



Comparative Analysis Of Convolutional Neural Networks, Long Short-Term Memory Networks, and Bert For Text-Based Emotion Classification

Porawat Visutsak^{1*}, Tanajak Tongbai², Duongduen Ongrungruaeng², Nuttiruj Phongwuttisak², Surapong Wiriya³, and Keun Ho Ryu³

¹ Faculty of Applied Science, King Mongkut's University of Technology North Bangkok, Bangkok, 10800, Thailand

² Faculty of Information Technology, King Mongkut's Institute of Technology Ladkrabang, Bangkok, 10520, Thailand

³ Chaopraya University, Nakhon Sawan, 60240, Thailand

⁴ Faculty of Information Technology, Ton Duc Thang University, Ho Chi Minh City, 700000, Vietnam

* Correspondence: porawatv@kmutnb.ac.th

Citation:

Visutsak, P.; Tongbai, T.; Ongrungruaeng, D.; Phongwuttisak, N.; Wiriya, S.; Ryu, K. H. Comparative analysis of convolutional neural networks, long short-term memory networks, and bert for text-based emotion classification. *ASEAN J. Sci. Tech. Report.* **2026**, 29(6), e262378. <https://doi.org/10.55164/ajstr.v29i6.262378>

Article history:

Received: November 17, 2025

Revised: April 11, 2026

Accepted: April 12, 2026

Available online: May 10, 2026

Publisher's Note:

This article is published and distributed under the terms of the Thaksin University

Abstract: This paper presents a detailed experimental study and comparative analysis of three popular deep learning architectures (CNN, LSTM, and BERT) for emotion classification in written messages. Using a publicly available dataset of six unique emotional states (sadness, joy, love, anger, fear, and surprise), an effective ablation study was conducted to determine optimal architectural configurations, including a sequence length of 66 tokens and an embedding size of 200. To validate the results of a comparative analysis of model performance, a bootstrap technique (30 trials) and the Wilcoxon Signed-Rank test were used to eliminate potential bias. As shown by experiments, the tuned BERT architecture (with a learning rate of $2e-5$) produced the most accurate and reliable result of 93.50% in classifying emotional states from texts. Moreover, with an appropriate sequence length configuration, the LSTM network (89.92%) significantly outperformed the CNN (89.65%), confirming the need to account for long-range dependencies in emotion classification. Overall, the research results show the key importance of hyperparameter tuning and the ability to handle complex information for emotion identification.

Keywords: Emotion classification; deep learning; convolutional neural networks (CNNs); long short-term memory (LSTM); BERT;

1. Introduction

The growth of digital communication platforms, including social media, email, and instant messaging, has generated a vast volume of textual data infused with human emotion. The ability to automatically detect and interpret these emotions is becoming increasingly important across various domains, from enhancing customer service and understanding public opinion to developing more empathetic human-computer interaction systems. However, automated emotion detection from text is inherently complex. Emotional expression in language is often subtle, highly context-dependent, and can be masked by ambiguity, sarcasm, or cultural nuances.

To address these complexities in natural language processing, Devlin et al. [1] introduced BERT (Bidirectional Encoder Representations from Transformers), a

conceptually simple yet powerful language representation model. Unlike previous models such as OpenAI GPT, which relied on unidirectional architectures, or ELMo, which used shallow concatenations, BERT is designed to pre-train deep bidirectional representations by jointly conditioning on both left and right context across all layers [1], [2]. It achieves this deep bidirectionality by utilizing a "masked language model" (MLM) objective to predict randomly masked tokens, alongside a "next sentence prediction" (NSP) task to capture sentence-level relationships. Pre-trained on a massive corpus of 3.3 billion words (BooksCorpus and Wikipedia), BERT can be fine-tuned with just one additional output layer. This approach has proven highly effective, achieving state-of-the-art results on eleven natural language processing tasks, including pushing the GLUE benchmark score to 80.5% and significantly advancing SQuAD question-answering performance [1-2]. Sun et al. [2] further demonstrated that scaling to extreme model sizes with BERT can yield large improvements even on small-scale downstream tasks, thereby minimizing the need for task-specific engineering.

Lawan et al. [3] present a comparative analysis of machine learning models for domain adaptation in multiclass sentiment classification, a notable challenge due to vocabulary and sentiment expression variations across domains. The paper reviews established techniques, including lexicon-based approaches and traditional machine learning algorithms such as Logistic Regression, SVM, and Random Forest, and notes their limitations in domain adaptation despite standard preprocessing steps such as lowercase conversion, stop-word removal, tokenization, stemming, and TF-IDF vectorization. Deep learning models, such as CNNs, known for capturing local patterns, and LSTMs, proficient at handling sequential data, are also examined. The study highlights that transformer-based models, particularly BERT, generally offer superior performance in capturing nuanced contextual information and adapting across domains. The research aims to empirically validate these models using Amazon review data from diverse categories. It suggests exploring hybrid models, other transformer architectures, and advanced training techniques to enhance domain adaptation.

The proliferation of online information necessitates a practical analysis of emotional tendencies in text to understand user needs. Current sentiment classification models often struggle to achieve meaningful performance gains due to their reliance on manually labeled features, which limits exploration of deep semantic features hidden within the text. Huang et al. [4] present a text sentiment classification model that integrates BERT with CNN. In this approach, BERT is used to obtain word embeddings for the text, which a CNN then processes to learn deep semantic information and ascertain the text's emotional orientation. Experimental validation on a large movie review dataset demonstrated that the BERT-CNN model achieved an accuracy of 86.67%, markedly outperforming traditional textCNN methods and indicating its strong performance in this domain.

Demszky et al. [5] introduce GoEmotions, the largest manually annotated dataset for fine-grained emotion classification, consisting of 58,000 English Reddit comments labeled across 27 distinct emotion categories, including a Neutral category. Designed to address the limitations of previous datasets that rely on small sample sizes or coarse taxonomies (such as the standard six basic emotions), the authors carefully curated Reddit data by filtering for length, balancing sentiment, and masking sensitive information to mitigate bias. Statistical evaluation via Principal Preserved Component Analysis demonstrated strong, reliable dissociation among all 27 emotion categories, validating the taxonomy's design. To establish a baseline, the authors fine-tuned a BERT model that achieved an average F1-score of 0.46 on the comprehensive taxonomy. Their transfer learning experiments further demonstrated that GoEmotions effectively generalizes across other domains and taxonomies, making it a highly valuable resource when target-domain training data is limited.

Saravia et al. [6] introduce Contextualized Affect Representations for Emotion Recognition (CARER), a semi-supervised, graph-based framework designed to capture evolving linguistic nuances, such as slang and coded words, for text-based emotion classification. Because traditional lexicon-based methods often fail to model these dynamic linguistic variations effectively, the authors propose representing a large-scale emotion corpus as a complex network. By applying graph measures—specifically eigenvector centrality and clustering coefficients—the system automatically categorizes tokens as "connector words" or "subject words," serving as the building blocks for pattern-based features. To preserve semantic relationships and improve classification, these basic patterns are enriched using continuous word embeddings and weighted using a custom metric called pattern frequency-inverse emotion frequency (pf-ief). When combined with a multi-layer Convolutional

Neural Network (CNN), the CARER model significantly outperforms conventional baselines and state-of-the-art models such as DeepMoji and EmoNet, while demonstrating robustness with reduced feature sets and applicability to multilingual tasks, such as Chinese emotion recognition.

