### **Time-Series Forecasting Using CNN-RNN Hybrid Model**

#### **Submitted by: Pritam Mukherjee**

#### **1. Objective**

The objective of this task was to develop a hybrid CNN-RNN model to forecast future time-series values. The model aimed to leverage:

* **CNN (Convolutional Neural Network)** for feature extraction, capturing short-term dependencies.
* **RNN (Recurrent Neural Network)** for sequential modeling, capturing long-term temporal trends.

This approach was implemented on a publicly available dataset of energy consumption to predict future values based on historical data.

#### **2. Dataset Overview**

The dataset used contains hourly energy consumption data, representing real-world time-series challenges with:

* Periodic fluctuations (e.g., daily and weekly patterns).
* Random spikes and troughs (e.g., holiday consumption, equipment malfunctions).
* Dependencies on time-of-day, weekends, and external factors.

The data preprocessing steps were tailored to address these characteristics and ensure robust forecasting.

#### **3. Methodology**

##### **Data Preprocessing**

1. **Aggregation**:
   1. Multiple .csv files were merged, and TxnDate and TxnTime columns were combined into a single Datetime index.
2. **Feature Engineering**:
   1. **Lag Features**: Incorporated historical data for up to 7 prior hours.
   2. **Rolling Statistics**: Added rolling mean, standard deviation, and median to capture local trends.
   3. **Time-Based Features**: Extracted day-of-week, month, and hour.
   4. **Daypart Features**: Included a binary indicator for weekends.
   5. **Exponential Moving Average (EMA)**: Captured short-term trends using EMA with a 7-hour span.
   6. **Seasonal Decomposition**: Split the time-series data into trend, seasonal, and residual components using the seasonal\_decompose method.
3. **Normalization**:
   1. Data was normalized to bring all features into a uniform scale for model training.
4. **Train-Test Split**:
   1. The dataset was divided into training, validation, and test sets for fair model evaluation.

##### **Model Design**

The hybrid CNN-RNN model was built to effectively process and predict time-series data:

* **CNN Layers**:
  + Extracted short-term patterns using a 1D convolutional layer.
  + Followed by dropout for regularization and batch normalization for stable training.
* **RNN Layers**:
  + A single LSTM layer captured long-term dependencies in the data.
* **Output Layer**:
  + A Dense layer provided the final prediction.

##### **Model Summary**

|  |  |  |
| --- | --- | --- |
| Layer (type) | Output Shape | Param |
| conv1d\_1 (Conv1D) | (None, 32, 32) | 128 |
| dropout\_1 (Dropout) | (None, 32, 32) | 0 |
| batch\_normalization\_1  (Batch Normalization) | (None, 32, 32) | 128 |
| lstm\_1 (LSTM) | (None, 50) | 16600 |
| dense\_1 (Dense) | (None, 1) | 51 |

Total params: 16,907 (66.04 KB)  
 Trainable params: 16,843 (65.79 KB)  
 Non-trainable params: 64 (256.00 B)

##### **Model Training**

* The model was trained using the **Mean Squared Error (MSE)** loss function, with Adam optimizer.
* Hyperparameters such as window size, learning rate, and layer configurations were optimized through experimentation.

