# Project Report

April 5, 2024

## 1 Abstract

## 2 Introduction

For this project, we investigated and compared the performance of 4 models on the "Emotion "dataset which contains around 400,000 English Twitter messages labelled with six fundamental emotions; anger, fear, joy, love, sadness, and surprise. Investigation of such a dataset is beneficial for understanding the underlying emotional expression spectrum of short English texts and thus such a sentimental analysis system can be used to help businessmen improve the quality of their goods via the reviews from customers. Those 4 models contain CNN, BERT, BiGRU and Long Short-Term memory network. The program started with data pre-processing including tokenization, padding, and constructing self-defined functions to balance the biased dataset.

## 3 System Design

### 3.1 Data Cleaning and Pre-processing

In order to successfully implement the sentimental analysis task for the given data set, we may need to first traverse the data set to check whether it contains inherent bias or not, and after checking the distribution of each class, we find that the first two classes contain most of data points compared with less than a tenth of data points belongs to the last class. Therefore, we need to address this bias existing within the data distribution and make the improved dataset have more equally distributed data. Besides, we could manually set the number of data points visible to the model so that we could accelerate the training phase with a small size of the dataset.
The second step for pre-processing contains tokenization and padding so that the human language collected can be interpreted and understood by our model. The Tokenization technique used by us is word tokenization so that the tokenizer chops a whole sentence into individual words or "tokens." These tokens are the basic building blocks of language, and tokenization helps computers understand and process human language by splitting it into manageable units.

### 3.2 Pretrained-BERT model

After we managed to create the input dataset, the following procedure will be the creation of a dataloader with a batch size of 32 and the training of our deep learning model. For the pre-trained BERT model, additional pooling and fully connected layers were added to reinforce the performance. During the training phase, the usage of backpropagation together with the Adam optimizer ensured we could find the optimal parameter for weights and bias. Adam optimizer is an improved version of SGD in terms of combining the advantages of two other extensions of stochastic gradient descent. Specifically: The adaptive Gradient Algorithm and Root Mean Square Propagation, the first algorithm was introduced to maintain a per-parameter learning rate that improves performance on problems with sparse gradients which suits natural language tasks well, the second algorithm was invented to address the problem coming from noisy samples (which is common for online texts) by adjust the learning rate dynamically according to the average of recent magnitudes of the gradients for the weight.With the help of a new optimizer and learning rate of $1 \times 10^{-5}$.

### 3.3 BiGRU model

For the BiGRU model, We began by structuring a sequential neural network for a text classification task, starting with some extra data preprocessing steps like stemming and lemmatization. Our model included an embedding layer for text vectorization, a bidirectional GRU layer for capturing sequential dependencies in both directions and dense layers with dropout for regularization to prevent overfitting. We then moved on to fitting the model with padded sequence data, using categorical cross-entropy as the loss function and 'adam' as the optimizer. During the training, we tracked the model's accuracy and loss on both the training and validation datasets to monitor its performance and generalization ability over epochs. The model achieved an impressive 0.9418 accuracy, indicating effective learning and robust predictive capability on the dataset used. To visualize the training process, we planned to plot the accuracy and loss curves using Matplotlib to provide insights into the model's behaviour during training, such as detecting overfitting or underfitting.

### 3.4 Convolutional Neural Network

After successfully preprocessing and preparing the input dataset, the next step was to implement an efficient data handling mechanism through the creation of a dataloader. We started with an Embedding layer for the CNN model. This layer transforms the tokenized words into dense vectors of fixed size, capturing semantic information. The model further employs convolutional layers (Conv1D) with varying kernel sizes to extract and learn different features from the embedded text, simulating how different words and their combinations contribute to the emotional tone of a message. Following the convolutional layers, a GlobalMaxPooling1D layer was introduced to reduce the dimensionality of the feature maps and to capture the most significant features extracted across the text. Activation functions played a pivotal role in CNN, with the use of the ReLU function across convolutional layers to encourage efficient learning and mitigate the issue of vanishing gradients. Finally in a softmax layer, assigning probabilities across the six targeted emotional states, ensuring a nuanced classification output based on textual input. Throughout the training phase, we leveraged the SGD optimizer, SGD provided a robust and straightforward approach to optimize the model's parameters—weights and biases, which is particularly advantageous for our text-based emotion classification task.

### 3.5 Long Short-Term memory

The data pre-processing flow of the model is basically the same as described in the data cleaning and pre-processing section. One more function on stop words has been added to make the data more concise by removing the words that appear frequently in a language but do not carry too much meaning hence improving the performance of the model. After tokenizing the data without stopwords, a vocabulary is built from the tokenized text. Assigning a unique integer index to each unique word in the dataset. This vocabulary is used to convert text tokens into numerical form, making it possible for the model to process the text. The dataset then is split into the train, val, and test sets for further training, validating and testing.

