Università di Perugia
Dipartimento di Matematica e Informatica

Corso di Laurea in Informatica

A Deep Learning approach for Time Series Imputation on Photovoltaic data

Relatori
Valentina Poggioni
Enrico Bellocchio
Alessandro Devo

Laureando
Nicolò Vescera

Anno Accademico 2022-2023

Contents

Abstract

The growing need for the adoption of tools capable of generating clean energy from renewable and sustainable sources has led to extensive generation and collection of energy production data, especially from photovoltaic panels installed worldwide. However, these data often have gaps and deficiencies due to various factors such as temporary failures, adverse weather conditions, or malfunctions of sensors and data collection instruments. Accurate imputation of these gaps is crucial to ensure the reliability of analyses and predictions based on this data. This thesis aims to address the problem of imputing time series data from photovoltaic panels using advanced deep learning techniques. In particular, a deep learning model based on Fully Connected Neural Networks (FCNN) and Recurrent Neural Networks (RNN) are proposed to capture the complex temporal relationships between the total energy produced (target feature) and various components of the system. The models were trained on a dataset consisting of real data from a photovoltaic system with a power capacity of approximately $1MW$.

Chapter 1

Background

1.1 Problem Definition

The following thesis aims to address the problem of imputing time series data from photovoltaic systems. Specifically, it often happens that data acquisition instruments in a system temporarily fail, causing a period of time, more or less extended, where the curve of the total energy produced is missing. To try to fill this “gap” a simple formula is not sufficient, as various factors such as solar irradiance, ambient temperature, presence of clouds, rainfall, etc., need to be taken into account.

Formally, we can define the problem as follows:

Definition 1.1.1. Given:

- A set of time series data representing a photovoltaic system, $S = S_1, S_2, ..., S_N$, where N represents the number of available time series;
- A target time series $t \in S$ that represent the Total Generated Energy;

Where each time series S_i is composed of ordered pairs (t_i, v_i), where t_i is a timestamp representing the moment when the value v_i was recorded.
The objective of the imputation problem is to estimate the missing or damaged values in the time series t.

1.2 Photovoltaic Implant

Solar photovoltaic (PV) energy systems are made up of different components. Each component has a specific role. The type of component in the system depends on the type of system and the purpose. For example, a simple PV-direct system is composed of a solar module or array (two or more modules wired together) and the load (energy-using device) it powers. A solar energy system produces direct current (DC). This is electricity which travels in one direction. The loads in a simple PV system also operate on direct current (DC). A stand-alone system with energy storage (a battery) will have more components than a PV- direct system.

1.2.1 Solar Module

The majority of solar modules available on the market and used for residential and commercial solar systems are silicon- crystalline. These modules consist of multiple strings of solar cells, wired in series (positive to negative), and are mounted in an aluminum frame. The size or area of the cell determines the amount of amperage. The larger the cell, the higher the amperage.

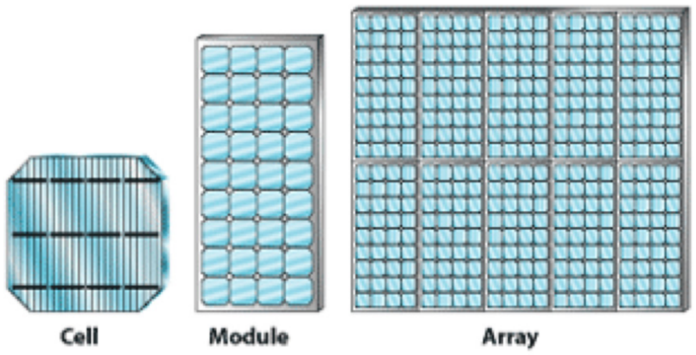


Figure 1.1: The solar cell is the basic component. Cells wired together and mounted in a frame compose a solar module. Several modules wired together form an array.

1.2.2 Solar Array

The solar array is made up of multiple PV modules wired together. Connecting the negative wire of one module to the positive wire of a second module is the beginning of a series string. Wiring modules in series results in the voltage of each of the two modules is added together. A series string represents the summed voltages of each individual module. The negative cable of one module is connected to the positive cable of the next module. In a large system, multiple strings are assembled and the non-connected ends are connected to homerun leads which are landed at the terminals of an enclosure located near the array. The goal is to wire modules in series to build voltage.

1.2.3 Junction Box

A PV system array with multiple strings of modules will have a positive lead and a negative lead on the end of each string. The positive leads will be connected to individual fuses and the negative leads will be connected to a negative busbar in an enclosure. This is called the source circuit. The junction box serves to "combine" multiple series strings into one parallel circuit. For example, an array with three strings of 10 modules wired in series would produce 300 volts (10 modules x 30 volts) per string and 4 amps per string. When the leads are landed in the combiner box, the circuit would produce 300 volts at 12 amps (3 strings x 4 amps/string). Once the circuits are combined, leaving the box it is referred to as the output circuit.

