Universiteit van Amsterdam

Group H

Digital Twin for Solar Power Plants

Luca Giandomenico

Jovana Vukmirovic

Gabriel Gil Pichelbauer

Yunxuan Tang

13 October 2025

Project Overview: From Data to Digital Twin

Goal: Developing a Digital Twin for solar power plants using real-world data. Aiming to simulate, monitor, and predict energy production with enhanced visualization and AI support.

The process is divided into five main tasks:

1. Dataset & preprocessing
2. Representation model
3. Historical replay
4. Machine learning-based forecasting
5. Visualization & system integration

Dataset & Preprocessing

Dataset

Time-stamped sensor and power generation data from two solar power plants in India

- Collected over a 34-day period, from 15-05-2020 00:00 until 17-06-2020 23:45
- Two pairs of power generation data/weather sensor readings, each pair corresponding to a solar power plant
- Power generation data gathered at the inverter level (22 per plant), where each inverter has multiple solar panels attached to it, while the weather sensor data is gathered at the plant level
- Time interval between readings is 15 minutes

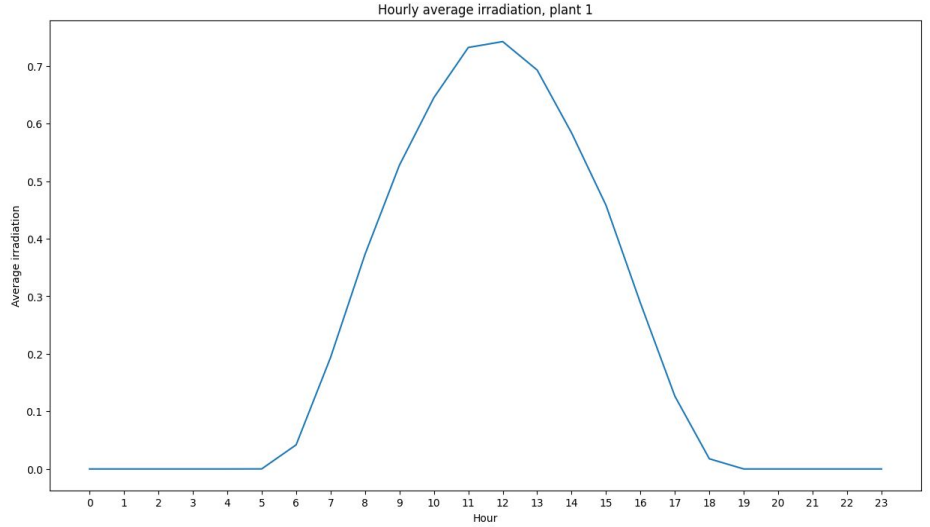
Dataset & Preprocessing

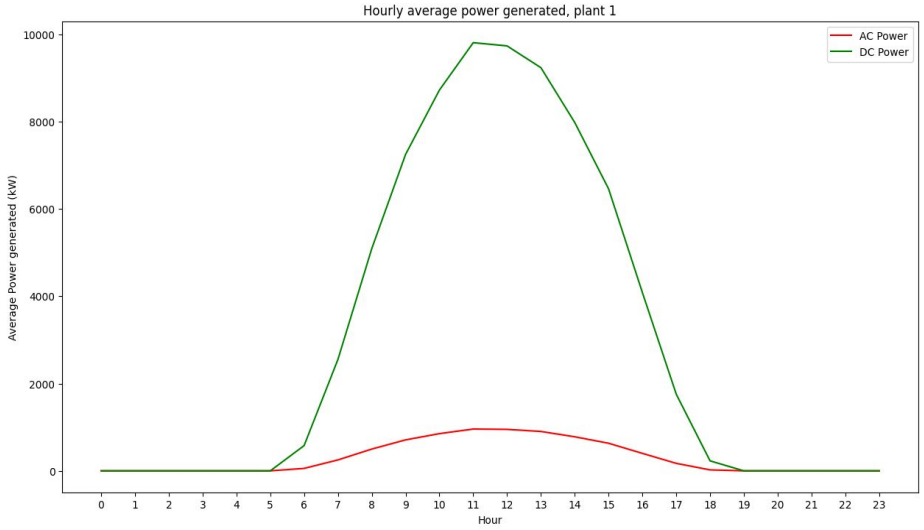
Preprocessing

- Corrected DATE_TIME strings to be the same across files
- Redundant columns removed (e.g. SOURCE_KEY in weather files)
- Power generation data merged with weather sensor data
- Complex inverter IDs replaced with integers ('1BY6WEcLGh8j5v7' → 1)
- Converted DATE_TIME fields into datetime objects
- Stored final data frames in private InfluxDB3 database