The model Long short term memory network is built with library pytorch in this project. An embedding layer takes token indices as input is the first layer of the network. The purpose of the embedding layer is to convert token indices into dense vectors of a fixed size. To prevent the presence of overfitting a dropout layer is added after the embedding layer. The rate is set as 0.1. A convolutional layer comes after the dropout layer to extract local features from the embedding vector sequence. Then comes a max pooling layer with kernel size 3 to reduce the dimensionality of the input. The core of the model is the LSTM layer. The layer is bidirectional to capture long-term dependencies in both forward and backward directions on outputs from the max pooling. Another dropout layer is placed here to further prevent the overfitting. The dropout rate is still 0.1. One fully connected layer projects the LSTM output to a higher-dimensional space. With another dropout layer, the last fully connected layer projects the feature to the number of classes in this problem which is 6. The cross-entropy loss and the RMSprop optimizer are used to train the model.

1

### 3.6 Hyperparameters in Pretrained-BERT model

The project explores two modelling approaches: the first leveraging BERT's capabilities enhanced with a fully connected layer, and the second introducing convolutional layers to capture more complex patterns, potentially improving performance. Both approaches utilize critical hyperparameters such as a learning rate of 1e-5, a batch size of 32, a max sequence length of 128, and vary in the number of training epochs to refine the model's accuracy and efficiency. The convolutional model variant additionally adjusts the kernel size and the number of filters to further fine-tune the model's capability. Impressively, both models achieve remarkable results, with the first approach yielding a testing accuracy of 0.97 and an F1 score of approximately 0.93, underscoring the efficacy and potential of combining BERT with CNN for emotion classification in textual data.
In addition, after comparing all of these models, we chose the Pretrained-BERT model as our primary model since it has the highest accuracy and a relatively easy training process. The data and figures used for comparison are discussed in detail in the next section.

## 4 Result

In evaluating our model, we used two primary metrics: accuracy and loss.
Accuracy is a common metric for classification tasks, representing the proportion of correct predictions out of all predictions made. It's intuitive and easy to understand; however, it might not be the sole indicator of performance, especially in cases where the class distribution is imbalanced. In our context, a high accuracy indicated that the model was correctly classifying a large proportion of the text samples into their respective categories.
Loss, specifically categorical cross-entropy loss in our case, quantifies how well the predicted probability distribution aligns with the true distribution. In simpler terms, it measures the "distance" between the model's predicted probabilities for each class and the actual label. It's a crucial metric during training as it's directly minimized through the optimization process. The goal is to decrease the loss, meaning our model's predictions are getting closer to the true labels.
Both these metrics are crucial throughout the training and validation process, providing insights into how well the model is learning and generalizing. A decreasing trend in loss and an increasing trend in accuracy are typically signs that the model is improving. If validation loss starts to increase while training loss continues to decrease, it could indicate overfitting, meaning that while the model performs well on the training data, it's failing to generalize to unseen data. Conversely, if both training and validation loss decrease but the validation loss decreases more slowly, it might suggest that the model is underfitting and too simplistic to capture the complexity of the data.
To ensure our model performs well not just on the training data but also on new, unseen data, we monitor the validation metrics closely. They help us decide when to stop training, adjust hyperparameters, or apply regularization techniques.

### 4.1 Pretrained-BERT model

**Modeling Approach 1:**Resulted in a testing accuracy of 0.97, with an F1 score of approximately 0.93. This approach established a robust foundation using BERT as the backbone, supplemented by a pooling layer and a fully connected layer for emotion classification. The iterative training and validation process under this architecture underscored its efficacy in accurately classifying emotions from text data. Such metrics indicate the model's ability to accurately identify the correct emotion labels from the text data, with minimal false positives and false negatives.

**Modeling Approach 2:** The additional convolutional layers appeared to have a positive effect, maintaining a high level of performance as seen in the first approach. The precision and recall metrics are very similar to those of the first model, with the averages being the same: a macro average a weighted average of 0.95 for precision, recall, and F1-score, and an overall accuracy of 0.95.