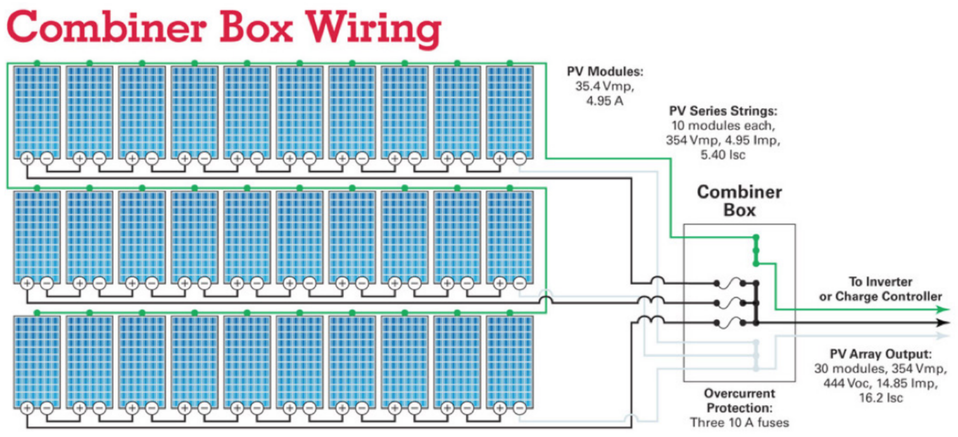

Combiner Box Wiring
PV Modules: 35.4 Vmp, 4.95 A
PV Series Strings: 10 modules each, 354 Vmp, 4.95 Imp, 5.40 Isc
Combiner Box
To Inverter or Charge Controller
PV Array Output: 30 modules, 354 Vmp, 444 Voc, 14.85 Imp, 16.2 Isc
Overcurrent Protection: Three 10 A fuses

Figure 1.2: This figure represent an output circuit made of 3 string, each one hosts 10 solar modules.

1.2.4 Inverter

Energy from an array or a battery bank is direct current (DC). This will provide for DC loads such a lights, fans, pumps, motors, and some specialty equipment. However, if the energy is to be used to power loads that operate on alternating current (AC), as what is found in a residence, the current needs to be converted. The inverter changes DC energy to AC energy. Inverters are available in many different sizes for various-sized loads. A string inverter is used to convert DC power from a solar array to AC power and can be connected to an AC distribution power panel (service panel) in a residence or facility.

1.2.5 System Metering

Several tools are available to help the solar user to monitor their system. On stand-alone or off-grid PV systems, the battery meter is used to measure the energy coming in and going out of the battery bank. Charging and discharging of batteries, and proper functioning of the charging system is important to alert the user to incomplete charging, battery decline, or possible system shutdown. System monitoring with web-based tools and apps allow the solar user to see system activity using a cell phone or tablet from a location away from their system.

1.3 Open-Meteo

Open-Meteo is an open-source weather data platform and community-driven weather forecasting project. Open-Meteo aims to provide access to weather data and forecasts that are openly accessible to the public and can be used for a variety of applications, including research, development, and personal use. It offers a diverse range of APIs that go beyond traditional weather forecasting such as past weather data, ocean data, air quality, ensemble forecasts, climate forecasts based on IPCC predictions, and even floods. Open-Meteo offers over 80 years of hourly weather data, covering any location on earth, all at a 10 kilometer resolution. This extensive dataset is very useful to delve into the past and analyze historical weather patterns.

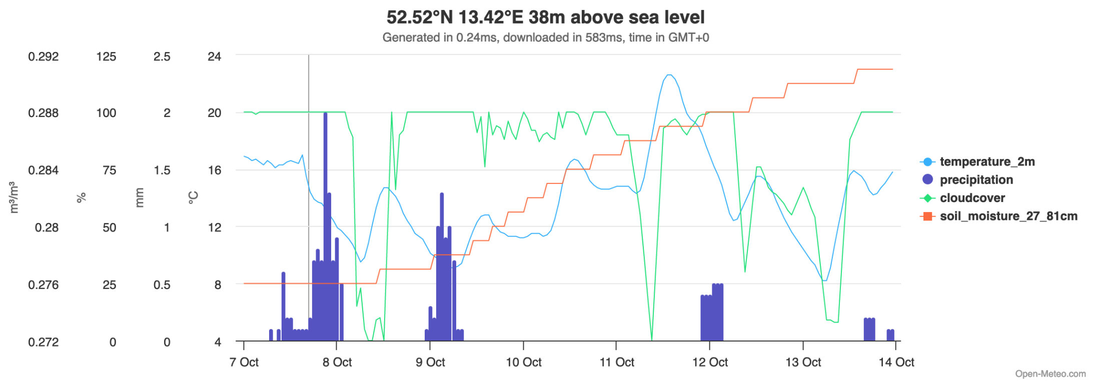
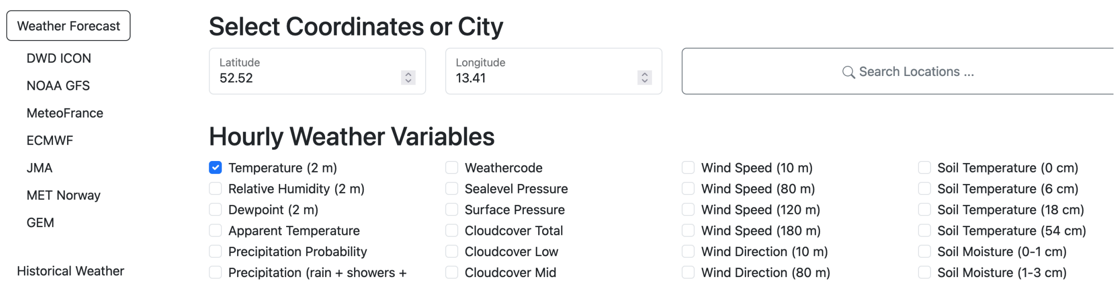


