**Batch: HO -DL -1**

**Roll Number: 16010421073 Experiment Number: 7**

**Name: Keyur Patel**

**Title of the Experiment: Recurrent Neural Network**

**Program:**

**• Import Requisite Libraries**

**import numpy as np**

**import pandas as pd**

**import matplotlib.pyplot as plt**

**from sklearn.model\_selection import train\_test\_split**

**from sklearn.preprocessing import MinMaxScaler**

**from keras.models import Sequential**

**from keras.layers import LSTM, Dense**

**• Load any time series dataset.**

**df = pd.read\_csv('https://raw.githubusercontent.com/jbrownlee/Datasets/master/airline-passengers.csv')**

**• Pre-process and visualize the dataset.**

**plt.figure(figsize=(10, 6))**

**plt.plot(df['Month'], df['Passengers'], marker='o')**

**plt.title('Airline Passengers Dataset')**

**plt.xlabel('Month')**

**plt.ylabel('Number of Passengers')**

**plt.xticks(np.arange(0, len(df), step=12), df['Month'][::12], rotation=45)**

**plt.grid(True)**

**plt.show()**

**• Form the Training and Testing Data.**

**data = df['Passengers'].values.reshape(-1, 1)**

**scaler = MinMaxScaler(feature\_range=(0, 1))**

**scaled\_data = scaler.fit\_transform(data)**

**def create\_dataset(dataset, time\_steps=1):**

**X, y = [], []**

**for i in range(len(dataset) - time\_steps):**

**X.append(dataset[i:(i + time\_steps), 0])**

**y.append(dataset[i + time\_steps, 0])**

**return np.array(X), np.array(y)**

**time\_steps = 12**

**X, y = create\_dataset(scaled\_data, time\_steps)**

