APPLIANCES ENERGY PREDICTION REPORT

Energy Data Time-Series Forecasting Assessment

Prepared on:

24th of November 2025

Prepared by: L. Mohamed Ilham

(lafeerilham123@gmail.com)

Table of Content

Appliances Energy Prediction Report

Date: November 23, 2025
Subject: Energy Data Time-Series Forecasting Assessment
By: L. Mohamed Ilham

1. Introduction

The efficient management of energy consumption in smart homes is a critical factor in reducing costs and environmental impact. The objective of this project is to develop a robust predictive model to forecast the energy consumption of household appliances.

Using a dataset collected from a low-energy building, we leverage historical sensor data including temperature, humidity, and weather conditions to predict the Appliances energy usage (in Wh). The approach progresses from statistical baselines to advanced Deep Learning architectures (LSTM, GRU, and CNN-LSTM) to capture complex temporal dependencies.

2. Data Insights (Exploratory Data Analysis)

The dataset consists of 10-minute interval readings spanning 4.5 months. Before modeling, an extensive Exploratory Data Analysis (EDA) was conducted to understand the underlying patterns.

2.1 Distribution Analysis

The target variable, Appliances, exhibited a **positive skew** (long right tail). Most readings cluster around low usage (standby mode), with intermittent spikes representing active appliance usage.

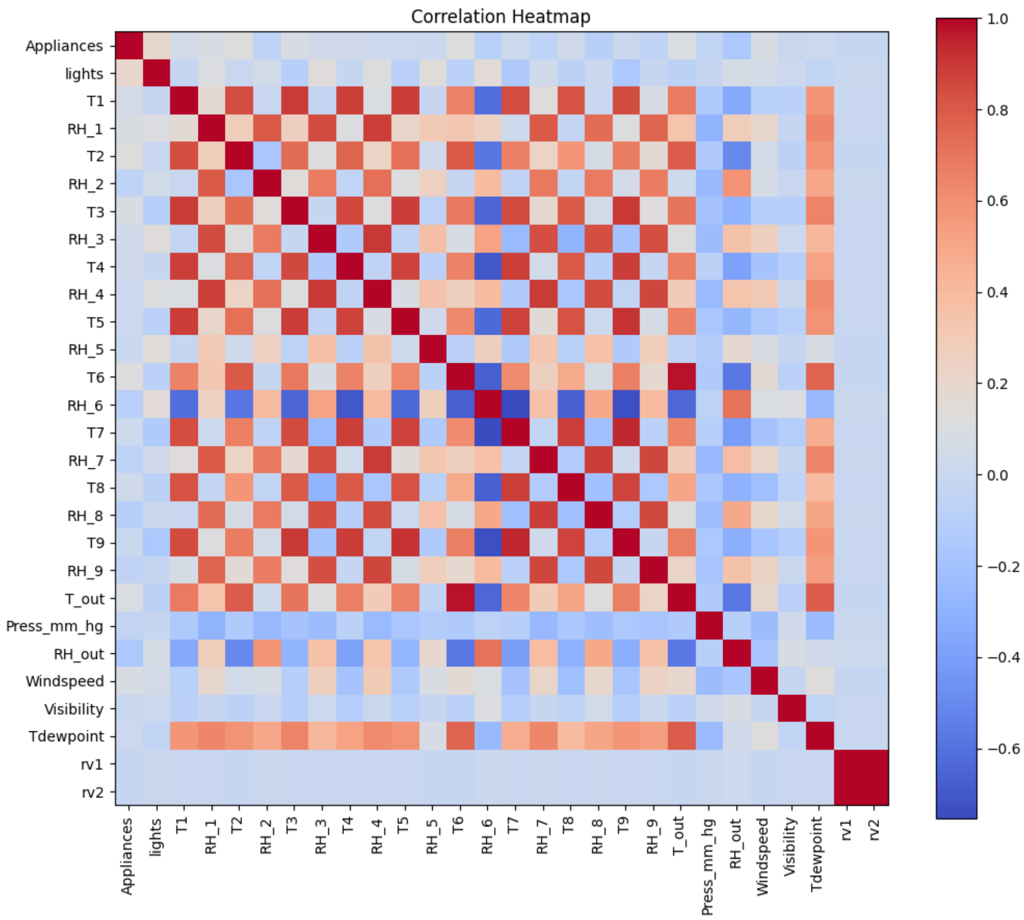
- **Observation:** Modeling raw skewed data often leads to poor convergence in neural networks.
- **Visual Evidence:** A histogram of the target variable showed a heavy concentration of values < 100 Wh.

2.2 Correlation Analysis

A correlation heatmap was generated to understand feature relationships.

- **Findings:** High multicollinearity was observed among indoor temperature sensors (e.g. T1 vs T2) and humidity sensors. However, lights showed a distinct but weak correlation with Appliances.

- **Visual:**



3. Preprocessing

To ensure data quality and model stability, the following preprocessing pipeline was implemented:

1. Log Transformation: To mitigate the impact of the skewed target distribution, I applied a logarithmic transformation:

$$y' = \log(1 + y)$$

   This successfully normalized the distribution, reducing the impact of outliers.

