A2024S-T3 AML 3104 - Neural Networks and Deep Learning 01

Assignement-5 Report

on

RNN and LSTM.

Submitted to:

Ishant Gupta

Submitted by:

Rewant Sharma: c0894265

Submission Date:

11th August 2024

Contents

1. **Introduction** 3

1.1 what isSentiment Analysis 3

1.2 Applications of Sentiment Analysis 3

2. **Recurrent Neural Networks (RNNs)** 3

2.1 How RNNs Differ from Traditional Feedforward Neural Networks 3

2.2 Common Issues with RNNs: 4

2.3Tools and techniques used for analysis: 4

3. **Data Preparation and Preprocessing** 5

3.1 Loading the IMDB 5

3.2 Tokenizing and Padding Sequences: 5

4. **Building the RNN Model** 5

4.1 Model Architecture 5

5.**Training the model** 6

5.1 Splitting the dataset 6

5.2 Training with early stoppin: 6

6. **Evaluate the model** 7

6.1 Evaluating model performancce 7

6.2 Plot the evaluation metrices: 7

7. **Hyperparameter tuning** 7

7.1 Experiment1: 7

7.2 Experiment2: 7

7.3 Experiment3: 7

8. **Evaluate the model** 8

6.1 Feed forward neural network 8

6.2 Training with early stopping: 8

9. **Conclusion** 9

1. Introduction

1.1 What is Sentiment Analysis?

Sentiment analysis is a branch of study that employs natural language processing (NLP), text analysis, and computational linguistics to uncover and quantify subjective information and affective states in text data. It is often referred to as opinion mining or emotion AI. It seeks to identify the sentiment conveyed in a text and classify it as neutral, positive, or negative. Sentiment analysis that is more sophisticated can also identify feelings like joy, rage, or grief.

1.2Applications of Sentiment Analysis

Sentiment analysis is widely used across various domains, including:

- **Social Media Monitoring**: Analyzing public sentiment about brands, products, or events on platforms like Twitter and Facebook.
- **Customer Feedback Analysis**: Understanding customer opinions from reviews, surveys, and support tickets to improve products and services.
- **Market Research**: Gauging public opinion on new product launches or marketing campaigns.
- **Brand Reputation Management**: Monitoring online mentions and reviews to manage brand image and respond to negative feedback promptly.
- **Employee Feedback**: Analyzing employee surveys and feedback to improve workplace satisfaction and productivity

2. Recurrent Neural Networks (RNNs)

2.1 How RNNs Differ from Traditional Feedforward Neural Networks

1.**Recurrent Neural Networks (RNNs)** are a class of neural networks designed to recognize patterns in sequences of data, such as time series or natural language. Unlike traditional feedforward neural networks, which process inputs independently, RNNs have connections that form directed cycles, enabling them to maintain a memory of previous inputs.

- **Sequential Data Handling:** RNNs are specifically designed to handle sequential data, making them suitable for tasks like language modeling, speech recognition, and time series prediction.
- **Memory:** RNNs have an internal state (hidden state) that can capture information about previous inputs, allowing them to maintain context over sequences.

2. **The Concept of Hidden States in RNNs**

In RNNs, hidden states are used to store information about previous time steps. The network updates its hidden state at each time step based on the current input and the previous hidden state.

- **Information Passing:** At each time step, the input data and the previous hidden state are combined to produce a new hidden state, which is then used to make predictions or pass information to the next time step.
- **Mathematical Representation:** The hidden state htht at time step tt is typically calculated as:

  $$\mathbf{ht=f(Wh\cdot ht-1+Wx\cdot xt+b)}\ \boldsymbol{ht=f(Wh\cdot ht-1+Wx\cdot xt+b)}$$

  where ff is an activation function (commonly tanh or ReLU), WhWh and WxWx are weight matrices, xtxt is the input at time tt, and bb is a bias term.

2.2.Common Issues with RNNs:

Vanishing and Exploding Gradients

RNNs can suffer from vanishing and exploding gradient problems, which affect their ability to learn long-range dependencies in sequences.

- **Vanishing Gradients:** Occurs when gradients become too small during backpropagation through time, causing the network to stop learning. This is common when using activation functions like sigmoid or tanh.
- **Exploding Gradients:** Occurs when gradients become too large, leading to unstable updates and divergence during training.

