**Assignment No: - 4**

**Name :- Isha Ghorpade**

**Enrollnment :- 22420020**

**Roll :- 381066**

**Problem Statement:**

Time Series Prediction using Recurrent Neural Networks (RNNs)

**Objectives**

* To study the architecture and working of Recurrent Neural Networks (RNNs).
* To preprocess and structure time series data for model training.
* To design and implement an RNN model using Keras and TensorFlow.
* To evaluate the predictive performance of the model on unseen test data.
* To visualize and compare predicted results with actual time series values.

**Software and Hardware Requirements**

* Operating System: Windows / Linux / macOS
* Programming Kernel: Python 3.x
* Tools/IDEs: Jupyter Notebook, Anaconda, or Google Colab
* Hardware: Minimum 4 GB RAM; GPU recommended for faster computation
* Libraries/Packages: TensorFlow, Keras, NumPy, Pandas, Matplotlib, Scikit-learn

**Theory**

**Definition**

Recurrent Neural Networks (RNNs) are a class of deep learning models designed to handle sequential data. Unlike feedforward networks, RNNs maintain a hidden state that captures dependencies from previous inputs, making them highly effective for time series forecasting.

Architecture

* Input Layer: Accepts sequential data points (e.g., stock prices, weather readings).
* Recurrent Layers: Composed of units such as SimpleRNN, LSTM (Long Short-Term Memory), or GRU (Gated Recurrent Unit) that capture temporal dependencies.
* Fully Connected (Dense) Layer: Processes the learned features into final outputs.
* Output Layer: Produces predicted values (e.g., next-day stock price).

Activation Functions

* Tanh: Keeps values within [-1,1] and stabilizes the flow of gradients.
* Sigmoid: Useful for gating mechanisms in LSTM/GRU.

Memory Cells

* LSTM and GRU architectures extend RNNs by mitigating the vanishing gradient problem.
* They allow the model to retain long-term dependencies, which are essential for accurate forecasting.

**Methodology**

1. Data Acquisition
   * Collect historical stock prices or weather data from sources like Yahoo Finance, Kaggle, or weather APIs.
2. Data Preparation
   * Select relevant features (e.g., closing stock prices).
   * Normalize data into a range [0,1] for stable training.
3. Sequence Creation
   * Convert raw data into sequences (e.g., last 60 time steps → predict next value).
4. Model Architecture
   * Build a Sequential model with:
     + RNN/LSTM/GRU layer(s) (e.g., 50 units).
     + Dense layer with one neuron for regression output.
5. Model Compilation
   * Optimizer: Adam
   * Loss Function: Mean Squared Error (MSE)
6. Model Training
   * Train using training dataset with validation on test set.
   * Define epochs (e.g., 10–20) and batch size (e.g., 32).
7. Evaluation
   * Test the model’s predictions on unseen data.
8. Prediction and Visualization
   * Forecast future values (e.g., next 20 days).
   * Compare predictions with actual values using line plots.

**Advantages**

* Sequential Data Handling: Specifically designed for time-dependent data.
* Captures Temporal Dependencies: Can learn patterns across multiple time steps.
* Flexibility: Works with variable-length input sequences.
* Long-Term Memory: LSTM and GRU handle long sequences effectively.

**Limitations**

* High Computational Cost: Training is resource-intensive for long sequences.
* Vanishing/Exploding Gradients: A challenge in traditional RNNs (solved partially by LSTM/GRU).
* Overfitting Risk: Small datasets may cause poor generalization.
* Large Data Requirement: Accurate predictions need a sufficient history of time series data.

**Applications**

* Stock Market Prediction: Forecasting future stock prices or trends.
* Weather Forecasting: Predicting temperature, rainfall, or humidity.
* Natural Language Processing (NLP): Text generation, speech recognition, and translation.
* Energy Demand Forecasting: Predicting electricity consumption patterns.
* Healthcare: Patient monitoring and disease trend prediction.

**Working / Algorithm**

**Step 1:** Import required libraries (NumPy, Pandas, TensorFlow, Matplotlib).  
**Step 2:** Load dataset (stock prices or weather data).  
**Step 3:** Convert date column to datetime format and set as index.  
**Step 4:** Extract target variable (e.g., closing prices).  
**Step 5:** Normalize data using MinMaxScaler.  
**Step 6:** Create sequences (e.g., last 60 values → next prediction).  
**Step 7:** Split into training (80%) and testing (20%) datasets.  
**Step 8:** Build RNN model using Keras Sequential API:

* Add **SimpleRNN/LSTM/GRU** layer.
* Add Dense output layer.  
  **Step 9:** Compile model with Adam optimizer and MSE loss.  
  **Step 10:** Train model with epochs and batch size.  
  **Step 11:** Predict next **N values** (e.g., 20 days).  
  **Step 12:** Inverse transform predictions to original scale.  
  **Step 13:** Compare with actual values.  
  **Step 14:** Plot **actual vs predicted values**.  
  **Step 15:** Print predicted vs actual values for evaluation.

**Conclusion**

Recurrent Neural Networks (RNNs) provide a strong framework for time series prediction by modeling sequential dependencies in data. Using specialized units like LSTMs or GRUs, they overcome limitations of traditional RNNs and excel in capturing both short-term and long-term patterns.

In this study, an RNN model was implemented for stock market forecasting / weather prediction. The model successfully demonstrated the ability to predict future values by learning from past data. While challenges such as high computational cost and overfitting exist, proper preprocessing, architecture selection, and hyperparameter tuning ensure robust and accurate predictions.

Thus, RNNs remain a powerful and versatile tool for solving real-world forecasting problems across multiple domains.