2. **Feature Scaling:** Deep Learning models are sensitive to the scale of input data. I used MinMaxScaler to normalize all features to the range [0, 1]. The scaler was saved (scaler.pkl) to ensure consistent transformation during inference.

3. **Time-Series Splitting:** Unlike standard random shuffling, time-series data requires preserving the temporal order.
   - **Training Set:** First 80% of the timeline.
   - **Test Set:** Last 20% of the timeline.

4. Feature Engineering

Raw sensor data often contains noise. I generated derived features to help the models capture context and trends.

1. **Temporal Features:** Extracted hour, day_of_week, and month. Added boolean flags for is_weekend and is_night to capture human behavioral cycles.
2. **Lag Features:** To capture autocorrelation, we introduced lags at t-1, t-6, t-12, and t-24 (representing 10 mins, 1 hour, 2 hours, and 4 hours prior).
3. **Rolling Statistics:** Calculated rolling_mean and rolling_std over 1-hour and 3-hour windows to smooth out short-term volatility.
4. **Domain Interactions:** Created interaction terms, such as T_RH_interaction (Temperature \times Humidity) and lights_x_lag1, to capture combined environmental effects.

Feature Selection: Using a Random Forest regressor, we identified the top important features, which included Appliances_log, lights, and specific temperature sensors (T6, T3).

5. Model Design

We designed and evaluated four distinct architectures. The Deep Learning models utilized a sliding window approach with a look-back period of **24 time steps** (4 hours).

5.1 Baselines

- **Linear Regression:** A simple baseline to establish minimum performance expectations.
- **Random Forest Regressor:** An ensemble method tuned using RandomizedSearchCV to capture non-linear relationships without temporal architectural components.

5.2 Deep Learning Architectures

1. **LSTM (Long Short-Term Memory):**
   - *Architecture:* Two stacked LSTM layers (64 & 32 units) - Dense Layer.
   - *Purpose:* Designed to learn long-term dependencies and handle the vanishing gradient problem.
2. **GRU (Gated Recurrent Unit):**
   - *Architecture:* Two stacked GRU layers (64 & 32 units) - Dense Layer.
   - *Purpose:* A more computationally efficient variant of LSTM.
3. **CNN-LSTM (Hybrid):**
   - *Architecture:* Conv1D (64 filters) \rightarrow MaxPooling \rightarrow LSTM (64 units) - Dense.
   - *Purpose:* The CNN layer acts as a feature extractor/filter to reduce noise, while the LSTM layer interprets the temporal dynamics of these features.

6. Model Optimization

To prevent overfitting and ensure convergence, the following optimization strategies were applied to the Deep Learning models:

- **Optimizer:** Adam (lr=0.001).
- **Loss Function:** Mean Squared Error (MSE).
- **Early Stopping:** Monitored validation loss with a patience of 8 epochs to stop training once generalization stopped improving.
- **Learning Rate Reduction:** ReduceLROnPlateau reduced the learning rate by a factor of 0.5 if validation loss stagnated for 4 epochs.
- **Regularization:** Dropout layers (0.2) were inserted between recurrent layers.

7. Results and Evaluation

The models were evaluated on the unseen Test Set using Mean Absolute Error (MAE), Root Mean Square Error (RMSE), and the Coefficient of Determination (R^2).