To mitigate these issues, techniques such as gradient clipping, using alternative architectures like Long Short-Term Memory (LSTM) networks or Gated Recurrent Units (GRUs), and employing more sophisticated optimization algorithms can be used.In summary, sentiment analysis and RNNs are powerful tools in the realm of NLP and machine learning. Sentiment analysis helps extract valuable insights from textual data, while RNNs enable the modeling of sequential dependencies, crucial for understanding context in language and other time-dependent data.

2.3 Tools and techniques used for analysis

The analysis was performed using a variety of tools and techniques that ensured a thorough examination of the data. Tools and techniques used include:

1. **Tools**:

- **Tensor flow IMDB database**: Used for data collection
- **Python**: Used to clean, manipulate and analyze data.
- **Visual Studio code**: Interactive environment used for coding and visualization.
- **Pandas**: Facilitated data manipulation and analysis.
- **Matplotlib**: Used for data visualization.

2. **Techniques**:

- **Data Cleaning and Preprocessing**: As the Data we chose from an already cleaned dataset which is pre-provided and tokenization which is already done by tensor flow when we extract the data into aur dataset.
- **Sentiment Analysis**: Performed on post comments to assess the overall sentiment (positive, negative or neutral) of the content.
- **Trend Analysis**: Identified trends in engagement metrics over time.
- **Visualization**: Created various graphs to visualize data and identify patterns.

This combination of tools and techniques provided a comprehensive analysis of social media data that resulted in valuable insights and strategic recommendations.

3. **Data Preparation and Preprocessing**

3.1 Loading the IMDB: Dataset The IMDB dataset is a well-known dataset for sentiment analysis, consisting of 50,000 movie reviews labeled as positive or negative.

3.2 Tokenizing and Padding Sequences: The dataset is already tokenized by TensorFlow. We need to pad the sequences to ensure uniform input length.

4. Building the RNN Model

4.1 Model Architecture: We will implement an RNN model using TensorFlow and Karas, including the following layers:

- **Input Layer** The input layer serves as the entry point for the data into the neural network. It doesn't perform any computations but defines the shape of the input data. *Implementation*: In Kera's, the input layer is implicitly defined by specifying the input shape parameter in the first layer of the model. In this case, the input shape is (250,), meaning each input sequence has 250 tokens.

- **Embedding Layer:** The embedding layer converts integer-encoded words into dense vectors of fixed size. This layer is essential for natural language processing tasks as it transforms sparse, high-dimensional input data into a lower-dimensional, dense representation.

    - **input_dim**: The size of the vocabulary (10,000 words in this case).
    - **output_dim**: The size of the embedding vectors (32 dimensions).
    - **input_length**: The length of input sequences (250 tokens).

    Role: The embedding layer helps capture semantic relationships between words by mapping them to continuous vector spaces where similar words have similar representations

- **RNN Layer (LSTM):** The Long Short-Term Memory (LSTM) layer is a type of recurrent neural network (RNN) layer designed to handle long-term dependencies in sequential data. LSTMs are effective at capturing temporal patterns and relationships in data.

- *Implementation*:
    - **units:** The number of LSTM units (64 in this case).
    - **return_sequences**: If True, the LSTM layer returns the full sequence of outputs for each input sequence. This is useful when stacking multiple RNN layers.
- *Role:* The embedding layer helps capture semantic relationships between words by mapping them to continuous vector spaces where similar words have similar representations

- **Fully Connected Layer:** The fully connected (Dense) layer is a standard neural network layer where each neuron is connected to every neuron in the previous layer. It is typically used for classification tasks.
- *Implementation*:
    - **units**: The number of neurons in the layer (16 in this case).
    - **activation**: The activation function applied to the output of the layer ('relu' in this case).
- *Role:* The Dense layer learns complex features and patterns by combining the outputs of the previous layers. The ReLU activation function introduces non-linearity, allowing the model to learn more complex relationships.

- **Output Layer:** The output layer produces the final predictions of the model. For binary classification tasks, a single neuron with a sigmoid activation function is typically used.
    - **units:** The number of neurons in the layer (1 in this case).
    - **activation:** The activation function applied to the output of the layer ('sigmoid' in this case).
- *Role:* The output layer produces a probability value between 0 and 1, indicating the likelihood of the input belonging to the positive class. The sigmoid activation function squashes the output to this range, making it suitable for binary classification.