Mokhamed et al. [7] investigate emoji prediction for Arabic text, a growing area in Natural Language Processing due to the prevalence of emojis in digital communication. The research compares the performance of traditional Machine Learning (ML) models (Support Vector Machines, Multinomial Naive Bayes, and Random Forest) with that of advanced Deep Learning (DL) models (LSTM, Bi-LSTM, AraBERT, and mBERT). Using a large, well-balanced Arabic Twitter dataset, the study found that ML models achieved accuracies ranging from 74% to 84%. Notably, DL models generally outperformed them, with LSTM and Bi-LSTM achieving the highest accuracies of 91.16% and 91%, respectively, while AraBERT and mBERT scored 85% and 80%, respectively. This suggests that while ML models provide a baseline, DL approaches, particularly LSTM and Bi-LSTM, offer enhanced performance for Arabic emoji prediction.

Chandra Sekhar et al. [8] focus on sentiment analysis of tweets that contain both text and emojis, using a Kaggle dataset. The study employed sentence embedding models, including DistilBERT, USE-Large3, and RoBERTa, to generate embeddings, which were then used to train dense neural networks (NNs) and LSTM models. While text classification accuracy was high at around 98%, it dropped significantly to 75% when the validation set included emojis absent from the training data. To enhance scalability, a distributed training methodology was implemented, resulting in a 17% reduction in runtime without compromising accuracy. The research also incorporated explainable AI algorithms, SHAP and LIME, to understand model behavior and potential biases. This work aims to bridge the gap in existing sentiment analysis by offering a more holistic understanding of sentiment in digital communications that combine text and emojis.

Gaafar et al. [9] investigate the performance of a Long Short-Term Memory-Recurrent Neural Network (LSTM-RNN) deep learning approach for classification tasks on both textual and image datasets. The study utilized the IMDB movie reviews dataset for textual analysis and the Fashion MNIST image dataset. The results indicated that the image-based LSTM-RNN model outperformed the sentiment-based textual model in terms of speed, with processing times of 3.49 minutes for images versus 18.25 minutes for text, an advantage attributable to the well-defined patterns in image data representation. Furthermore, the LSTM-RNN model achieved higher classification accuracy for images (96.50%) than for text (85.69%), which the authors attribute to the complex network architecture, processing elements, and features of the image datasets. The study concludes that the LSTM-RNN performed better on image classification tasks than on textual data classification, both in terms of speed and accuracy.

Atlas et al. [10] introduce a novel sentence-level sentiment analysis model for online product reviews, employing natural language processing and a hybrid deep learning approach. The methodology includes web crawling for review collection, preprocessing steps like tokenization and lemmatization, feature extraction using a bidirectional gated recurrent unit (Bi-GRU), and sentiment classification into positive, negative, or neutral polarities via a hybrid RNN-based LSTM classifier. Performance evaluation using metrics such as accuracy, precision, recall, F-measure, and AUC revealed that the proposed model outperformed existing techniques, including Deep Convolutional Neural Network (DCNN), Multilayer Perceptron (MLP), CapsuleNet, and Generative Adversarial Networks (GAN). The study concludes that this hybrid Bi-GRU and LSTM-based RNN model offers a promising, superior approach for sentiment analysis of product reviews.

This research by Balakrishnan et al. [11] provides a comparative benchmark of deep learning architectures, including CNNs, RNNs, and Bi-LSTM networks. These models were evaluated using multiple word embedding techniques, including BERT and its derivatives, FastText, and Word2Vec. The Easy Data Augmentation method was employed to create two distinct datasets (original and augmented). The performance of each model was tested in both 5-class and 3-class (compressed) classification scenarios. The results indicate that Neural Network-based models utilizing Word2Vec yielded the top predictions, with a combined CNN-RNN-Bi-LSTM architecture achieving the highest accuracy (96%) and F1-score (91.1%). Among standalone models, RNN performed best, achieving 87.5% accuracy and an F1-score of 83.5%, while RoBERTa had the highest F1-score (73.1%) among the Transformer-based models. This study concludes that

deep learning methods are more effective than traditional supervised machine learning approaches for text-based sentiment analysis and outlines potential avenues for future research.

Cortiz [12] explores the application of Natural Language Understanding (NLU) to emotion recognition by fine-tuning and evaluating five Transformer-based language models: Bidirectional Encoder Representations from BERT, DistilBERT, RoBERTa, XLNet, and ELECTRA. Using the GoEmotion dataset, which features 27 fine-grained emotion categories, the study assessed models based on F1-score and completion time. RoBERTa achieved the highest macro F1 Score of 0.49, followed by DistilBERT and XLNet at 0.48, and BERT at 0.46. ELECTRA had the lowest F1-score at 0.33. Although RoBERTa performed best, the paper suggests that DistilBERT is the most efficient, given its faster training time relative to RoBERTa and XLNet, as well as its superior F1-score compared to BERT. The research concludes that while Transformer models are effective for fine-grained emotion recognition, there is potential for further enhancements in both model performance and dataset development.

Nanyonga et al. [13] evaluate the performance of BERT, CNN, and LSTM deep learning models for classifying aviation incident reports from the Australian Transport Safety Bureau (2013-2023) into four injury severity levels: Nil, Minor, Serious, and Fatal. After standardized preprocessing and hyperparameter tuning, BERT achieved perfect scores (1.00) across all evaluation metrics (accuracy, precision, recall, and F1-score) for all classes. In comparison, LSTM achieved 99.01% accuracy, with strong performance on the Nil class but weaker performance on the Minor class. Meanwhile, CNN achieved 98.99% accuracy, excelling in the Fatal and Serious classes while performing moderately in the Minor class. The findings highlight the advantages of transformer-based architectures, such as BERT, for complex textual classification in aviation safety and suggest future research directions, including ensemble methods and model interpretability, to further enhance predictive performance.

Abas et al. [14] introduce a novel deep learning model, BERT-CNN, for detecting emotions from text, leveraging the strengths of BERT and CNN. The model first utilizes BERT to generate dynamic semantic vector representations of words based on their context. These vectors are then fed into a CNN for emotion classification. Evaluated on the SemEval 2019 Task 3 and ISEAR datasets, the BERT-CNN model demonstrated superior performance compared to existing state-of-the-art models. It achieved an accuracy of 94.7% and an F1-score of 94% on the SemEval dataset (for happy, angry, and sad emotions, after removing the other classes to balance the classes) and an accuracy of 75.8% with an F1-score of 76% on the ISEAR dataset (for seven emotions). The study highlights that a learning rate of 4×10^{-5} yielded the best results for the SemEval dataset.

Talaat [15] aims to enhance sentiment analysis accuracy by proposing four hybrid deep learning models that combine BERT variants (DistilBERT and RoBERTa) with Bi-LSTM and Bi-GRU. Evaluated on three datasets for sentiment classification into positive, negative, and neutral, with and without considering emojis, these models use pre-trained word embeddings for fine-tuning. The study found that hybrid architectures with Bi-GRU layers generally yielded the best results; for example, DistilBERT-GLG without emojis increased accuracy by 1.84% over DistilBERT alone on the Apple dataset. RoBERTa-based models typically outperformed DistilBERT, and the inclusion of emojis affected performance, demonstrating the efficacy of the proposed hybrid methods over standalone BERT models and classical machine learning approaches.

