Advancing Tweet Sentiment Analysis with BERT-EFRI: From Bag of Words to Representational Transfer Learning in Large Language Models

Matteo Boglioni, Federica Bruni, Francesco Rita, Christopher Zanoli
Team: CIL'em All
Department of Computer Science, ETH Zurich, Switzerland

***Abstract*—This paper presents a comprehensive approach to sentiment classification using a dataset of 2.5 million tweets labeled as positive or negative. As social media continues to grow rapidly, platforms like Twitter generate vast amounts of textual data daily. However, the immense volume and the informal, often ambiguous nature of tweets make manual analysis impractical, highlighting the necessity for effective automated sentiment analysis methods. Our approach progresses from text representations using Bag of Words (BOW) or word embeddings (GloVe, FastText) and traditional classifiers to advanced deep learning models, including Convolutional Neural Networks (CNNs), Long Short-Term Memory (LSTM) networks, and Large Language Models (LLMs).We evaluate each model based on classification accuracy, demonstrating consistent performance improvements over baseline models. The final model, leveraging our novel BERT-Enhanced Random Forest Integration (BERT-EFRI), achieved the highest accuracy of 91.84%, demonstrating the effectiveness of combining embeddings from LLMs with random forests to improve sentiment analysis.**

I. Introduction

Sentiment Analysis (SA) is a branch of Natural Language Processing (NLP) that aims to identify and extract sentiment from text. The unstructured nature of text data poses significant challenges for accurately determining sentiment polarity. SA typically involves categorizing text into predefined sentiment categories, such as positive or negative [1].

Tweet Sentiment Analysis, a specialized area within SA, focuses on evaluating the sentiment expressed in tweets. This task is particularly challenging due to the short and informal nature of tweets, which often contain slang, abbreviations, and hashtags. These elements add layers of complexity to the sentiment classification process [2].

In this study, we address the problem of tweet sentiment classification using a dataset of 2.5 million tweets labeled as positive or negative, sourced from the Kaggle competition "ETHZ CIL Text Classification 2024"[3]. Our approach evolves from traditional machine learning techniques to more sophisticated deep learning models, aiming to enhance classification accuracy at each stage.

Our contributions can be summarized as follows:

- **Comparison of Traditional Methods**: We implement and compare several traditional models, including classic machine learning classifiers using text representations such as Bag of Words (BoW), and word embeddings: GloVe, and FastText.
- **Implementation of Advanced Methods**: We develop advanced neural network models, including Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks, to capture spatial and temporal features in the text data. Additionally, we integrate and fine-tune Large Language Models (LLMs) to leverage their superior contextual understanding, achieving higher performance in sentiment classification.
- **Novel Contribution**: We present a novel method, BERT-EFRI, which leverages embeddings from RoBERTa [4] (robustly optimized BERT [5]) to enhance the performance of a random forest classifier.
- **Comprehensive Evaluation**: We provide a comprehensive evaluation of each model, highlighting incremental improvements and discussing the strengths and weaknesses of our approach.

The remainder of this paper is organized as follows. Section II describes the preprocessing pipeline, models, and methods employed in this study. Section III presents the experimental results, followed by a discussion in Section IV. Finally, we summarize our findings and suggest directions for future research in Section V.

Our code is publicly available at this link.

II. Models and Methods

A. Data Preprocessing

Data preprocessing is essential for tweet sentiment analysis due to tweets' short length and irregular structure, which challenge accurate sentiment classification. Effective preprocessing can significantly enhance classification accuracy [2]. Current approaches highlight the need for comprehensive, application-specific preprocessing pipelines [6].

Accordingly, in our implementation, we adopted the following preprocessing steps:

- Addressing null values and managing duplicates to maintain data integrity.
- Resolving conflicting tweets (identical content with different labels) to prevent confusion during model training.
- Text normalization by:
  1) Converting all text to lowercase;
  2) Removing user mentions, URLs, and unnecessary whitespaces;
  3) Expanding contractions;
  4) Transforming emojis and emoticons into descriptive text to preserve their sentiment value;
  5) Splitting hashtags into meaningful words;
  6) Removing retweet symbols, digits, and non-alphabetic characters;
  7) Reducing repeated letters;
  8) Replacing slang words with standard equivalents;
  9) Correcting spelling errors;
  10) Removing stopwords to minimize noise;
  11) Applying lemmatization to reduce words to their base forms.

