Name: Aziz Sayyad
Roll No: 281069
Batch: A3

Assignment No: - 4

Problem Statement:
Implement a Recurrent Neural Network (RNN) in Python using Keras and TensorFlow for time series prediction, focusing on applications such as stock market price analysis or weather forecasting.

Objective:

- To understand the architecture and working of Recurrent Neural Networks.
- To preprocess time series data for sequential modeling.
- To implement RNN-based models for prediction tasks.
- To evaluate model performance using appropriate metrics.
- To visualize predicted results against actual data.

S/W Packages and H/W apparatus used:

- **Operating System:** Windows/Linux/MacOS
- **Kernel:** Python 3.x
- **Tools:** Jupyter Notebook, Anaconda, or Google Colab
- **Hardware:** CPU with minimum 8GB RAM; GPU recommended for large datasets
- **Libraries and packages used:** TensorFlow, Keras, NumPy, Pandas, Matplotlib, Scikit-learn, yfinance (for stock data), OpenWeather API (for weather data)

Theory:
Recurrent Neural Networks (RNNs) are specialized neural networks designed for sequential data. Unlike feedforward networks, RNNs maintain a **hidden state** that captures dependencies between time steps, making them suitable for tasks like stock price prediction and weather forecasting.

Structure:

- **Input Layer:** Sequential time series data (e.g., past stock prices, temperature readings).
- **RNN/GRU/LSTM Layers:** Capture temporal dependencies and patterns in sequential data.
- **Dense Layer:** Produces predicted value(s).
- **Activation Functions:** ReLU/Tanh in hidden layers, linear activation in output layer.
- **Loss Function:** Mean Squared Error (MSE) or Mean Absolute Error (MAE).
- **Backpropagation Through Time (BPTT):** Used for training by unfolding the network across time steps.

Methodology:

1. **Data Acquisition:**
   - For stock market: Use APIs like Yahoo Finance (`yfinance`).
   - For weather: Use APIs such as OpenWeather or historical weather datasets.
2. **Data Preparation:**
   - Handle missing values.
   - Normalize data between 0 and 1.
   - Create time windows (sliding window approach).
   - Split into training and testing datasets.
3. **Model Architecture:**
   - RNN / LSTM / GRU layers for sequence modeling.
   - Dense layer for prediction output.
4. **Model Compilation:** Optimizer (Adam), Loss function (MSE/MAE).
5. **Model Training:** Train on historical data, validate with unseen data.
6. **Model Evaluation:** Evaluate using RMSE, MAE, and prediction plots.
7. **Visualization:** Plot actual vs predicted values to analyze performance.

Advantages:

- Captures sequential patterns in time series data.
- Suitable for financial and environmental forecasting.
- Can handle long-term dependencies (with LSTM/GRU).
- High accuracy with sufficient data and proper tuning.
- Flexible for univariate and multivariate forecasting.

Limitations:

- Training can be slow and computationally expensive.
- Prone to vanishing/exploding gradient problems in standard RNNs.
- Requires large historical datasets for accuracy.
- Sensitive to noisy or non-stationary data (common in stock prices).
- Less interpretable compared to traditional statistical models (e.g., ARIMA).

Applications:

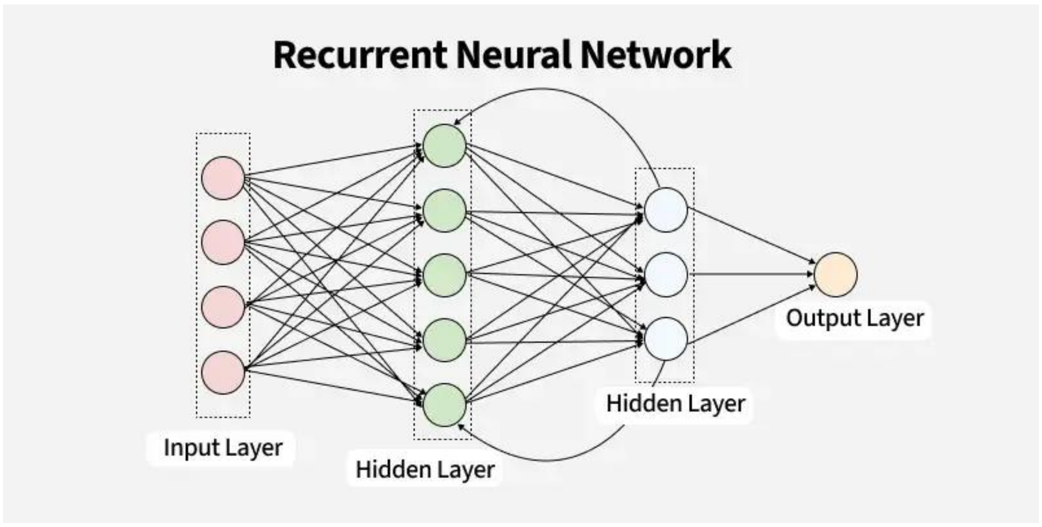
- **Stock Market Prediction:** Forecasting stock prices, indices, and trends.
- **Weather Forecasting:** Predicting temperature, rainfall, and humidity.
- **Energy Demand Forecasting:** Predicting electricity/gas consumption.
- **Traffic Prediction:** Estimating congestion trends.
- **Healthcare:** Predicting patient vital signs over time.

Working / Algorithm:

1. Import required libraries (TensorFlow, NumPy, Pandas, etc.).
2. Acquire dataset (stock prices or weather data).
3. Preprocess data (clean, normalize, and create time windows).
4. Split dataset into training and testing sets.
5. Build RNN/LSTM model with sequential layers.
6. Compile with optimizer (Adam) and loss (MSE/MAE).

7. Train the model with time series data.
8. Evaluate on test dataset using RMSE/MAE.
9. Predict future values based on historical input.
10. Visualize actual vs predicted trends using plots.

Diagram:



Conclusion:
RNN-based models, particularly LSTMs and GRUs, are effective for time series prediction tasks such as stock market analysis and weather forecasting. By capturing sequential dependencies, these models can forecast future trends with high accuracy. Although they require significant computational resources and careful data preprocessing, RNNs remain a powerful solution for sequential data modeling and predictive analytics.