**Deep Learning Project Report**

**Topic--** **Predicting PM25 Concentration with LSTM**

**--Biswajit Sarkar 2101AI11**

**Introduction:**

This report provides a comprehensive overview of the application of Long Short-Term Memory (LSTM) networks to predict PM2.5 concentration in four different cities: G, B, S, and T. The dataset employed for this analysis contains historical records of diverse environmental factors such as weather conditions, humidity, temperature, and pressure, among others. The primary goal of this project is to develop LSTM models capable of accurately forecasting PM2.5 concentrations, with a focus on the cities mentioned.

The entire process, from data pre-processing to model evaluation, is detailed within this report. Key performance metrics, including Mean Squared Error (MSE) and Mean Absolute Error (MAE), are employed to quantitatively assess the accuracy of the developed LSTM models.

**Data Preprocessing:**

The data is read using Pandas from CSV files for both training (df\_train) and testing (df\_test).

A scaling function (scaling) is defined to normalize specific columns using Min-Max scaling.

Function for scaling the dataset

def scaling(dataset):

  dataset.iloc[:,0]

  del dataset[dataset.columns[0]]

  cols\_to\_scale = ['weather','PM25\_Concentration','wind\_direction','temperature','pressure','humidity','PM10\_Concentration','NO2\_Concentration','CO\_Concentration','O3\_Concentration','SO2\_Concentration']

  from sklearn.preprocessing import MinMaxScaler

  scaler = MinMaxScaler()

  dataset[cols\_to\_scale] = scaler.fit\_transform(dataset[cols\_to\_scale])

  return dataset

**Data Splitting:**

The splitting function is defined to create sequences of input features (X) and target values (y) with a specified window size.

Function for splitting the dataset by considering step size into X train and Y train dataset.

def splitting(dataset,win):

  window\_size = win

# Initialize empty lists to store X and Y

  X\_sequences = []

  Y\_values = []

# Iterate through the DataFrame to create sequences

  for i in range(len(dataset) - window\_size):

      X\_seq = dataset.iloc[i:i+window\_size].values

      Y\_val = dataset.iloc[i+window\_size]['PM25\_Concentration']

      X\_sequences.append(X\_seq)

      Y\_values.append(Y\_val)

# Convert the lists to NumPy arrays for modeling

  X\_train = np.array(X\_sequences)

  y\_train = np.array(Y\_values)

  return X\_train,y\_train

**Model Architecture:**

**LSTM Neural Network:** The create\_model function establishes the architecture of the LSTM-based neural network using the Keras library.

**Sequential Model:** The model is sequential, meaning that layers are added one after the other.

**LSTM Layers:** Two LSTM layers are used with 100 and 50 units, respectively. LSTM layers are well-suited for capturing patterns in time series data.

**Dense Layer:** A dense layer with one unit and a ReLU activation function is added for the final output.

**Compilation:** The model is compiled with the Adam optimizer and mean squared error loss.

Code for creation of model

def create\_model(X\_train, y\_train):

  X\_train, Y\_train = np.array(X\_train), np.array(y\_train)

# Reshape the data

  X\_train = np.reshape(X\_train, (X\_train.shape[0], X\_train.shape[1], X\_train.shape[2]))

  model = keras.Sequential()

  model.add(layers.LSTM(100, input\_shape=(X\_train.shape[1], X\_train.shape[2]), return\_sequences=True))

  model.add(layers.LSTM(50, return\_sequences=False))

  model.add(layers.Dense(1,activation='relu'))

# Compile the model

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

  return model

Model: "sequential"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

lstm (LSTM) (None, 10, 100) 45200

lstm\_1 (LSTM) (None, 50) 30200

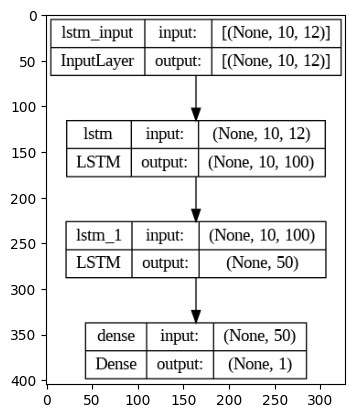
dense (Dense) (None, 1) 51

=================================================================

Total params: 75451 (294.73 KB)

Trainable params: 75451 (294.73 KB)

Non-trainable params: 0 (0.00 Byte)



**Training Process:**

The model is created for different window sizes (1, 7, 14, 30, and 60).