Bi and Zhang [16] address challenges in cross-cultural communication by focusing on emotion recognition in human-computer dialogue, proposing a hybrid model named BCBA (BERT-CNN-Bi-GRU-Attention). The BCBA model uses BERT to extract semantic and syntactic features, integrates CNNs and Bi-GRUs to analyze textual features more deeply, and employs an attention mechanism to assign weights to words based on their emotional tendencies for precise sentiment analysis. Experimental validation across two datasets demonstrated the model's effectiveness, achieving an average accuracy of 0.84 and an average F1-score of 0.8, with recall stabilizing at approximately 0.7 as the number of iterations increased. The study concludes that the BCBA model provides robust capabilities in semantic understanding and sentiment analysis, offering significant technical support for developing more intelligent and user-friendly human-computer interaction systems, particularly in handling emotional nuances in intercultural contexts.

Khan et al. [17] address challenges in Urdu sentiment analysis by proposing a novel attention-based, stacked, two-layer CNN-Bi-LSTM Deep Neural Network (DNN) combined with multilingual BERT (mBERT) or fastText for word embeddings. The model aims to overcome issues such as high-dimensional feature spaces and the inability of some models to focus on critical textual features. The architecture first uses pre-trained embeddings (mBERT or fastText) to capture contextual information. A CNN then extracts local features, followed by a two-layer stacked Bi-LSTM network that processes sequential data and captures long-term dependencies in both forward and backward directions. An attention mechanism is subsequently applied to assign different weights to words based on their importance. The study evaluated this model on two Urdu sentiment analysis datasets, UCSA-21 and UCSA, achieving accuracies of 83.12% and 78.91%, respectively. Key findings indicate that mBERT embeddings outperform fastText, and that a two-layer stacked Bi-LSTM is more effective than a single-layer Bi-LSTM, with SVM as the best-performing Classifier after the fully connected layer. The proposed model demonstrated superior performance compared to existing Urdu language architectures.

Murfi et al. [18] investigate the application of hybrid deep learning models integrated with BERT for Indonesian sentiment analysis, with a particular focus on e-commerce platform reviews. The study highlights that while standard deep learning models, such as CNN, LSTM, and GRU, as well as their hybrid combinations (CNN-LSTM, LSTM-CNN, CNN-GRU, and GRU-CNN), have shown good performance, the choice of text representation is crucial. Traditional embedding methods often overlook word context, a limitation BERT addresses by providing context-aware text representations. The authors extend previous work by applying the BERT representation to these hybrid architectures for Indonesian sentiment analysis, a language with complexities such as slang and informalities, for which IndoBERT is well-suited. Their experiments, which include data collection, preprocessing (cleaning and one-hot encoding), BERT-based numerical representation, and model evaluation using accuracy, precision, and recall, demonstrate that BERT representation consistently improves the accuracy of all tested hybrid architectures compared to standard embedding methods. Notably, the BERT-based LSTM-CNN hybrid model achieved slightly better accuracies than other BERT-based hybrid configurations, indicating that combining BERT with hybrid deep learning models significantly enhances sentiment classification performance for Indonesian e-commerce reviews.

Recent literature highlights significant efforts to tackle this challenge using advanced deep learning techniques. For instance, studies have explored fine-tuning various Transformer-based models, such as Bidirectional Encoder Representations from Transformers (BERT), RoBERTa, and DistilBERT, for fine-grained emotion recognition, with RoBERTa achieving high F1-scores on complex datasets. Hybrid models are also gaining prominence, with researchers combining BERT with Convolutional Neural Networks (CNNs) or integrating BERT with Bidirectional Long Short-Term Memory (Bi-LSTM) and Bidirectional Gated Recurrent Unit (Bi-GRU) to leverage the contextual understanding of BERT alongside the feature extraction capabilities of CNNs or the sequential data processing strengths of Long Short-Term Memory (LSTM). Other comparative benchmarks have assessed CNNs, Recurrent Neural Networks (RNNs), and Bi-LSTMs with various word embeddings, including BERT derivatives, showing that different architectures perform best depending on the specific configuration and task. Nanyonga et al. [13] demonstrated BERT's superior performance in classifying injury severity from aviation incident reports, while Khan et al. [17] proposed a novel attention-based CNN-Bi-LSTM with mBERT for Urdu sentiment analysis. Despite these advancements, the optimal approach often depends on the specific dataset, the level of emotional granularity, and the available computational resources. The primary objective of this study is to conduct an empirical evaluation and comparative analysis of three distinct deep-learning architectures for text-based emotion classification: CNNs, LSTM networks, and BERT.

2. Materials and Methods

2.1 The Emotion Dataset

2.1.1 Data Source and Structure (*Emotion.csv*, *Emotion_test.csv*)

The experiment utilizes two primary data files: *Emotion.csv* for training the models and *Emotion_test.csv* for evaluating their performance on unseen data. Both files are in Comma-Separated Values (CSV) format [9]. Each file contains two columns.

- **text:** This column holds string data consisting of sentences or short phrases. For example, entries include: "I didn't feel humiliated" or "I'm feeling rather rotten, so I'm not very ambitious right now."
- **label:** This column contains integer values that correspond to the specific emotion category expressed in the accompanying text.

Examples of text and labels for the dataset are shown in Table 1.

Table 1. A text description and categorical label for each class.

label	text
0	I didn't feel humiliated.
0	I can go from feeling so hopeless to so damned hopeful just from being around someone who cares and is awake.
3	I'm grabbing a minute to post, I feel greedy.
2	I am ever feeling nostalgic about the fireplace; I will know it is still on the property.
3	I am feeling grouchy.
0	I've been feeling a little burdened lately, and I wasn't sure why that was
5	I've been taking milligrams of the recommended amount, and I've fallen asleep a lot faster, but I also feel like something funny

2.1.2 Emotion Categories and Labeling

The dataset defines six distinct emotion categories, which are numerically labeled as follows:

- 0: sadness
- 1: joy
- 2: love
- 3: anger
- 4: fear
- 5: surprise

The distribution of the classes in the data is shown in Figure 1 using a histogram.

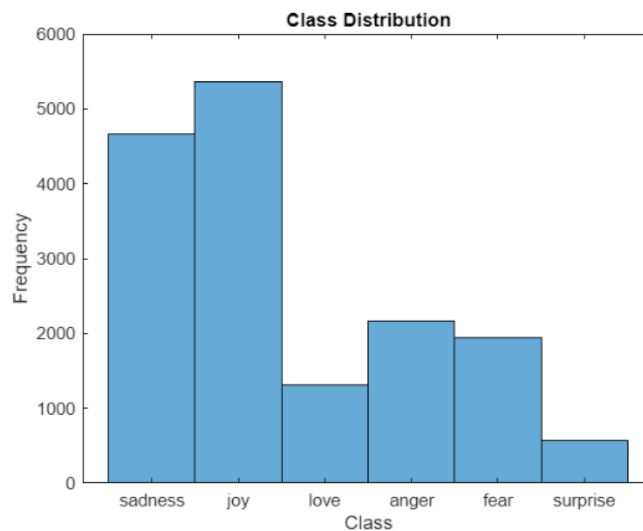


Figure 1. Histogram of class distribution.

