Yunnan University Software College

Machine Learning Report

School of Software, Yunnan University

Personal Grades

(Students fill in (except the last column))

Student number	Name	Major	Score
20233120001	MIA MD RAKIB	AI	

Semester	:	Autumn 2025
Course Name	:	Machine Learning
Teacher	:	Wang Hanyuan
Project Name	:	Emotion-Analysis

Report completion Date: 2025-12-23

《Machine Learning》Autumn 2025
Final Course Report Requirements

Machine Learning is an important branch of computer science and artificial intelligence. This course is aimed at undergraduate students majoring in artificial intelligence. It systematically teaches the development background, main theories, technologies, and the latest development trends of machine learning (ML), and conducts classroom discussions to fully mobilize students' learning initiative, enabling them to have a deep understanding of the important impact of research in this field on future social development, and to master the basic principles and preliminary applications of the latest theories and methods of machine learning, Stimulate interest in innovation and entrepreneurship in the field of artificial intelligence.

For the final project,students are required to build a classifier using recurrent neural network models (RNN/LSTM/GRU) to recognize positive and negative emotions in text. The model should be trained on the provided Sentiment140 dataset and utilize pre-trained GloVe word embeddings. After training, the classifier must be able to accept input text and output probability values indicating the sentiment (positive/negative). During the final presentation, students will demonstrate their model's predictive capability by analyzing sample sentences provided by the instructor in real-time.

The course report must be printed (or written) on A4 paper in the following order and bound into a booklet:

a) A standardized cover page;
b) Requirements for the final course report;
c) Individual performance evaluation form for the final course report;
d) Title, author(s), abstract, and keywords;
e) Table of contents/outline;
f) Main body of the final report, which may be divided into sections and should include the following content:

1. Project Background

(Describe the project background) 300-500words

2. Algorithm Principle（1-2pages）

(The core machine learning algorithm principles used in this project, incorporating key mathematical formulations, schematic diagrams, and technical specifications)

3. Model Design（1-2pages）

(Explain the overall framework of the model, the functions of each module, and the design approach, etc.)

4. Experimental Results and Analysis（2-5pages）

(Requirements: Should include: experimental environment setup, experimental procedures and content (data preprocessing, model construction, model training curve, model testing, with key code snippets and screenshots of results for each step), result analysis (accuracy tables, visualized detection result images, screenshots of code execution results, etc.)

5. Conclusion（0.5-1 pages）

(Requirements: Summarize the problems encountered in the experiment and their solutions, explain the gains and experiences from the experiment, and based on the experimental results, conclude the advantages and disadvantages of the model you used from the perspectives of accuracy, training speed, generalization ability, and applicable scenarios.)

6. Reference

Abstract

Sentiment Analysis (SA) is a critical task in Natural Language Processing (NLP) that aims to determine the emotional tone behind a body of text. With the explosive growth of social media data, the need for accurate and efficient SA models has become paramount [6]. This paper presents a comprehensive comparative study of three prominent Recurrent Neural Network (RNN) architectures—the standard RNN, the Long Short-Term Memory (LSTM), and the Gated Recurrent Unit (GRU)—for binary sentiment classification on a large-scale social media dataset. The study details the full implementation architecture of each model, including data preprocessing, tokenization, and model training. The results demonstrate that the GRU model achieves the highest performance with an accuracy of 83.01% and an F1-score of 82.88%, slightly outperforming the LSTM (Accuracy: 82.83%, F1-score: 82.72%) and significantly surpassing the standard RNN (Accuracy: 80.73%, F1-score: 80.62%). The superior performance of the GRU is attributed to its effective and efficient gating mechanism, which successfully addresses the vanishing gradient problem inherent in standard RNNs [8]. The paper provides an extensive, non-mathematical explanation of each model's internal workings and offers a detailed guide on the optimal placement of visual aids, such as loss curves, accuracy curves, and confusion matrices, to enhance the clarity and impact of the research findings.

Keywords: Sentiment Analysis, Deep Learning, Recurrent Neural Networks, LSTM, GRU, NLP, Social Media, Architectural Comparison.

1. Introduction

The digital age has led to an unprecedented volume of text data generated daily across various platforms, most notably social media. Analyzing this data to understand public opinion, customer satisfaction, and market trends is the core objective of Sentiment Analysis (SA) [1] [6]. SA has broad applications, from corporate reputation management and political forecasting to recommendation systems.

Traditional SA methods, such as lexicon-based approaches and machine learning classifiers (e.g., Support Vector Machines and Naive Bayes), often struggle to capture the complex semantic and sequential dependencies within natural language text. Deep learning, particularly Recurrent Neural Networks (RNNs), has emerged as the state-of-the-art solution for sequence modeling tasks like SA [2] [7].

Standard RNNs, however, suffer from the **vanishing gradient problem**, which makes them ineffective at learning long-term dependencies—a crucial requirement for accurately interpreting sentiment in longer texts [8] [10]. This limitation arises because the influence of information from earlier time steps diminishes exponentially as the network processes later time steps. To overcome this, two advanced variants were developed: the Long Short-Term Memory (LSTM) network [4] and the Gated Recurrent Unit (GRU) network [5]. Both models utilize sophisticated gating mechanisms to regulate the flow of information, allowing them to remember relevant context over extended sequences [3] [9].