Figure 1.3: A plot about some Open-Meteo data.


Weather Forecast
DWD ICON
NOAA GFS
MeteoFrance
ECMWF
JMA
MET Norway
GEM

Historical Weather

Select Coordinates or City

Latitude: 52.52
Longitude: 13.41
Search Locations ...

Hourly Weather Variables

[✓ Temperature (2 m)]	Weathercode	Wind Speed (10 m)	Soil Temperature (0 cm)
Relative Humidity (2 m)	Sealevel Pressure	Wind Speed (80 m)	Soil Temperature (6 cm)
Dewpoint (2 m)	Surface Pressure	Wind Speed (120 m)	Soil Temperature (18 cm)
Apparent Temperature	Cloudcover Total	Wind Speed (180 m)	Soil Temperature (54 cm)
Precipitation Probability	Cloudcover Low	Wind Direction (10 m)	Soil Moisture (0-1 cm)
Precipitation (rain + showers +	Cloudcover Mid	Wind Direction (80 m)	Soil Moisture (1-3 cm)

Figure 1.4: Open-Meteo forecast features.

1.4 Multi Layer Perceptron

The Multi Layer Perceptron is the most known and most frequently used type of neural network. On most occasions, the signals are transmitted within the network in one direction: from input to output. There is no loop, the output of each neuron does not affect the neuron itself. This architecture is called feed- forward. Layers which are not directly connected to the environment are called hidden. There are also feed-back networks, which can transmit impulses in both directions, due to reaction connections in the network. These types of networks are very powerful and can be extremely complicated. Introduction of several layers was determined by the need to increase the complexity of decision regions. A perceptron with a single layer and one input generates decision regions under the form of semi planes. By adding another layer, each neuron acts as a standard perceptron for the outputs of the neurons in the anterior layer, thus the output of the network can estimate convex decision regions, resulting from the intersection of the semi planes generated by the neurons. The power of the multilayer perceptron comes precisely from non-linear activation functions. Almost any non-linear function can be used for this purpose, except for polynomial functions. Currently, the functions most commonly used today are the single-pole (or logistic) sigmoid.

