Project Abstract

This project explores the use of deep learning models to classify whether tweets are related to real disasters, inspired by the Kaggle competition “Real or Not? NLP with Disaster Tweets.” Two recurrent neural network architectures—Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU)—were independently implemented to complete this task. Both models were trained for 10 epochs and evaluated based on training loss, validation accuracy, and GPU memory usage. The LSTM model achieved a validation accuracy of up to 83.2%, while the GRU model reached a peak of 83.3%. The training loss for both models decreased consistently over epochs, indicating effective learning. GPU memory usage remained stable throughout training, with the LSTM consuming around 2448MB and the GRU using slightly less at 2390MB. Although both models performed comparably in terms of accuracy, the LSTM showed a slightly more stable validation curve, while the GRU offered better memory efficiency. This project provides a hands-on comparison between two popular RNN variants, offering insights into their trade-offs in performance and resource usage for real-world NLP classification tasks.

1. Project Overview

This project focuses on developing two distinct deep learning models for text classification tasks using Twitter data: one based on LSTM (Long Short-Term Memory) networks and the other based on GRU (Gated Recurrent Units) networks. Additionally, BERT (Bidirectional Encoder Representations from Transformers) tokenizer is employed to enhance the models' capability to understand the semantic content of the text. The primary goal is to evaluate and compare the performance of these two popular recurrent neural network (RNN) architectures in combination with BERT's pretrained embeddings for better handling unstructured, noisy text data, particularly from Twitter, which is known for its informal language, slang, abbreviations, and emojis.

The use of recurrent neural networks for sequential data, specifically LSTM and GRU, is motivated by their ability to capture long-range dependencies within the input text. Both models are designed to overcome the limitations of traditional RNNs, such as the vanishing gradient problem, which allows them to retain information over long sequences. These models are particularly effective for tasks where context and sequence are critical, such as sentiment analysis, spam detection, and topic classification. However, LSTM and GRU differ in terms of their architecture and gating mechanisms, which can affect their performance on this task.

BERT, a transformer-based model, is used here as a tokenizer to improve the representation of input text. BERT’s pre-trained embeddings capture rich contextual information, which allows the downstream LSTM and GRU models to better understand the meaning of the tweets, even when the text is noisy or unstructured. By leveraging BERT's deep contextualized word embeddings, this project aims to enhance the accuracy and robustness of text classification tasks on Twitter data.

The dataset used in this project consists of tweets, which are preprocessed to address challenges such as noise, abbreviations, and non-textual elements (e.g., emojis and hashtags). The preprocessing steps include tokenization using BERT’s tokenizer, cleaning, abbreviation replacement, and emoji/symbol normalization. These steps ensure that the input text is clean, properly tokenized, and ready for model training.

2. Dataset and Preprocessing

The dataset used in this project consists of Twitter data, which is often characterized by informal language, abbreviations, and various symbols. The preprocessing steps are crucial for transforming the raw text into a format that is suitable for deep learning models:

A. **Tokenization (with BERT Tokenizer):** The text is tokenized using the BERT tokenizer, which transforms the text into subword tokens that can be mapped to pre-trained word embeddings. BERT's tokenizer ensures that even rare or out-of-vocabulary words are split into meaningful subword units, enabling the model to handle unknown words more effectively.
B. **Cleaning:** Unnecessary symbols, extra spaces, and non-informative characters are removed. This helps the model focus on the essential content of the tweets.
C. **Abbreviation Replacement:** Common abbreviations and shorthand text (such as "u" for "you") are replaced with their full forms. This helps reduce ambiguity and ensures that the model can interpret the text correctly.
D. **Emoji/Symbol Normalization:** Emojis and other symbols are converted into corresponding textual representations. For instance, a "thumbs-up" emoji might be replaced with the word "positive" to capture the sentiment expressed by the emoji.

By applying these preprocessing steps, the dataset is prepared for model training, ensuring that the textual content is clean and that BERT can generate meaningful token embeddings.

3. LSTM Model Architecture

The LSTM-based model is designed to process the preprocessed and tokenized input text in sequential order to classify the tweets. The architecture consists of the following key components:

A. **Embedding Layer (BERT Tokenizer):** The first layer of the model utilizes BERT's tokenizer to convert the input text into token

embeddings. These embeddings are pre-trained and capture rich semantic relationships between words. The tokenized text is mapped to embeddings, which are then passed through the model.