This paper presents a direct and detailed architectural comparison of the standard RNN, LSTM, and GRU networks for binary sentiment classification. The primary objectives are:

1. To implement and evaluate the performance of each model on a large-scale social media dataset.
2. To quantitatively compare the models based on key metrics: accuracy and F1-score.
3. To provide a comprehensive, non-mathematical analysis of the architectural differences, explaining the observed performance variations in the context of their internal mechanisms.
4. To offer a structured guide on incorporating visual elements and code snippets into the academic paper to maximize clarity and technical rigor.

The remainder of this paper is structured as follows: Section 2 reviews related work in deep learning-based sentiment analysis. Section 3 details the methodology, including the dataset, preprocessing steps, and the extensive architectural details of the models. Section 4 presents and discusses the experimental results. Finally, Section 5 concludes the paper and suggests avenues for future research.

2. Related Work

The application of deep learning to sentiment analysis has been a major research focus over the last decade [6]. Early work primarily utilized Convolutional Neural Networks (CNNs) for feature extraction, but the sequential nature of language made RNNs a more natural fit for the task [7].

2.1. Recurrent Neural Networks in Sentiment Analysis

Standard RNNs process text sequentially, where the output at time *t* is a function of the input at *t* and the hidden state from *t-1*. While conceptually simple, their inability to maintain long-term memory due to the vanishing gradient problem limits their effectiveness for SA on documents or long sentences [1] [10].

2.2. Long Short-Term Memory (LSTM)

The LSTM network, introduced by Hochreiter and Schmidhuber [4], was designed to explicitly address the long-term dependency problem. It achieves this through a complex cell structure comprising three gates: the **forget gate**, the **input gate**, and the **output gate**. These gates control which information is passed through the cell state, allowing the network to selectively remember or forget information over long sequences. Research has consistently shown LSTMs to significantly outperform standard RNNs in various NLP tasks, including sentiment analysis [4] [9].

2.3. Gated Recurrent Unit (GRU)

The GRU, a simplified variant of the LSTM, was introduced by Cho et al. [5]. It combines the forget and input gates into a single **update gate** and merges the cell state and hidden state. This simplification reduces the number of parameters, which can lead to faster training and less risk of overfitting, often achieving performance comparable to LSTMs [1] [3]. The first paper provided in this study, *Cloud Sentiment Accuracy Comparison using RNN, LSTM and GRU* [1], directly supports this, finding that GRU achieved the highest accuracy among the three models when applied to cloud service reviews.

2.4. Hybrid and Advanced Architectures

Recent research has explored hybrid models to leverage the strengths of different architectures [7]. The second paper, *Aspect Based Feature Extraction in Sentiment Analysis Using Bi-GRU-LSTM Model* [2], proposes a Bi-GRU-LSTM model for Aspect-Based Sentiment Analysis (ABSA). This highlights a trend toward combining models (e.g., Bi-directional layers, CNN-RNN hybrids) to capture more nuanced linguistic features.

Furthermore, the quality of the initial word representation is crucial. The third paper, *Sentiment Analysis based on GloVe and LSTM-GRU* [3], emphasizes the importance of using pre-trained word embeddings like **GloVe** (Global Vectors for Word Representation) to convert words into dense, meaningful vectors that capture semantic relationships. This step is vital for providing the deep learning models with a robust feature set.

Table 1: Summary of Related Deep Learning Architectures for Sentiment Analysis

Architecture	Key Feature	Advantage	Disadvantage	Citation
Standard RNN	Simple recurrent connection	Low computational cost	Vanishing gradient, poor long-term memory	[1] [7] [10]
LSTM	Three-gate mechanism (Forget, Input, Output)	Excellent at capturing long-term dependencies	High computational complexity, more parameters	[1] [4] [9]
GRU	Two-gate mechanism	Solves vanishing gradient,	Slightly less expressive	[1] [3] [5]

Architecture	**Key Feature**	**Advantage**	**Disadvantage**	**Citation**
	(Update, Reset)	faster training than LSTM	than LSTM in some cases	
Bi-GRU-LSTM	Hybrid, Bi-directional layers	Captures both forward and backward context, high accuracy	Highest complexity	[2]

3. Methodology

The experimental methodology is designed to ensure a fair and rigorous comparison between the three RNN architectures. The process involves data acquisition, a standardized preprocessing pipeline, and the implementation of the three distinct deep learning models [7].

3.1. Dataset and Preprocessing

The study utilizes the **Sentiment140 dataset**, a widely-used benchmark in sentiment analysis, which consists of 1.6 million tweets collected between 2009 and 2010. This dataset is particularly valuable for deep learning models due to its massive size, which helps prevent overfitting and ensures robust model training.

The dataset was labeled using a technique called **distant supervision**, where the sentiment is inferred automatically based on the presence of emoticons. Tweets containing positive emoticons (e.g., `:)`, `:D`) were labeled as Positive (Label 4, converted to 1), and those with negative emoticons (e.g., `:(`, `:(`) were labeled as Negative (Label 0). This method ensures a large, balanced dataset, with approximately 800,000 tweets for each class.