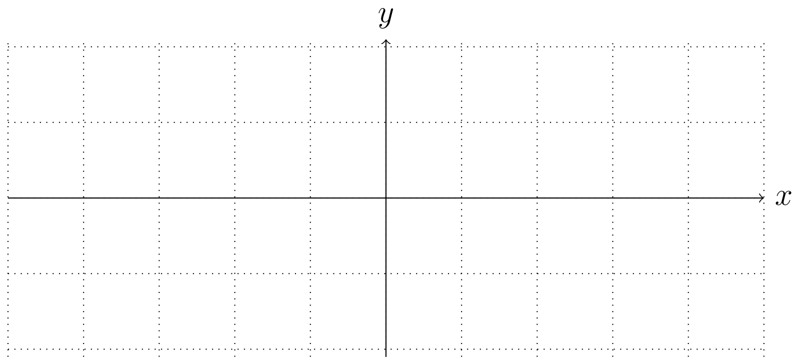


Figure 1.5: Sigmoid function $\sigma(x) = \frac{1}{1+e^{-x}}$

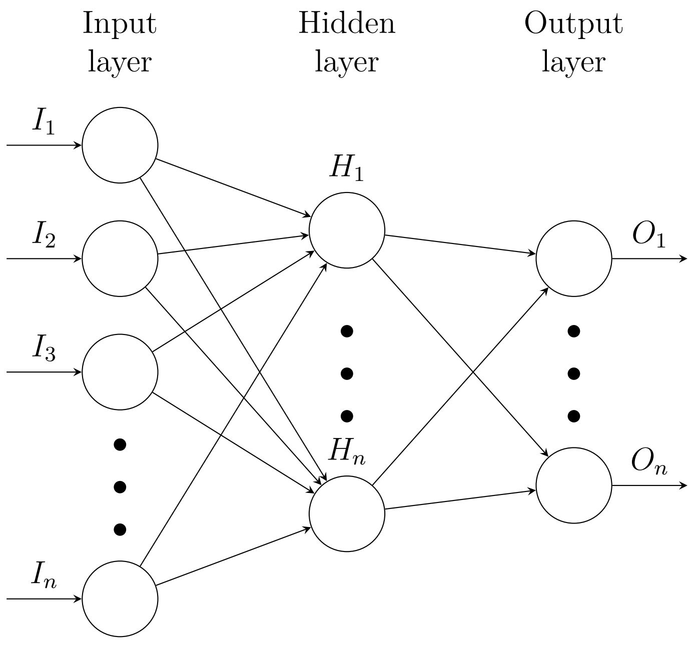


Figure 1.6: Multi Layer Perceprton architecture.

1.5 Recurrent Neural Network

Recurrent Neural Networks (RNNs) are a type of neural network architecture which is mainly used to detect patterns in a sequence of data. Such data can be handwriting, genomes, text or numerical time series which are often produced in industry settings (e.g. stock markets or sensors). However, they are also applicable to images if these get respectively decomposed into a series of patches and treated as a sequence. On a higher level, RNNs find applications in Language Modelling and Generating Text, Speech Recognition, Generating Image Descriptions or Video Tagging. What differentiates Recurrent Neural Networks from Feedforward Neural Networks also known as Multi-Layer Perceptrons (MLPs) is how information gets passed through the network. While Feedforward Networks pass information through the network without cycles, the RNN has cycles and transmits information back into itself. This enables them to extend the functionality of Feedforward Networks to also take into account previous inputs $X_{0:t-1}$ and not only the current input X_t. This difference is visualised on a high level in Figure 1.7. Note, that here the option of having multiple hidden layers is aggregated to one Hidden Layer block H. This block can obviously be extended to multiple hidden layers.

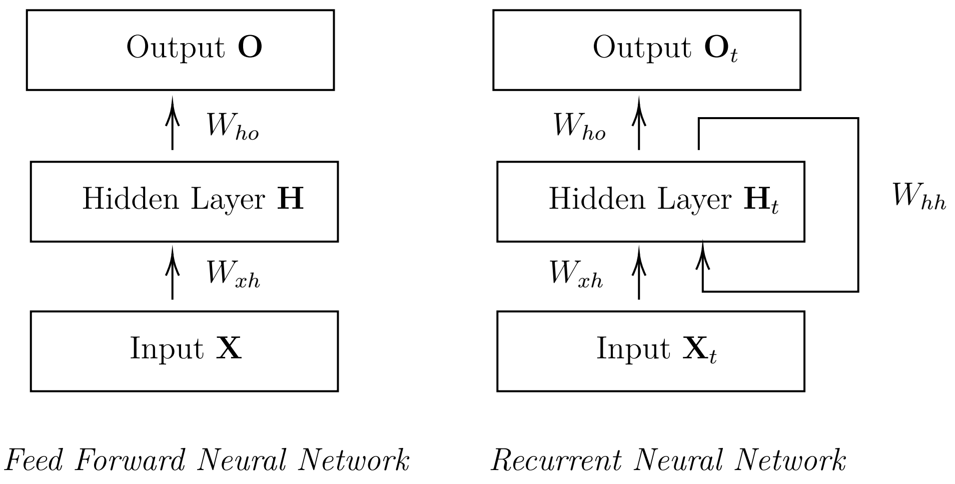


Figure 1.7: Visualization of difference between MLPs and RNN.

1.5.1 Gated Recurrent Unit

Gated Recurrent Units (GRU) are an advanced variation of RRNs (Recurrent Neural Network). GRUs use update gate and reset get for solving a standard RNNs vanishing gradient issue. These are essentially 2 vectors that decide the type of information to be passed towards the output. What makes these vectors special is that programmers can train them to store information, especially from long ago. They do all of this by utilizing its gated units which help solve vanishing/exploding gradient problems often found in traditional recurrent neural networks. These gates are helpful for controlling any information to be maintained or discarded for each step. It is also worth keeping in mind that gated recurrent units make use of reset and update gates.

- **Update Gates Function**: The main function of the update gate is to determine the ideal amount of earlier info that is important for the future.

One of the main reasons why this function is so important is that the model can copy every single past detail to eliminate fading gradient issue.

- **Reset Gate's Function**: A major reason why reset gate is vital because it determines how much information should be ignored. It would be fair to compare reset gate to LSTMs forget gate because it tends to classify unrelated data, followed by getting the model to ignore and proceed without it.