Each model is trained using the training data (X\_train, y\_train) with a specified number of epochs and batch size.

**Testing Process:**

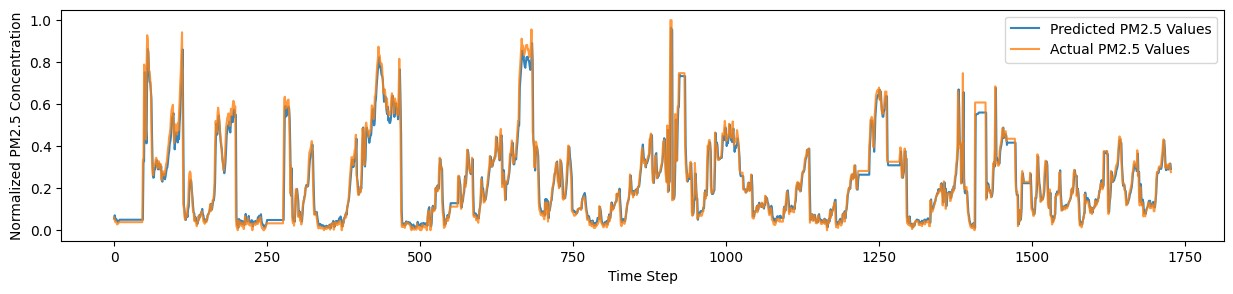
The trained models are used to make predictions on the test data (X\_test).

Predictions are stored in variables (y\_pred).

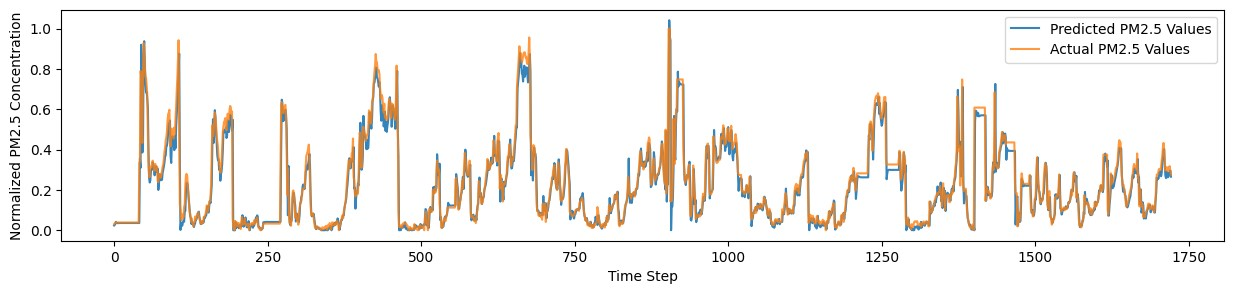
**Evaluation Results:**

The Mean Squared Error (MSE) and Mean Absolute Error (MAE) are calculated for each step size (1, 7, 14, 30, and 60) and printed. These metrics provide insights into the accuracy and performance of the models on the test data.

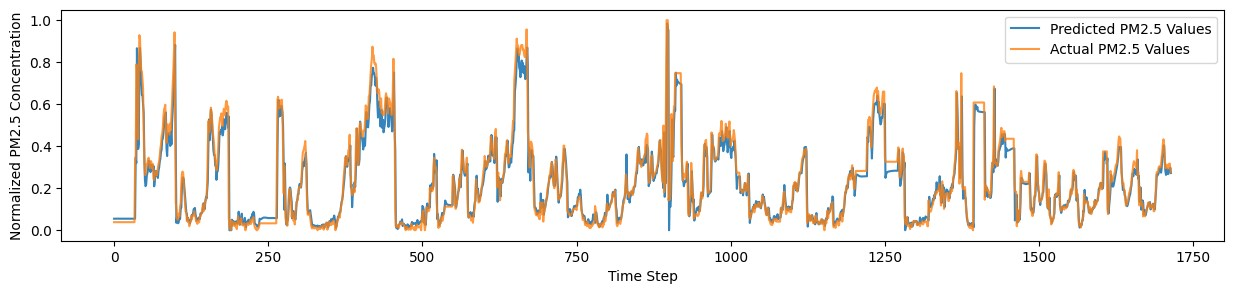
**City B - Step 1 - MSE: 0.004146558233821172, MAE: 0.03185439376070695**