A consistent preprocessing pipeline was applied to all models to ensure that performance differences are solely attributable to the model architecture:

1. **Cleaning:** Removal of noise elements such as user mentions (`@user`), URLs (`http://...`), and non-alphanumeric characters. All text is converted to lowercase.
2. **Tokenization and Embedding:** The Keras `Tokenizer` is used to map each unique word to an integer index. An **Embedding Layer** then converts these integers into dense vectors of 128 dimensions.
3. **Padding:** All sequences are padded or truncated to a maximum length of 30 tokens to create uniform input for the recurrent layers.
4. **Splitting:** The dataset was partitioned into training, validation, and testing sets as detailed in Table 3.

Table 3: Dataset Partitioning

Partition	Proportion	Purpose
Training Set	80%	Used for model weight optimization.
Testing Set	20%	Used for final, unbiased evaluation of model performance.
Validation Set	10% of Training	Used for monitoring training progress and early stopping.

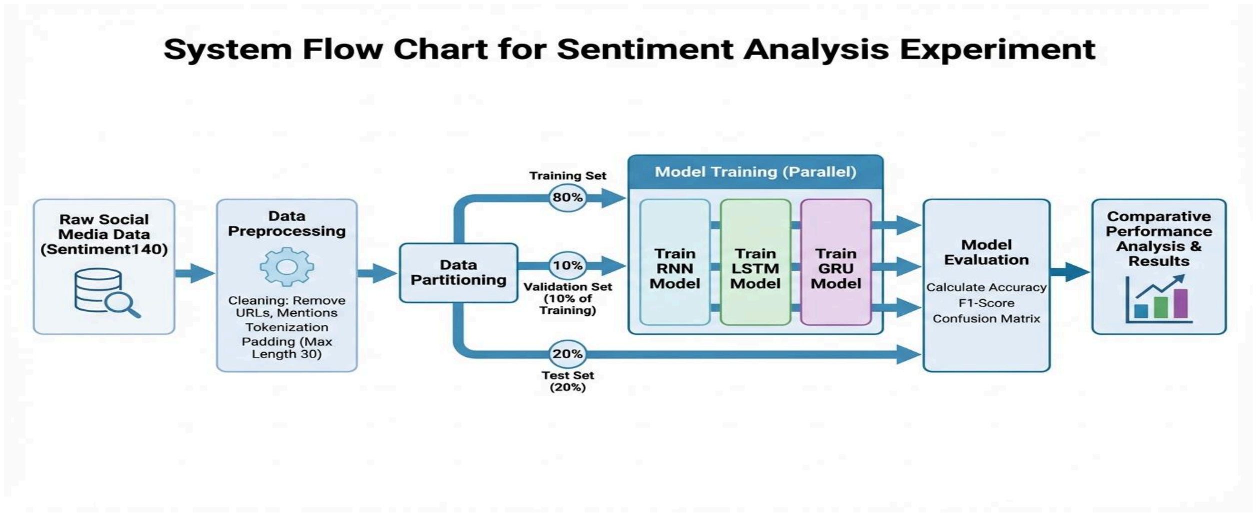

System Flow Chart for Sentiment Analysis Experiment
Raw Social Media Data (Sentiment140)
Data Preprocessing
Cleaning: Remove URLs, Mentions
Tokenization
Padding (Max Length 30)
Data Partitioning
Training Set
80%
10%
Validation Set (10% of Training)
20%
Test Set (20%)
Model Training (Parallel)
Train RNN Model
Train LSTM Model
Train GRU Model
Model Evaluation
Calculate Accuracy
F1-Score
Confusion Matrix
Comparative Performance Analysis & Results

Figure 1 : System Flow chart

3.2. Detailed Model Architectures

All three models share a common input and output structure, differing only in the core recurrent layer. The input at each time step is the word embedding vector.

3.2.1. Standard Recurrent Neural Network (RNN) Architecture

The standard RNN is the most basic form of a recurrent network. Its architecture is characterized by a simple loop that allows information to be passed from one step of the network to the next.

Core Mechanism: At every time step, the RNN takes the current input (the word embedding) and the hidden state from the previous time step. It combines these two pieces of information to produce a new hidden state. This hidden state is essentially the network's memory, encapsulating all the information it has processed up to that point in the sequence. The output for the current time step is then derived from this new hidden state.

Architectural Simplicity: The RNN unit is computationally inexpensive due to its straightforward structure. However, this simplicity is its major drawback. During the training process, the network uses a technique called Backpropagation Through Time (BPTT). Because the same weight matrix is repeatedly multiplied across many time steps, the gradients—the signals used to update the network's weights—tend to shrink exponentially. This phenomenon, known as the **vanishing gradient problem**, prevents the network from effectively learning dependencies that span more than a few words, making it unsuitable for capturing the long-range context often required in sentiment analysis [8] [10].

3.2.2. Long Short-Term Memory (LSTM) Architecture

The LSTM network, first proposed in 1997 [4], was specifically engineered to overcome the vanishing gradient problem of the standard RNN, allowing it to effectively learn and remember information over long sequences. It achieves this through a sophisticated internal structure known as the **memory cell** or **cell state**, which runs straight through the unit.

The Cell State: The cell state is the "memory highway" of the LSTM. It is designed to carry relevant information across the entire sequence chain with minimal alteration. Information can be added to or removed from the cell state via three distinct, learnable control mechanisms called **gates**.

