CMPSC 497 - Midterm Project Report
Emotion Classification with CNNs and RNNs

Hrishikesh Shenai — hms6207@psu.edu

1 Introduction

In today's digital age, social media platforms like X (previously known as Twitter), Facebook, etc. have become powerful mediums for users to express their thoughts, opinions, and emotions. The vast amount of real-time data generated on such platforms present an opportunity for analyzing public sentiment and emotional trends. **Emotion detection** is a subfield of Natural Language Processing (NLP) that focuses on identifying and categorizing emotions such as happiness, sadness, anger, and fear from textual data. Emotion detection plays a crucial role in various applications, including brand sentiment analysis, mental health monitoring, crisis management, and political sentiment analysis. Traditional sentiment analysis classifies text as positive, negative, or neutral, whereas emotion detection provides a deeper understanding by identifying specific emotions.

This project explores two families of neural networks for emotion detection: **Convolutional Neural Networks** (CNNs) and **Recurrent Neural Networks** (RNNs). These architectures are widely used in NLP due to their ability to capture textual patterns and contextual relationships.

CNNs, originally designed for image processing, have proven effective in text classification tasks, such as in the work of LeCun et al. in their 1998 paper "Gradient-Based Learning Applied to Document Recognition" [1]. They apply convolutional filters to extract important n-gram features from text, allowing the model to identify relevant patterns that contribute to emotion classification. CNNs are particularly useful for capturing **local dependencies within a text**, making them efficient for emotion detection tasks. Their ability to process data in parallel also makes them computationally efficient compared to sequential models.

Unlike CNNs, RNNs are designed to process sequential data by **maintaining a memory of past inputs**, making them well-suited for understanding context in text. Variants like Long Short-Term Memory (LSTM), a concept pioneered by Hochreiter & Schmidhuber in their 1997 paper "Long Short-Term Memory" [2], and Gated Recurrent Units (GRU) improve upon traditional RNNs by addressing the vanishing gradient problem, enabling the model to retain information over longer sequences. This project adapts innovation in the form of bidirectional LSTMs (BiLSTMs) to process the sequential nature of language in human texts effectively. The BiLSTM architecture enhances the LSTM's capabilities by analyzing data in both forward and backward directions, ensuring a richer understanding of context.

1.1 Problem Statement

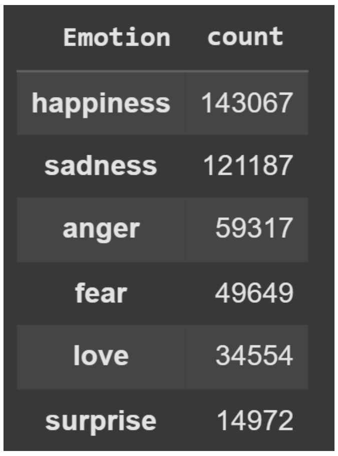
This project aims to develop an emotion detection system for Natural Language Data using deep learning techniques. The system will pre-process the texts, extract meaningful features and classify them into different emotion categories.

The project is guided by several key objectives:

- **Accuracy and Efficiency:** Develop a model that accurately classifies emotions into the correct categories with high efficiency. Minimize misclassification to ensure reliable categorization.
- **Contextual Understanding:** Model should be able to understand the contexts effectively to accurately classify emotions as words can have different meanings based on the context.
- **Handling Slang and Sarcasm:** The model should minimize being misled during classification by slang terms and sarcasms, which is commonly used by today's generation on social media platforms.
- **Generalization Capability:** Ensure the model generalizes well to unseen data, demonstrating its applicability to real-world scenarios beyond the confines of the training dataset.
- **Data Imbalance:** Some emotions are more frequently expressed than others, leading to an imbalance in the dataset that can affect model performance, which should be mitigated.

2 Dataset and Data Curation

This section provides an overview of the dataset used: **Sentiment and Emotion Analysis Dataset** [3]. The data set is a carefully curated collection of textual data that aims to enable researchers and data scientists to explore human emotions in text. The dataset consists of **422,000 labeled sentences**, categorized into six distinct emotions: happiness, sadness, anger, fear, love, and surprise. Fig 1 displays the number of text samples present in the dataset for each of the classes.



Emotion	count
happiness	143067
sadness	121187
anger	59317
fear	49649
love	34554
surprise	14972

Figure 1: Emotion Counts

A comprehensive analysis of text lengths during the **Exploratory Data Analysis (EDA)** phase provided key insights into the structure of the dataset. As seen in in Fig 2, the average text content length was found to be around 20 words, with a maximum length close to 75 words. However, the majority of texts contained fewer than 25 words, while only a small fraction exceeded 50 words. This highlights the brevity and conciseness of text content, reinforcing the importance of designing models that can effectively extract meaningful patterns from short texts.

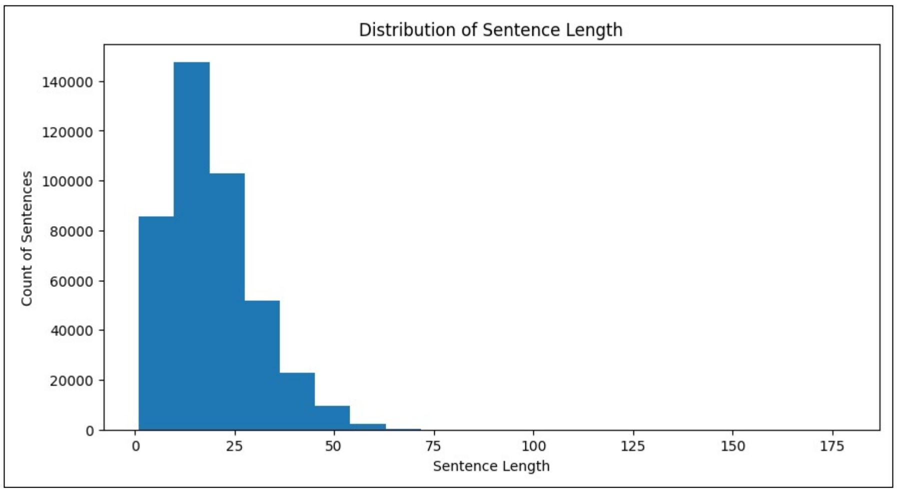


Figure 2: Distribution of Length of Sentences

A significant observation during the EDA was the imbalance in emotion distribution across the dataset as shown in the Fig 3. The "Happiness" and "Sadness" categories contained the highest number of samples, with over 250,000 instances combined, whereas the other categories had a much lower representation. This imbalance introduced potential biases in model training, where the model might favor majority classes, leading to misclassification of underrepresented emotions.