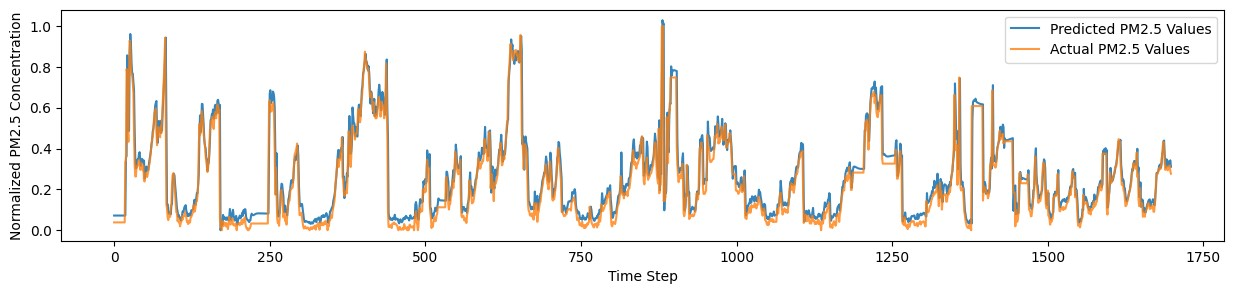
**City B - Step 7 - MSE: 0.004028300110587754, MAE: 0.03174049904128986**

****

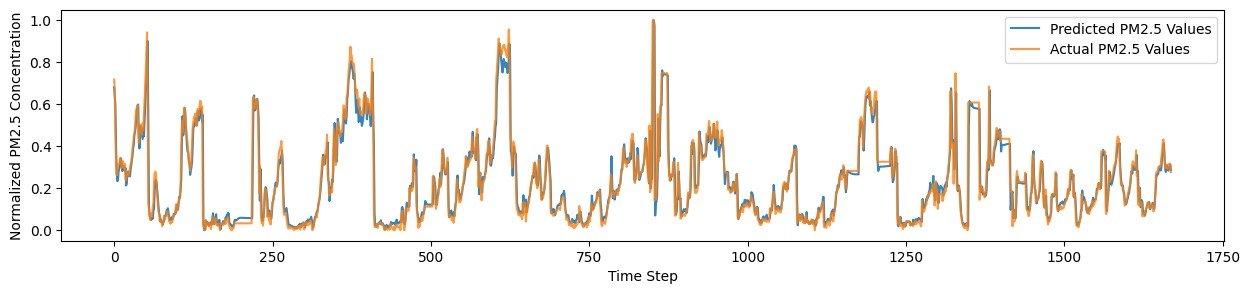
**City B - Step 14 - MSE: 0.004059170493576804, MAE: 0.03404374150533642**

****

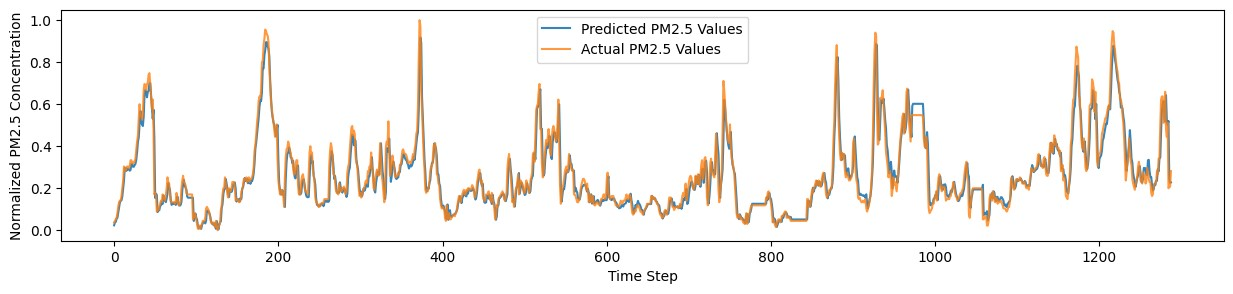
**City B - Step 30 - MSE: 0.004125953858820335, MAE: 0.03287698498170364**