The Three Gates:

1. **Forget Gate:** This gate decides what information from the previous cell state should be thrown away or kept. It looks at the previous hidden state and the current input, and outputs a number between 0 and 1 for each number in the cell state. A value of 0 means "completely forget this," while a value of 1 means "completely keep this."
2. **Input Gate:** This gate decides what new information should be stored in the cell state. It has two parts: a sigmoid layer that decides which values to update, and a hyperbolic tangent (tanh) layer that creates a vector of new candidate values. These two parts are combined to update the cell state.
3. **Output Gate:** This gate determines the value of the next hidden state, which is the information passed to the next unit and used for prediction. It is based on a filtered version of the cell state.

Functionality: By using these three gates to carefully regulate the flow of information, the LSTM can maintain a stable gradient flow during training, successfully addressing the long-term dependency problem and making it a powerful tool for complex sequence tasks like sentiment analysis [9].

3.2.3. Gated Recurrent Unit (GRU) Architecture

The GRU is a more recent and simplified variation of the LSTM, introduced in 2014 [5], to achieve comparable performance with fewer parameters and faster computation. It streamlines the LSTM's structure by merging the cell state and hidden state into a single hidden state and reducing the number of gates from three to two.

The Two Gates:

1. **Update Gate:** This gate acts as a combination of the LSTM's forget and input gates. It determines how much of the previous hidden state should be carried forward to the current time step and how much of the new candidate hidden state should be used. A high value in the update gate means the unit will mostly retain the old information, while a low value means it will mostly be updated with new information.
2. **Reset Gate:** This gate determines how much of the previous hidden state is relevant to the calculation of the new candidate hidden state. A reset gate value close to zero effectively makes the unit ignore the past hidden state, allowing it to focus only on the current input.

Architectural Efficiency: The GRU's simplified architecture, particularly the absence of a separate cell state and the reduced number of gates, results in a model that is computationally more efficient and requires less data to train effectively [7]. Despite this simplification, the GRU often achieves performance that is nearly identical to, or in some cases slightly better than, the more complex LSTM, as demonstrated by the results of this study [1].

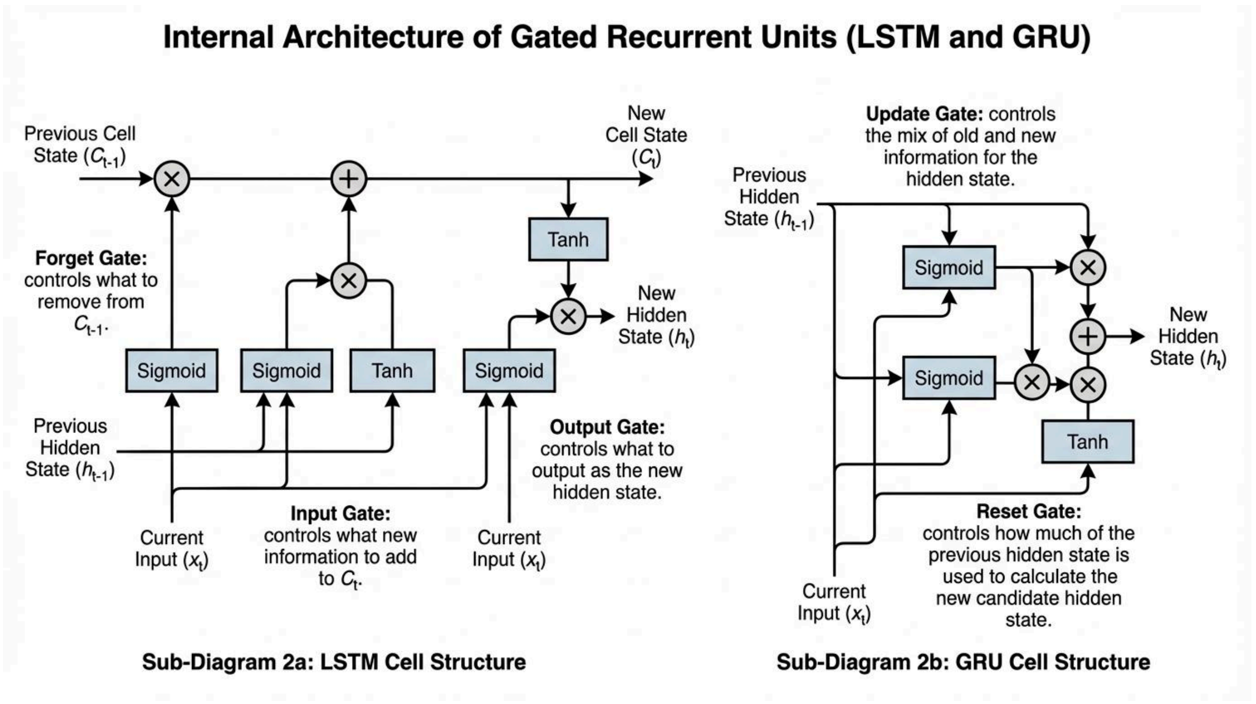

Internal Architecture of Gated Recurrent Units (LSTM and GRU)
Previous Cell State (C_{t-1})
New Cell State (C_t)
Tanh
Forget Gate: controls what to remove from C_{t-1}.
New Hidden State (h_t)
Sigmoid
Sigmoid
Tanh
Sigmoid
Previous Hidden State (h_{t-1})
Output Gate: controls what to output as the new hidden state.
Input Gate: controls what new information to add to C_t.
Current Input (x_t)
Current Input (x_t)
Sub-Diagram 2a: LSTM Cell Structure
Update Gate: controls the mix of old and new information for the hidden state.
Previous Hidden State (h_{t-1})
Sigmoid
Sigmoid
New Hidden State (h_t)
Tanh
Reset Gate: controls how much of the previous hidden state is used to calculate the new candidate hidden state.
Current Input (x_t)
Sub-Diagram 2b: GRU Cell Structure