Brief analysis

- Irradiation, AC/DC power generated and temperature all peak during the same hours (6:00-18:00) and days





Representation Model

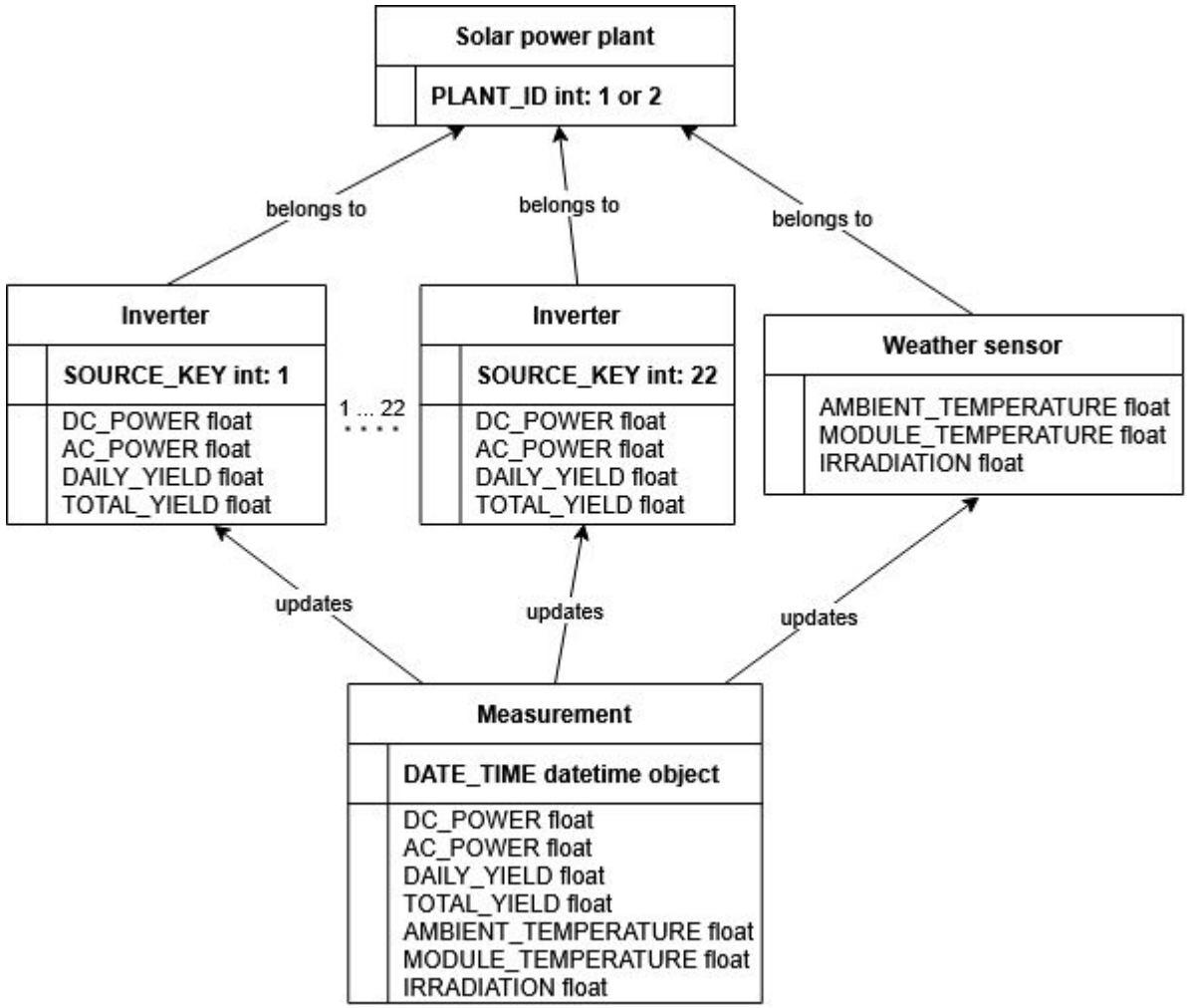
State

Conceptually, the state of the system at any given time can be seen as a concatenation of the current data from all inverters and sensors of both plants