Class imbalance was a significant challenge in this project, as certain emotions like happiness and sadness had a much higher representation compared to fear, love and surprise. To ensure a balanced

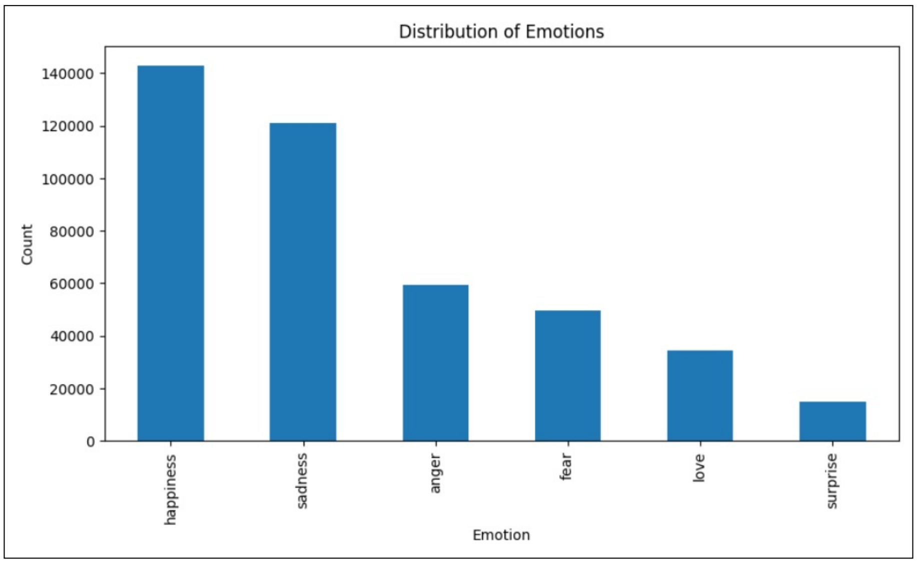


Figure 3: Distribution of Text Samples per Category

dataset and improve model performance, additional data points were sourced from another dataset [4] containing **English Twitter messages** labeled with the same six basic emotions: anger, fear, joy (happiness), love, sadness, and surprise. Additional samples were extracted for the underrepresented classes (love and surprise) from this dataset. At the same time, some randomly selected samples pertaining to "Happiness" and "Sadness" categories were dropped. This helped mitigate the skewed distribution and provided the model with a richer variety of emotion-labeled text. The Fig 4 shows the distribution of the emotion categories after resampling the dataset. While the counts for the "Surprise" category remains low, the dataset is relatively balanced and ready for pre-processing.

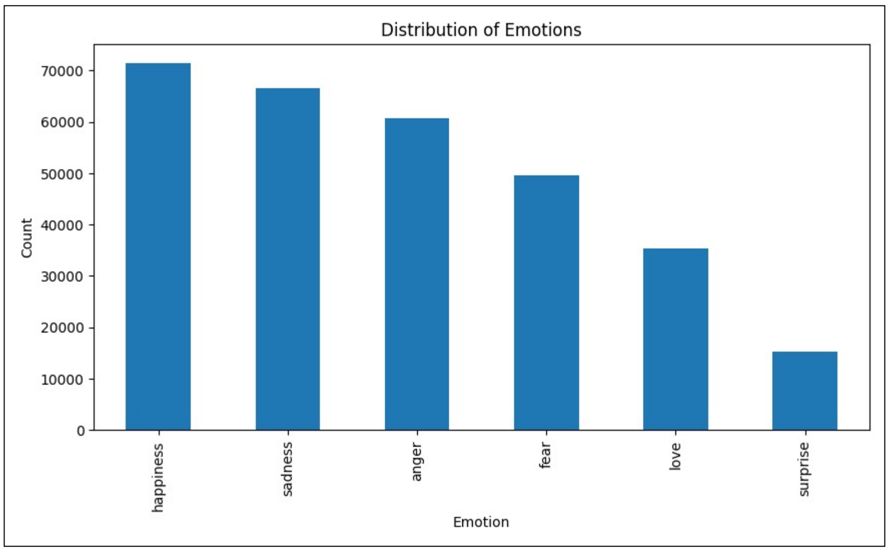


Figure 4: Distribution of Text Samples per Category after resampling

2.1 Data Pre-Processing

To enhance the model's ability to understand textual data, word embeddings were employed. Word embeddings transform raw text into dense vector representations that capture semantic meaning, context, and relationships between words. Data pre-processing is crucial before generating word embedding as raw textual data often contains noise, inconsistencies, and irrelevant elements that can negatively impact model performance.

- **Lowercasing:** Standardized all text to lowercase to ensure consistency.

- **Text Cleaning:** Stopwords, HTML tags, special characters (emoticons, emojis), punctuation marks, numbers and symbols were removed from raw text data. Unclean text with unnecessary characters can lead to misleading vector representations, reducing the effectiveness of embeddings in capturing semantic meaning.
- **Expanding Contractions:** To provide explicit representation to models and avoid ambiguity, expanded contractions (shortened forms of words or combinations) to their full forms. Texts often contain misspellings and slangs which can introduce unwanted variability in embeddings.
- **Word Lemmatization:** Reduced words to their base or root form through lemmatization, treating different variations of words as the same. This reduces redundancy, making the embedding space more efficient.
- **Empty Strings Removal:** Removed empty strings or whitespaces that do not contribute meaningful information to the data.
- **Tokenization:** The text was split into individual words or subword units, preparing it for embedding representation.

3 Word Embeddings

As mentioned, word embeddings play a crucial role in transforming raw text into **dense vector representations**, capturing semantic relationships, contextual meaning, and word similarity. They help improve model performance by providing a structured way to understand words beyond simple one-hot encoding.