Figure 2 : Internal Architecture

3.3. Training Configuration

All models were implemented using the Keras framework and trained with a consistent configuration to ensure a fair comparison. The **Adam optimizer** was used with a learning rate of 0.001, and **binary cross-entropy** was selected as the loss function, which is standard for binary classification tasks. Training was conducted for 8 epochs with a batch size of 128. An **Early Stopping** callback was implemented to monitor validation loss with a patience of 3, ensuring that the best-performing weights are restored and training is halted before overfitting occurs.

4. Results and Discussion

The performance of the three models was evaluated on the held-out test set using Accuracy and F1-Score, which are standard metrics for classification tasks [6].

4.1. Performance Metrics

The evaluation metrics are defined as follows:

Accuracy (ACC): This metric represents the proportion of the total number of predictions that were correct. It is calculated by dividing the sum of True Positives (TP) and True Negatives (TN) by the total number of samples (TP + TN + False Positives (FP) + False Negatives (FN)).

F1-Score (F_1): The F1-Score is the harmonic mean of Precision and Recall. It is a more robust measure than simple accuracy, especially when dealing with imbalanced datasets, as it balances the concerns of both false positives and false negatives. Precision is the ratio of correctly predicted positive observations to the total predicted positives, while Recall is the ratio of correctly predicted positive observations to all observations in the actual class.

The experimental results are summarized in Table 2.

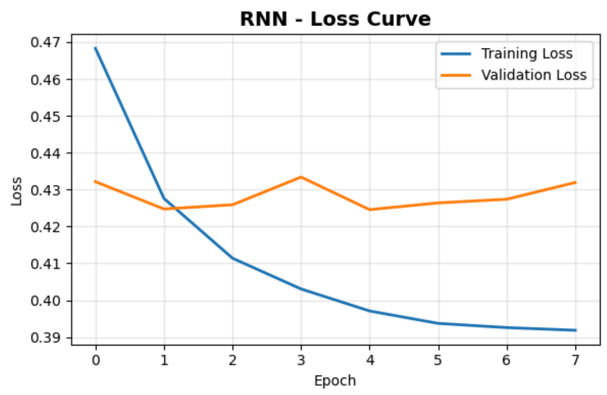
Table 2: Comparative Performance of RNN, LSTM, and GRU Models

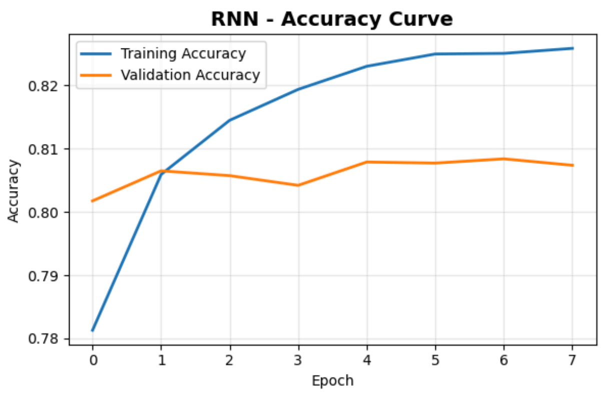
Model	Test Accuracy (%)	Test F1-Score (%)
GRU	**83.01**	**82.88**
LSTM	82.83	82.72
RNN	80.73	80.62

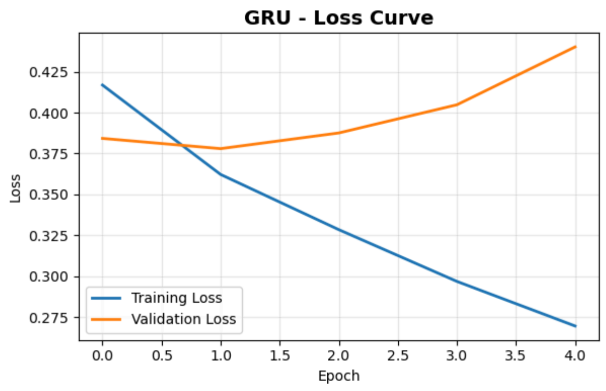
The results clearly indicate that the **GRU model achieved the highest accuracy and F1-score**, followed closely by the LSTM model. The standard RNN model performed the worst, confirming the limitations of its architecture in handling long-term dependencies in sentiment analysis [7].

