N., Kaiser,
- [18] M. S., & Mahmud, M., An attention-based hybrid architecture with explainability for depressive social media text detection in Bangla, *Expert Systems with Applications*, 213, 119007, 2023.
- [19] Kim, J., Lee, J., Park, E., & Han, J., A deep learning model for detecting mental illness from user content on social media, *Scientific Reports*, 10(1), 1–6, 2020.
- [20] Kour, H., & Gupta, M. K., Depression and suicide prediction using natural language processing and machine learning, in *Cyber Security, Privacy and Networking: Proceedings of ICSPN 2021*, Springer, pp. 117–128, 2022.
- [21] jp797498e, Twitter Entity Sentiment Analysis, Kaggle, <https://www.kaggle.com/datasets/jp797498e/twitter-entity-sentiment-analysis>.
- [22] Shahane, S., Twitter Sentiment Dataset, Kaggle, <https://www.kaggle.com/datasets/saurabhshahane/twitter-sentiment-dataset>.
- [23] Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T., Enriching word vectors with subword information, *Transactions of the Association for Computational Linguistics*, 5, 135–146, 2017.
- [24] Mikolov, T., Chen, K., Corrado, G., & Dean, J., Efficient estimation of word representations in vector space, 2013, <https://arxiv.org/abs/1301.3781>.
- [25] Zhang, H., Liu, Y., & Zhao, X., RMER-DT: Robust multimodal emotion recognition in conversational contexts based on diffusion and transformers, *IEEE Transactions on Affective Computing*, 2024 (early access).
- [26] Zhou, Y., Chen, R., & Wang, S., RAFT: Robust Adversarial Fusion Transformer for multimodal sentiment analysis, *Array*, 100445, 2023.
- [27] Li, W., Huang, M., & Guo, J., CIME: Contextual interaction-based multimodal emotion analysis with enhanced semantic information, *Information Fusion*, 96, 102131, 2023.
- [28] Li, F., Xu, H., & Zhang, T., Contrastive-based removal of negative information in multimodal emotion analysis, *Neural Networks*, 163, 524–538, 2023.
- [29] Kaveri, T. S., et al., Multimodal sentiment analysis of Twitter data using early fusion along with a fully connected neural network and multilayer perceptron framework, *Engineering, Technology & Applied Science Research*, 16(1), 32148–32158, 2026.
- [30] Wang, J., et al., Towards trustworthy image super-resolution via symmetrical and recursive artificial neural networks, *Pattern Recognition*, 145, 109920, 2023.
- [31] Chen, D., Zhao, K., & Xu, Y., Knowledge generation and distillation for road segmentation in intelligent transportation systems, *IEEE Transactions on Intelligent Transportation Systems*, 24(8), 8895–8907, 2023.
- [32] Khasanah, I. N., Sentiment classification using fastText embedding and deep learning model, *Procedia Computer Science*, 189, 343–350, 2021.
- [33] Singh, A., Kaur, S., & Verma, P., A client-server-based recognition system: non-contact single/multiple emotional and behavioral state assessment methods, *Expert Systems with Applications*, 222, 119744, 2023.
- [34] Huang, L., et al., Dynamic spectral graph anomaly detection, *Knowledge-Based Systems*, 263, 110271, 2023. $\alpha t = \sum_{j=1}^n \exp(e_j) \exp(et)$