Notably, steps 4 and 5 are specific additions to the standard pre-processing techniques. These steps represent original contributions of this study, significantly enhancing the preservation of sentiment value and the meaningful interpretation

of hashtags.
Indeed, upon examining the dataset, it was observed that it contained numerous emojis, emoticons, and hashtags, all of which are crucial for Tweet SA. We translated emojis and emoticons into their alphabetical equivalents, thereby preserving their sentiment value. Furthermore, we implemented a method to split hashtags into meaningful words. This task was particularly challenging due to the lengthy, continuous sequences of lowercase letters commonly found in tweets, making it difficult for algorithms to distinguish individual words within such sequences.

After defining this complex preprocessing pipeline, we investigated the effects of each step by testing various preprocessing configurations on our dataset and evaluating the accuracy across multiple models. Each configuration was created by selecting a subset of the complete preprocessing steps, excluding certain steps by setting them to `False`.

Surprisingly, not all preprocessing steps improved the results. Our tests revealed that the most effective configuration was the one detailed in Table I. As shown in Table II, the validation accuracies for different preprocessing configurations varied significantly. Contrary to our initial hypotheses, the complete preprocessing configuration, which enabled all steps (`True`), performed worse than the raw configuration, which disabled all steps (`False`).

Step	Flag	Step	Flag
Handling null values	True	De-emojizing/emoticonizing	True
Handling duplicates	False	Hashtag handling	True
Conflicting tweets	False	Handling punctuation	False
Lowercasing	True	Replacing slang	False
Tag removal	True	Correcting spelling	False
Whitespace stripping	False	Removing stopwords	False
Handling contractions	True	Lemmatization	False

Table I
OPTIMAL PRE-PROCESSING POLICY.

Policy	BoW+best classifier	GloVe+best classifier	FastText
Raw	81.92%	80.65%	86.17%
Total	77.27%	76.18%	78.35%
Optimal	**82.16%**	**81.16%**	**86.22%**

Table II
VALIDATION ACCURACIES FOR DIFFERENT PRE-PROCESSING POLICIES AND BASELINE MODELS.

B. Word Embeddings and Classifiers

For the task of tweet sentiment analysis, our first choice was to use a combination of text representation/word embedding techniques and traditional classifiers. This approach is a logical starting point due to its simplicity, effectiveness, and wide acceptance in natural language processing (NLP). We implemented three main models: Bag of Words (BoW) with traditional classifiers II-B1, GloVe embeddings with traditional classifiers II-B2, and FastText II-B3.

1) Bag of Words (BoW) with traditional classifiers: The Bag of Words (BoW) model [7] represents text as a collection of word frequencies, ignoring grammar and word order but capturing the presence of words effectively. This makes BoW a sensible first choice for text classification tasks, including tweet sentiment analysis. We implemented the BoW model using the `CountVectorizer` class from `scikit-learn`. For classification, we trained multiple commonly used classifiers for tweet SA [1]: Logistic Regression, Support Vector Machines (SVM), Ridge Classifier, Stochastic Gradient Descent (SGD) Classifier, Extra Trees, and Multi-Layer Perceptron (MLP).

2) GloVe embeddings with traditional classifiers: Global Vectors for Word Representation (GloVe) [8] [9] is a widely used word embedding technique that captures semantic relationships between words by mapping them into continuous vector spaces. Unlike BoW, GloVe retains contextual information, making it more powerful for sentiment analysis. We utilized pre-trained GloVe vectors specific to Twitter (*glove.twitter.27B.zip*) [8] to leverage domain-specific knowledge. Similar to the BoW approach, we applied traditional classifiers (Logistic Regression, SVM, Ridge Classifier, SGD Classifier, Extra Trees, and MLP) to the GloVe embeddings.

3) FastText: FastText [10] is designed for efficient learning of word representations and sentence classification. FastText enhances traditional word embeddings by considering subword information, which captures finer-grained nuances and morphological structures of words. This capability is particularly useful for the informal and diverse language of tweets. We trained FastText models using the `-autotune-validation` flag, which allows for automatic hyperparameter optimization.