3.1 GloVe Word Embeddings

In this project, GloVe (Global Vectors for Word Representation) [5] embeddings were used due to their effectiveness in capturing both local and global co-occurrence information. Unlike Word2Vec, which learns embeddings based on context windows, GloVe constructs word vectors by analyzing word co-occurrence statistics across the entire corpus, making it particularly powerful for sentiment and emotion classification tasks.

3.2 Key characteristics of GloVe embeddings:

- **Trained on a Vast Corpus:** The embeddings were pretrained on the Common Crawl dataset, comprising 400K unique word vectors, ensuring strong generalization across diverse text data.
- **Preserves Semantic Relationships:** GloVe effectively captures the contextual similarities between words, positioning happy closer to joyful than sad in the vector space.
- **Compact and Computationally Efficient:** Unlike sparse representations, GloVe generates dense, low-dimensional word vectors, enhancing both memory efficiency and model performance.

3.3 Integrating GloVe embeddings into the Neural Network models:

- An embedding dimension of 200 was chosen to balance computational efficiency with accuracy.
- The GloVe embeddings were stored in an embedding matrix, where each word in the dataset was mapped to its corresponding 200-dimensional vector.
- If a word was not found in the pretrained embeddings, its vector was initialized with zeros to maintain uniform input dimensions.
- The embedding matrix was incorporated into the model using TensorFlow's Embedding Layer, allowing the network to learn refined word representations during training. By leveraging GloVe embeddings, the model gained a richer contextual understanding of words, improving its ability to accurately classify emotions.

4 Neural Network Architectures

4.1 Convolution Neural Network

CNNs are adept at spatial hierarchies and feature extraction, primarily through convolution filters that capture patterns in spatial data. The architecture 5 excels in image recognition, object detection, and certain natural language processing tasks.

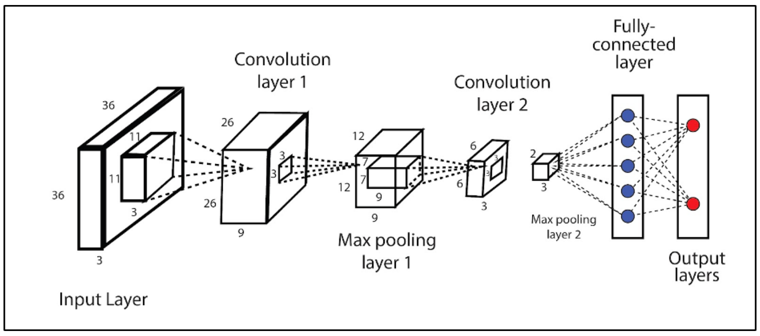
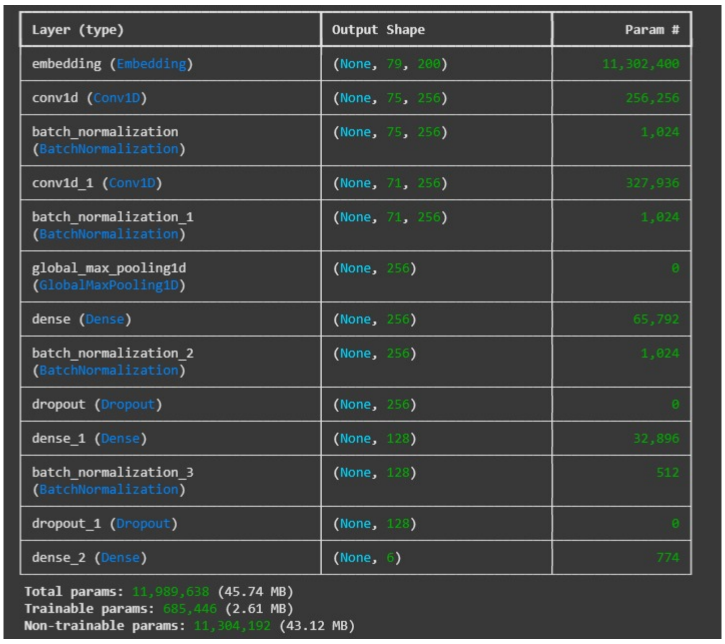


Figure 5: Architecture of a Convolution Neural Network [6]



```
Layer (type)                      Output Shape               Param #
embedding (Embedding)             (None, 79, 200)         11,302,400
conv1d (Conv1D)                   (None, 75, 256)            256,256
batch_normalization               (None, 75, 256)              1,024
(BatchNormalization)
conv1d_1 (Conv1D)                 (None, 71, 256)            327,936
batch_normalization_1             (None, 71, 256)              1,024
(BatchNormalization)
global_max_pooling1d              (None, 256)                      0
(GlobalMaxPooling1D)
dense (Dense)                     (None, 256)                 65,792
batch_normalization_2             (None, 256)                  1,024
(BatchNormalization)
dropout (Dropout)                 (None, 256)                      0
dense_1 (Dense)                   (None, 128)                 32,896
batch_normalization_3             (None, 128)                    512
(BatchNormalization)
dropout_1 (Dropout)               (None, 128)                      0
dense_2 (Dense)                   (None, 6)                      774
Total params: 11,989,638 (45.74 MB)
Trainable params: 685,446 (2.61 MB)
Non-trainable params: 11,304,192 (43.12 MB)
```

Figure 6: CNN Model Summary

Model Architecture:

- **Embedding Layer:** Maps 56,512 words into 200-dimensional vectors using GloVe pre-trained embeddings.
- **Convolution Blocks:** Two blocks, each with Conv1d layers with filter sizes of 256, ReLU activations and BatchNormalization. Filters of varying sizes extract textual features at multiple scales.
- **Global Max Pooling:** Reduces dimensional while retaining critical features.
- **Batch Normalization:** Helps stabilize training and improves performance. It normalizes the activations of a layer (Dense) across a batch of data. It often helps in faster convergence and can sometimes lead to better generalization.
- **Dropout and Dense Layer:** A dropout rate of 0.3 prevents over-fitting. The Dense (fully connected) layer maps features to 6 classification categories. Softmax activation is used here.