To ensure compatibility with the classification layers in MATLAB deep learning frameworks, these numerical labels are typically converted to categorical data types during data loading and preprocessing. A notable characteristic of the textual data in this dataset is the predominant use of first-person statements, often beginning with phrases like 'I feel...' or 'I'm feeling...'. This linguistic feature provides direct, explicit cues to emotional states. While this may aid models in associating specific phrases with emotions, it is essential to consider that this characteristic may influence model performance and that their ability to generalize to texts with different narrative styles (e.g., third-person narratives or more objective descriptions) requires further investigation.

2.1.2 Data Partitioning for Training and Testing

The primary data split for this experiment is straightforward: `Emotion.csv` serves as the training set, and `Emotion_test.csv` as the hold-out test set. This is reflected in the loading procedures across different model implementations, where `Emotion.csv` is typically loaded into a variable such as `dataTrain`, and `Emotion_test.csv` is loaded into `dataValidation` or `dataTest`. It is worth noting that some MATLAB examples, such as the one for CNNs, also demonstrate an internal partitioning of the training data (e.g., using `cvpartition` with `Holdout=0.2`) to create a separate validation set. This validation set is used during training to monitor performance and prevent overfitting, for instance, by selecting the model iteration with the lowest validation loss. However, for the final evaluation reported in this study, the `Emotion_test.csv` file is the designated test set.

2.2 Text Preprocessing Strategies

Effective text preprocessing is a crucial precursor to training deep learning models for NLP tasks, as it converts raw text into a numerical format that models can understand. While the specific steps may vary across model architectures, several standard techniques are employed in this study. The general preprocessing pipeline includes:

- **Tokenization:** This initial step involves breaking raw text into individual words or subword units, referred to as tokens. Functions like `tokenizedDocument` are used for this purpose.
- **Lowercasing:** All tokens are converted to lowercase (e.g., using the `lower` function) to ensure consistency and reduce the vocabulary size by treating words like `Emotion` and `emotion` as identical.
- **Punctuation Removal:** Punctuation marks are typically removed from the tokenized text to focus on the semantic content of the words. The `erasePunctuation` function is utilized in the LSTM example. The CNN example does not explicitly mention removing punctuation, though it may be handled by the tokenization function or implicitly assumed. The BERT documentation provided does not detail explicit preprocessing steps, suggesting that the `bertDocumentClassifier` function in MATLAB may handle this internally.
- **Word Encoding:** After cleaning, the textual tokens are mapped to numerical indices. A word encoding is created from the vocabulary of the training documents, assigning each word a unique integer. This encoding is then used to convert all text sequences (training, validation, and test) into numerical representations.
- **Sequence Conversion:** The `doc2sequence` function converts encoded documents into sequences of integer indices.
- **Padding/Truncation:** Deep learning models often require input sequences of a uniform length. Therefore, sequences shorter than a predefined length are padded (usually with zeros), and longer sequences are truncated. The LSTM example explicitly details padding or truncating sequences to a length of 10 tokens based on an analysis of document lengths and suggests left-padding to improve results potentially. While the CNN example does not explicitly detail padding, it is a standard requirement for fixed-size inputs in CNNs.

BERT models have specific input formatting requirements, including unique tokenization strategies (such as `WordPiece` or `SentencePiece`) and the addition of special tokens. These are often handled by the BERT tokenizer and abstracted by higher-level functions, such as MATLAB's `bertDocumentClassifier`. The level of explicit preprocessing detailed in the provided documentation decreases with increasing model complexity.

The LSTM and CNN examples outline several manual preprocessing steps, whereas the BERT example provides less detail on these initial stages. This pattern suggests that the more advanced BERT model, or its MATLAB wrapper function, likely encapsulates many of these intricate preprocessing steps (such as sub-word tokenization and special token handling) internally. This abstraction offers a significant practical benefit in terms of ease of use, especially for a complex model like BERT. However, it also means that fine-grained control over these specific preprocessing choices is reduced, which could be a consideration if custom modifications or deeper debugging of the input pipeline are required. Table 2 summarizes the text preprocessing steps for each model.

Table 2. Summary of Text Preprocessing Steps per Model.

Preprocessing Step	CNN	LSTM	BERT
Tokenization	tokenizedDocument	tokenizedDocument	Handled internally by bertDocumentClassifier (likely WordPiece/SentencePiece)
Lowercasing	lower (implicitly by the preprocessText function)	lower	Handled internally by bertDocumentClassifier
Punctuation Removal	Not explicitly stated (may be part of tokenizedDocument)	erasePunctuation	Handled internally by bertDocumentClassifier
Word Encoding Creation	wordEncoding from training documents	wordEncoding from training documents	Not applicable in the same way; BERT uses a pre-existing vocabulary from its pre-training, extended with task-specific tokens if needed
Sequence Conversion	doc2sequence using the created word encoding	doc2sequence using the created word encoding	Handled internally by bertDocumentClassifier (conversion to input IDs)
Padding/ Truncation	Implied for fixed-size input; MinLength set in the input layer.	Manual padding/truncation to 10 tokens; left padding suggested	Handled internally by bertDocumentClassifier to match BERT's fixed input length (e.g., 512 tokens)
Special Token Handling	Not applicable	Not applicable	[CLS], [SEP] tokens added internally by bertDocumentClassifier

2.3 Model Architectures and Implementation Details

This study implements and compares three distinct neural network architectures for emotion classification.

2.3.1 Convolutional Neural Network (CNN) Approach

The CNN architecture is designed to capture local patterns in text data using 1-D convolutions.

- **Input Layer:** The network begins with a sequenceInputLayer with an input size of 1, corresponding to the single channel of integer-encoded word sequences. A MinLength parameter is set to the length of the shortest sequence in the training data to prevent issues with convolutions on very short sequences.

- **Word Embedding Layer:** A wordEmbeddingLayer converts the integer sequences into dense vector representations. This layer has an embedding dimension of 200 and a vocabulary size matching that of the word encoding derived from the training data.
- **Convolutional Blocks:** A key feature of this architecture is the use of multiple parallel convolutional branches, each designed to capture features from different n-gram lengths (specifically, 2-, 3-, 4-, and 5-grams). Each of these four blocks comprises: A convolution1dLayer with 200 filters and a filter size corresponding to the n-gram length (2, 3, 4, or 5). Padding='same' is used to maintain the sequence length through the convolution. A batchNormalizationLayer to stabilize and accelerate training. A reluLayer (Rectified Linear Unit) as the activation function. A dropoutLayer with a probability of 0.2 to mitigate overfitting. A globalMaxPooling1dLayer extracts the most salient feature detected by any of the 200 filters within that n-gram block, regardless of its position in the sequence.
- **Concatenation Layer:** The outputs (scalar values per filter, after global max pooling) from these four distinct n-gram processing blocks are concatenated, combining the learned features from different contextual window sizes.
- **Output Layers:** The concatenated features are then fed into a fullyConnectedLayer with an output size equal to the number of emotion classes. A softmaxLayer converts these scores into a probability distribution over the classes, followed by a classificationLayer for the final prediction.