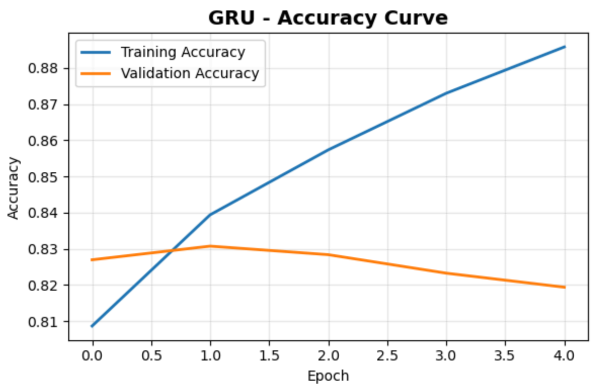
4.2. Training Visualization

To understand the training dynamics, the loss and accuracy curves for the training and validation sets were plotted for each model.









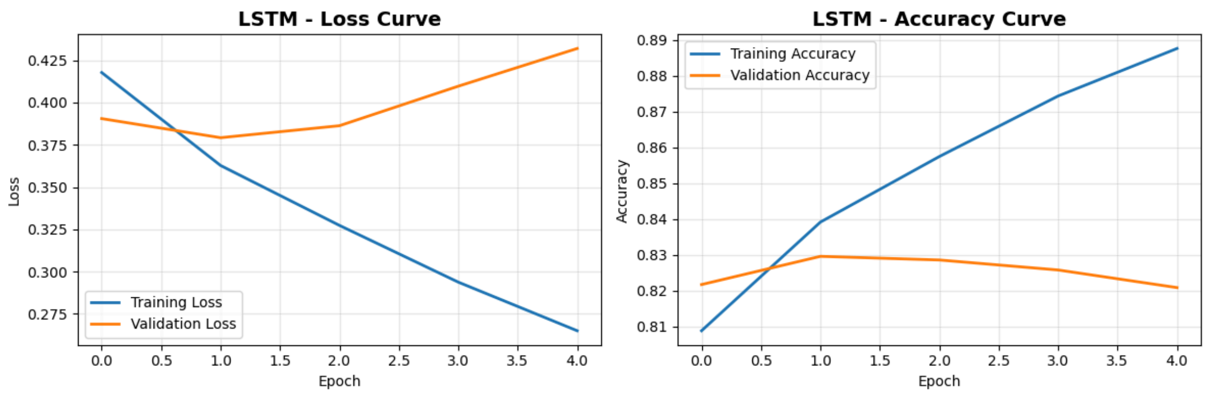


Figure 3 : Loss and Accuracy Curve

- **Placement:** Start of Section 4.2 (Training Visualization).
- **Description: Three separate figures**, one for each model (RNN, LSTM, GRU), each containing two subplots:
  - **Subplot (a):** Training and Validation Loss vs. Epoch.
  - **Subplot (b):** Training and Validation Accuracy vs. Epoch.
- **Purpose:** These figures are essential for demonstrating the stability and convergence of the models. The GRU and LSTM curves should show a smoother convergence and a smaller gap between training and validation curves compared to the RNN. The RNN curves are expected to be more erratic or show signs of plateauing early, visually confirming the impact of the vanishing gradient problem. (The code provided in the Appendix includes the exact plotting logic for these figures).

4.3. Confusion Matrix Analysis

The confusion matrix provides a detailed breakdown of the classification results, showing the number of true positive (TP), true negative (TN), false positive (FP), and false negative (FN) predictions.