4.2 RNN (SimpleRNN)

RNNs 7 have the concept of "memory" that helps them store the states or information of previous inputs to generate the next output of the sequence. In the feedforward pass of an RNN, the network computes the values of the hidden units and the output after 'k' time steps. The weights associated with the network are shared temporally.

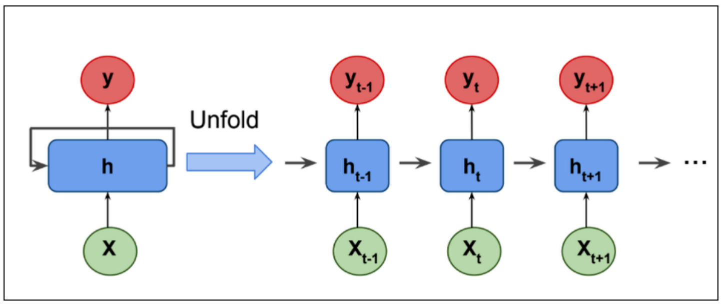
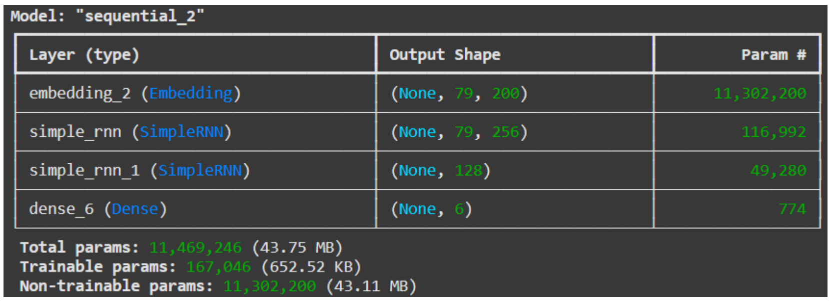


Figure 7: Architecture of Simple RNN [8]



```
Model: "sequential_2"
┌──────────────────────────────────┬──────────────────────────┬──────────────┐
│ Layer (type)                     │ Output Shape             │      Param # │
├──────────────────────────────────┼──────────────────────────┼──────────────┤
│ embedding_2 (Embedding)          │ (None, 79, 200)          │   11,302,200 │
├──────────────────────────────────┼──────────────────────────┼──────────────┤
│ simple_rnn (SimpleRNN)           │ (None, 79, 256)          │      116,992 │
├──────────────────────────────────┼──────────────────────────┼──────────────┤
│ simple_rnn_1 (SimpleRNN)         │ (None, 128)              │       49,280 │
├──────────────────────────────────┼──────────────────────────┼──────────────┤
│ dense_6 (Dense)                  │ (None, 6)                │          774 │
└──────────────────────────────────┴──────────────────────────┴──────────────┘
 Total params: 11,469,246 (43.75 MB)
 Trainable params: 167,046 (652.52 KB)
 Non-trainable params: 11,302,200 (43.11 MB)
```

Figure 8: Simple RNN Model Summary

Model Architecture:

- **Embedding Layer:** Mirrors the CNN's embedding specifications.
- **RNN Layers:** Fully-connected SimpleRNN layers with filter sizes 256 and 128, where the output is fed back as the new input.
- **Linear Transformation and Dropout:** A linear layer with ReLU activation interpret LSTM features. A dropout rate of 0.20 offers regularization.
- **Output Layer:** A dense (fully-connected) layer that maps features to 6 emotion categories.

4.3 Bi-Directional Long Short Term Memory

RNNs shine in sequential data handling, leveraging memory elements to maintain context. BiLSTMs shown in fig 9 extend RNN capabilities, analyzing sequences in both directions for a comprehensive understanding, crucial for tasks like sentiment analysis.

Model Architecture:

- **Embedding Layer:** Mirrors the CNN's embedding specifications.
- **Bidirectional LSTM Layers:** BiLSTM layers with filter sizes of 256 capture contextual information from text sequences.
- **Dropout:** A dropout rate of 0.20 offers regularization.
- **Output Layer:** Aligns features to 6 emotion categories.

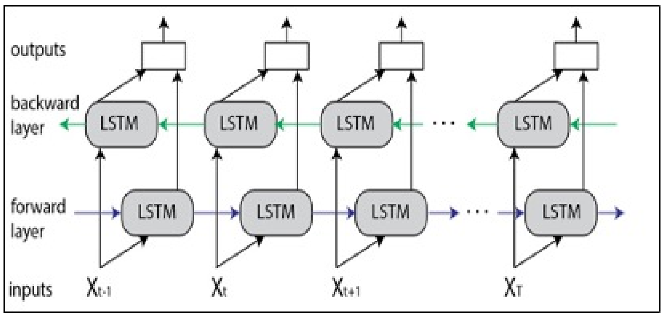
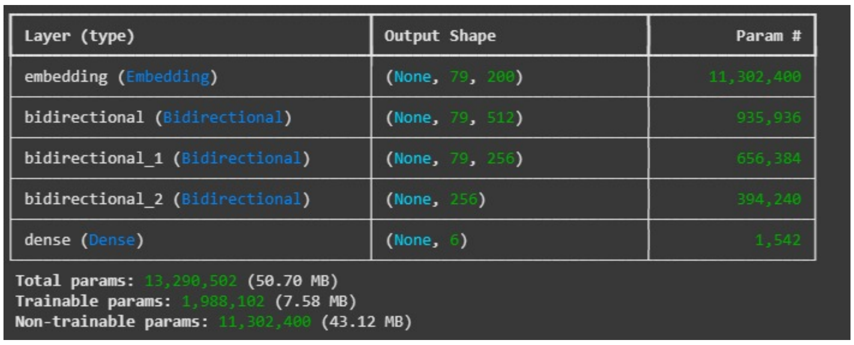


Figure 9: Architecture of Bi-Directional LSTMs [7]