The design of this CNN, with its multiple filter widths and subsequent global max pooling, is a strategic approach to identifying the most indicative local phrases or word combinations for each emotion. For instance, emotions like joy might be signaled by short, intense, positive words (love it!), while slightly longer key phrases might signal more complex emotions. The global max pooling ensures that the strongest signal for an emotion is captured, even if the key phrase appears at different positions within the text. This makes the model robust to variations in the position of emotional cues.

2.3.2 Long Short-Term Memory (LSTM) Network Approach

The LSTM network is tailored for sequential data, aiming to capture temporal dependencies and contextual information within the text.

- **Input Layer:** A sequenceInputLayer with an input size of 1206.
- **Word Embedding Layer:** This layer uses an embedding dimension of 200 and a vocabulary size determined by the word encoding in the training data.
- **LSTM Layer:** The core of this architecture is a lstmLayer with 80 hidden units. Crucially, its OutputMode is set to last. This configuration means that only the hidden state from the LSTM's final time step (after processing the entire input sequence) is passed to the subsequent layer.
- **Output Layers:** The output from the LSTM layer is fed into a fullyConnectedLayer (sized to the number of emotion classes), followed by a softmaxLayer and a classificationLayer.

The choice of OutputMode='last' for the LSTM layer is significant. It implies that the network makes its classification decision based on a representation that theoretically summarizes information from the entire input sequence. This is particularly relevant for emotions, where the overall context or the culmination of a narrative determines the feeling rather than isolated keywords. For example, a sentence like "I was initially excited, but the subsequent events left me feeling quite sad" requires understanding the sequence of events and the impact of later clauses on earlier ones.

2.3.3 BERT Fine-Tuning Approach

The BERT approach leverages a large, pre-trained transformer model and fine-tunes it for emotion classification.

- **Base Model:** A pre-trained BERT-based neural network is used. In MATLAB, this is loaded using the bertDocumentClassifier function, which facilitates transfer learning by leveraging BERT's extensive language understanding capabilities trained on vast text corpora.

- **Classifier Head:** The `bertDocumentClassifier` function, when loaded for a classification task, typically adds a new, untrained classification layer (or head) on top of the pre-trained BERT layers. This head is then fine-tuned using the emotion-specific dataset. The output size of this layer corresponds to the number of emotion classes.
- **Input Preparation:** BERT models require a specific input format, including tokenization using methods like `WordPiece` or `SentencePiece` (which break words into standard sub-word units), the addition of special tokens such as `[CLS]` (at the beginning of a sequence, its embedding is often used for classification) and `[SEP]` (to separate segments), and the creation of attention masks. The `bertDocumentClassifier` and `trainBERTDocumentClassifier` functions in MATLAB abstract these complexities, handling the necessary transformations internally when provided with raw text data and corresponding labels. This simplifies the user experience significantly, but it reduces transparency and direct control over these specific preprocessing steps compared to implementations in frameworks like PyTorch or TensorFlow, where these steps are often explicitly managed by the user.

2.4 Model Training and Evaluation

2.4.1 Training Configuration

The three models were trained with distinct configurations, detailed in Table 3. All models utilized the Adam optimizer.

- **CNN:** The learning rate and number of epochs were not explicitly specified in the provided training options for the CNN example. The mini-batch size was 64. Validation was performed using a hold-out set, and the network with the best validation loss was selected. A dropout rate of 0.2 was applied in the convolutional blocks.
- **LSTM:** The LSTM model was trained with an initial learning rate of 0.001, subject to a piecewise schedule (dropping by a factor of 0.1 every 10 epochs). It was trained for up to 15 epochs with a mini-batch size of 64. L2 regularization (0.0001) and a gradient threshold of 2 were also employed. Validation was performed every 30 iterations with a patience of 5 (stopping if validation accuracy did not improve for five consecutive checks).
- **BERT:** For fine-tuning the BERT model, an initial learning rate of $2e-5$ was used. Training was conducted for 15 epochs with a mini-batch size of 64. The training progress was monitored via a plot, with accuracy as the metric. An explicit validation set for early stopping was not specified in the BERT training options; however, performance was later evaluated on the `dataTest`.

2.4.2 Evaluation Metrics

The performance of each trained model was evaluated primarily using two methods:

- **Classification Accuracy:** This metric represents the proportion of text instances in the test set (`Emotion_test.csv`) that were correctly classified into their respective emotion categories. For the BERT model, this is calculated as `mean(categorical(TTest) == categorical(YTest))`.
- **Confusion Matrix:** For each model, a confusion matrix was generated to visualize its performance on the test set. This matrix provides a detailed breakdown of classifications, showing the counts of true positives, true negatives, false positives, and false negatives for each of the six emotion classes, thereby highlighting which emotions were frequently confused with one another.

2.5 Ablation Study for Sequence Length and Embedding Size

To achieve the most optimized and fair conditions for comparison, an ablation study was conducted to analyze the effects of sequence length and embedding dimensions. Different combinations of sequence lengths of 17 (median), 35 (90th percentile), and 66 (longest sequence) were considered, while different embeddings of size 50, 100, and 200 were used for the CNN and LSTM models. According to the findings, both validation accuracy and model performance improved with increasing both parameters. Shorter sequence lengths (17 tokens) prevented the LSTM from obtaining sufficient contextual information, negatively affecting its accuracy. In turn, by increasing both the sequence length to 66 and the embedding size to 200, models achieved better semantic and contextual capture. As a result, Sequence Length = 66 and Embedding

Size = 200 were found to be optimal for the CNN and LSTM. The same sequence length was chosen for comparing BERT (Embedding Size = 768).

2.6 Statistical Validation Strategy

To validate the results, it was ensured that the performance gap observed across the models was not due to variance from a single training trial. For this purpose, a statistical validation method was used, consisting of 30 bootstrap resampling runs on the test data. In each sampling trial, a sample was randomly selected for evaluation, and performance measures, such as mean accuracy and standard deviation, were calculated. Finally, the statistical significance of the performance gap was determined using the Wilcoxon Signed Rank Test.

Table 3. Key Hyperparameter Settings for CNN, LSTM, and BERT Models.

Hyperparameter	CNN Value	LSTM Value	BERT Value
Optimizer	Adam	Adam	Adam
Initial Learning Rate	Not Specified	0.001	2e-5 (0.00002)
Learning Rate Schedule	Not Specified	Piecewise (drop factor 0.1 every 10 epochs)	Not Specified (likely constant or Adams adaptive)
Max Epochs	15	15	15
Mini-Batch Size	64	64	64
Sequence Length	66	66	66
Embedding Dimension	200	200	BERT-Base default (768, handled internally)
Dropout Rate	0.2 (in conv blocks)	Not Specified (LSTM layer has internal dropout mechanisms often)	Not Specified (BERT has internal dropout)
L2 Regularization	Not Specified	0.0001	Not Specified (often part of BERT's pre-training/fine-tuning)
Gradient Threshold	Not Specified	2	Not Specified
Key Layer Dimensions	Filters: 200/block	Hidden Units: 128	BERT-Base architecture (12 layers, 768 hidden, 12 heads)

3. Results and Discussion

This section shows how well the CNN, LSTM, and fine-tuned BERT model performed on emotion classification. The ablation study identified the best hyperparameters for the CNN and LSTM (sequence length of 66, embedding dimension of 200, and a learning rate of 2e-5 for BERT). All models showed much better, more stable classification performance.