$$S(t) = \{s_i(t) | i = 1, 2, 3, ..., 44\}^*$$

For example, s_1 could be represented as:

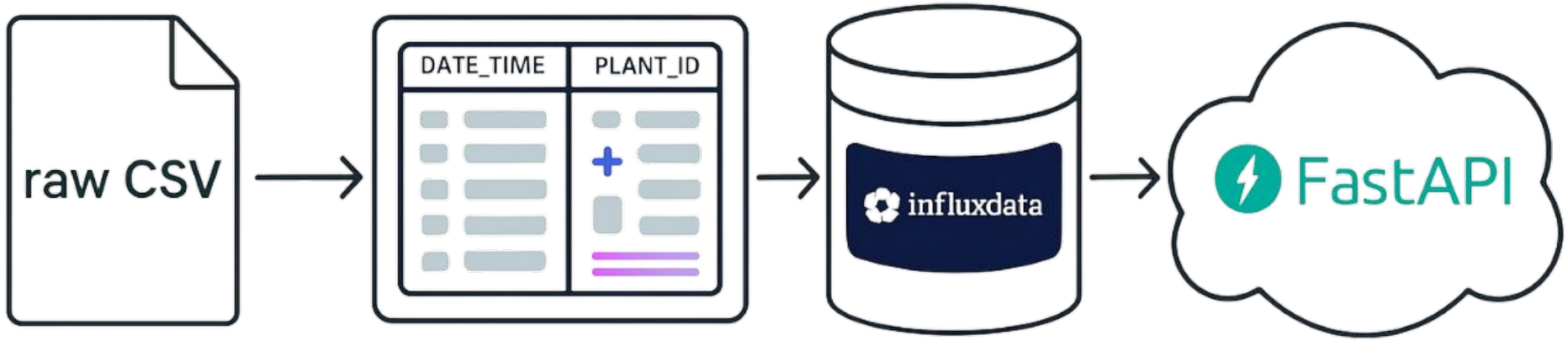

```
{
  "PLANT_ID": 1,
  "SOURCE_KEY": 1,
  "DC_POWER": 0.0,
  "AC_POWER": 0.0,
  "DAILY_YIELD": 0.0,
  "TOTAL_YIELD": 6259559.0,
  "AMBIENT_TEMPERATURE": 25.184316133333333,
  "MODULE_TEMPERATURE": 22.8575074,
  "IRRADIATION": 0.0
}
```

* Visualization filters this and only displays 1 inverter, then $S(t) = s_i(t)$

System Architecture

- Python-based backend using FastAPI
- Modular design connecting data, ML, and visualization
- Supports replay and real-time system updates



Backend Design & Functionality

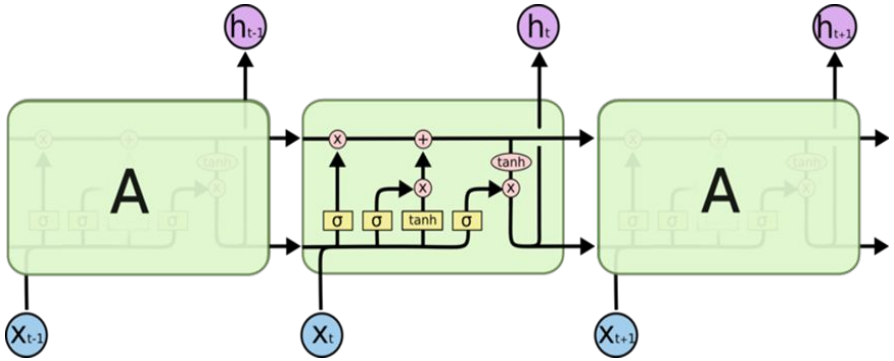
- FastAPI modularized using API Router
- Live data querying through **InfluxDB**
- SQL-like time filtering (WHERE time >= ...)
- Robust error handling (HTTP status + JSON Response)

```
@router.get("/replay")
def replay(plant: int, timestamp: str):
    if client is None:
        return JSONResponse(status_code=503, content={"error":
        "Database connection unavailable"})
    ts = pd.to_datetime(timestamp)
    query = f"""
        SELECT * FROM plant{plant}
        WHERE time = '{ts.isoformat()}Z'
        ORDER BY time
    """
    result = client.query(query=query, language="sql", mode="pandas")
    if result.empty:
        return {"message": "No data found for that timestamp."}
    return result.to_dict(orient="records")
```