5. **Training the model**

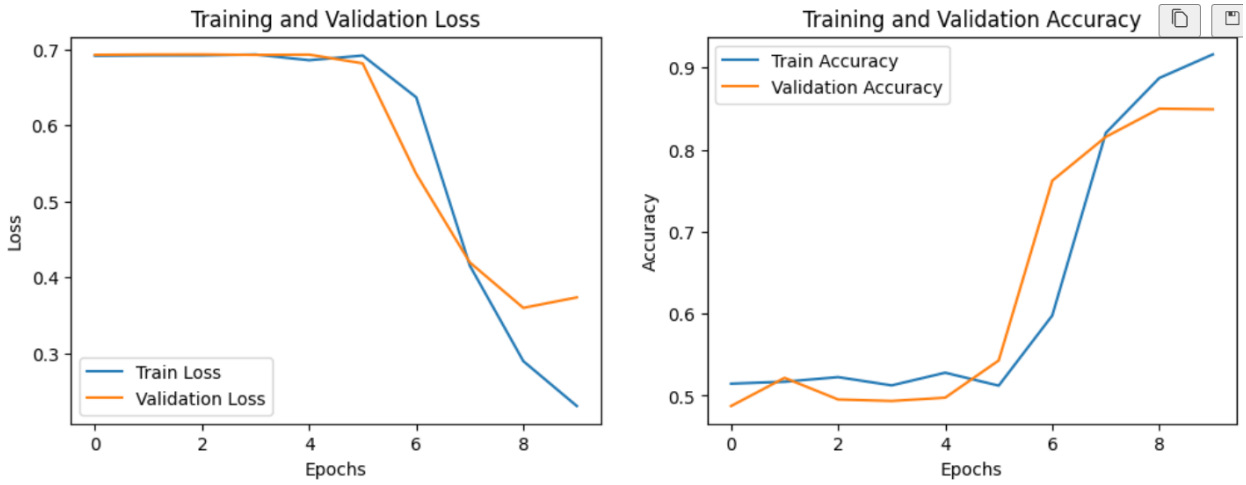
5.1 Splitting the Dataset: Split the dataset into training and validation sets.

5.2 Training with early stopping: Train the model and use early stopping to prevent overfitting.

6. Evaluating the Model

6.1 Evaluating Model Performance: Evaluate the model's performance on the test set.

6.2 Plot Training and Validation Metrics: The dataset is already tokenized by TensorFlow. We need to pad the sequences to ensure uniform input length.



The Above Graph shows that as we move forward with our epoch the loss for both training and validation decrease significantly and saturates after 8th epoch.

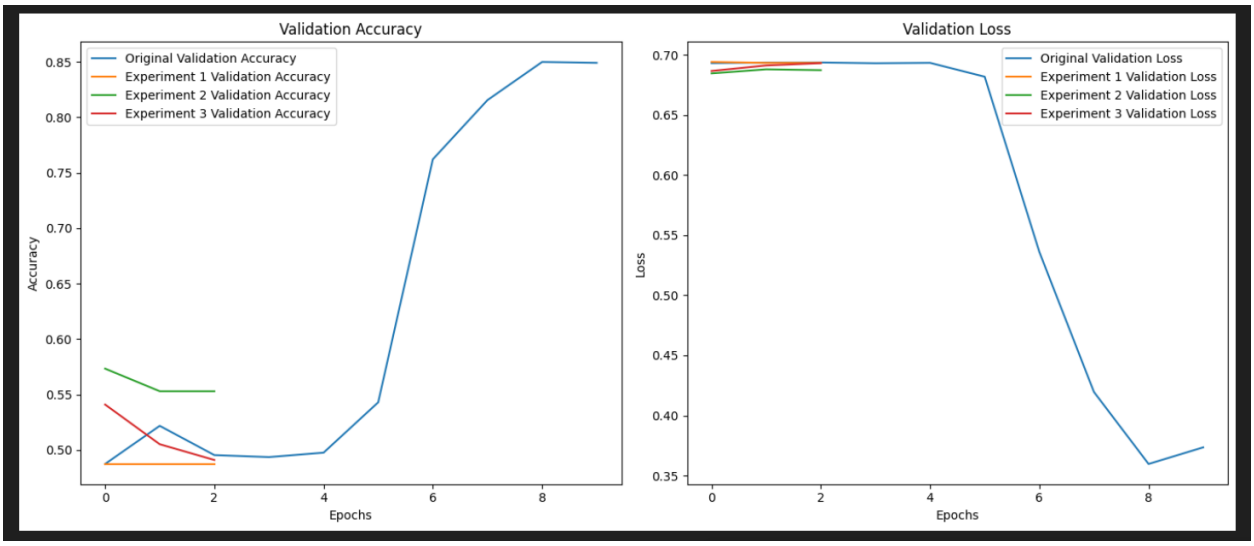
For the Accuracy we can see that as we move forward with the epochs the training and validation accuracy also increases.

