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.

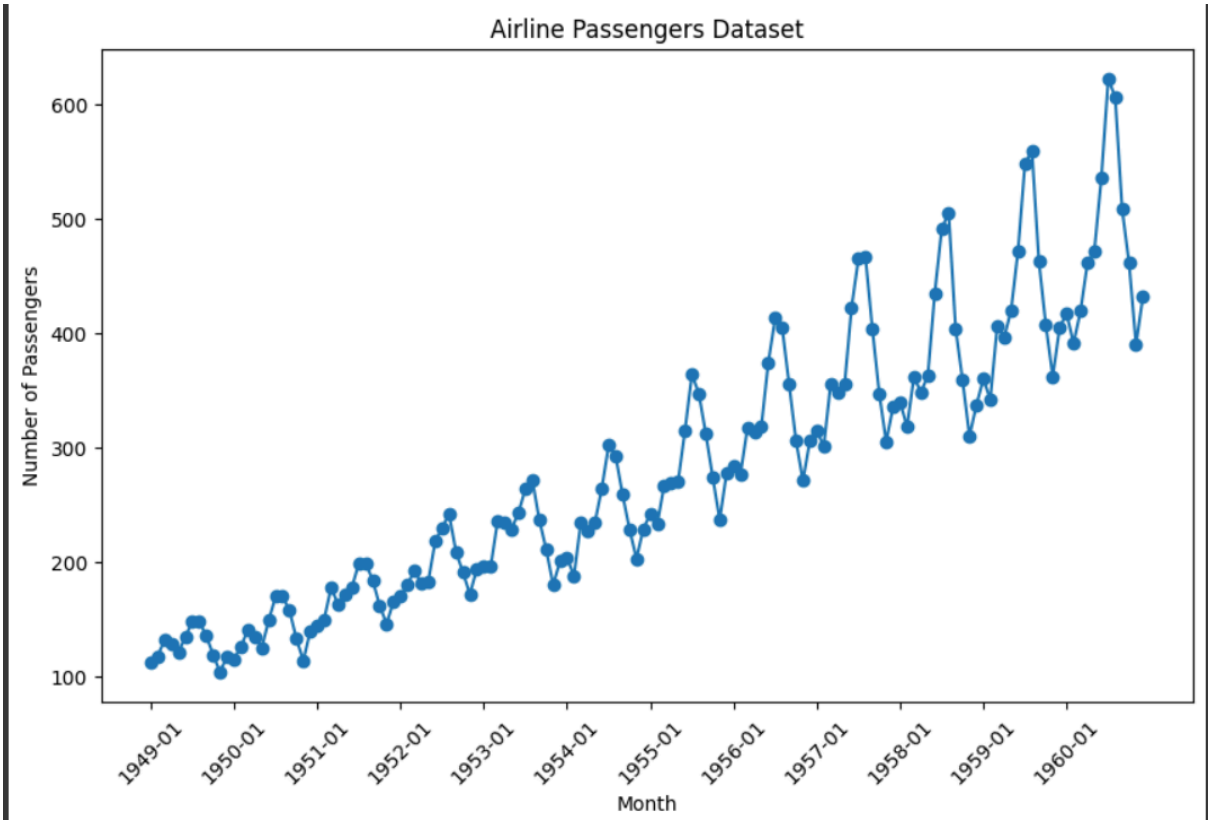
```
train_predict = model.predict(X_train)

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.

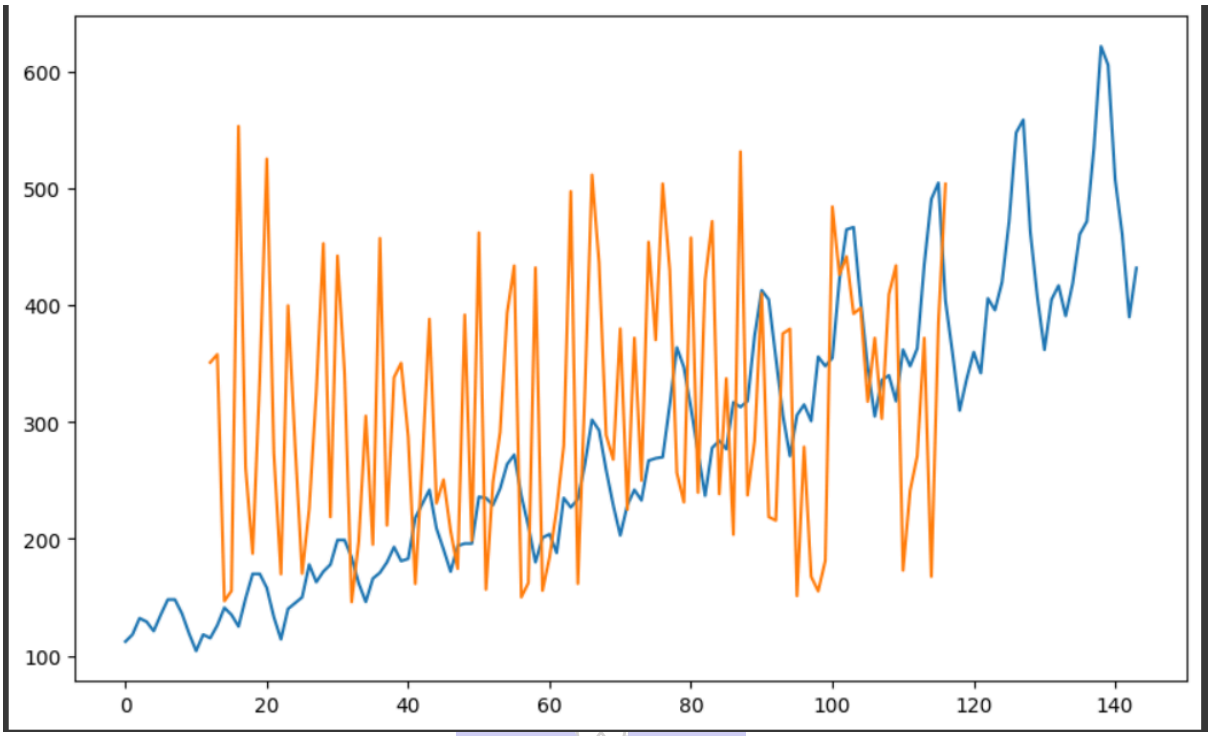


Develop and train the model