**X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)**

**Reshape input to be [samples, time steps, features]**

**X\_train = X\_train.reshape(X\_train.shape[0], X\_train.shape[1], 1)**

**X\_test = X\_test.reshape(X\_test.shape[0], X\_test.shape[1], 1)**

**• Develop and train the model.**

**model = Sequential()**

**model.add(LSTM(units=50, return\_sequences=True, input\_shape=(X\_train.shape[1], 1)))**

**model.add(LSTM(units=50))**

**model.add(Dense(1))**

**model.compile(optimizer='adam', loss='mean\_squared\_error')**

**model.fit(X\_train, y\_train, epochs=100, batch\_size=32)**

**Plot the predictions for training and testing data.**

**train\_predict = model.predict(X\_train)**

**test\_predict = model.predict(X\_test)**

**train\_predict = scaler.inverse\_transform(train\_predict)**

**test\_predict = scaler.inverse\_transform(test\_predict)**

**plt.figure(figsize=(10, 6))**

**plt.plot(df['Passengers'], label='Actual')**

**plt.plot(np.arange(time\_steps, len(train\_predict) + time\_steps), train\_predict, label='Train Prediction')**

**plt.plot(np.arange(len(train\_predict) + 2\*time\_steps, len(df)), test\_predict, label='Test Prediction')**

**plt.title('Airline Passengers Prediction')**

**plt.xlabel('Month')**

**plt.ylabel('Number of Passengers')**

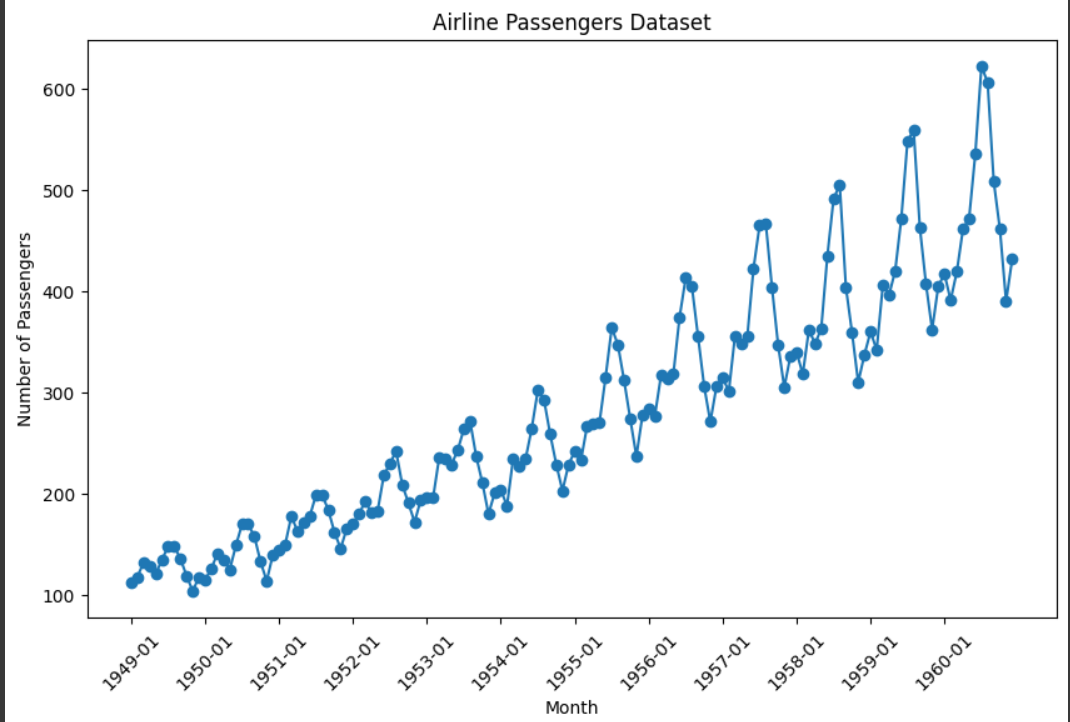
**plt.legend()**

**plt.grid(True)**

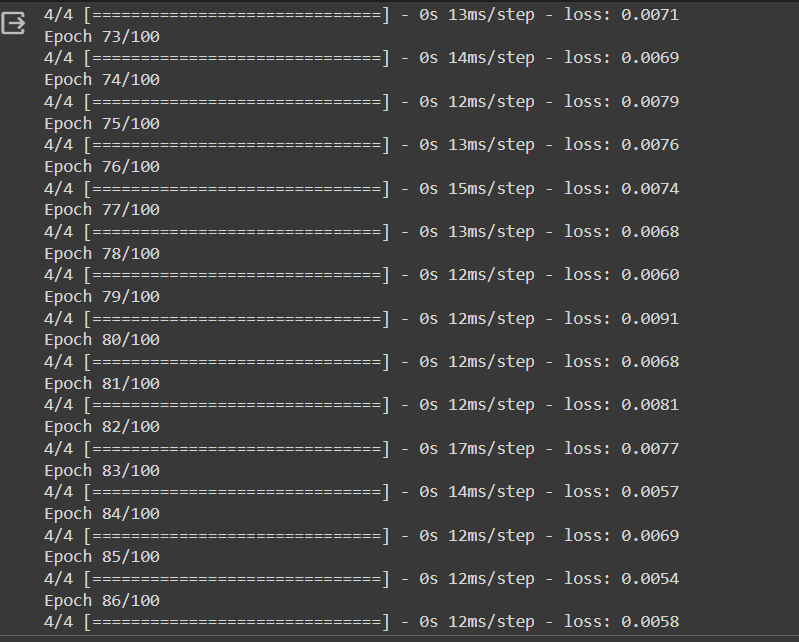
**plt.show()**

**Output:**

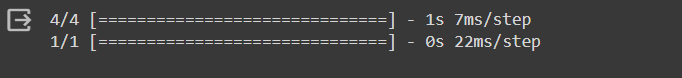
**Pre-process and visualize the dataset.**