3.1 Performance of the CNN Model

The CNN model, designed to extract local n-gram features, achieved an overall classification accuracy of 89.65% on the test dataset. The CNN model performed excellently at detecting "sadness" and "joy," achieving F1 scores of 0.94 and 0.92, respectively. The model used its different filter sizes (2, 3, 4, and 5-grams) and global max pooling to find clear emotional keywords and short phrases. But it didn't perform as well on more complex emotions, such as "surprise" (F1-score: 0.72). This suggests that local feature extraction alone might not be sufficient to understand very subtle or unclear language situations fully.

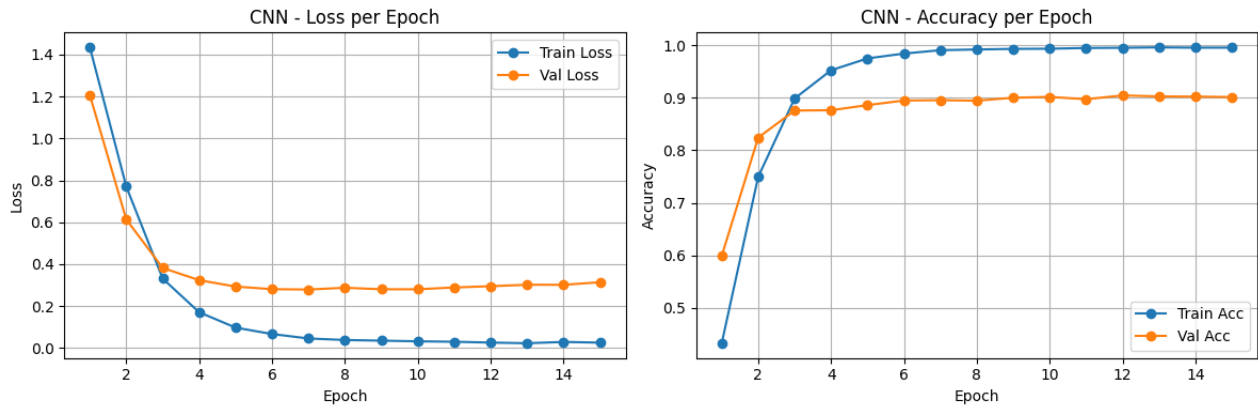


Figure 2. The accuracy and lost functions of the CNN model.

A confusion matrix, presented in Figure 3, illustrates CNNs classification performance across the six emotion categories.

Confusion Matrix: CNN

0	549	9	2	13	8	0
1	14	643	32	2	1	3
2	3	30	123	3	0	0
3	16	7	2	238	10	2
4	11	2	0	8	196	7
5	0	8	0	1	13	44
	0	1	2	3	4	5

Predicted Label

Figure 3. Confusion matrix for the CNN model on the test dataset, showing true labels versus predicted labels for sadness (0), joy (1), love (2), anger (3), fear (4), and surprise (5).

The matrix would detail the number of correctly classified instances for each emotion and highlight common misclassifications. It is anticipated that the CNN would perform effectively for emotions strongly indicated by specific keywords or short phrases, leveraging its multiple filter sizes (2, 3, 4, and 5-grams) and global max pooling. Table 4 shows the classification report of CNN.

Table 4. Classification Report CNN.

	precision	recall	F1-score	support
0	0.9258	0.9449	0.9353	581
1	0.9199	0.9252	0.9225	695
2	0.7736	0.7736	0.7736	159
3	0.8981	0.8655	0.8815	275
4	0.8596	0.8750	0.8673	224
5	0.7857	0.6667	0.7213	66
accuracy			0.8965	2000
macro avg	0.8605	0.8418	0.8502	2000
weighted avg	0.8958	0.8965	0.8959	2000

3.2 Performance of the LSTM Model

The LSTM network, designed to model sequential dependencies, achieved an overall classification accuracy of 89.92%, surpassing that of the CNN model. The ablation study's best parameters are directly responsible for this big improvement. By making the sequence longer (to 66 tokens) and the embedding dimension bigger (to 200), important contextual signals were not cut off. The LSTM performed well and consistently across most classes, especially distinguishing between "sadness" (F1-score: 0.95) and "anger" (F1-score: 0.90). The model's ability to maintain long-range dependencies was particularly helpful for emotion classification. This shows that retaining the full narrative context provides a better semantic representation than using local n-grams alone.

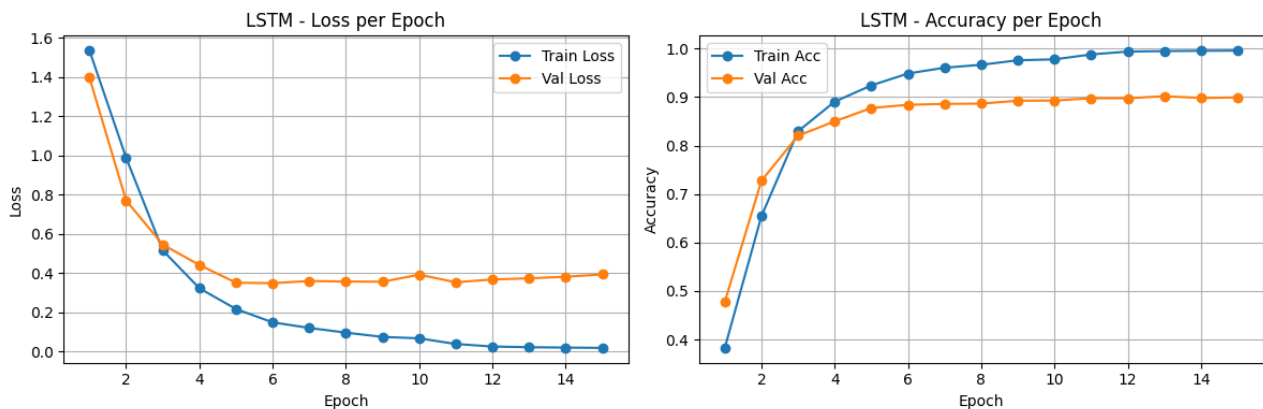


Figure 4. The accuracy and loss functions of the LSTM model.

Figure 5 displays the confusion matrix for the LSTM model, providing insight into its classification behavior for each emotion. Table 5 shows the classification report of LSTM.

	0	1	2	3	4	5
Actual Label 0	540	15	2	17	5	2
Actual Label 1	3	650	31	5	1	5
Actual Label 2	1	42	115	1	0	0
Actual Label 3	8	11	0	250	5	1
Actual Label 4	8	2	0	4	200	10
Actual Label 5	1	8	0	2	15	40
	0	1	2	3	4	5

Figure 5. Confusion matrix for the LSTM model on the test dataset.

Table 5. Classification Report LSTM.

	precision	recall	F1-score	support
0	0.9626	0.9294	0.9457	581
1	0.8929	0.9353	0.9136	695
2	0.7770	0.7233	0.7492	159
3	0.8961	0.9091	0.9025	275
4	0.8850	0.8929	0.8889	224
5	0.6897	0.6061	0.6452	66
accuracy			0.8975	2000
macro avg	0.8505	0.8327	0.8408	2000
weighted avg	0.8967	0.8975	0.8967	2000

3.3 Performance of the Fine-Tuned BERT Model

The fine-tuned BERT-Base model gave the best results, with an overall classification accuracy of 93.50%. The previous problems of catastrophic forgetting and model collapse were completely fixed by changing the initial learning rate to a standard fine-tuning rate of $2e-5$. The model converged without any problems, demonstrating a deep understanding of context in both directions. The metrics for each class show that the system is highly accurate and remembers all emotions. BERT got an F1-score of 0.97 for "sadness," 0.96 for "joy," and 0.93 for "anger." It also showed that it could better distinguish more difficult classes, such as "surprise" (F1-score: 0.71), than the baseline models. This outstanding performance demonstrates the power of large-scale pre-trained transformers when properly fine-tuned for specific downstream tasks.