****

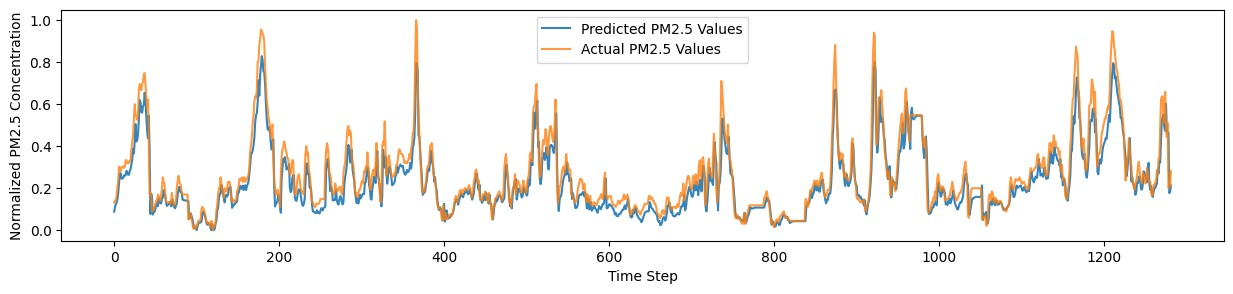
**City B - Step 60 - MSE: 0.00430872632575428, MAE: 0.03949034117335655**

****

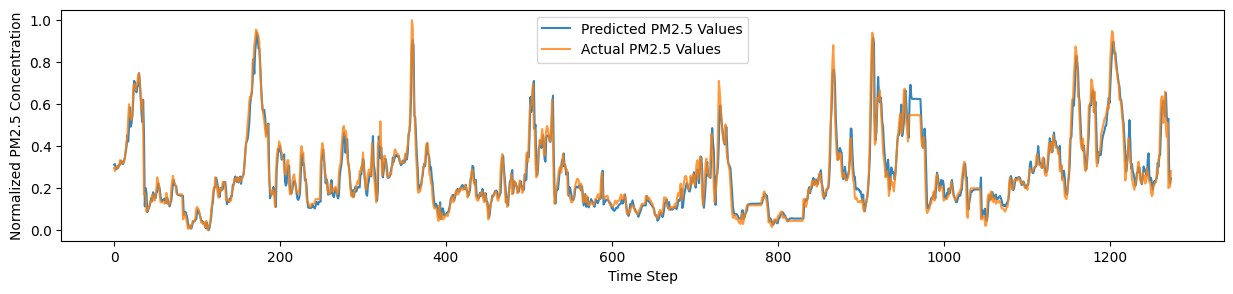
**City G - Step 1 - MSE: 0.002701813214881749, MAE: 0.03414587267344676**

****

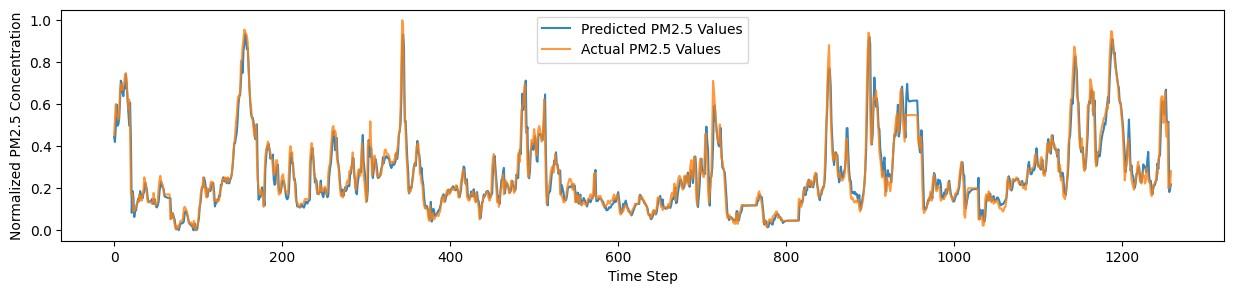
**City G - Step 7 - MSE: 0.002621322314776081, MAE: 0.03447750034085714**