This demonstrates the model's capability to accurately classify emotional states across a balanced dataset with high consistency. The addition of convolutional layers in the second approach did not significantly change the performance metrics, suggesting that the initial BERT architecture was already quite robust for this particular task.

| Class | Precision | Recall | F1-Score | Support |
|---|---|---|---|---|
| Class 0 | 0.97 | 0.95 | 0.96 | 839 |
| Class 1 | 0.98 | 0.89 | 0.94 | 811 |
| Class 2 | 0.92 | 0.99 | 0.95 | 804 |
| Class 3 | 0.96 | 0.95 | 0.95 | 869 |
| Class 4 | 0.94 | 0.91 | 0.92 | 820 |
| Class 5 | 0.92 | 1.00 | 0.96 | 857 |
| | **Accuracy** | | 0.95 | 5000 |
| | **Macro Avg** | 0.95 | 0.95 | 5000 |
| | **Weighted Avg** | 0.95 | 0.95 | 5000 |

Table 1: Classification Report for Pretrained-BERT Model Approach 1

| Class | Precision | Recall | F1-Score | Support |
|---|---|---|---|---|
| Class 0 | 0.97 | 0.94 | 0.96 | 839 |
| Class 1 | 0.99 | 0.89 | 0.93 | 811 |
| Class 2 | 0.92 | 0.99 | 0.95 | 804 |
| Class 3 | 0.96 | 0.96 | 0.96 | 869 |
| Class 4 | 0.94 | 0.91 | 0.92 | 820 |
| Class 5 | 0.92 | 1.00 | 0.96 | 857 |
| | **Accuracy** | | 0.95 | 5000 |
| | **Macro Avg** | 0.95 | 0.95 | 5000 |
| | **Weighted Avg** | 0.95 | 0.95 | 5000 |

Table 2: Classification Report for Modeling Approach 2

## 4.2 BiGRU model

The training process involved 5 epochs with a batch size of 128, and it included validation using a separate test set. The results indicate that the model achieved high accuracy, with a training accuracy approaching 0.9418 and a validation accuracy close to 0.9391 by the end of the 5th epoch. The loss decreased over epochs, indicating good learning progress.

| Epoch | Step/Total Steps | Loss | Accuracy | Val Loss | Val Accuracy |
|---|---|---|---|---|---|
| 1/5 | 2280/2280 | 0.3117 | 0.8766 | 0.2013 | 0.9093 |
| 2/5 | 2280/2280 | 0.1183 | 0.9336 | 0.1288 | 0.9318 |
| 3/5 | 2280/2280 | 0.1007 | 0.9394 | 0.0941 | 0.9392 |
| 4/5 | 2280/2280 | 0.0956 | 0.9403 | 0.1077 | 0.9371 |
| 5/5 | 2280/2280 | 0.0915 | 0.9418 | 0.0945 | 0.9391 |

Table 3: Model training and validation results for BiGRU Model.

Finally, the accuracy and loss during training and validation were plotted using Matplotlib. The plots likely show curves that depict the improvement of model accuracy and the reduction of loss over each epoch, suggesting effective learning and model convergence. These results would be indicative of a well-performing model on this particular emotion classification task.

## 4.3 CNN

The model has trained for 38 epochs using the early stopping method. On the left, we have the training and validation loss. Both curves decline sharply in the initial epochs, indicating that the model is quickly learning from the dataset. After this initial phase, both losses level off, with the training loss continuing to decrease at a slower rate and the validation loss becomes flat which means the model is converging and generalizing well without overfitting. On the right graph, we observe the training and validation accuracy and both of them are around 91 percent. These curves ascend at the beginning, showing rapid learning and improvement in model performance. They plateau soon after, indicating a point where the model has likely optimized its weights to the possible data.