C. Convolutional Neural Network (CNN)

Building on our initial approach of using word embeddings and traditional classifiers, implementing Convolutional Neural Networks (CNNs) is a natural progression to enhance accuracy in tweet sentiment analysis.[11]

Unlike simpler models that rely on pre-defined word vectors and linear classifiers, CNNs can automatically learn hierarchical feature representations from raw text data, capturing local patterns like n-grams. This makes them particularly suited for text classification tasks, including sentiment analysis. By applying convolutional filters, CNNs detect word and phrase patterns, handling spatial relationships in text. Pooling layers reduce the dimensionality of feature maps, retaining significant information and improving robustness to text variations. These characteristics enable CNNs to recognize contextual nuances and dependencies that simpler models might miss.

Our proposed Convolutional Neural Network (CNN) for tweet sentiment analysis leverages an embedding layer to transform input tokens into dense vectors, capturing semantic relationships between words. The architecture includes three successive Conv1D layers, each followed by MaxPool1D layers to reduce dimensionality and highlight salient features. The final Conv1D layer output undergoes global max pooling, ensuring a fixed-size representation that captures the most significant features across tweets. This is followed by a fully connected layer with ReLU activation and a dropout layer to prevent overfitting. The architecture of the proposed CNN is displayed in Table VIII.

D. Hybrid Models with CNN-LSTM and LSTM-CNN Architectures

The integration of Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks in hybrid models has proven effective for enhancing sentiment classification [12], [13]. CNNs excel at extracting local features, while LSTMs are adept at capturing long-term dependencies in sequential data, enabling these hybrid models to comprehend both local patterns and global context effectively. Two configurations have been explored: the LSTM-CNN variant begins with an embedding layer followed by LSTM layers to capture sequential dependencies, which are then refined by convolutional layers for hierarchical feature extraction and max-pooling to highlight significant features (see Table IX). In contrast, the CNN-LSTM variant starts with convolutional layers for local feature extraction, followed by LSTM layers to model sequence dependencies (see Table X). Both architectures include fully connected layers and dropout regularization to mitigate overfitting, resulting in accurate sentiment classifications.

E. LLM

In this study, we utilized four models from Hugging Face's model repository, chosen for their training on Twitter data, which is suitable for our sentiment analysis task.

1) `twitter-roberta-base-sentiment-latest`: This model[14] [15] originally performs sentiment analysis with three labels (-1, 0, 1). We modified it for binary classification by adding a fully connected linear layer, reducing the output dimension to 1, followed by a sigmoid function to produce a probability between 0 and 1. This model underwent full fine-tuning, adjusting all its weights.

2) `twitter-roberta-large-topic-sentiment-latest`: Initially designed for sentiment analysis with five labels (-2, -1, 0, 1, 2), we adapted this model [16] similarly by adding a linear layer to reduce the output dimension to 1 and applying a sigmoid function. Due to the model's large size, we performed parameter-efficient fine-tuning using the LoRA[17](see Table III) implementation from the Adapter library. This approach allowed us to train only the adapter and the classification layer.

3) `bertweet-base`: As this model[18] does not inherently perform sentiment analysis, we added a classification layer consisting of a linear layer from hidden size to 1, followed by a sigmoid function. This model also underwent full fine-tuning.

4) `bertweet-large`: This model[18] was adapted similarly to bertweet-base with the addition of a classification layer. Given its large size, we applied LoRA for parameter-efficient fine-tuning with the same adapter configuration used for the `twitter-roberta-large` model.

Parameter	Value	Parameter	Value
selfattn_lora	True	attn_matrices	['q','k','v']
intermediate_lora	True	r	8
output_lora	True	alpha	16

Table III
LoRA CONFIGURATION PARAMETERS

To carry on the analysis of state of the art techniques, we also combined those models together into different ensemble architectures (see Table XI). Unfortunately, training multiple models simultaneously was infeasible due to resource constraints and this forced the decision to train only the ensembles' classification layers. In practice, we performed the forward pass on a smaller dataset (c.a. 200000 samples) separately for each fine-tuned model, keeping the last hidden state and saving it in an embedding dataset. The concatenation of different models' embeddings was then fed to a simple classifier, incorporating non-linearities between the linear layers to enhance expressiveness.