#### **4. Evaluation and Results**

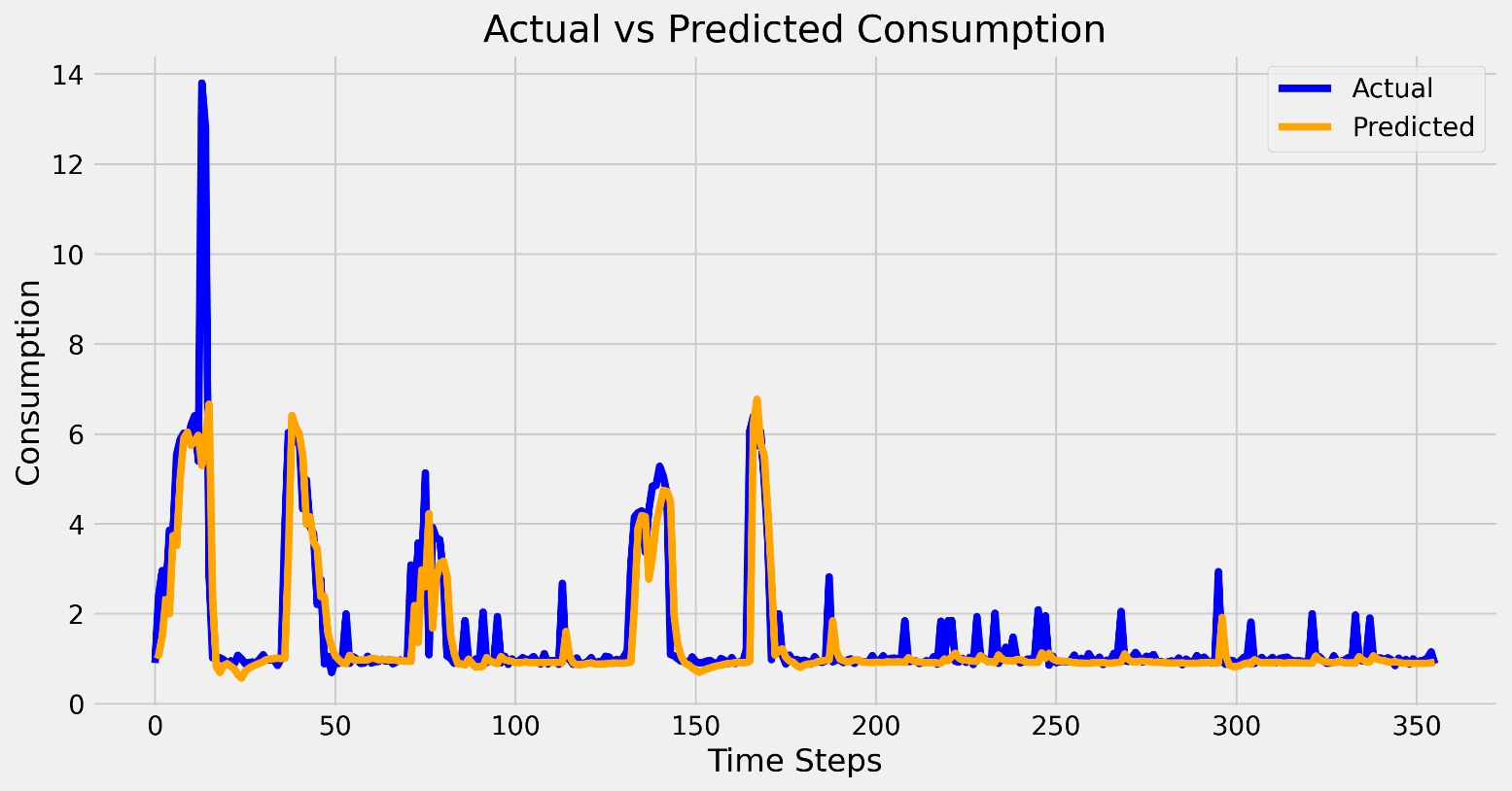
**Evaluation Metrics**:

The model's performance was evaluated using RMSE, MAE, and R² metrics:

* **Root Mean Square Error (RMSE)**:
  + Normalized Scale: **0.0333**
  + Original Scale: **0.9094** (significantly less compared to the standard deviation (**2.9612**).
* **Mean Absolute Error (MAE)**:
  + Normalized Scale: **0.0144**
  + Original Scale: **0.3937**
* **R-Squared (R²)**:
  + Achieved an R² score of **0. 6626**, demonstrating moderate correlation between predictions and actual values.

**Visualizations**:

* Here is a plot comparing predicted vs. actual consumption values highlighted the model's ability to capture both short-term and long-term patterns effectively.



#### **5. Challenges Faced**

1. **Preprocessing Complexities**:
   1. The data's dependence on external factors like time of day, day of week and weekends made preprocessing non-trivial.
   2. Techniques such as lag features, rolling statistics, and decomposition were necessary to provide meaningful inputs for the model.
2. **Model Selection**:
   1. Designing a hybrid CNN-RNN model required extensive experimentation due to limited available resources and prior implementations for this architecture.
3. **Capturing Spikes**:
   1. Sudden peaks and falls in the data posed challenges for earlier models but were mitigated by the hybrid model and feature engineering.