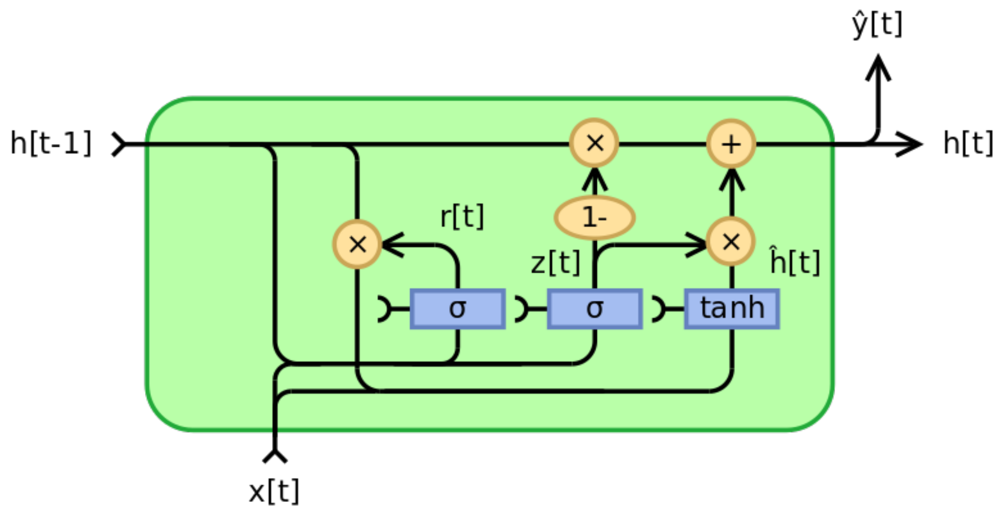


Figure 1.8: Gated Recurrent Unit (GRU) architecture.

1.6 Transformer

Recurrent neural networks, long short-term memory and gated recurrent neural networks in particular, have been firmly established as state of the art approaches in sequence modeling and transduction problems such as language modeling and machine translation. Recurrent models typically factor computation along the symbol positions of the input and output sequences. Aligning the positions to steps in computation time, they generate a sequence of hidden states h_t, as a function of the previous hidden state h_{t-1} and the input for position t. This inherently sequential nature precludes parallelization within training examples, which becomes critical at longer sequence lengths, as memory constraints limit batching across examples. Attention mechanisms have become an integral part of compelling sequence modeling and transduction models in various tasks, allowing modeling of dependencies without regard to their distance in the input or output sequences. The Transformer allows for significantly more parallelization.

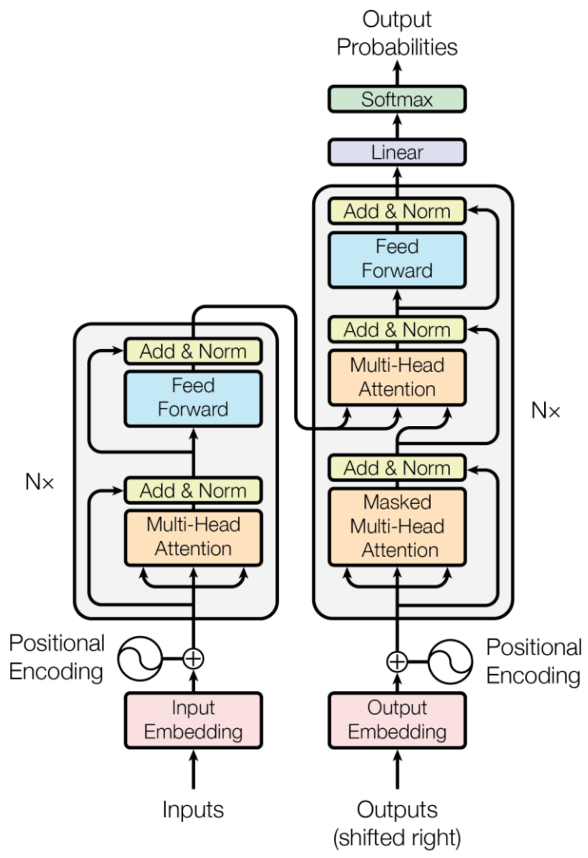


Figure 1.9: This figure shows the Transformers model architecture.

Most competitive neural sequence transduction models have an encoder-decoder structure. Here, the encoder maps an input sequence of symbol representations $(x_1, \ldots, x_n)$ to a sequence of continuous representations $z = (z_1, \ldots, z_n)$. Given z, the decoder then generates an output sequence $(y_1, \ldots, y_m)$ of symbols one element at a time. At each step the model is auto-regressive, consuming the previously generated symbols as additional input when generating the next. The Transformer follows this overall architecture using stacked self-attention and point-wise, fully connected layers for both the encoder and decoder, shown in the left and right halves of Figure 1.9, respectively.