****

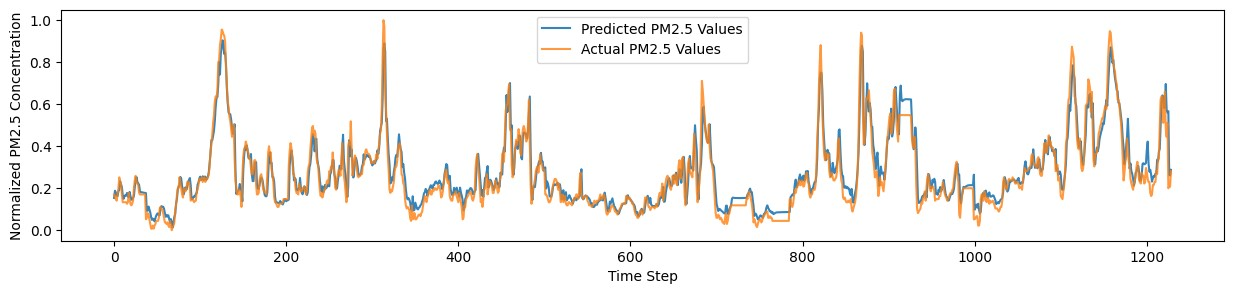
**City G - Step 14 - MSE: 0.0025500599016619638, MAE: 0.03397357240896848**

****

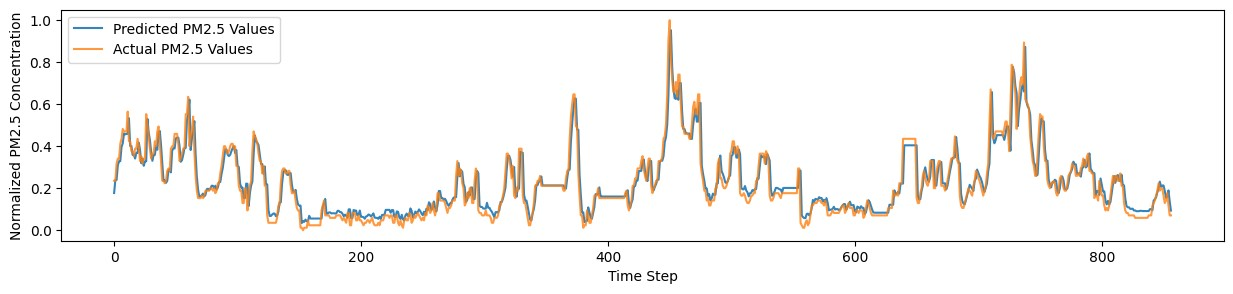
**City G - Step 30 - MSE: 0.0027578483780994305, MAE: 0.03581920134348722**

****

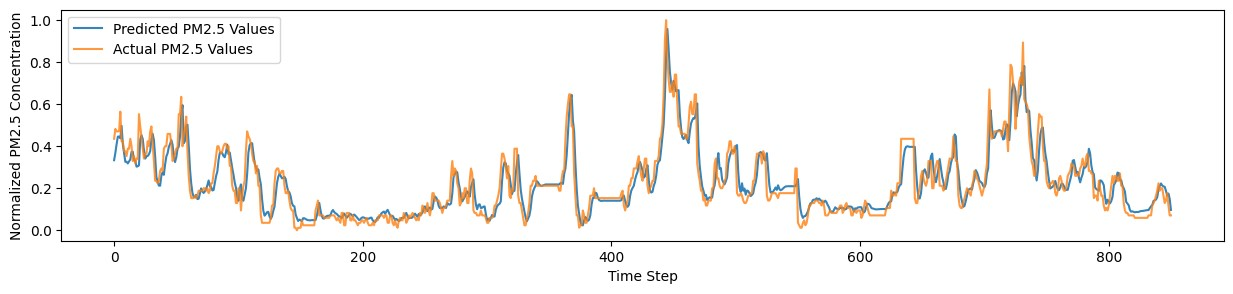
**City G - Step 60 - MSE: 0.0028050721542638504, MAE: 0.03658683089401267**