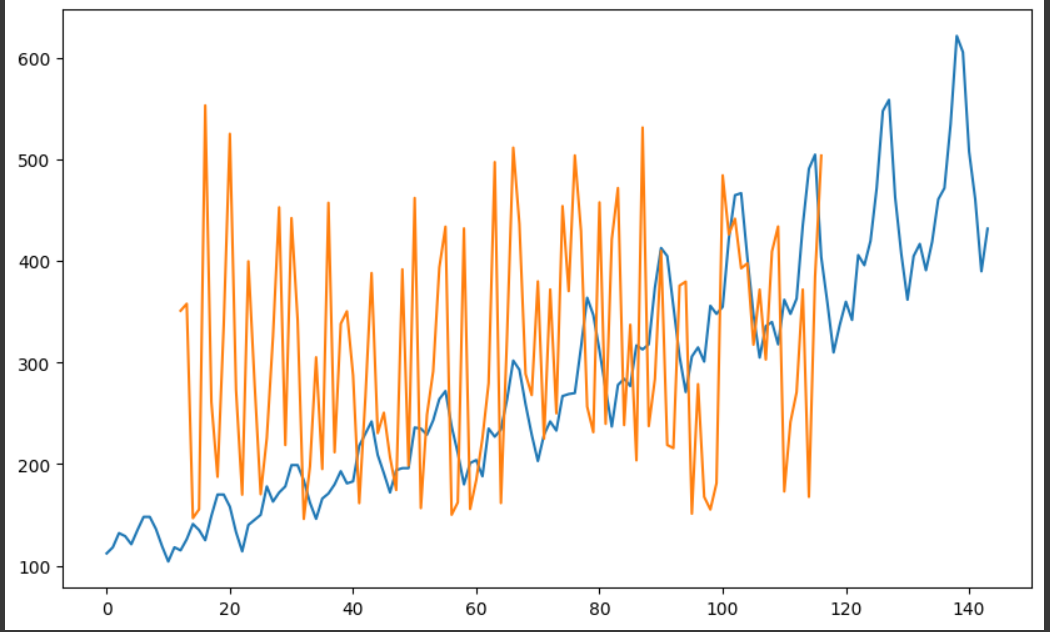
****

**Develop and train the model**

****

**Plot the predictions for training and testing data**

****

****

**Post Lab Question- Answers (If Any):**

**1. Differentiate between recurrent Neural Network and Feedforward Neural Network.**

**Ans:**

| **Feature** | **Feedforward Neural Network (FNN)** | **Recurrent Neural Network (RNN)** |
| --- | --- | --- |
| **Information Flow** | Unidirectional | Bidirectional (through time) |
| **Feedback Connections** | Absent | Present (Loops/cycles) |
| **Memory** | No explicit memory of past inputs | Maintains memory of past inputs |
| **Processing** | Fixed input size, output depends only on current input | Variable input size, output depends on both current and past inputs |
| **Suitable For** | Static data like images, static classification tasks | Sequential data like time series, natural language processing, speech recognition |
| **Architecture Complexity** | Simpler architecture | More complex architecture |
| **Training Difficulty** | Generally easier | More challenging due to vanishing/exploding gradients, and long-term dependencies |
| **Computationally Efficient** | More efficient | Less efficient, especially for long sequences |
| **Applications** | Image recognition, classification | Time series prediction, language modeling, translation |

**2. What are the problems associated with RNN.**

**Ans: Recurrent Neural Networks (RNNs) are powerful tools for handling sequential data, but they also come with several challenges:**

**1.** **Vanishing and Exploding Gradients:**

* RNNs suffer from the vanishing or exploding gradient problem during training.
* This occurs when gradients either diminish to zero or explode to extremely large values as they are back-propagated through time. It can hinder the training process, especially for long sequences.

**2.** **Difficulty in Capturing Long-Term Dependencies:**

* Traditional RNN architectures struggle to capture dependencies across long sequences.
* This is because the influence of early inputs diminishes rapidly as it propagates through time due to the recurrent nature of the network.
* As a result, RNNs may not effectively model long-range dependencies in sequential data.

**3.** **Training Instability:**

* RNNs are more prone to training instability compared to other architectures like feedforward networks.
* The dynamics of the recurrent connections can lead to oscillations or chaotic behavior during training, making convergence difficult.

**CO: 4 Underhand the essentials of Recurrent and Recursive Nets.**

**Conclusion: Thus we successfully implemented recurrent neural networks.**