Time-Series Forecasting using RNN

1. Provide brief details about the nature of your dataset. What is it about? What type of data are we encountering? How many entries and variables does the dataset comprise?

→ The Individual Household Electric Power Consumption dataset, sourced from the UCI Machine Learning Repository, serves as a comprehensive reservoir of data related to electric power consumption within a singular household. Spanning an extensive temporal horizon of nearly four years, this dataset meticulously captures minute-by-minute measurements, providing an intricate depiction of electric power dynamics and consumption patterns.

Comprising an array of metrics, the dataset encapsulates crucial parameters including voltage variations, global active power, and global reactive power, among others. These metrics, meticulously sampled at a rate of one minute, afford an unprecedented granularity, enabling nuanced insights into household electricity utilization over time.

2. Provide the details related to your RNN architecture.

→ Below is a brief overview of the RNN architecture used in our code:

1. **Architecture Overview:**

The RNN consists of three LSTM layers followed by fully connected layers.

2. **LSTM Layers:**

self.lstm1: First LSTM layer with input size input_size, hidden size hidden_size, and bidirectional=True.

self.lstm2: Second LSTM layer with input size hidden_size * 2 (twice the hidden size because it's bidirectional), and hidden size hidden_size, also bidirectional.

self.lstm3: Third LSTM layer with input size hidden_size * 2 and output size 32, bidirectional.

3. **Fully Connected Layers:**

self.fc1: Linear layer with input size 32 * 2 (twice the output size of the third LSTM layer) and output size 128.

self.fc2: Linear layer with input size 128 and output size 64.

self.fc3: Linear layer with input size 64 and output size 32.

self.fc4: Linear layer with input size 32 and output size 1.

4. **Activation and Dropout:**

ReLU activation function is used after each fully connected layer. Dropout with a probability of 0.25 is applied after each ReLU activation.

5. **Forward Pass:**

The forward pass involves passing the input tensor x through the three LSTM layers.

The output of the last time step is selected using out[:, -1, :].

Then, the output is passed through the fully connected layers with ReLU activation and dropout.

The final output is passed through a sigmoid function to produce a value between 0 and 1.

6. **Model Initialization:**

The model is initialized with input size inferred from the shape of the training data, hidden size of 128, and one layer of LSTM.

3. Discuss the results and provide relevant graphs:

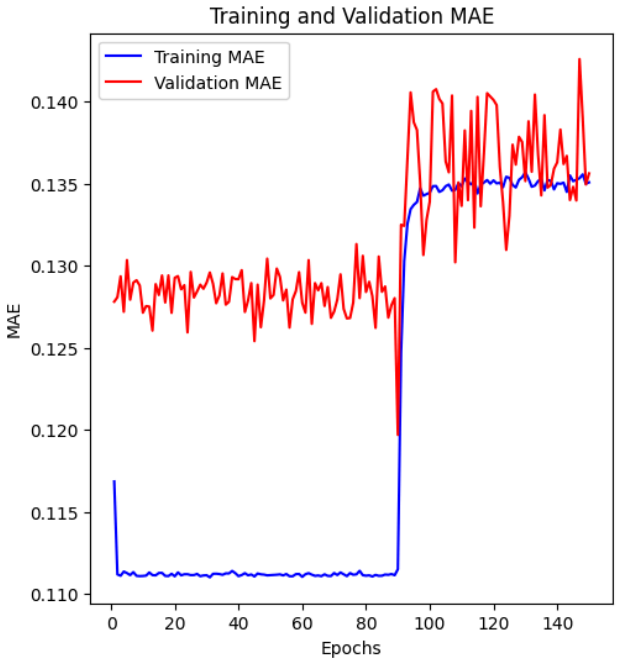
a. Report training accuracy, training loss, validation accuracy, validation loss, testing accuracy, and testing loss.

→ As our model solved regression problem, we have used Loss and Mean absolute error as our evaluation metrics, below were the results:

Training Loss: 0.0640,
Training MAE: 0.0640,
Val Loss: 0.1334,
Val MAE: 0.1334,
Test Loss: 0.0960,
Test MAE: 0.0960

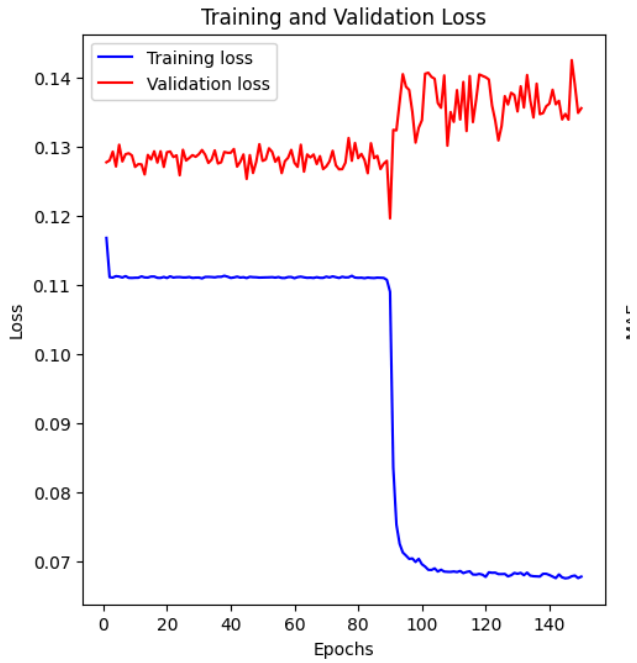
b. Plot the training and validation accuracy over time (epochs).

→ As our model solved regression problem, we have plotted Mean absolute error over time:



c. Plot the training and validation loss over time (epochs).

→



d. Generate a confusion matrix using the model's predictions on the test set.

→ A confusion matrix is typically used for classification problems, where the goal is to categorize data points into discrete classes or categories. In regression problems, the target variable is continuous, and the goal is to predict a numerical value rather than assigning instances to distinct classes. Therefore, confusion matrices are not applicable in regression scenarios.
As it was a regression problem, instead of a confusion matrix we evaluated our model using metrics such as Mean Absolute Error (MAE)

e. Report any other evaluation metrics used to analyze the model's performance on the test set.

→ We have used Mean absolute error as our evaluation metrics, below were the results:
Training MAE: 0.0640,
Val MAE: 0.1334,
Test MAE: 0.0960