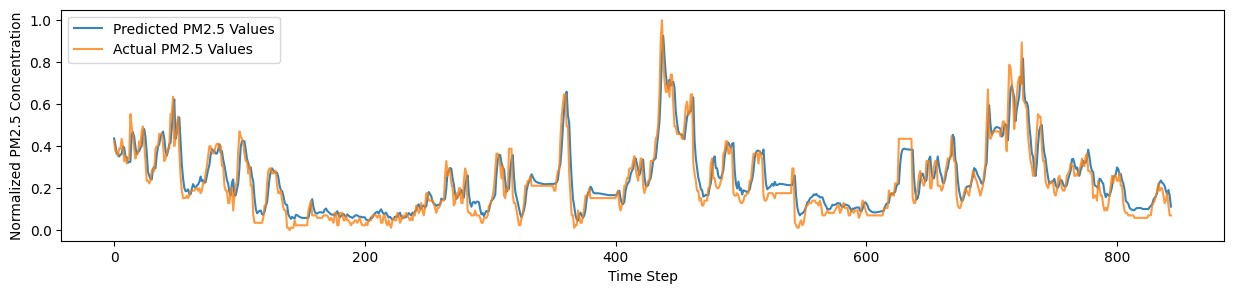
**City S - Step 1 - MSE: 0.0027792199804348755, MAE: 0.033369773609886015**

****

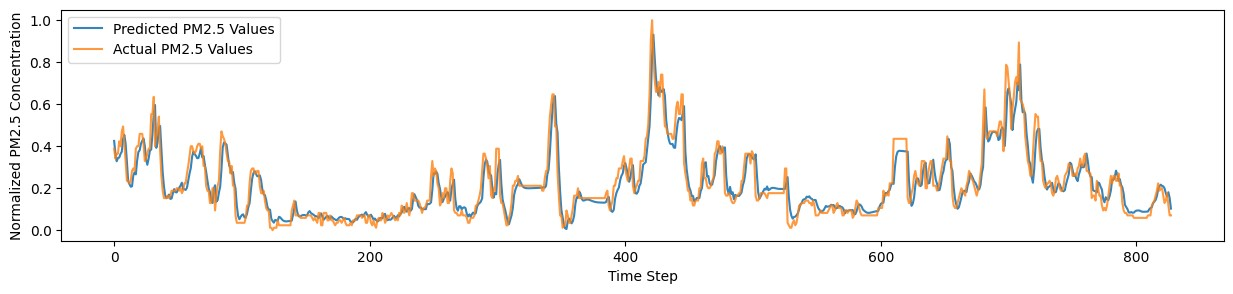
**City S - Step 7 - MSE: 0.0036876475235110085, MAE: 0.04269329242481921**

****

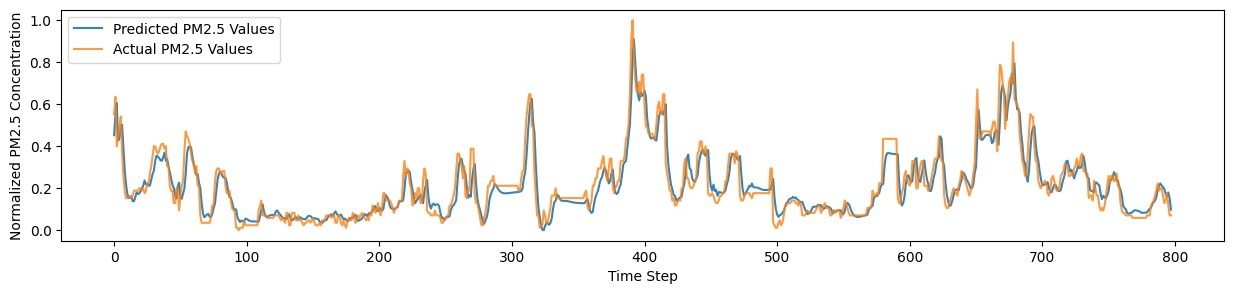
**City S - Step 14 - MSE: 0.003804500489080533, MAE: 0.043213266104689285**