```
4/4 [==============================] - 0s 13ms/step - loss: 0.0071
Epoch 73/100
4/4 [==============================] - 0s 14ms/step - loss: 0.0069
Epoch 74/100
4/4 [==============================] - 0s 12ms/step - loss: 0.0079
Epoch 75/100
4/4 [==============================] - 0s 13ms/step - loss: 0.0076
Epoch 76/100
4/4 [==============================] - 0s 15ms/step - loss: 0.0074
Epoch 77/100
4/4 [==============================] - 0s 13ms/step - loss: 0.0068
Epoch 78/100
4/4 [==============================] - 0s 12ms/step - loss: 0.0060
Epoch 79/100
4/4 [==============================] - 0s 12ms/step - loss: 0.0091
Epoch 80/100
4/4 [==============================] - 0s 12ms/step - loss: 0.0068
Epoch 81/100
4/4 [==============================] - 0s 12ms/step - loss: 0.0081
Epoch 82/100
4/4 [==============================] - 0s 17ms/step - loss: 0.0077
Epoch 83/100
4/4 [==============================] - 0s 14ms/step - loss: 0.0057
Epoch 84/100
4/4 [==============================] - 0s 12ms/step - loss: 0.0069
Epoch 85/100
4/4 [==============================] - 0s 12ms/step - loss: 0.0054
Epoch 86/100
4/4 [==============================] - 0s 12ms/step - loss: 0.0058
```

Plot the predictions for training and testing data

```
4/4 [==============================] - 1s 7ms/step
1/1 [==============================] - 0s 22ms/step
```



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.