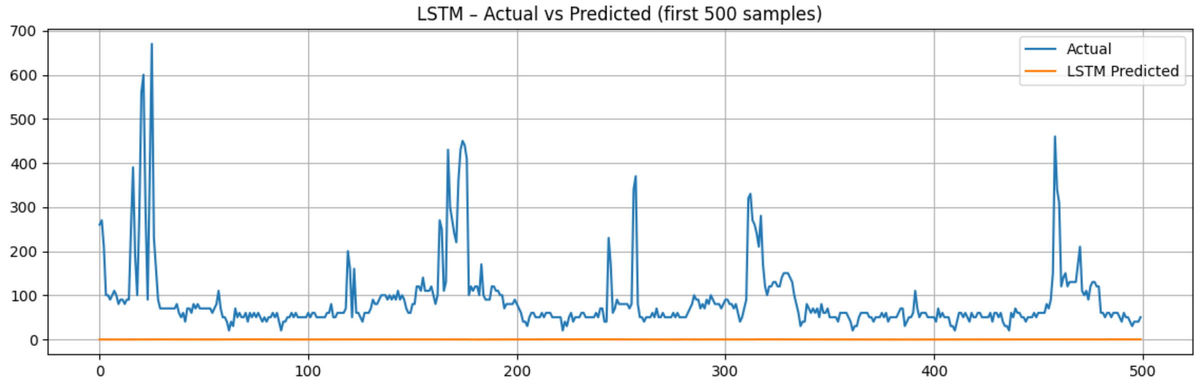
Performance Comparison Table

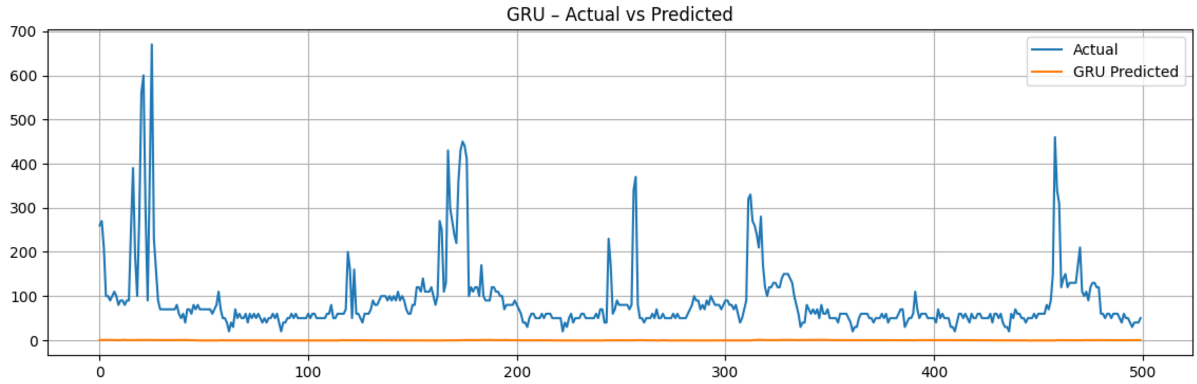
Model	MAE	RMSE	R2 Score
CNN-LSTM	**30.12**	**65.07**	**0.45**
Random Forest	60.15	116.73	-0.76
LSTM	95.12	129.45	-1.17
GRU	94.81	129.19	-1.16

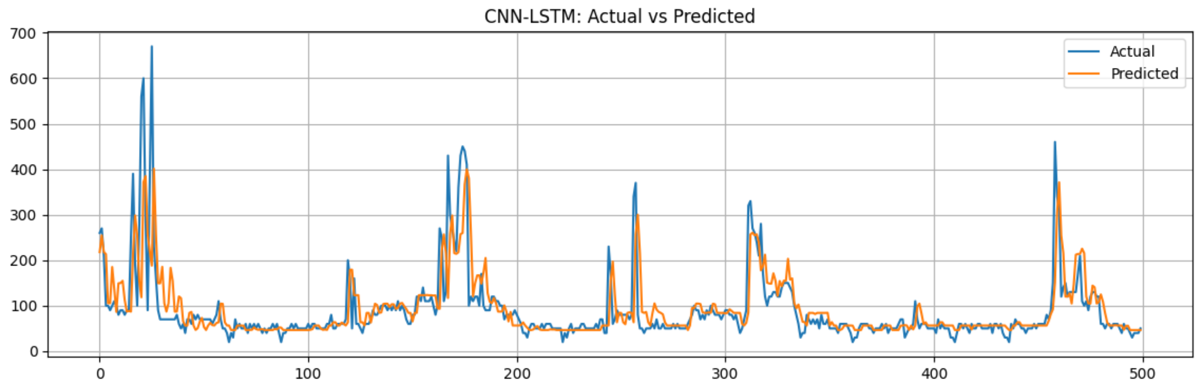
Analysis of Results

- **Winner:** The **CNN-LSTM** hybrid model was the clear winner, achieving the lowest error and the only positive R^2 score. This demonstrates that pre-filtering the sensor noise via CNN layers is crucial for this specific dataset.
- **Issues with Pure RNNs:** The standard LSTM and GRU models performed poorly (negative R^2), likely failing to distinguish signal from noise in the volatile energy spikes.
- **Random Forest:** While it outperformed pure RNNs, it lacked the ability to effectively model the sequential transitions compared to the hybrid approach.

Visualization: Actual vs. Predicted







- The **CNN-LSTM** plot showed the predicted line following the general trend of the actual data, capturing the periodic rises in energy usage much better than the flat-line predictions of the standard LSTM.

8. Challenges and Solutions

Challenge	Solution
Data Skewness	The target variable was heavily skewed. We applied np.log1p transformation to normalize the distribution, which significantly assisted the neural networks in learning loss gradients.
Overfitting	Deep models tend to memorize training noise. We implemented **Dropout** layers and **Early Stopping** to prevent this.
High Volatility	Appliance usage is erratic (spiky). Standard RNNs struggled. The **Hybrid CNN-LSTM** architecture solved this by using 1D Convolutions to extract local patterns before temporal processing.

9. Conclusion

This assessment highlighted that while energy consumption data is time-dependent, it is also noisy and volatile. Traditional time-series models (LSTM/GRU) may struggle without assistance.

The **Hybrid CNN-LSTM** model proved to be the most effective solution, leveraging Convolutional layers to smooth features and LSTM layers to remember history. Future work recommends increasing the window size beyond 4 hours and exploring Transformer-based architectures (e.g., Temporal Fusion Transformers) to further improve accuracy on energy spikes.

10. References

Dataset: *Appliances Energy Prediction Data Set*.