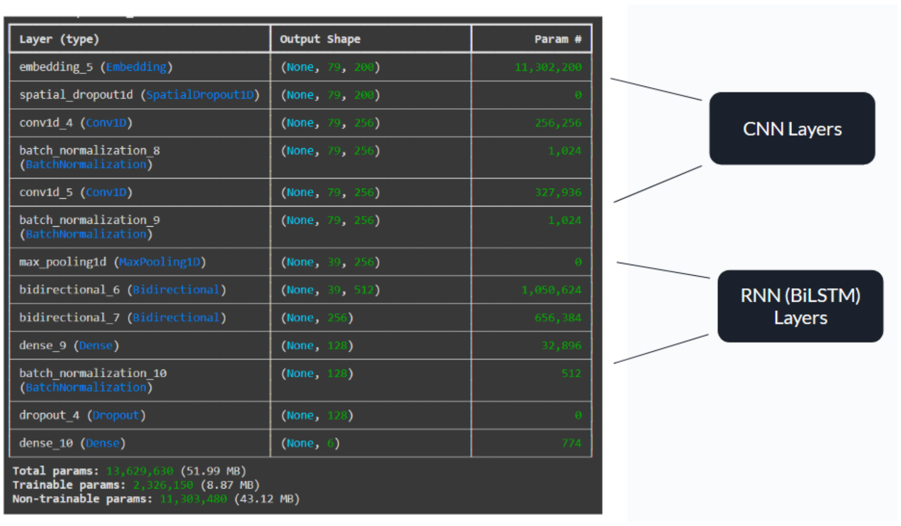
Layer (type)	Output Shape	Param #
embedding (Embedding)	(None, 79, 200)	11,302,400
bidirectional (Bidirectional)	(None, 79, 512)	935,936
bidirectional_1 (Bidirectional)	(None, 79, 256)	656,384
bidirectional_2 (Bidirectional)	(None, 256)	394,240
dense (Dense)	(None, 6)	1,542

```
Total params: 13,290,502 (50.70 MB)
Trainable params: 1,988,102 (7.58 MB)
Non-trainable params: 11,302,400 (43.12 MB)
```

Figure 10: RNN (Bi-Directional LSTMs) Model Summary

4.4 Combined CNN and BiLSTM Neural Network

This neural network architecture aims to combine the merits of CNNs and BiLSTM (RNNs). The CNN layers are used to extract local features (e.g., n-grams) from the text. These features are then fed to the BiLSTM layers to capture longer-range dependencies.



Layer (type)	Output Shape	Param #
embedding_5 (Embedding)	(None, 79, 200)	11,302,200
spatial_dropout1d (SpatialDropout1D)	(None, 79, 200)	0
conv1d_4 (Conv1D)	(None, 79, 256)	256,256
batch_normalization_8 (BatchNormalization)	(None, 79, 256)	1,024
conv1d_5 (Conv1D)	(None, 79, 256)	327,936
batch_normalization_9 (BatchNormalization)	(None, 79, 256)	1,024
max_pooling1d (MaxPooling1D)	(None, 39, 256)	0
bidirectional_6 (Bidirectional)	(None, 39, 512)	1,050,624
bidirectional_7 (Bidirectional)	(None, 256)	656,384
dense_9 (Dense)	(None, 128)	32,896
batch_normalization_10 (BatchNormalization)	(None, 128)	512
dropout_4 (Dropout)	(None, 128)	0
dense_10 (Dense)	(None, 6)	774

```
Total params: 13,629,630 (51.99 MB)
Trainable params: 2,326,150 (8.87 MB)
Non-trainable params: 11,303,480 (43.12 MB)
```

Figure 11: Combined CNN-RNN Model Summary

Model Architecture:

- **Embedding Layer:** Same as before, maps word indices to dense vector representations (GloVe).
- **CNN Layers:**
  - Conv1D: 2 convolution layer with 256 filters and a kernel size of 5. These extract local features from the text.

  - BatchNormalization: Same purpose as in CNN model
  - GlobalMaxPooling1D(): Extracts the most important features from each filter map, reducing the dimensionality to feed into the RNN.

- **RNN Layers:** Two bidirectional LSTM layers. These process the output of the CNN in both forward and backward directions, capturing sequential dependencies.
- **Dense and Dropout Layers:** These layers add non-linearity and help prevent overfitting.
- **Output Layer:** With Softmax activation, aligns model features to 6 classification categories.

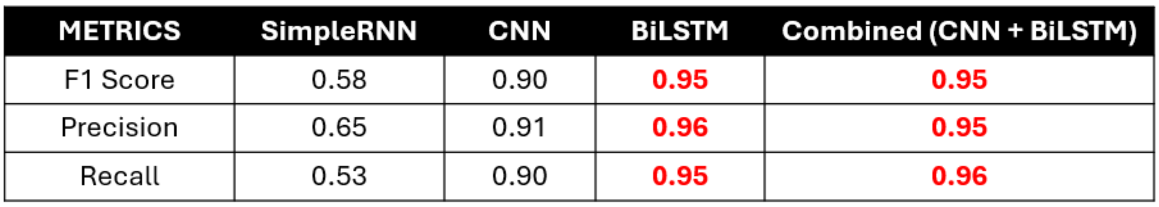
5 Methodology and Training Details

- **Common Data Pre-processing:**
  - **Training Environment:** All models were trained in Google Colab with the T4 GPU (15 GB RAM).
  - **Data Preparation:** Pre-processed the data following the steps mentioned in section 2.1. The data was then split into Train, Validation and Test sets with sizes in ratio 7:1:2.
  - **Label Encoding:** Emotion target categories are encoded to numeric values using LabelEncoder.
  - **Tokenization and Padding:** Text data is converted into padded integer sequences using Keras Tokenizer.
  - **Word Embeddings:** Loaded the 200-dimensional pre-trained GloVe embedding for enhanced word representation.
  - **Early Stopping:** Monitors validation loss and keeps track of the 4 latest model performances to determine when to stop training and prevent overfitting.
  - **Loss and Optimization:** Adopted Categorical Cross-Entropy Loss and Adam [9] optimizer.
  - **Evaluation:** Performance assessed on validation data, monitoring metrics such as loss and accuracy over 30 epochs, with batch size of 128. Post training, the model is evaluated using the test set.
- **CNN Model Training:**
  - **Architecture:** The CNN model employs a range of filter sizes and multiple convolution layers as described in the previous section.
  - **Hyper-parameters:** Learning rate, Filters (256), (0.001), dropout rates (0.5, 0.3), Kernel Size (5) and ReLU activations, optimized using GridSearchCV.
- **SimpleRNN Model Training:**
  - **Architecture:** The RNN model uses Keras SimpleRNN layers with dropout (0.20) for regularization as described in the previous section.
  - **Hyper-parameters:** Learning rate (0.005), Filters (256) and dropout rates (0.2).
- **Bi-LSTM Model Training:**
  - **Architecture:** The BiLSTM model integrates bidirectional LSTM layers with dropout of 0.20 for regularization and ReLU activation as described in the previous section.
  - **Hyper-parameters:** Learning rate (0.005), Filters (256, 128) and dropout rates (0.2), optimized using GridSearchCV.
- **Combined (CNN + BiLSTM) Model Training:**
  - **Architecture:** The model employs two Convolution Layers with BatchNormalization, passing features to two BiLSTM layers with 0.2 dropout as described in the previous section.
  - **Hyper-parameters:** Learning rate (0.001), dropout rates (0.5, 0.3), Kernel Size (5) and ReLU activations, optimized using GridSearchCV.