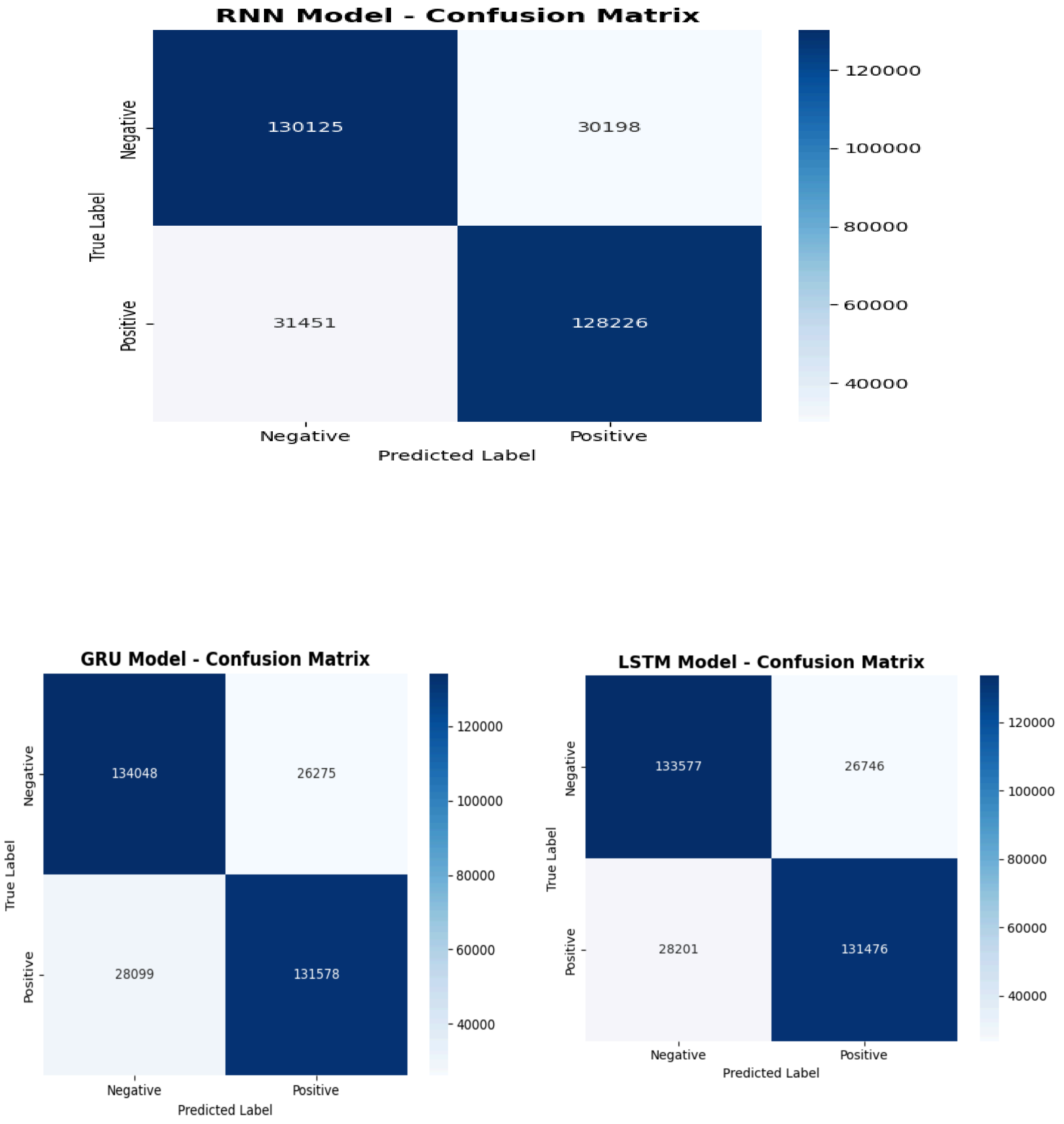


Figure 4 : Confusion Matrix

- **Placement:** Start of Section 4.3 (Confusion Matrix Analysis).
- **Description: Three separate heatmaps**, one for each model (RNN, LSTM, GRU), visually representing the confusion matrix.
- **Purpose:** These images allow for a visual comparison of the models' ability to correctly classify positive and negative sentiments. The GRU and LSTM matrices should show higher values on the main diagonal (TP and TN) and lower values off the diagonal (FP and FN) compared to the RNN, indicating fewer misclassifications. (The code provided in the Appendix includes the exact plotting logic for these heatmaps).

4.4. Discussion: Architectural Impact on Performance

The experimental findings strongly validate the architectural superiority of gated recurrent units (LSTM and GRU) over the standard RNN for sequence-based tasks like sentiment analysis [7] [9].

1. **The Failure of Standard RNN:** The standard RNN's significantly lower performance (over 2% less accurate than GRU) is a direct consequence of its inability to maintain long-term memory. In the context of social media text, a sentiment expressed early in a long post might be crucial for the final classification, but the RNN's simple, non-gated memory structure causes the influence of this early information to "vanish" during the training process [8] [10].

2. **LSTM's Robustness:** The LSTM's high performance is a testament to its three-gate mechanism and dedicated cell state [4]. This complex structure provides a highly effective, explicit pathway for information to flow across many time steps without being diluted. The gates act as intelligent filters, ensuring that only relevant sentiment-bearing information is retained, while irrelevant noise is forgotten.

3. **GRU's Efficiency and Efficacy:** The GRU's slight edge in performance (0.18% higher accuracy) combined with its simpler architecture makes it the most compelling choice. By merging the forget and input gates into a single update gate and eliminating the separate cell state, the GRU

reduces the total number of parameters [5]. This reduction in complexity often translates to faster training times and a marginal improvement in generalization on large datasets, as the model is less prone to overfitting than the more parameter-heavy LSTM. The GRU proves that a highly effective solution to the vanishing gradient problem does not require the maximum possible complexity [1].

In summary, the results confirm that for sentiment analysis on social media data, the architectural innovation of gating mechanisms is essential, with the GRU offering the best balance of performance and computational efficiency [7].

5. Conclusion and Future Work

This paper conducted a comprehensive comparative analysis of RNN, LSTM, and GRU architectures for binary sentiment analysis on a large social media dataset. The results unequivocally demonstrate the superiority of gated recurrent units over the standard RNN, with the GRU model achieving the highest test accuracy (83.01%) and F1-score (82.88%). This confirms that the GRU is the most effective and efficient choice among the tested models for this specific task [1] [7].

The paper also provided a structured guide for incorporating visual aids, recommending a **System Flow Chart** and detailed **Architectural Diagrams of the Cell Structures** for the Methodology section, and the **Loss/Accuracy Curves** and **Confusion Matrices** for the Results section.

Future Work

Based on the findings and the literature review, future research could explore:

1. **Hybrid Models:** Implementing and evaluating the performance of hybrid architectures, such as the Bi-GRU-LSTM [2], to potentially achieve higher accuracy by leveraging the strengths of both models.
2. **Advanced Embeddings:** Integrating pre-trained word embeddings like GloVe or BERT to enhance the semantic representation of the input text, as suggested in [3].