****

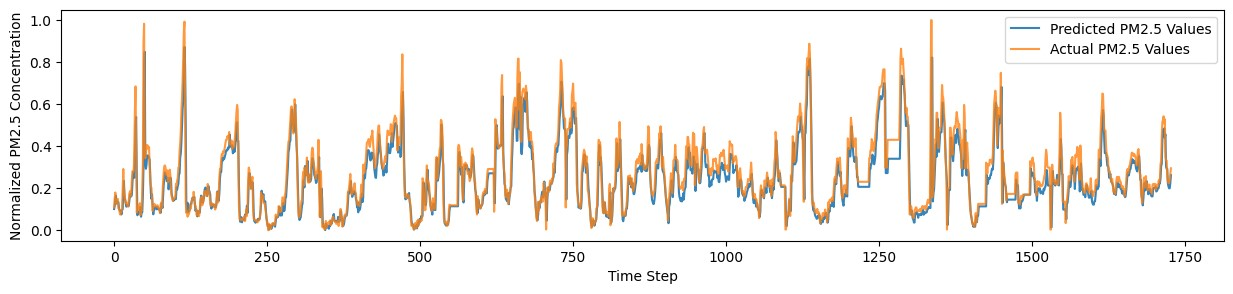
**City S - Step 30 - MSE: 0.004135327183613024, MAE: 0.045571450241941945**

****

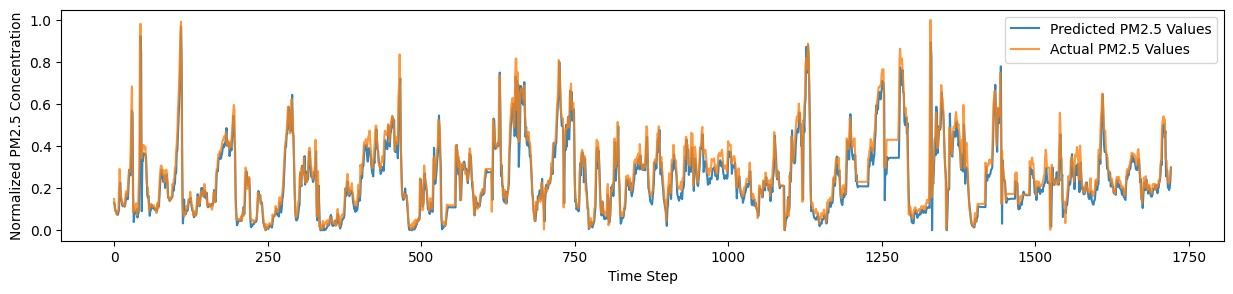
**City S - Step 60 - MSE: 0.003753906963325069, MAE: 0.04518638181559313**

****

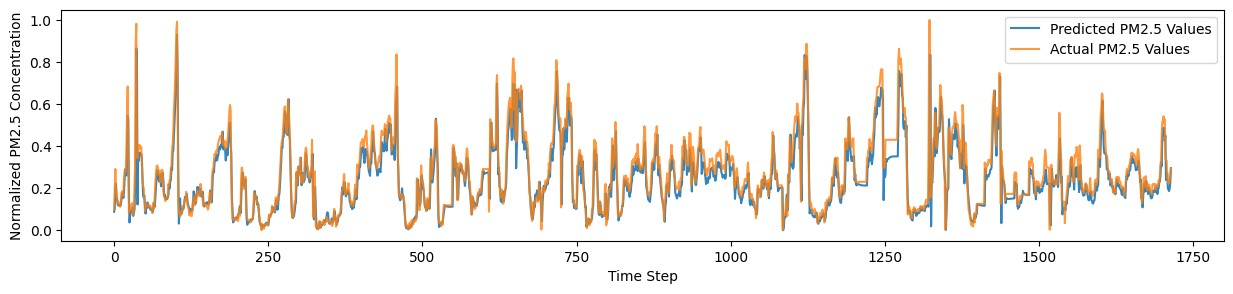
**City T - Step 1 - MSE: 0.004583750607658867, MAE: 0.0411623463094226**