- *Encoder*: The encoder is composed of a stack of $N = 6$ identical layers. Each layer has two sub-layers. The first is a multi-head self-attention mechanism, and the second is a simple, position- wise fully connected feed-forward network. Residual connection are employed around each of the two sub-layers, followed by layer normalization. That is, the output of each sub-layer is LayerNorm(x+Sublayer(x)), where Sublayer(x) is the function implemented by the sub-layer itself. To facilitate these residual connections, all sub-layers in the model, as well as the embedding layers, produce outputs of dimension $dmodel = 512$.

- *Decoder*: The decoder is also composed of a stack of $N = 6$ identical layers. In addition to the two sub-layers in each encoder layer, the decoder inserts a third sub-layer, which performs multi-head attention over the output of the encoder stack. Similar to the encoder, there are residual connections around each of the sub-layers, followed by layer normalization. The self-attention sub-layer is modified in the decoder stack to prevent positions from attending to subsequent positions. This masking, combined with fact that the output embeddings are offset by one position, ensures that the predictions for position i can depend only on the known outputs at positions less than i.

- *Attention*: An attention function can be described as mapping a query and a set of key-value pairs to an output, where the query, keys, values, and output are all vectors. The output is computed as a weighted sum of the values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key.

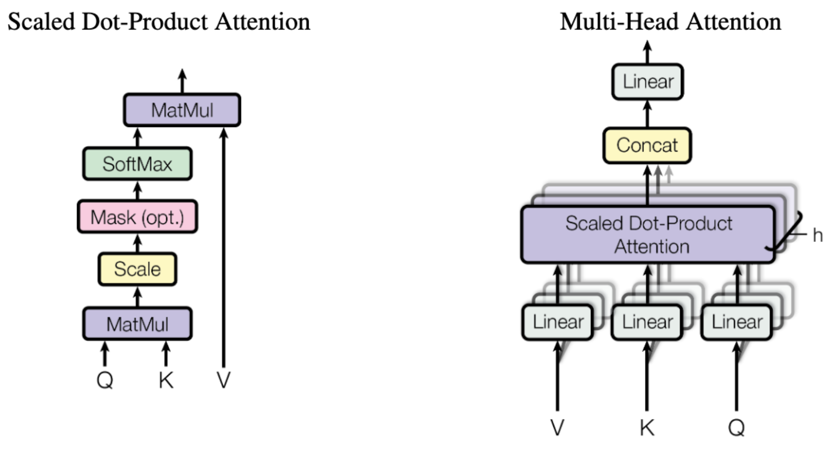


Figure 1.10: Scaled Dot-Product Attention (left). Multi-Head Attention consists of several attention layers running in parallel (right).

1.7 Dataset

The dataset at our disposal describes a period of almost two years (from 02/02/2022 to 16/06/2023) and comes from a 978 kW photovoltaic plant located in the province of Bari. It consists of 3 inverters, 27 field panels, 1 meter, 1 solarimeter, 2 interface protections and 1 "system" device in which data from Solargis are stored. The dataset is organized into files, one for each type of device, representing each individual day, and the data is aggregated every 5 minutes. Each row contains a reference to the device it belongs to (`deviceName` and `deviceId`).

File Name
`2022_02_02_Rofilo_NP00003174_inverter.csv`
`2022_02_02_Rofilo_NP00003174_meteorology.csv`
`2022_02_02_Rofilo_NP00003174_meter.csv`
`2022_02_02_Rofilo_NP00003174_other.csv`
`2022_02_02_Rofilo_NP00003174_plantDevice.csv`
`2022_02_02_Rofilo_NP00003174_stringbox.csv`
`2022_02_03_Rofilo_NP00003174_inverter.csv`
`2022_02_03_Rofilo_NP00003174_meteorology.csv`
`2022_02_03_Rofilo_NP00003174_meter.csv`
`2022_02_03_Rofilo_NP00003174_other.csv`
`2022_02_03_Rofilo_NP00003174_plantDevice.csv`
`2022_02_03_Rofilo_NP00003174_stringbox.csv`
...

Table 1.1: Some files from our dataset.

timestamp	**serial**	...	**TotalEnergy(kWh)**	**Frequency(Hz)**	**deviceid**
2022-10-23 04:30:00	INV01	...	357196.88	50.06	83204
2022-10-23 04:35:00	INV01	...	357196.88	50.06	83204
...	...	...	...	...	...
2022-10-23 13:00:00	INV01	...	357921.16	49.95	83204
2022-10-23 13:05:00	INV01	...	357935.24	49.95	83204
...	...	...	...	...	...
01/02/2023 23:45	INV01	...	431324.36	49.88	83204
01/02/2023 23:50	INV01	...	431324.36	49.88	83204

Table 1.2: Some lines from an Inverter file.