6 Results

For quantitative analysis, the **F1 score** metric was used, a metric that strikes a balance between recall and precision. Recall measures the model's capacity to capture all pertinent instances of a class, whereas precision indicates the accuracy of positive predictions. Because it is the harmonic mean of recall and precision, the F1 score provides a detailed analysis that is especially useful when class distributions are unbalanced. It is robust as it takes into account both false positives and false negatives, giving a nuanced understanding of a model's effectiveness across a range of class outcomes. Complementing this, the **confusion matrix** was also taken into consideration as a versatile metric for assessing the performance of multi-class classification models. It provides a comprehensive breakdown of predictions, showcasing instances correctly or incorrectly classified per class.

6.1 Quantitative Analysis



METRICS	SimpleRNN	CNN	BiLSTM	Combined (CNN + BiLSTM)
F1 Score	0.58	0.90	**0.95**	**0.95**
Precision	0.65	0.91	**0.96**	**0.95**
Recall	0.53	0.90	**0.95**	**0.96**

Figure 12: Classification metrics across models

The results of all four trained models across the test set for 6 emotion categories are shown in Table 12. Among the models evaluated, SimpleRNN performed the worst, achieving only 58% accuracy, likely due to its inability to effectively capture long-range dependencies and vanishing gradient issues. In contrast, CNN demonstrated strong performance with 92% accuracy, benefiting from its ability to extract local features efficiently. However, BiLSTM and the combined CNN + BiLSTM model outperformed all others, achieving 95% accuracy, thanks to their ability to capture both sequential dependencies and spatial features. This indicates that while CNNs are efficient and effective, RNN-based architectures (especially BiLSTM) are better suited for tasks requiring deeper contextual understanding of text. The CNN + BiLSTM model strikes a balance, leveraging CNN's fast feature extraction and BiLSTM's ability to capture sequential patterns, making it the most effective model for emotion detection in texts.

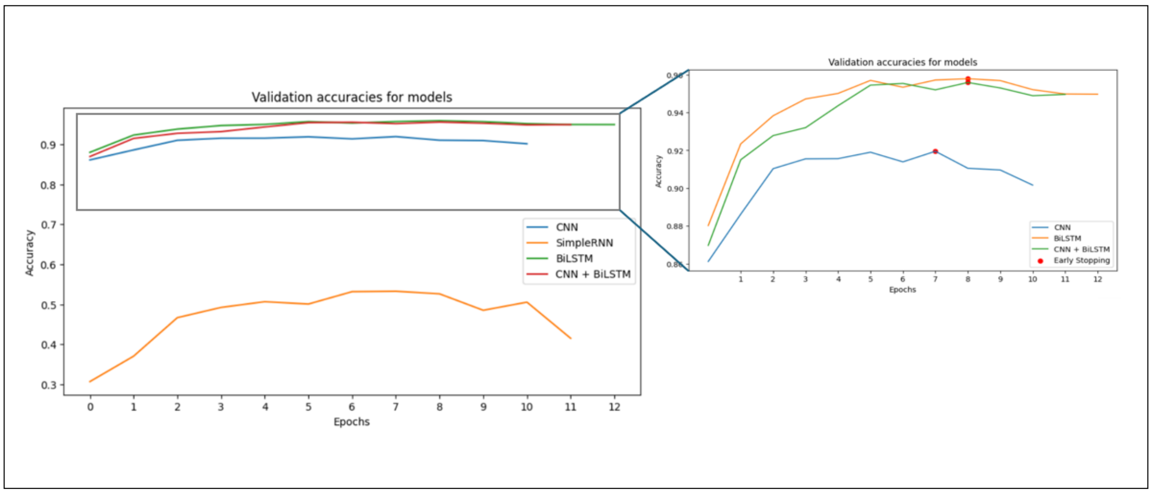


Figure 13: Validation Accuracy per Epoch

Furthermore, the validation accuracy per epoch for the BiLSTM and Combined Models, as shown in Figure 13, is consistently higher than that of the other model implementations. This consistent trend further reiterates their effectiveness in multi-class text classification tasks. The models' ability to maintain higher accuracy throughout the training process indicates its robust learning capacity and adaptability to the complexities present in the dataset. It is also notable to observe early stopping callback getting activated to stop the training as the validation accuracies started dropping, to prevent overfitting.

To delve deeper into the analysis, the confusion matrices for BiLSTM and Combined models are considered. Both models show high accuracy for happiness, sadness, and anger, with the majority of samples correctly classified. For example, in both matrices, 'Happiness' and 'Sadness' categories have the highest correct classifications (diagonal values). 'Anger' also has a high correct classification rate, with over 10,000 correctly identified samples in both models. The models perform best in these categories, likely due to the distinctive word usage associated with these emotions. 'Fear' and 'Surprise' categories show noticeable misclassification. In both matrices, 'Fear' is sometimes misclassified as 'Anger' or 'Surprise', indicating that the model struggles to differentiate between these emotions. The 'Surprise' category has the fewest samples and is often misclassified into other emotions, demonstrating the impact of class imbalance.