****

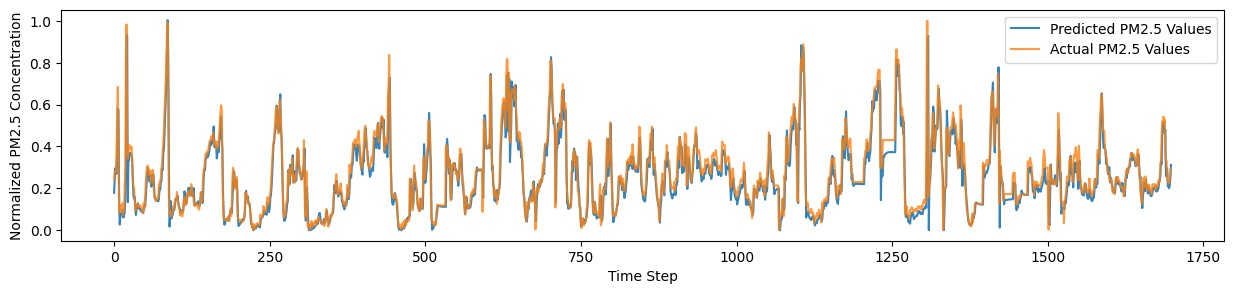
**City T - Step 7 - MSE: 0.004219069798457633, MAE: 0.039631912565864125**

****

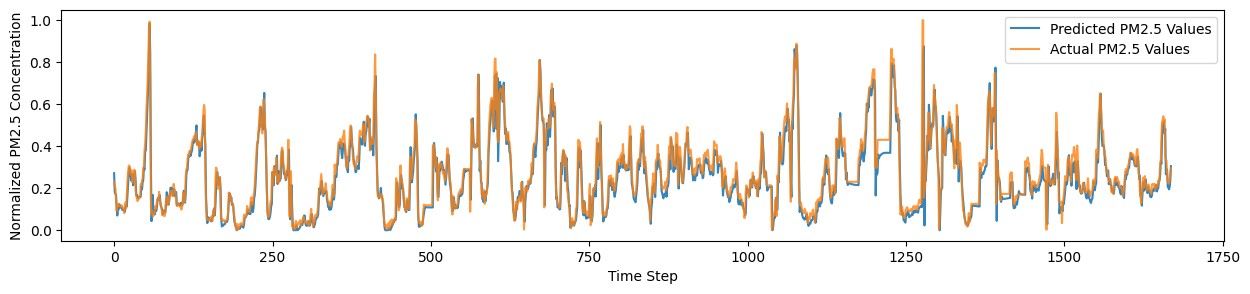
**City T - Step 14 - MSE: 0.0040898633090318505, MAE: 0.038605057491331404**

****

**City T - Step 30 - MSE: 0.10446582380530715, MAE: 0.27346095908859647**

****

**City T - Step 60 - MSE: 0.005960732085101698, MAE: 0.055978565759045244**

****

**Model Saving and Loading:**

**Model Saving**: Trained models are saved in the H5 format for future use or deployment.

**Model Loading:** The code demonstrates how to load the saved models back into memory using TensorFlow/Keras.

I have saved 20 models for each city B, G, S, and T with step sizes 1,7,14,30,60 separately.

**Conclusion:**

The successful application of LSTM networks for PM2.5 concentration prediction represents a valuable contribution to the field of air quality forecasting. The insights gained from this project pave the way for future endeavors aimed at refining forecasting models, considering the unique characteristics of each city. As urban environments continue to evolve, the ongoing optimization of predictive models remains essential for effective environmental management and public health planning.