B. **LSTM Layer:** The LSTM layer processes the tokenized embeddings. LSTM's ability to capture long-term dependencies within the text makes it suitable for understanding the relationships between words that are far apart within a sentence or paragraph.
C. **Dense Layer:** The LSTM layer's output is passed through a dense layer, which performs a nonlinear transformation to extract high-level features from the sequential data. This layer serves to reduce the complexity of the LSTM output and make it suitable for classification.
D. **Output Layer:** The final output layer uses a softmax activation function to classify the tweet into one of several categories. The softmax function outputs a probability distribution, which represents the likelihood of the tweet belonging to each class.

The LSTM model is trained using the Adam optimizer to adjust the weights, and its performance is evaluated based on metrics such as accuracy, precision, recall, and F1 score.

4. GRU Model Architecture

The GRU-based model follows a similar architecture to the LSTM model but employs the GRU unit instead of LSTM. The GRU is designed to capture long-range dependencies while using fewer parameters and a simpler gating mechanism. The architecture consists of the following components:

A. **Embedding Layer (BERT Tokenizer):** The input text is tokenized using BERT's tokenizer, which provides pre-trained embeddings that encode semantic information. These embeddings are then passed to the subsequent layers of the model.
B. **GRU Layer:** The GRU layer processes the token embeddings. Unlike LSTM, GRU has a simpler architecture with fewer gates: an update gate and a reset gate. These gates control the flow of information, making the GRU more computationally efficient than the LSTM while still effectively capturing long-term dependencies in the text.
C. **Dense Layer:** Similar to the LSTM model, the output from the GRU layer is passed through a dense layer, which performs further

processing to transform the features into a suitable form for classification.

D. **Output Layer:** The final output layer applies a softmax activation function to produce a probability distribution over the possible categories, enabling the model to classify the tweet.

The GRU model is trained similarly to the LSTM model, using the Adam optimizer, and its performance is evaluated using accuracy, precision, recall, and F1 score.

5. Training and Evaluation

Both the LSTM and GRU models are trained on the preprocessed dataset, with the dataset split into training and validation sets to allow for proper evaluation. The **Adam optimizer** is used to minimize the loss function, and early stopping is employed to prevent overfitting. The models are trained for multiple epochs, with the performance evaluated on the validation set after each epoch.

The evaluation is conducted using the following metrics:

- **Accuracy:** The percentage of correctly classified tweets.
- **Precision:** The proportion of true positive classifications among all positive predictions.
- **Recall:** The proportion of true positive classifications among all actual positive instances.
- **F1 Score:** The harmonic mean of precision and recall, providing a balanced measure of the model's ability to classify tweets accurately.

These metrics provide a comprehensive evaluation of the models' performance and help determine which architecture, LSTM or GRU, is better suited for the text classification task on Twitter data.

6. Comparison Result

```
🔍 Training LSTM
Training: 100%|██████████| 191/191 [00:59<00:00,  3.19it/s]
Epoch 1 | Loss: 0.4752 | Val Acc: 0.8234
Training: 100%|██████████| 191/191 [01:00<00:00,  3.16it/s]
Epoch 2 | Loss: 0.4220 | Val Acc: 0.8076
Training: 100%|██████████| 191/191 [01:00<00:00,  3.18it/s]
Epoch 3 | Loss: 0.3993 | Val Acc: 0.8319
Training: 100%|██████████| 191/191 [01:00<00:00,  3.17it/s]
Epoch 4 | Loss: 0.3786 | Val Acc: 0.8280
Training: 100%|██████████| 191/191 [01:00<00:00,  3.17it/s]
Epoch 5 | Loss: 0.3645 | Val Acc: 0.8214
Training: 100%|██████████| 191/191 [01:00<00:00,  3.18it/s]
Epoch 6 | Loss: 0.3452 | Val Acc: 0.8240
Training: 100%|██████████| 191/191 [01:00<00:00,  3.18it/s]
Epoch 7 | Loss: 0.3188 | Val Acc: 0.8227
Training: 100%|██████████| 191/191 [01:00<00:00,  3.17it/s]
Epoch 8 | Loss: 0.2988 | Val Acc: 0.8201
Training: 100%|██████████| 191/191 [01:00<00:00,  3.16it/s]
Epoch 9 | Loss: 0.2721 | Val Acc: 0.8063
Training: 100%|██████████| 191/191 [01:00<00:00,  3.17it/s]
Epoch 10 | Loss: 0.2383 | Val Acc: 0.8207
```