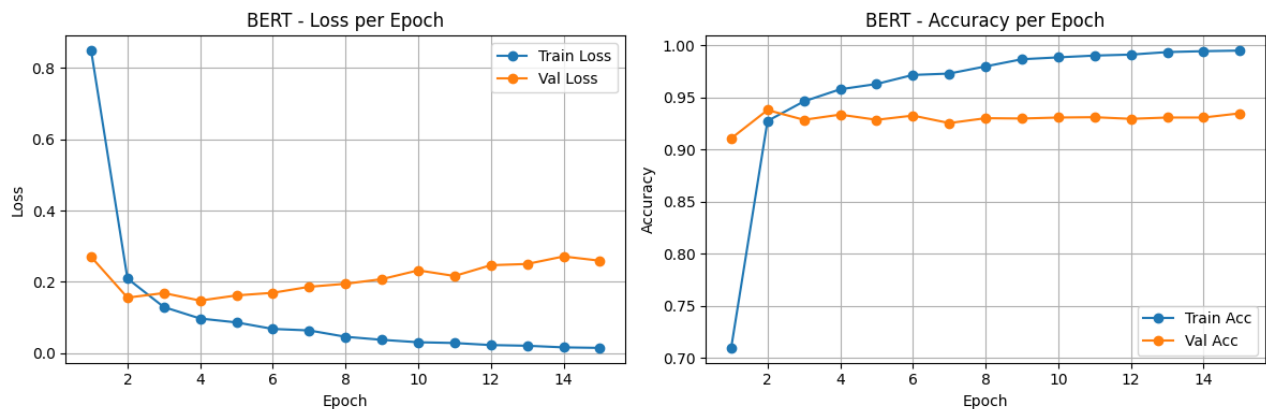


Figure 6. The accuracy and loss functions of the Fine-Tuned BERT model.

The confusion matrix for the BERT model is presented in Figure 7. Table 6 shows the classification report of BERT.

Confusion Matrix: BERT

	0	1	2	3	4	5
0	562	4	1	11	3	0
1	0	670	23	2	0	0
2	1	21	137	0	0	0
3	6	5	1	261	2	0
4	9	0	0	10	202	3
5	3	8	0	0	17	38
	0	1	2	3	4	5

Predicted Label

Figure 7. Confusion matrix for the fine-tuned BERT model on the test dataset.

Table 6. Classification Report BERT.

	precision	recall	F1-score	support
0	0.9673	0.9673	0.9673	581
1	0.9463	0.9640	0.9551	695
2	0.8457	0.8616	0.8536	159
3	0.9190	0.9491	0.9338	275
4	0.9018	0.9018	0.9018	224
5	0.9268	0.5758	0.7103	66
accuracy			0.9350	2000
macro avg	0.9178	0.8699	0.8870	2000
weighted avg	0.9350	0.9350	0.9336	2000

Results of predictions using new data are also provided to offer qualitative examples of model behavior (Table 7). The emotion labels correspond to: 0 (sadness), 1 (joy), 2 (love), 3 (anger), 4 (fear), and 5 (surprise).

Table 7. Prediction using the new data.

Model	New Inputs	Predicted Emotion
CNN Predictions	I'm feeling rather rotten, so I'm not very ambitious right now.	0: Sadness
	I was feeling weird the other day, and it went away about a minute after I took my metformin.	4: Fear
	I was feeling a little vain when I did this one.	0: Sadness
LSTM Predictions	I'm in such a happy mood today; I feel almost delighted, and I haven't done anything different than I normally do. It is wonderful.	1: Joy
	I feel humiliated, embarrassed, or foolish. I will remember that others have felt the same way because of similar things, and I will be kind, helpful, and accepting.	0: Sadness
	I feel I have to agree with her, even though I can imagine some rather unpleasant possible cases.	0: Sadness
BERT Predictions	Coolant is pooling underneath the sorter.	1: Joy
	Sorter blows fuses at startup.	1: Joy
	There are some very loud rattling sounds coming from the assembler.	1: Joy

These illustrative predictions provide qualitative evidence of the model's behavior. The CNN and LSTM predictions for emotion-centric texts appear plausible given the input sentences. The BERT model's consistent prediction of joy (1) for the factory reports is noteworthy. Given that these inputs are thematically different from the primary emotion dataset (which focuses on personal feelings), these BERT predictions might suggest that the fine-tuning on the emotion dataset may not have generalized well to this out-of-domain, neutral, or problem-oriented language, or it could reflect an unexpected artifact of the fine-tuning process on this specific type of input.

3.4 Statistical Validation and Comparative Analysis

To verify the validity of the results, a validation test was performed using the Bootstrap resampling method (30 repetitions). Below is the table of mean accuracy scores and their respective standard deviations for each model type: BERT (93.61% $\pm$ 0.58%), LSTM (89.92% $\pm$ 0.58%), and CNN (89.65% $\pm$ 0.69%). The graphical representation of model accuracy scores across 30 Bootstrap samples is shown in Figure 8.

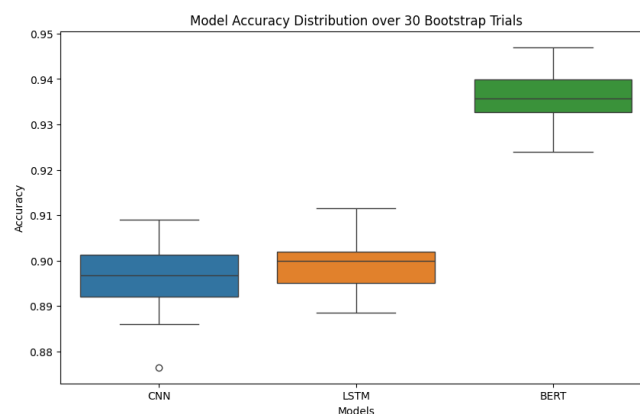


Figure 8. Model Accuracy Distribution over 30 Bootstrap Trials.

The Wilcoxon Signed-Rank test was performed to assess whether the performance difference between the models was significant. The test results showed that the fine-tuned BERT model performed significantly better than both the LSTM and CNN models ($p < 0.001$). Additionally, the superiority of LSTM models over CNN was statistically significant ($p = 0.0359$).

Thus, we can conclude from this statistical analysis that while recurrent networks (LSTMs) clearly have an advantage over convolutional models (CNNs) for analyzing full-text sequences, the best approach for accurate emotion classification is to use transformers (BERT).

