Exp 8 Design and implement a deep learning network for forecasting time series data.

import numpy as np

import matplotlib.pyplot as plt

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import LSTM, Dense

from sklearn.preprocessing import MinMaxScaler

# Generate or load data (for example: sine wave)

time\_steps = 500

t = np.arange(time\_steps)

data = np.sin(0.02 \* t) + 0.5 \* np.random.normal(size=time\_steps)  # noisy sine wave

# Plot the data

plt.plot(t, data)

plt.title("Sample Time Series Data")

plt.show()

# Normalize data

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

data = data.reshape(-1, 1)

data\_scaled = scaler.fit\_transform(data)

# Create sequences

def create\_sequences(data, seq\_length):

    xs, ys = [], []

    for i in range(len(data) - seq\_length):

        x = data[i:i+seq\_length]

        y = data[i+seq\_length]

        xs.append(x)

        ys.append(y)

    return np.array(xs), np.array(ys)

SEQ\_LENGTH = 30

X, y = create\_sequences(data\_scaled, SEQ\_LENGTH)

# Train-test split

split = int(0.8 \* len(X))

X\_train, X\_test = X[:split], X[split:]

y\_train, y\_test = y[:split], y[split:]

# Build LSTM model

model = Sequential([

    LSTM(50, activation='tanh', input\_shape=(SEQ\_LENGTH, 1)),

    Dense(1)

])

model.compile(optimizer='adam', loss='mse')

# Train the model

history = model.fit(X\_train, y\_train, epochs=30, batch\_size=16, validation\_split=0.1)

# Predict on test data

y\_pred = model.predict(X\_test)

# Inverse transform to original scale

y\_test\_inv = scaler.inverse\_transform(y\_test)

y\_pred\_inv = scaler.inverse\_transform(y\_pred)

# Plot predictions vs actual

plt.plot(y\_test\_inv, label='Actual')

plt.plot(y\_pred\_inv, label='Predicted')

plt.title("Time Series Forecasting with LSTM")

plt.legend()

plt.show()