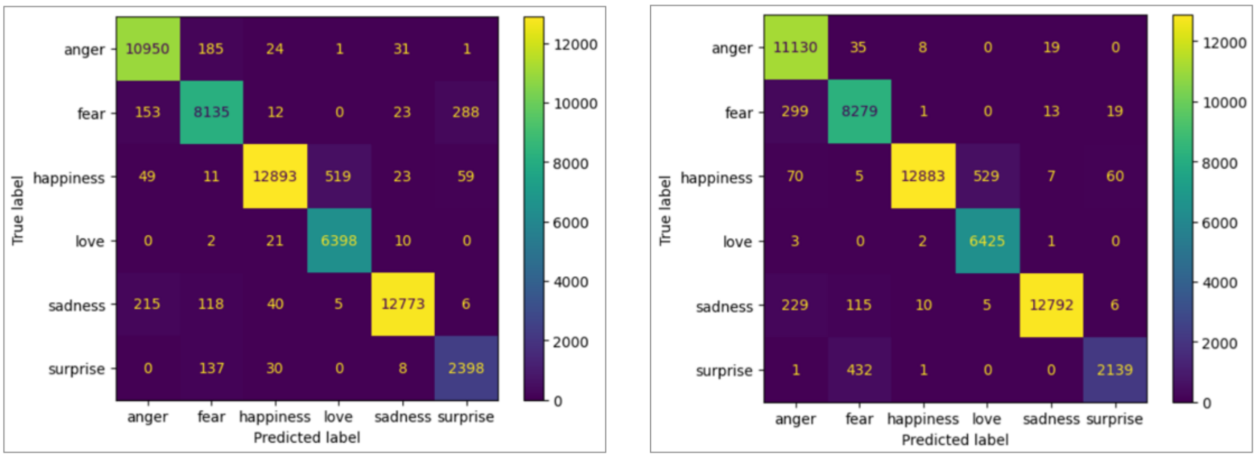


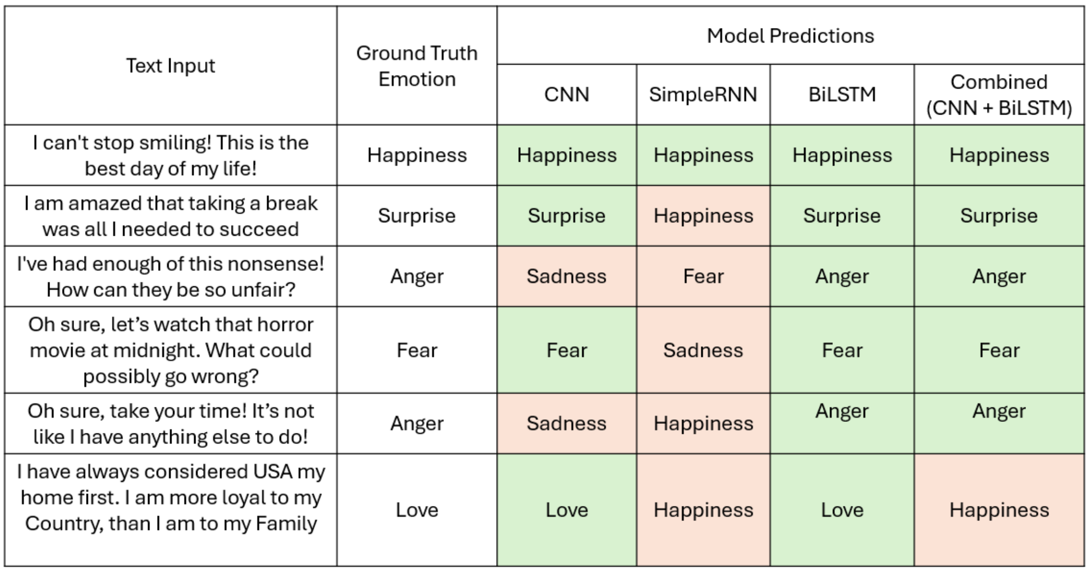
Figure 14: Confusion Matrix for Combined Model (on the left) and BiLSTM Model (on the right)

6.2 Qualitative Analysis

The table presented in 15 provides a comparison of predictions made by all of the models on a set of randomly selected posts from the internet. Some of the texts were sarcastic in nature, providing a challenge for the models. As observed from the table, the BiLSTM and Combined models consistently outperform the other models in accurately predicting the emotions. The SimpleRNN model seems to be affected by the unbalanced dataset, as it can be seen predicting mostly the majority classes. Further analysis of a larger set of texts and posts reaffirms the BiLSTM and Combined Models' superior performance.

6.3 Performance Analysis

The graph 16 shows average time taken per epoch to train the models. As observed from the graph, the CNN and SimpleRNN models exhibited the fastest training times, each completing an epoch in under 60 seconds, making them suitable for real-time applications. In contrast, the BiLSTM model had the longest training time, averaging over 1800 seconds per epoch, due to its bidirectional processing and sequential dependency handling. The hybrid CNN + RNN model achieved a balance, taking around 800 seconds per epoch, leveraging CNN's computational efficiency while incorporating RNN's sequential learning capabilities. Based on this, CNN models are preferable for speed-sensitive applications, while BiLSTM models, despite being computationally expensive, provide higher accuracy in capturing context. The CNN + RNN hybrid approach offers a trade-off between speed and accuracy, making it a viable option for practical deployments where both factors are crucial.