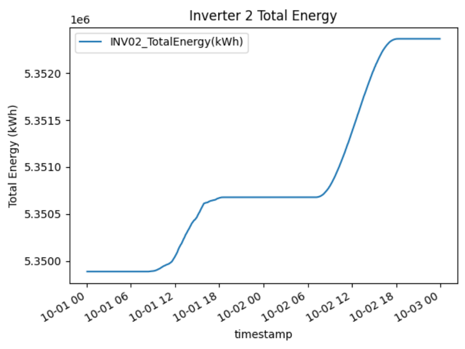
1.7.1 Inverter

Inside the files related to the inverters, we can find data such as the processor's operating temperature (`InternalTemperature`), various Alternating and Direct currents produced (`CurrentAC` and `CurrentDC`), Delivered Power, total energy produced by the individual inverter and other status and control bits that indicate various stages of the inverter's life (boot, reset, ready, etc.).

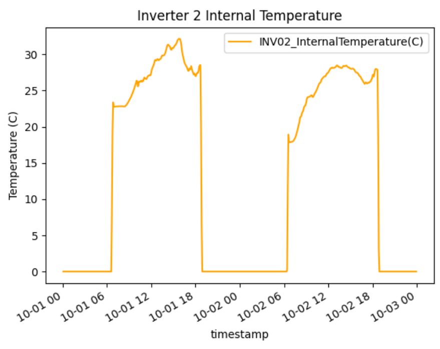
Down below a table with all the available features:

Name	**Unit Symbol**	**Description**
CommunicationCode	-	Communication Code
Failure 3	-	Active Alarm
Failure 4	-	Isolation Alarm
CurrentDC	A	Photovoltaic field Current
CurrentAC	A	Network Current
CurrentAC Phase1	A	Line RMS Current Phase R
CurrentAC Phase2	A	Line RMS Current Phase S
CurrentAC Phase3	A	Line RMS Current Phase T
TotalEnergy	kWh	Active Energy Delivered
Frequency	Hz	Network Frequency
PowerAC Phase1	kW	Line PA Phase R
PowerAC Phase2	kW	Line PA Phase S
PowerAC Phase3	kW	Line PA Phase T
PowerAC	kW	Delivered PA
PowerDC	kW	Photovoltaic field Power
Status	-	Inverter State
Failure	-	Network docking PLL State
Failure 2	-	Network State 1
Failure 1	-	Network State 2
InternalTemperature	C	CPU Temp.
HeatSinkTemperature	C	IGBT Temp.
VoltageDC	V	Photovoltaic field Voltage
VoltageAC	V	Network Voltage
VoltageAC Phase1	V	Line RMS Voltage Phase R
VoltageAC Phase2	V	Line RMS Voltage Phase S
VoltageAC Phase3	V	Line RMS Voltage Phase T

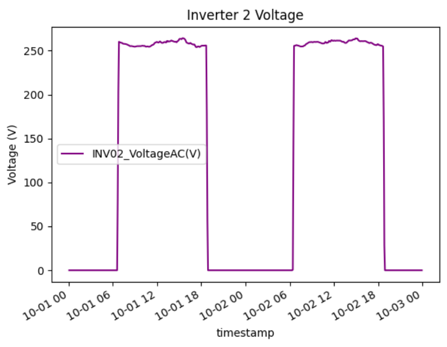
Table 1.3: All available features from an `inverter` file.



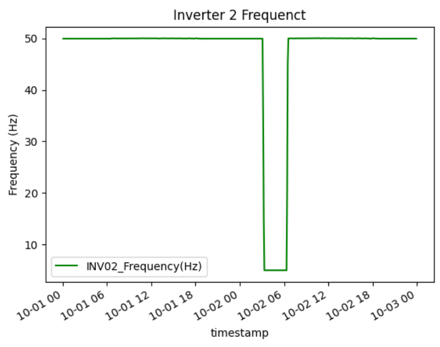
(a) Inverter 2 Total Energy plot.



(b) Inverter 2 CPU Temperature plot.



(c) Inverter 2 Voltage AC plot.



(d) Inverter 2 Frequency plot.

1.7.2 Junction Box

In the files related to the Junction Box or Combiner Box, we find data that describes the current production of the various strings they are connected to (`CurrentString1-7`, in general, each Junction Box manages 7 strings), some temperatures (such as `ModuleTemperature`), and some control bits to check proper operation.

Name	Unit Symbol	Description
CommunicationCode	-	Communication Code
Failure	-	Strings Alarm
CurrentString1	A	Current I1
CurrentString2	A	Current I2
CurrentString3	A	Current I3
CurrentString4	A	Current I4
CurrentString5	A	Current I5
CurrentString6	A	Current I6
CurrentString7	A	Current I7
AverageStringCurrent	A	Average Current
Irradiance	W/m^2	Modules Irradiation
Failure 1	-	Open Strings
Failure 2	-	Not Perform. Strings
EnvironmentTemperature	C	Environment Temperature
ModuleTemperature	C	Modules Temperature
InternalTemperature	C	Board Temperature

Table 1.4: All available features form a `stringbox` file.

