Comprehensive Technical Documentation

Deep Learning for Energy Production and Consumption Forecasting

Implementing LSTM Neural Networks for Stable Time-Series Analytics

December 10, 2025

System Architecture: Python-Based LSTM Workflow

1. Introduction

The global transition to sustainable energy requires not only the generation of green power but also the intelligent management of its consumption. Predicting when energy will be produced by solar panels and when it will be consumed by buildings is critical for grid stability and billing efficiency. Traditional statistical methods often fail to capture the long-term dependencies and non-linear patterns inherent in energy data. This project introduces a deep learning solution based on Long Short-Term Memory (LSTM) networks to address these challenges.

The system developed here is a full-stack data science pipeline. It covers everything from raw data cleaning and physical constraint enforcement to the training of specialized neural models and the deployment of a future-forecasting engine.

2. Problem Statement

Energy data is notoriously noisy and unpredictable. Buildings like Civic Centers or Sports Areas exhibit 'spiky' consumption patterns where usage might jump from 0.5 kWh to 15.0 kWh within a single hour due to an event. Similarly, solar production follows a strict daily cycle but is highly dependent on weather noise.

Key challenges addressed in this project includes:

- High Variance: Large spikes causing standard models to predict a 'flat line' average.
- Physical Deviations: Preventing models from predicting solar production at midnight.
- Multi-Series Correlation: Capturing the relationship between production and consumption.
- Cold Start Problems: Providing models enough context with 90 or 30 days of history.

3. Deep Learning: The LSTM Architecture

Traditional Neural Networks treat every input as independent. However, energy usage at 10:00 AM is heavily influenced by the usage at 09:00 AM and 08:00 AM. This 'sequence memory' is where Recurrent Neural Networks (RNNs) enter the picture.

The Long Short-Term Memory (LSTM) is a special type of RNN designed to overcome the 'Vanishing Gradient' problem. It does this through a series of internal 'gates' that control the flow of information:

- The Forget Gate: Decides what information from previous hours is no longer relevant.
- The Input Gate: Decides which new energy patterns from the current hour should be stored in the cell memory.
- The Cell State: Acts as a 'conveyor belt' that carries long-term trends throughout the entire time-series.
- The Output Gate: Filters the internal memory to provide the final kWh prediction.

4. Technology Architecture

The project is built on the following professional technology stack:

- TensorFlow/Keras: The primary brain of the system, handling the neural network computation.
- Pandas: Used for high-speed time-series manipulation and data cleaning.
- Scikit-Learn: Provides the MinMaxScaler, ensuring all data is normalized between 0 and 1 for neural stability.
- Holidays (Spain): Crucial for feature engineering, allowing the model to recognize school holidays.
- Joblib: Used for persisting scalers, ensuring that future predictions use the exact same math as the training phase.

5. Data Processing Workflow

Clean data is more important than a complex model. Our preprocessing pipeline follows a strict 'Physical Realism' protocol:

- ### Balancing and Interpolation

Missing hours are identified and filled using linear interpolation. This ensures a continuous stream of 24 data points per day, which is a mathematical requirement for the LSTM's fixed input window.

- ### Night Production Logic

A hard constraint is applied to solar data: any production values recorded between 21:00 and 06:00 are forced to 0.00. This prevents the model from hallucinating solar energy during non-daylight hours.

6. Advanced Feature Engineering

Raw timestamps are useless for a neural network. We engineer 'Smart Features' to provide temporal context.

- ### Cyclical Encoding (Sin/Cos)

Hour 23 and Hour 0 are numerically far apart (23 vs 0), yet they are chronologically neighbors. We solve this by projecting the hour onto a circle using Sine and Cosine transformations. This allows the model to perceive time as a continuous loop.

- ### Lagged Features

We provide the model with 'Lags'—explicit values from t-1 (previous hour) and t-24 (same hour yesterday). This gives the LSTM a shortcut to recognize daily rhythms instantly.

7. Model Architecture and Design

The system uses a 'Stacked LSTM' architecture, meaning multiple layers of memory are layered on top of each other.

- Layer 1 (128 Units): Captures high-frequency fluctuations and immediate transitions.
- Dropout (20%): Randomly disables 20% of neurons during training to prevent the model from 'memorizing' data (overfitting).
- Layer 2 (64 Units): Compresses information into long-term seasonal trends.
- Dense Output: A fully connected layer that outputs the final forecast for the next 720 hours (30 days).

8. Loss Functions and Optimization

Standard models use Mean Squared Error (MSE). However, for buildings with huge energy spikes (Civic Centers), MSE can be too sensitive to outliers. We implement Huber Loss.

Huber Loss acts like MSE for small errors but like Mean Absolute Error (MAE) for large spikes. This prevents the weight updates from becoming unstable when the building hosts a large energy-intensive event.

9. Training Implementation Detailed Analysis

1. General_Model_Train.py

This is the dual-column engine. It predicts Production and Consumption simultaneously. It uses a larger input window of 3 months to understand the correlation between the two. If production is high (sunny day), it helps contextually understand why consumption might be lower (less heating/lighting needed).

2. SingleCol_Model_Train.py

Specialized for buildings with only consumption data. We found that a 30-day input window is often better for these buildings as it allows the model to focus on the 'most recent' habits without being distracted by patterns from 3 months ago.

10. The Forecasting Engine (Prediction)

Deployment involves taking the 'best weights' from training and feeding them the most recent real-world data.

The prediction scripts (General_Predict.py and SingleCol_Predict.py) perform several crucial safety tasks:

- Scaling Alignment: Ensures the new data is scaled using the EXACT same Mean/Std as the training data.

- 3D Tensor Reshaping: Formats the 1D CSV data into the 3D block [1, 720, Features] expected by the LSTM.
- Non-Negative Enforcement: Using a ReLU-style clamp to ensure predicted energy can never be negative.

11. Synthetic Data Validation (SyntheticDataGen.py)

Testing a model requires data. We created a synthetic generator class to simulate building behavior. This is scientifically important for validating that the model has actually learned the laws of physics.

- The generator simulates:
- Solar Cycles: Sinusoidal production based on time of day (noon peak).
- Stochastic Noise: Random weather fluctuations.
- Consumption Spikes: Random heavy load events to test model robustness.

12. User Operational Manual

Follow these steps to operate the LSTM Energy System successfully:

1. Setup Environment

Ensure Python 3.10+ is installed. Activate the virtual environment:

```
.\venv\Scripts\activate
```

and install requirements.

2. Prepare Data

Place your raw building data inside `Data_Cleaning/Raw_data`. Run the cleaning scripts to generate the `_Cleaned.csv` files.

3. Execute Training

Run

```
python General_Model_Train.py
```

for buildings with solar, or

```
python SingleCol_Model_Train.py
```

for consumption-only sites.

4. Generate Forecast

Run the prediction scripts. Check the `Model_output/` folder for your CSV forecasts and trend plots.

13. Conclusion and Future Roadmap

This project successfully bridges the gap between raw unstructured energy data and actionable future insights. The use of LSTMs ensures that long-term transitions in building usage are captured, whiles the refined feature engineering provides the stability needed for reliable billing and grid planning.

Future Roadmap:

- Implementation on raspberry pi
- Connect each raspberry to the main hub