F. BERT-EFRI

Inspired by representation learning [19], where embeddings learned for a task are used to train a downstream model on a different one, our novel solution leverages the embeddings learned by fine-tuned LLM models to fit a different type of classifier. Although our task remains the same, we aim to utilize the representations generated by multiple models to fit a random forest classifier [20]. The choice of this specific classifier was motivated mainly by the nature of LLM-generated embeddings, which leverage their high-dimensionaity to encode text semantic over different directions. In particular, the meaningful aspect of this representation for our solution concerns the way two opposite concept are represented, since they often assume opposite directions, easily separable. The objective of our classifier is indeed to let nodes in the decision tree distinguish between "positive" and "negative" directions for different concepts.

Concerning the implementation, we fed scikit-learn random forest classifier (100 estimators) with the same concatenated embeddings used to train the aforementioned ensembles, enabling us to obtain comparable results.

III. RESULTS

We evaluate the performance of our sentiment analysis models using test accuracy as the metric.

*A. **Word Embeddings and Classifiers***

Table IV shows the obtained test scores on 50% of the kaggle test data. Each model was instantiated with the optimal hyper-parameters identified through k-fold cross-validation (`GridSearchCV`, `k=5`):

Model	Accuracy (%)
BoW + Logistic Regressor	80.04
BoW + SVM	79.74
BoW + Ridge Classifier	79.58
BoW + SGD Classifier	79.08
BoW + Extra Trees	79.74
BoW + MLP	83.30
GloVe + Logistic Regressor	77.70
GloVe + SVM	77.82
GloVe + Ridge Classifier	78.04
GloVe + SGD Classifier	77.98
GloVe + Extra Trees	78.66
GloVe + MLP	83.90
FastText	**85.88**

Table IV
TEST SCORES FOR "WORD EMBEDDINGS WITH CLASSIFIER" MODELS.

Observations:

- **BoW Models**: Achieve moderate accuracies ranging from 79.08% to 83.30%, with MLP performing the best among BoW-based models.
- **GloVe Models**: Show similar performance to BoW models, with MLP achieving the highest accuracy of 83.90%.
- **FastText**: Outperforms all other models with an accuracy of 85.88%, highlighting its effectiveness in capturing subword information.

*B. **CNN and Hybrid Models with CNN-LSTM and LSTM-CNN***

Table V shows the obtained test scores on 50% of the kaggle test data:

Model	Accuracy (%)
CNN	84.14
CNN-LSTM	85.74
LSTM-CNN	**87.12**

Table V
TEST SCORE FOR CNN AND HYBRID CNN-LSTM AND LSTM-CNN MODELS.

Observations:

- **CNN**: The CNN model achieves an accuracy of 84.14%, demonstrating its capability to capture local patterns and features in tweets effectively.
- **CNN-LSTM**: Achieves an accuracy of 85.74%, leveraging LSTM's ability to capture sequential dependencies.
- **LSTM-CNN**: Outperforms CNN-LSTM with an accuracy of 87.12%, highlighting the effectiveness of starting with CNN for local pattern recognition followed by LSTM for sequential understanding. Moreover, LSTM-CNN models demonstrate lower susceptibility to overfitting compared to CNN-LSTM by prioritizing sequential understanding before local feature extraction. Empirical studies consistently validate the superiority of LSTM-CNN models over CNN-LSTM models in sentiment analysis tasks [12], [13], leveraging the strengths of LSTM and CNN layers in an order that maximizes their combined effectiveness for sentiment analysis on Twitter data.

C. LLM

Table VI shows the obtained test scores on 50% of the kaggle test data:

Model	Accuracy (%)
RoBERTa-base	90.88
RoBERTa-large	**91.66**
BERTweet-base	91.20
BERTweet-large	91.64
Ensemble-base	91.60
Ensemble-large	**91.66**
Ensemble-base&large	90.60

Table VI
TEST SCORE FOR LLM MODELS.