Text Input	Ground Truth Emotion	Model Predictions			
		CNN	SimpleRNN	BiLSTM	Combined (CNN + BiLSTM)
I can't stop smiling! This is the best day of my life!	Happiness	Happiness	Happiness	Happiness	Happiness
I am amazed that taking a break was all I needed to succeed	Surprise	Surprise	Happiness	Surprise	Surprise
I've had enough of this nonsense! How can they be so unfair?	Anger	Sadness	Fear	Anger	Anger
Oh sure, let's watch that horror movie at midnight. What could possibly go wrong?	Fear	Fear	Sadness	Fear	Fear
Oh sure, take your time! It's not like I have anything else to do!	Anger	Sadness	Happiness	Anger	Anger
I have always considered USA my home first. I am more loyal to my Country, than I am to my Family	Love	Love	Happiness	Love	Happiness

Figure 15: Testing the models with some sample Texts

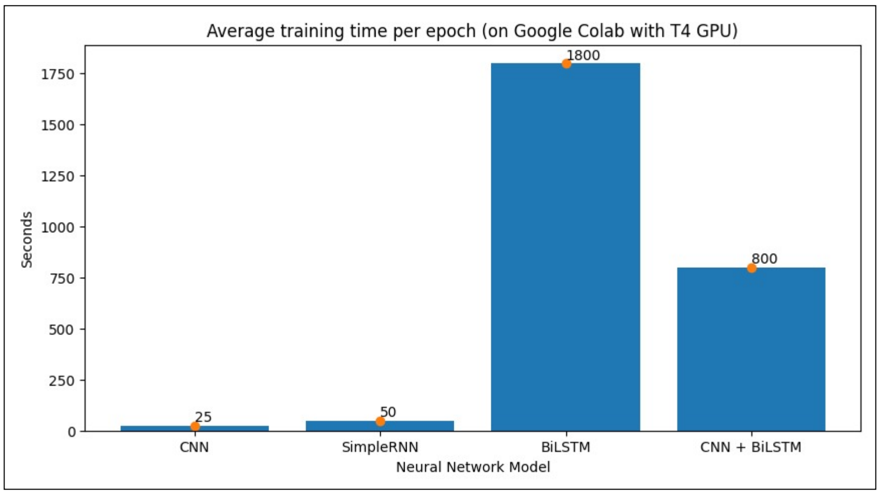


Figure 16: Distribution of Model Performance (Avg. Training time per epoch.)

7 Challenges and Lessons learned

Throughout the development of the emotion classification model, several challenges were encountered, each leading to valuable insights that helped refine the approach.

- **Handling an Unbalanced Dataset Through Resampling:** One of the primary challenges was the class imbalance in the dataset, where emotions like happiness and sadness had significantly more samples compared to love and surprise. This imbalance led to biased predictions, where the model favored majority classes. Oversampling the minority classes and Undersampling the majority classes were explored to create a more balanced dataset.
- **Preventing Overfitting with Early Stopping:** Deep learning models tend to overfit when trained for too many epochs, memorizing the training data instead of learning generalizable patterns. This was evident when validation accuracy started declining while training accuracy continued to improve. Using Early Stopping helped prevent overfitting by monitoring validation loss and stopping training when no further improvement was observed.
- **Importance of Cleaning Data for Word Embeddings:** Since word embeddings capture semantic meanings, unclean data such as misspellings, unnecessary symbols, and inconsistent text formats reduced the effectiveness of pretrained embeddings.

- **CNN vs. RNN: Speed vs. Accuracy:** The project explored two major deep learning architectures: CNNs and RNNs (LSTMs/BiLSTM) for emotion classification. While CNNs processed text faster due to parallel computations, RNN-based models provided better accuracy by capturing sequential dependencies in text.
- **Challenges in Identifying Human Emotions from Text:** Unlike structured data, human emotions are complex and highly context-dependent. Certain emotions like sarcasm, irony, and subtle sentiment shifts are difficult to detect purely from text. Additionally, pre-processing sometimes removed key indicators of emotion, such as:
  - Capitalized words (e.g., “I AM HAPPY” vs. “I am happy”)
  - Punctuation and special characters (e.g., “Great job!” vs. “Great job?” can convey different sentiments)
  - Contextual dependencies (e.g., sarcasm is often indicated through tone rather than words)

  A future scope can be to extract such information as features and feed them to the model to assess its impact.

8 Conclusion

This project successfully explored emotion detection from Text data using deep learning models, leveraging CNNs, RNNs (LSTMs/BiLSTMs), and hybrid architectures. Through data pre-processing, class imbalance handling, and model evaluation, challenges and strategies to improve performance were identified. The results highlighted that CNNs provided fast computations, while BiLSTMs achieved the highest accuracy (95%) by capturing contextual dependencies in text. Additionally, the trade-off between training time and accuracy revealed that CNN + BiLSTM offers an optimal balance for real-world deployment.

The project highlighted the complexity of emotion detection in textual data and the importance of balancing computational efficiency with model interpretability. Key takeaways included handling imbalanced data effectively, choosing appropriate deep learning architectures, and being mindful of text preprocessing techniques to retain emotional context. These insights will be invaluable for further research in sentiment analysis and emotion-aware AI applications.

References

[1] Lecun, Yann & Bottou, Leon & Bengio, Y. & Haffner, Patrick. (1998). Gradient-Based Learning Applied to Document Recognition. Proceedings of the IEEE. 86. 2278 - 2324. 10.1109/5.726791.

[2] S. Hochreiter and J. Schmidhuber, ”Long Short-Term Memory,” in Neural Computation, vol. 9, no. 8, pp. 1735-1780, 15 Nov. 1997, doi: 10.1162/neco.1997.9.8.1735.

[3] Agarwal, Kushagra. ”Sentiment and Emotion Analysis Dataset.” https://www.kaggle.com/datasets/kushagra3204/sentiment-and-emotion-analysis-dataset.

[4] Pandey, Parul. ”Emotion Dataset for Emotion Recognition Tasks.” https://www.kaggle.com/datasets/parulpandey/emotion-dataset.

[5] Pennington, Jeffrey & Socher, Richard & Manning, Christopher. (2014). ”Glove: Global Vectors for Word Representation.” EMNLP. 14. 1532-1543. 10.3115/v1/D14-1162.

[6] https://www.analyticssteps.com/blogs/convolutional-neural-network-cnn-graphical-visualization-code-explanation

[7] Yichen Zhao. ”Complete Guide to RNN, LSTM, and Bidirectional LSTM.” March 12, 2023. https://dagshub.com/blog/rnn-lstm-bidirectional-lstm

[8] Utebayeva, Dana & Ilipbayeva, Lyazzat & Matson, Eric. (2022). Practical Study of Recurrent Neural Networks for Efficient Real-Time Drone Sound Detection: A Review. Drones. 7. 26. 10.3390/drones7010026.

[9] Kingma, Diederik P., and Jimmy Ba. ‘Adam: A Method for Stochastic Optimization’. arXiv [Cs.LG], 2017, http://arxiv.org/abs/1412.6980. arXiv.