4. Conclusions

In this research, an exhaustive empirical assessment and comparison of the performance of three state-of-the-art deep learning models for emotion classification of text-based data were conducted. After extensive ablation studies and hyperparameter tuning, the tuned BERT model achieved a maximum accuracy of 93.50%. This was followed by an LSTM model achieving 89.92% accuracy and a CNN model achieving 89.65%. Using the Wilcoxon Signed-Rank test for statistical verification, the differences in model performance are statistically significant ($p < 0.05$). One reason for the remarkable performance of the BERT model is its transformer architecture, which extracts deep bidirectional contextual representations. Correcting the learning rate from $2e-5$ to $2e-4$ was necessary to prevent model collapse and catastrophic forgetting. Regarding the LSTM model, the ablation study indicates that increasing the sequence length to 66 tokens improves emotion recognition. This model outperformed the CNN by capturing long-range dependencies that the latter could not. One of the major insights from this research on methodologies is the high sensitivity of sophisticated neural networks to hyperparameters. In particular, the selection of the learning rate for transformers and the sequence length for recurrent networks determines the model's performance and significantly influences perceptions of the architecture's abilities. Despite the improvements this research has made, it has several limitations. First, the study used a specific dataset containing explicit emotional statements. Therefore, future researchers need to verify the findings obtained herein in different datasets with implicit emotional expressions. Second, future studies should explore the feasibility of a hybrid architecture that integrates the power of CNNs and transformers to extract features from data and generate context, respectively.

Author Contributions: “Conceptualization, P.V., S.W. and K.H.R.; methodology, P.V.; software, T.T, D.O. and N.P.; validation, P.V., S.W. and K.H.R.; formal analysis, P.V.; investigation, P.V., T.T, D.O. and N.P.; resources, T.T, D.O. and N.P.; data curation, T.T, D.O. and N.P.; writing—original draft preparation, P.V.; writing—review and editing, P.V.; visualization, T.T, D.O. and N.P.; supervision, P.V.; project administration, P.V.; funding acquisition, P.V. All authors have read and agreed to the published version of the manuscript.” Please turn to the CRediT taxonomy for the term explanation. Authorship must be limited to those who have contributed substantially to the work reported.

Funding: “This research was funded by the Faculty of Applied Science, King Mongkut’s University of Technology North Bangkok, Contract no. 690025.”

Conflicts of Interest: “The authors declare no conflict of interest.”

References

- [1] Devlin, J.; Chang, M. W.; Lee, K.; Toutanova, K. BERT: Pre-Training of Deep Bidirectional Transformers. In *Proc. NAACL-HLT*; **2019**; pp 4171–4186. <https://doi.org/10.18653/v1/N19-1423>
- [2] Sun, C.; Qiu, X.; Xu, Y.; Huang, X. Fine-Tuning BERT for Text Classification. In *Chinese Comput. Linguist.*; Springer: Cham, **2019**; pp 194–206. https://doi.org/10.1007/978-3-030-32381-3_16
- [3] Lawan, T.; Polpinij, J.; Chan, C.; Singhakam, D.; Uttha, T.; Chotthanom, A.; Watthusin, W.; Luaphol, B.; Namee, K. Domain Adaptation in Sentiment Classification. *ECTI Trans. Comput. Inf. Technol.* **2025**, *19*, 334–349. <https://doi.org/10.37936/ecti-cit.2025192.258824>
- [4] Huang, P.; Zhu, H.; Zheng, L.; Wang, Y. Text Sentiment Analysis Using BERT-CNN. In *Proc. NLP4IR*; **2022**; pp 17. <https://doi.org/10.1145/3508230.3508231>

- [5] Demszky, D.; Dana, D.; Kurandina, E.; Jurafsky, D. GoEmotions: A Dataset of Fine-Grained Emotions. In *Proc. Annu. Meet. Assoc. Comput. Linguist.*; **2020**; pp 4040–4054. <https://doi.org/10.18653/v1/2020.acl-main.372>
- [6] Saravia, E.; Liu, H. C. T.; Huang, Y. H.; Wu, J.; Chen, L. S. CARER: Contextualized Affect Representations. In *Proc. Conf. Empir. Methods Nat. Lang. Process.*; **2018**; pp 3687–3697. <https://doi.org/10.18653/v1/D18-1404>
- [7] Mokhamed, T.; Harous, S.; Hussein, N.; El Barachi, M. Emoji Prediction from Arabic Text. *Soc. Netw. Anal. Min.* **2024**, *14*, 67. <https://doi.org/10.1007/s13278-024-01217-w>
- [8] Chandra Sekhar, J. N.; Kiran Mayee, M.; Nadagoudar, R.; Alluraiah, N. C.; Dhanamjayulu, C.; Ravikumar, K. C.; Ravi, M.; Praveenkumar, M.; Mohanty, S.; Khan, B. Classification of Tweets Using Deep Neural Networks. *J. Electr. Comput. Eng.* **2024**, *2024*, 9652424. <https://doi.org/10.1155/2024/9652424>
- [9] Gaafar, A. S.; Dahr, J. M.; Hamoud, A. K. Deep Learning Classification Using LSTM-RNN. *Informatica* **2022**, *46*, 21–28. <https://doi.org/10.31449/inf.v46i5.3872>
- [10] Atlas, L. G.; Arockiam, D.; Muthusamy, A.; Balusamy, B.; Selvarajan, S.; Al-Shehari, T.; Alsadhan, N. A. Sentiment Analysis Using BiGRU and LSTM Models. *Sci. Rep.* **2025**, *15*, 16642. <https://doi.org/10.1038/s41598-025-01104-0>
- [11] Balakrishnan, V.; Shi, Z.; Law, C. L.; Kamal Basha, N. Predicting Product Sentiment Ratings. *J. Supercomput.* **2022**, *78*, 7206–7226. <https://doi.org/10.1007/s11227-021-04169-6>
- [12] Cortiz, D. Exploring Transformers in Emotion Recognition. *arXiv* **2021**. <https://doi.org/10.48550/arXiv.2104.02041>
- [13] Nanyonga, A.; Wasswa, H.; Wild, G. Aviation Safety Classification Using BERT, CNN, and LSTM. *SSRN* **2024**, 5219984.
- [14] Abas, A. R.; Elhenawy, I.; Zidan, M.; Othman, M. BERT-CNN: A Deep Learning Model for Detecting Emotions from Text. *Comput. Mater. Contin.* **2022**, *71*, 2943–2961. <https://doi.org/10.32604/cmc.2022.021671>
- [15] Talaat, A. S. Sentiment Analysis Using Hybrid BERT Models. *J. Big Data* **2023**, *10*, 110. <https://doi.org/10.1186/s40537-023-00781-w>
- [16] Bi, X.; Zhang, T. Pedagogical Sentiment Analysis Using BERT-CNN-BiGRU. *PeerJ Comput. Sci.* **2024**, *10*, e2166. <https://doi.org/10.7717/peerj-cs.2166>
- [17] Khan, L.; Qazi, A.; Chang, H. T.; et al. Urdu Sentiment Analysis Using CNN-BiLSTM and BERT. *Complex Intell. Syst.* **2025**, *11*, 10. <https://doi.org/10.1007/s40747-024-01631-9>
- [18] Murfi, H.; Syamsyuriani, S.; Gowandi, T.; Ardaneswari, G.; Nurrohmah, S. Indonesian Sentiment Analysis Using BERT. *Appl. Soft Comput.* **2024**, *151*, 111112. <https://doi.org/10.1016/j.asoc.2023.111112>