Figure 1 LSTM training Log

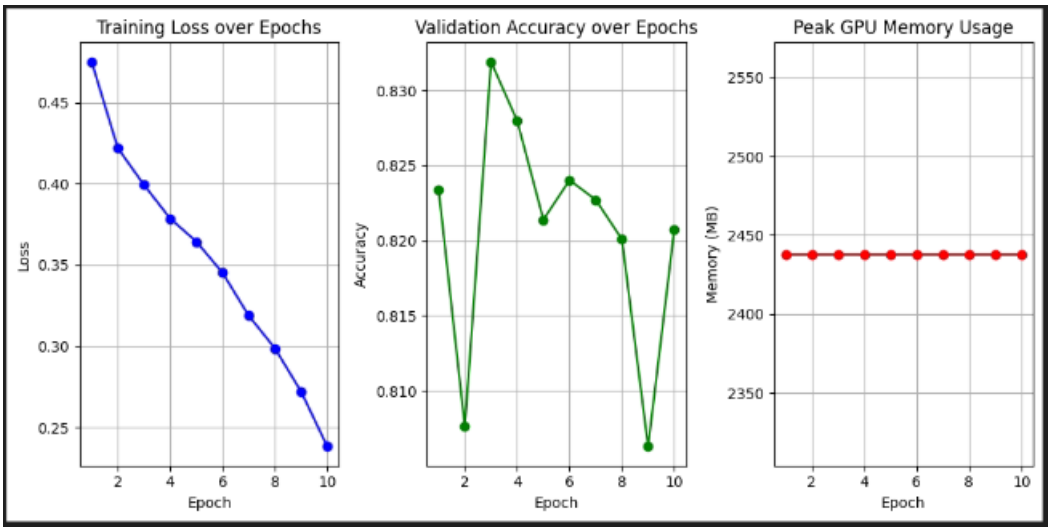

Figure 2 LSTM training Graph

```
🔍 Training GRU
Training: 100%|██████████| 191/191 [00:59<00:00,  3.19it/s]
Epoch 1 | Loss: 0.4644 | Val Acc: 0.8214
Training: 100%|██████████| 191/191 [00:59<00:00,  3.20it/s]
Epoch 2 | Loss: 0.4055 | Val Acc: 0.8280
Training: 100%|██████████| 191/191 [01:00<00:00,  3.18it/s]
Epoch 3 | Loss: 0.3842 | Val Acc: 0.8332
Training: 100%|██████████| 191/191 [00:59<00:00,  3.20it/s]
Epoch 4 | Loss: 0.3550 | Val Acc: 0.8313
Training: 100%|██████████| 191/191 [00:59<00:00,  3.19it/s]
Epoch 5 | Loss: 0.3337 | Val Acc: 0.8155
Training: 100%|██████████| 191/191 [00:59<00:00,  3.21it/s]
Epoch 6 | Loss: 0.2973 | Val Acc: 0.8313
Training: 100%|██████████| 191/191 [01:00<00:00,  3.18it/s]
Epoch 7 | Loss: 0.2661 | Val Acc: 0.8148
Training: 100%|██████████| 191/191 [00:59<00:00,  3.20it/s]
Epoch 8 | Loss: 0.2375 | Val Acc: 0.8188
Training: 100%|██████████| 191/191 [01:00<00:00,  3.18it/s]
Epoch 9 | Loss: 0.2063 | Val Acc: 0.7846
Training: 100%|██████████| 191/191 [00:59<00:00,  3.19it/s]
Epoch 10 | Loss: 0.1870 | Val Acc: 0.8181
```