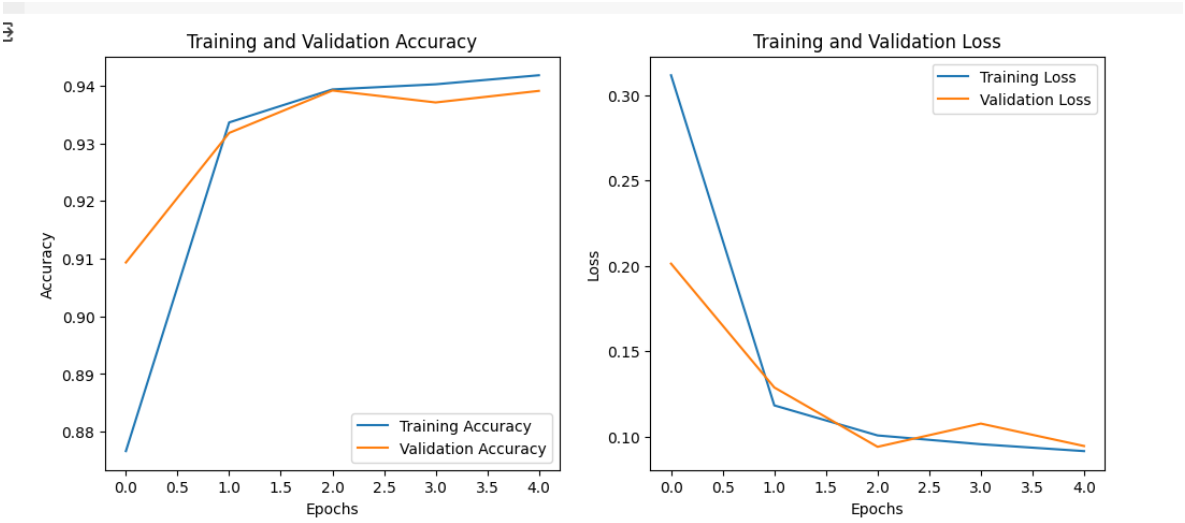


Figure 1: Training and Validation Accuracy and Loss

| | **Precision** | **Recall** | **F1 score** | **Support** |
|---|---|---|---|---|
| Sadness (0) | 0.97 | 0.94 | 0.96 | 24121 |
| Joy (1) | 0.92 | 0.91 | 0.94 | 28220 |
| Love (2) | 0.90 | 0.99 | 0.78 | 6824 |
| Anger (3) | 0.88 | 0.97 | 0.92 | 11448 |
| Fear (4) | 0.86 | 0.92 | 0.89 | 9574 |
| Surprise (5) | 0.85 | 0.73 | 0.78 | 3038 |
| | | | | |
| Accuracy | | | 0.92 | 83225 |
| macro avg | 0.90 | 0.87 | 0.88 | 83225 |
| weighted avg | 0.92 | 0.92 | 0.92 | 83225 |

Table 4: Test on CNN

## 4.4 LSTM

The model has been trained for 30 epochs with a learning rate of 0.002. There is a big improvement in the accuracy at the second epoch. Then the accuracy and loss tend to be stable. The training The results show an increasing trend in both training and validation accuracy over epochs, indicating the model is learning and generalizing well to unseen data. The presence of a validation set and the tracking of validation accuracy are crucial for monitoring overfitting. The slight variations in validation accuracy and loss compared to the training metrics suggest the model is relatively stable. The training process demonstrates effective learning, as evidenced by increasing accuracies and decreasing losses. The consistent improvement without large discrepancies between training and validation metrics suggests the model is not significantly overfitting.

The test result is presented in the table below. The result demonstrates a high degree of accuracy(0.94) across all the classes, indicating the model has learned to generalize well from the training dataset. The F1 scores are consistently high across classes which shows that the model is reliable in its predictions without significantly favoring some classes. Even the precision on class joy and surprise are seemingly lower than in other classes.

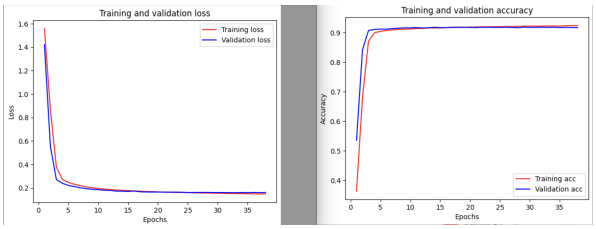


Figure 2: Train and Val vs. epoch

| | **Precision** | **Recall** | **F1 score** | **Support** |
|---|---|---|---|---|
| Sadness (0) | 0.95 | 0.99 | 0.97 | 24238 |
| Joy (1) | 0.99 | 0.91 | 0.95 | 28214 |
| Love (2) | 0.78 | 0.99 | 0.87 | 6911 |
| Anger (3) | 0.96 | 0.91 | 0.94 | 11463 |
| Fear (4) | 0.94 | 0.85 | 0.89 | 9542 |
| Surprise (5) | 0.73 | 0.97 | 0.84 | 2994 |
| | | | | |
| Accuracy | | | 0.94 | 83362 |
| macro avg | 0.89 | 0.94 | 0.91 | 83362 |
| weighted avg | 0.94 | 0.94 | 0.94 | 83362 |

Table 5: Test on LSTM

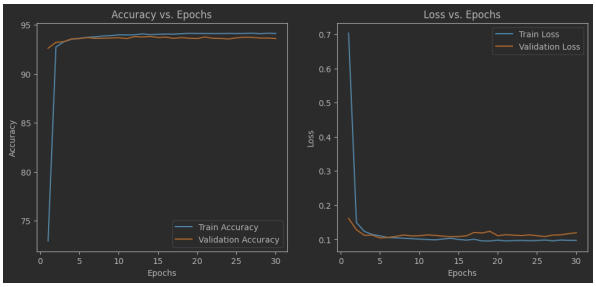


Figure 3: Train and Val vs. epoch