7. Hyperparameter Tunning

7.1 Experiment1: Increase LSTM Units: Split the dataset into training and validation sets.

7.2 Experiment:2Increase Dropouts: Train the model and use early stopping to prevent overfitting.

7.3 Experiment:3Change Learning Rate: Train the model and use early stopping to prevent

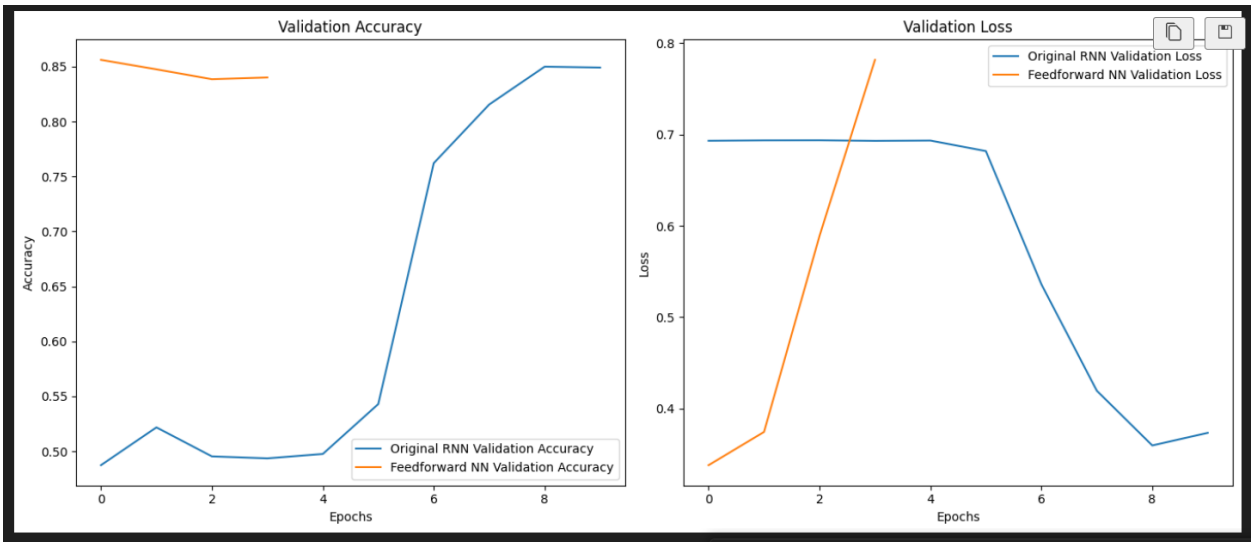


In the above experiment we can see the originally selected Later with less embedding , less layers and less dropouts out performs everyone .

8. **Comparative Analysis**

8.1 Feed Forward Neural Network: Split the dataset into training and validation sets.

8.2 Training with early stopping: Train the model and use early stopping to prevent overfitting.



From the above Graph we can tell that the validation accuracy for an RNN is high and increases as compared to a feed forward network. On the flip side, the validation loss for an original RNN would decrease as compared a feed forward network, which increases as the number of epochs increases.

By comparing the test accuracy and validation performance of both the original RNN and the simple feedforward neural network, we can determine which model performs better for the sentiment analysis task. This approach helps us understand the strengths and weaknesses of each architecture and make informed decisions about model selection

9.**Conclusion**

The project demonstrated the effectiveness of RNNs, particularly LSTM networks, in handling sentiment analysis tasks. By leveraging the sequential nature of text data, RNNs were able to capture temporal dependencies and provide better performance compared to traditional feedforward neural networks. Through hyperparameter tuning, we optimized the RNN model and achieved improved performance. The comparative analysis highlighted the strengths of RNNs in handling sequential data, making them a suitable choice for tasks like sentiment analysis. Overall, this project provided a comprehensive understanding of building, training, and optimizing RNN models for sentiment analysis, showcasing their potential in real-world applications. The insights gained from this project can be applied to other NLP tasks, further enhancing the capabilities of machine learning models in understanding and processing human language