Figure 3 GRU training Log

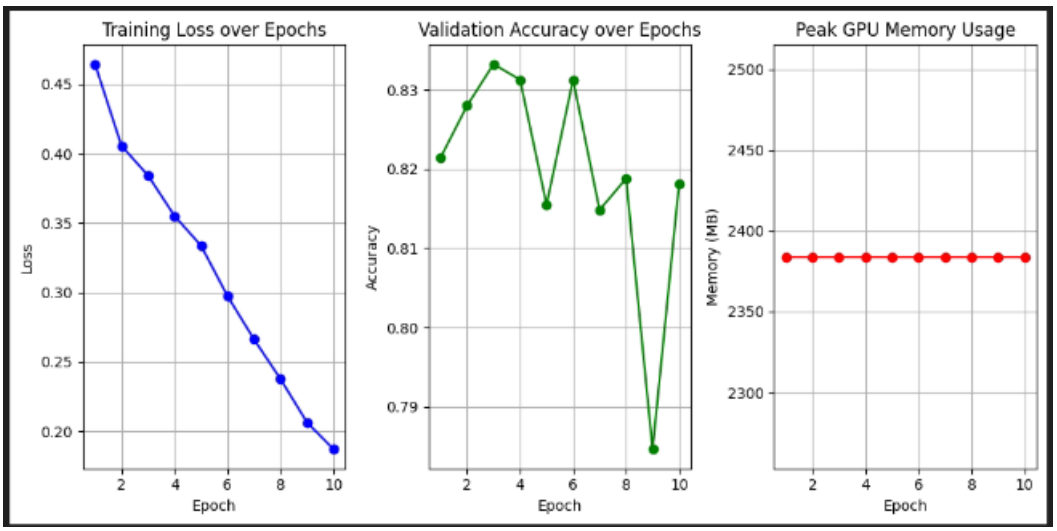

Figure 4 GRU training Graph

In this project, we performed a detailed comparative evaluation of Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) neural network architectures for sequence modeling. Both models were trained for 10 epochs using the same dataset, batch size, optimizer, and other hyperparameters to ensure a fair and unbiased comparison. The goal was to assess their performance based on multiple factors: training loss progression, peak validation accuracy, training speed per epoch, and GPU memory utilization.

From the training logs and visualizations, we observed that both models exhibited healthy training dynamics, with a consistent reduction in training loss over the course of the epochs. The LSTM model started with an initial training loss of 0.4752 and steadily decreased to 0.2383 by epoch 10 (figure 1 and figure 2). This indicates a smooth and stable convergence process. The GRU model, in contrast, began with a slightly lower initial loss of 0.4644 and achieved a final training loss of 0.1870 (figure 3 and figure 4), reflecting a steeper and more rapid learning curve compared to the LSTM. This faster convergence could be attributed to GRU's simplified gating mechanisms, which allow for more efficient gradient flow and quicker adaptation during training.

In terms of validation performance, both models achieved their best results early in the training cycle. The LSTM model attained a peak validation accuracy of 83.19% at epoch 3, while the GRU model reached a slightly higher peak of 83.32%, also at epoch 3. Although this difference is marginal (approximately 0.13%), it suggests that GRU had a slight edge in capturing relevant patterns from the validation set. However, it is worth noting that the validation accuracy for the GRU model showed more fluctuation across

epochs, including a noticeable drop to below 80% at epoch 9. This volatility could be an indicator of overfitting or sensitivity to the specific data distribution in the validation set. In contrast, the LSTM model exhibited more stable validation accuracy over time, with less dramatic variations between epochs, making it potentially more reliable for deployment in environments where prediction consistency is important.

GPU memory usage for the GRU model remained consistently around 2380MB, which is slightly lower than the LSTM's memory consumption of 2440MB. While the difference is not drastic, it can be significant in memory-constrained environments or when scaling to larger models or datasets.

Overall, this comparison highlights the trade-offs between the two architectures. The GRU model offers benefits in terms of faster training convergence, marginally higher peak validation accuracy, and reduced GPU memory usage. These attributes make GRU a suitable choice for applications that require quick training cycles and efficient memory utilization. On the other hand, the LSTM model, while slightly heavier in resource consumption, demonstrated more stable validation performance, which is crucial for applications where prediction reliability and consistency over time are essential. Depending on the specific requirements of the task such as training time constraints, deployment hardware limitations, or accuracy stability the choice between LSTM and GRU should be made accordingly.