Observations:

- **Full vs. Parameter-efficient Fine-tuning**: `RoBERTa-base` and `BERTweet-base` models, which underwent full fine-tuning, already show significant accuracy improvements. However, they are outperformed by large models (`RoBERTa-large` and `BERTweet-large`) despite only a small percentage of their weights was fine-tuned, proving LoRA's efficiency in reducing computational needs while maintaining high accuracy.
- **Efficiency Metrics**: LoRA reduced the trainable parameters to about 0.89% of the total model parameters, significantly decreasing training time and resource consumption, making it a practical approach for large models.

- **Ensembles**: `Ensemble-base` (using `RoBERTa-base` and `BERTweet-base` embeddings) significantly improves scores achieved by the base models, being a valid alternative to a large model. On the other hand, `Ensemble-large` (using `RoBERTa-large` and `BERTweet-large` embeddings) and `Ensemble-base&large` (using embeddings from all models) score poorly, probably due to over-parametrization caused by new weights introduced to handle larger inputs.

D. BERT-EFRI

Table VII shows the obtained test scores on 50% of the kaggle test data:

Model	Accuracy (%)
BERT-EFRI-base	91.64
BERT-EFRI-large	**91.84**
BERT-EFRI-base&large	91.14

Table VII
TEST ACCURACY OF BERT-EFRI.

Observation: The BERT-EFRI models outperform their respective ensemble models with a peak of 91.84% of accuracy achieved by BERT-EFRI-large, demonstrating the effctiveness of representational transfer learning in SA.

IV. DISCUSSION

The results show that utilizing the highly-expressive embeddings generated by LLMs greatly improves the performance on the task. This strategy enables the random forest's decision trees to discriminate between positive and negative sentiment directions with high accuracy. Our approach outperforms classical ensemble classifiers in terms of accuracy while maintaining comparable training time, employing embeddings from fine-tuned models. However, the reliance on pre-trained embeddings means our approach may inherit biases present in the original models, potentially impacting fairness and generalization across different datasets.

The implications of our novel approach for sentiment analysis are significant. By demonstrating that embeddings from LLMs can be effectively utilized with random forest classifiers, we open the door to more efficient and scalable sentiment analysis solutions. This method can be particularly beneficial if computational resources are limited, or rapid deployment is necessary.

V. SUMMARY

In this paper, we have explored various methods for SA. Our approach ranged from traditional machine learning techniques to advanced deep learning models, including CNNs, LSTMs, and Large Language Models. Through evaluation, we demonstrated the progressive improvements in classification accuracy, culminating in our novel BERT-EFRI model, which achieved the highest test accuracy of 91.84%. This highlights the efficacy of integrating embeddings from advanced language models with traditional classifiers. Future research will focus on further refining these integrations by exploring additional combinations of classifiers and LLM embeddings, as well as optimizing the training processes to reduce computational costs, thereby making the models more efficient and accessible for real-world applications.

References

[1] J. R. Jim, M. A. R. Talukder, P. Malakar, M. M. Kabir, K. Nur, and M. Mridha, "Recent advancements and challenges of nlp-based sentiment analysis: A state-of-the-art review," *Natural Language Processing Journal*, vol. 6, p. 100059, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2949719124000074

[2] E. Dritsas, G. Vonitsanos, I. E. Livieris, A. Kanavos, A. Ilias, C. Makris, and A. Tsakalidis, "Pre-processing framework for twitter sentiment classification," in *Artificial Intelligence Applications and Innovations*, J. MacIntyre, I. Maglogiannis, L. Iliadis, and E. Pimenidis, Eds. Cham: Springer International Publishing, 2019, pp. 138–149.

[3] Kaggle, "Ethz cil text classification 2024," 2024, accessed: 2024-07-15. [Online]. Available: https://www.kaggle.com/competitions/ethz-cil-text-classification-2024/overview

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

[5] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2019. [Online]. Available: https://arxiv.org/abs/1810.04805

[6] S. Symeonidis, D. Effrosynidis, and A. Arampatzis, "A comparative evaluation of pre-processing techniques and their interactions for twitter sentiment analysis," *Expert Systems with Applications*, vol. 110, pp. 298–310, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417418303683

[7] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. Cambridge University Press, 2008.