Why LSTMs and how we handled time

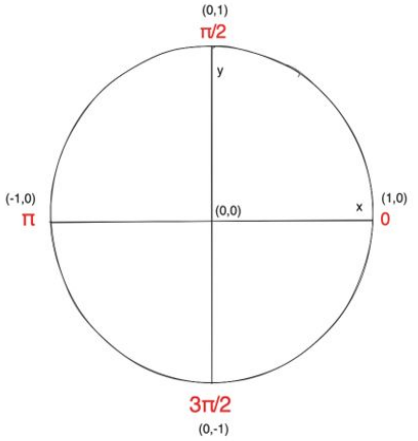
Why LSTM Network?

- Solar power generation is a time-series problem
- We chose an **LSTM (Long Short-Term Memory)** network because it's specifically designed to learn and remember long-term patterns in sequential data



Smart Time Representation

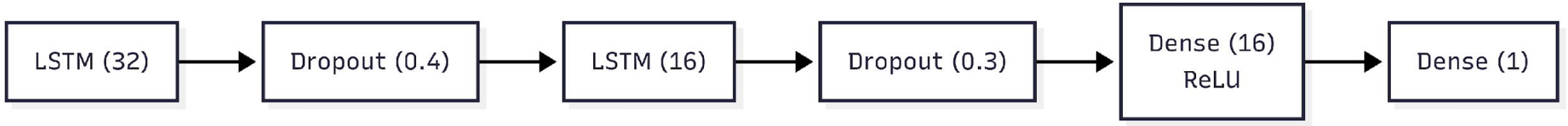
- We employed cyclical time encoding to help model understand cyclical time
- This maps time onto a circle, preserving the natural closeness of cyclical values
- Learning daily and seasonal patterns



Model Architecture and Implementation

Prediction Strategy & Data Handling

- **Inverter-Level prediction strategy:** Each inverter is treated independently to learn its unique characteristics
- **Per-inverter data processing:** Sequences were generated for each inverter separately to preserve unique patterns and prevent data leakage.
- **Sequence Generation:** Input sequence of 24 consecutive measurements to predict the 25th timestamp



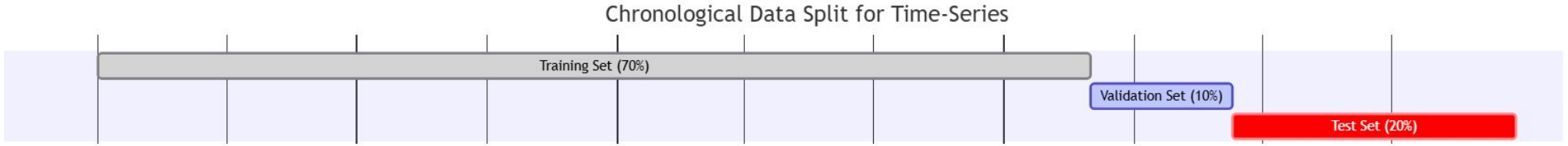
Training Configuration:

- **Optimizer: Adam** with a learning rate of 0.001, which is ideal for noisy, non-stationary solar data.
- **Batch Size:** A size of **32** was used to balance stable training with computational efficiency.