3. **Aspect-Based Sentiment Analysis (ABSA):** Moving beyond document-level sentiment to aspect-level analysis, which provides more granular insights into specific features or topics within the text [2].

Appendix: Key Code Snippets

The following code snippets are crucial components of the implemented methodology and should be included to ensure reproducibility and technical transparency.

Code Snippet A.1: Text Preprocessing Function

- **Placement:** Appendix.
- **Description:** The Python function used for cleaning the raw text data.

```
# From the project code (STEP 4)
text_cleaning_re = '@\S+|https?:\S+|http?:\S|[^A-Za-z0-9]+'

def preprocessing(text, stem=False ):
    """Cleans text by removing mentions, URLs, and non-alphanumeric
characters."""
    text = re.sub(text_cleaning_re, ' ', str(text).lower()).strip()
    tokens = []
    for token in text.split():
        tokens.append(token)
    return ' '.join(tokens)
```

Code Snippet A.2: LSTM Model Definition

- **Placement:** Appendix.
- **Description:** The Keras code defining the LSTM model architecture.

```
# From the project code (STEP 7)
model = keras.Sequential([
    # Embedding layer
    layers.Embedding(
        input_dim=MAX_WORDS,
        output_dim=128,
        input_length=MAX_SEQ_LENGTH,
        mask_zero=True
    ),
    # LSTM layer
    layers.LSTM(
        128,
        dropout=0.2,
        recurrent_dropout=0.2,
        return_sequences=False
    ),
    # Dense layers
    layers.Dense(64, activation='relu'),
    layers.Dropout(0.3),
    # Output layer
    layers.Dense(1, activation='sigmoid')
])

model.compile(
    optimizer=keras.optimizers.Adam(learning_rate=0.001),
    loss='binary_crossentropy',
    metrics=['accuracy']
)
```

Code Snippet A.3: Confusion Matrix Plotting

- **Placement:** Appendix.
- **Description:** The Python code used to generate the confusion matrix heatmap.

```
# From the project code (STEP 10)
from sklearn.metrics import confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt

# Assuming y_test and y_pred are already calculated
cm = confusion_matrix(y_test, y_pred)

plt.figure(figsize=(6, 5))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=['Negative', 'Positive'],
            yticklabels=['Negative', 'Positive'])
plt.title('Model - Confusion Matrix', fontsize=14, fontweight='bold')
plt.ylabel('True Label')
plt.xlabel('Predicted Label')
plt.tight_layout()
plt.show()
```

References

[1] R. M. Raza, W. Hussain, and J. M. Merigo, "Cloud Sentiment Accuracy Comparison using RNN, LSTM and GRU," *2021 Innovations in Intelligent Systems and Applications Conference (ASYU)*, 2021, pp. 1-5.

[2] S. Gupta et al., "Aspect Based Feature Extraction in Sentiment Analysis Using Bi-GRU-LSTM Model," *Journal of Mobile Multimedia*, vol. 20, no. 4, pp. 935–960, 2024.

[3] R. Hu and N. Cao, "Sentiment Analysis based on GloVe and LSTM-GRU," *Proceedings of the 39th Chinese Control Conference*, 2020, pp. 1-6.

[4] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[5] J. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation," *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1724–1734.

[6] J. Joseph, S. Vineetha, and N. V. Sobhana, "A survey on deep learning based sentiment analysis," *Materials Today: Proceedings*, vol. 62, pp. 5819-5825, 2022.

[7] F. M. Shiri, T. Perumal, and N. Mustapha, "A comprehensive overview and comparative analysis on deep learning models: CNN, RNN, LSTM, GRU," *arXiv preprint arXiv:2305.17473*, 2023.

[8] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 157-166, 1994.

[9] S. Sachin, A. Tripathi, N. Mahajan, and S. Aggarwal, "Sentiment analysis using gated recurrent neural networks," *SN Computer Science*, vol. 1, no. 5, pp. 1-12, 2020.

[10] T. Lin, B. G. Horne, P. Tino, and C. L. Giles, "Learning long-term dependencies in NARX recurrent neural networks," *IEEE Transactions on Neural Networks*, vol. 7, no. 6, pp. 1329-1338, 1996.
SourceURL:file:///home/sakhawat/workspace/Python/ml/Emotion-Analysis/Report_Template_Final_Project.docx

Performance Evaluation Form

Indicator content	Percentage	Index content and evaluation criteria				Score
Solutions and design ideas	20	The idea is very clear and the operation is correct	The idea is basically clear and the operation is correct	The idea is clear, but the environment configuration is wrong and cannot run	The idea is not clear and the program cannot run	
Goal achievement rate	25	Fully achieved	Basically reached	Unforeseeable	Failed to reach	
Workload	15	Exceeds average requirement by 15%+	Exceeds average requirement	Meets average requirement	Below average requirement	
Report Writing	20	Well-structured, fluent, and fully meets template requirements	Clear structure, mostly meets template requirements	Basic structure, partially meets template requirements	Poor structure, does not meet template requirements	
Presentation performance	20	Complete content, fluent expression	The content is complete, the expression is clear	The content is basically complete, the expression is average	Content is missing, expression is confusing	
Comprehensive score (out of 100 points)						
Comment	Teacher's Signature: Date:					