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

[9] U. N. Wisesty, R. Rismala, W. Munggana, and A. Purwarianti, "Comparative study of covid-19 tweets sentiment classification methods," in *2021 9th International Conference on Information and Communication Technology (ICoICT)*, 2021, pp. 588–593.

[10] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov, "Bag of tricks for efficient text classification," *arXiv preprint arXiv:1607.01759*, 2016.

[11] Y. Kim, "Convolutional neural networks for sentence classification," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1746–1751.

[12] P. M. Sosa, "Twitter sentiment analysis using combined lstm-cnn models," *Eprint Arxiv*, vol. 2017, pp. 1–9, 2017.

[13] N. I. Ramzi, M. Yusoff, and N. M. Noh, "Comparison analysis of lstm and cnn variants with embedding word methods for sentiment analysis on food consumption behavior," in *International Conference on Soft Computing in Data Science*. Springer, 2023, pp. 193–207.

[14] J. Camacho-collados, K. Rezaee, T. Riahi, A. Ushio, D. Loureiro, D. Antypas, J. Boisson, L. Espinosa Anke, F. Liu, E. Martínez Cámara *et al.*, "TweetNLP: Cutting-edge natural language processing for social media," in *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Abu Dhabi, UAE: Association for Computational Linguistics, Dec. 2022, pp. 38–49. [Online]. Available: https://aclanthology.org/2022.emnlp-demos.5

[15] D. Loureiro, F. Barbieri, L. Neves, L. Espinosa Anke, and J. Camacho-collados, "TimeLMs: Diachronic language models from Twitter," in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. Dublin, Ireland: Association for Computational Linguistics, May 2022, pp. 251–260. [Online]. Available: https://aclanthology.org/2022.acl-demo.25

[16] D. Antypas, A. Ushio, F. Barbieri, L. Neves, K. Rezaee, L. Espinosa-Anke, J. Pei, and J. Camacho-Collados, "Supertweeteval: A challenging, unified and heterogeneous benchmark for social media nlp research," in *Findings of the Association for Computational Linguistics: EMNLP 2023*, 2023.

[17] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, and W. Chen, "Lora: Low-rank adaptation of large language models," *CoRR*, vol. abs/2106.09685, 2021. [Online]. Available: https://arxiv.org/abs/2106.09685

[18] D. Q. Nguyen, T. Vu, and A. T. Nguyen, "BERTweet: A pre-trained language model for English Tweets," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 2020, pp. 9–14.

[19] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[20] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

Appendix

Layer	Parameters
Embedding	55546×20
Conv1D_1	$20 \times 32 \times 3$
MaxPool1D_1	0
Conv1D_2	$32 \times 64 \times 3$
MaxPool1D_2	0
Conv1D_3	$64 \times 128 \times 3$
GlobalMaxPool1D	0
FC_1	128×256
Dropout	0
FC_2	256×1

Table VIII
Poposed CNN Model Architecture.

Layer	Parameters
Embedding	55546×20
LSTM	$20 \times 4 \times 128$
Conv1D_1	$128 \times 32 \times 3$
MaxPool1D_1	0
Conv1D_2	$32 \times 64 \times 3$
MaxPool1D_2	0
Conv1D_3	$64 \times 128 \times 3$
GlobalMaxPool1D	0
FC_1	128×256
Dropout	0
FC_2	256×1

Table IX
LSTM-CNN Model Architecture.

Layer	**Parameters**
Embedding	55546×20
Conv1D_1	$20 \times 32 \times 3$
MaxPool1D_1	0
Conv1D_2	$32 \times 64 \times 3$
MaxPool1D_2	0
Conv1D_3	$64 \times 128 \times 3$
GlobalMaxPool1D	0
LSTM	$128 \times 4 \times 128$
FC_1	128×256
Dropout	0
FC_2	256×1

Table X
CNN-LSTM MODEL ARCHITECTURE.

Layer	**Parameters**
LL1	input size $\times$ input size / 2
ReLU	0
LL2	input size / 2 $\times$ 1
Sigmoid	0

Table XI
ENSEMBLE CLASSIFIER ARCHITECTURE.