Ensure Robust & Reliable Model

Data Handling & Training

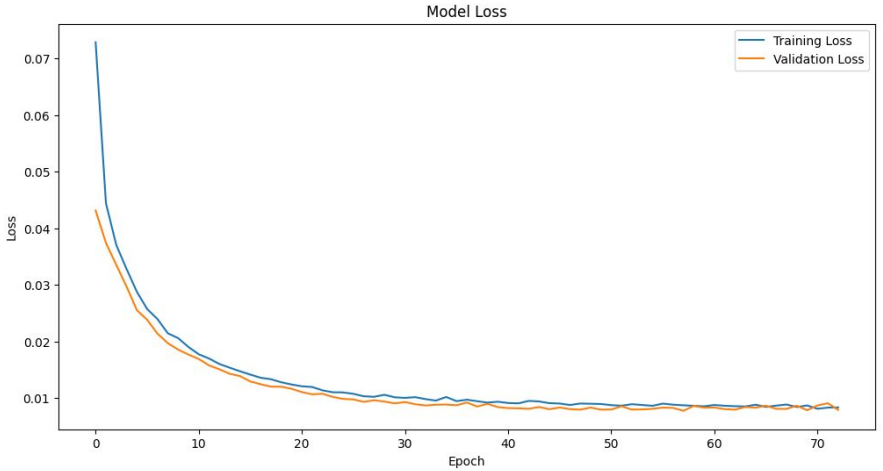
- **Scaling**: All features scaled to a [0,1] range to ensure stable and fast training.
- **Data Split**: We used a strict chronological split (70% train, 10% validation, 20% test) to prevent data leakage
- **Overfitting prevention**: Early stopping halts the training if the model stops improving



How we measured success?

We used standard regression metrics to evaluate our forecast accuracy:

- **RMSE (Root Mean Squared Error)**: To see the error in the same units as power output.
- **R^2 (R-Squared):** o understand how much of the power variation our model can explain



Outputs of the Digital Twin

Predictive outputs

- **Predicted AC Power Output**: Real-time and forecasted generation based on irradiance, temperature, and system conditions
- **Performance Ratio & Efficiency**: Daily/weekly KPIs showing how effectively the plant converts sunlight to power
- **Energy Yield Forecasts**: Short-term (hourly/daily) production estimates
- **Degradation Tracking**: Identification of efficiency loss in panels over time

Diagnostic and monitoring outputs

Used for real-time insights and fault detection:

- **Anomaly Detection**: Identify underperforming strings/inverters/panels automatically
- **Thermal or Electrical Fault Alerts**: Based on digital twin vs. sensor data deviation
- **Weather & Soiling Impact Analysis**: Quantify how much losses are due to shading, dust, or temperature

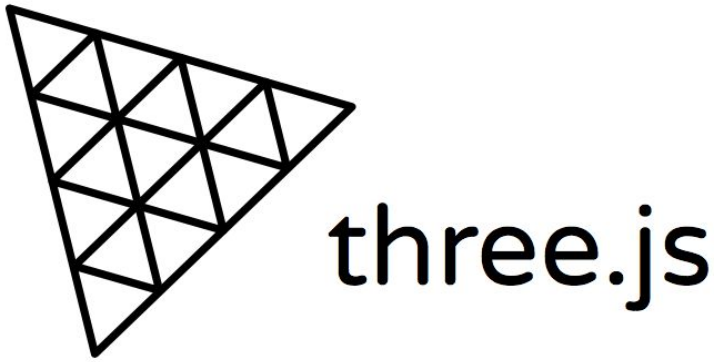
Strategic & Planning Outputs

Longer-term or business-level uses:

- **Financial Forecasting**: Revenue, ROI, and payback time predictions based on performance data
- **Carbon Savings Estimation**: CO_2 emissions avoided through clean generation
- **Scenario Analysis**: “What-if” simulations (e.g., adding panels, changing orientation, upgrading components)

Visualization Environment

- Web environment made using three.js
- Connects to the backend through http
- Chosen for compatibility with other frameworks as well as device compatibility



Replay System

- Replays historically data with a scaled playback speed.
- Includes interactive time controls.
- Real-Time data Visualization on the panels.

Prediction System

- AI powered forecasting of power generation.
- Dual panel comparison of actual vs predicted data.
- Smart time navigation.

Q&A

- Thanks~
- Any Questions?