1.7.3 Solargis

Solargis is a company specialized in providing solar data and forecasting services for photovoltaic installations and solar energy-related projects. Their main goal is to provide precise and reliable information on solar irradiation and solar weather conditions anywhere in the world. This data is essential for the design, optimization, and management of photovoltaic systems. Solargis collects and provides detailed data on global, direct, and diffuse solar irradiation at every geographical location. This data allows photovoltaic system developers to assess the amount of available solar energy in a given area, which is crucial for properly sizing the system and calculating production forecasts.

timestamp	...	**SolargisGHI(W/m2)**	**SolargisGTI(W/m2)**
2022-08-01 11:40:00	...	896	978
2022-08-01 11:45:00	...	896	978
2022-08-01 11:50:00	...	896	978
2022-08-01 11:55:00	...	914	1001
2022-08-01 12:00:00	...	914	1001
2022-08-01 12:05:00	...	914	1001
2022-08-01 12:10:00	...	928	1019
2022-08-01 12:15:00	...	928	1019
2022-08-01 12:20:00	...	928	1019

Table 1.5: Some Solargis data from `2022_08_01_Rofilo_NP00003174_plantDevice.csv` file

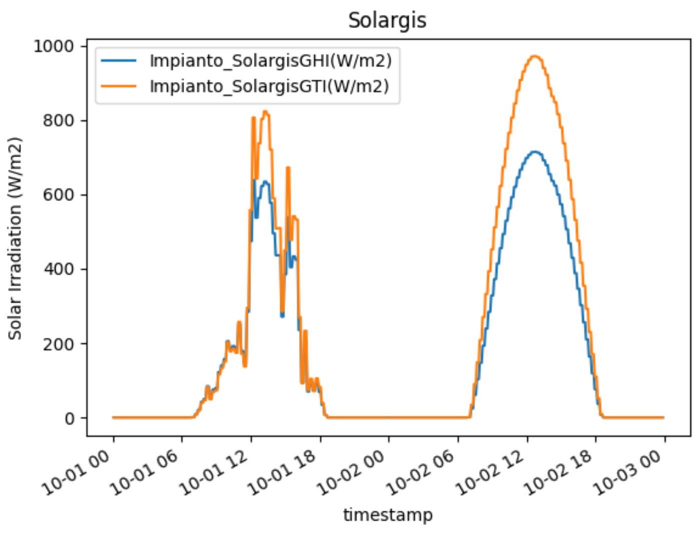


Figure 1.12: Solargis GHI & GTI plot.

Solar radiation takes a long journey until it reaches Earths surface. So when modelling solar radiation, various interactions of extra-terrestrial solar radiation with the Earths atmosphere, surface and objects are to be taken into account. The component that is neither reflected nor scattered, and which directly reaches the surface, is called direct radiation; this is the component that produces shadows. The component that is scattered by the atmosphere, and which reaches the ground is called diffuse radiation. The small part of the radiation reflected by the surface and reaching an inclined plane is called the reflected radiation. These three components together create global radiation.

In solar energy applications, the following parameters are commonly used in practice:

- Direct Normal Irradiation/Irradiance (DNI) is the component that is involved in thermal (concentrating solar power, CSP) and photovoltaic concentration technology (concentrated photovoltaic, CPV).
- Global Horizontal Irradiation/Irradiance (GHI) is the sum of direct and diffuse radiation received on a horizontal plane. GHI is a reference radiation for the comparison of climatic zones; it is also essential parameter for calculation of radiation on a tilted plane.
- Global Tilted Irradiation/Irradiance (GTI), or total radiation received on a surface with defined tilt and azimuth, fixed or sun-tracking. This is the sum of the scattered radiation, direct and reflected. It is a reference for photovoltaic (PV) applications, and can be occasionally affected by shadow.

Name	Unit Symbol	Description
SolargisGHI	W/m^2	Solargis Global Horizontal Irradiation
SolargisGTI	W/m^2	Solargis Global Tilted Irradiation

Table 1.6: All available features from a `plantDevice` file.

1.7.4 Meteorology

In the Meteo files, we can find some environment temperature and solar irradiance data. Is important to mention that these features are not as powerful as weather forecast or Solargis data for the Imputation task.

Name	Unit Symbol	Description
COMMUNICATION_CODE SOL	-	Communication Code
Irradiance SOL	W/m^2	Irradiance
Module Temperature HEX SOL	-	All Registers
Module Temperature SOL	C	Module Temperature

Table 1.7: All available features from a `meteo` file.

1.7.5 Meter

The Meter files contain information about the current injected into and drawn from the network. It is from here that we get the values of our target feature `Cont_TotalEnergy(kWh)`.

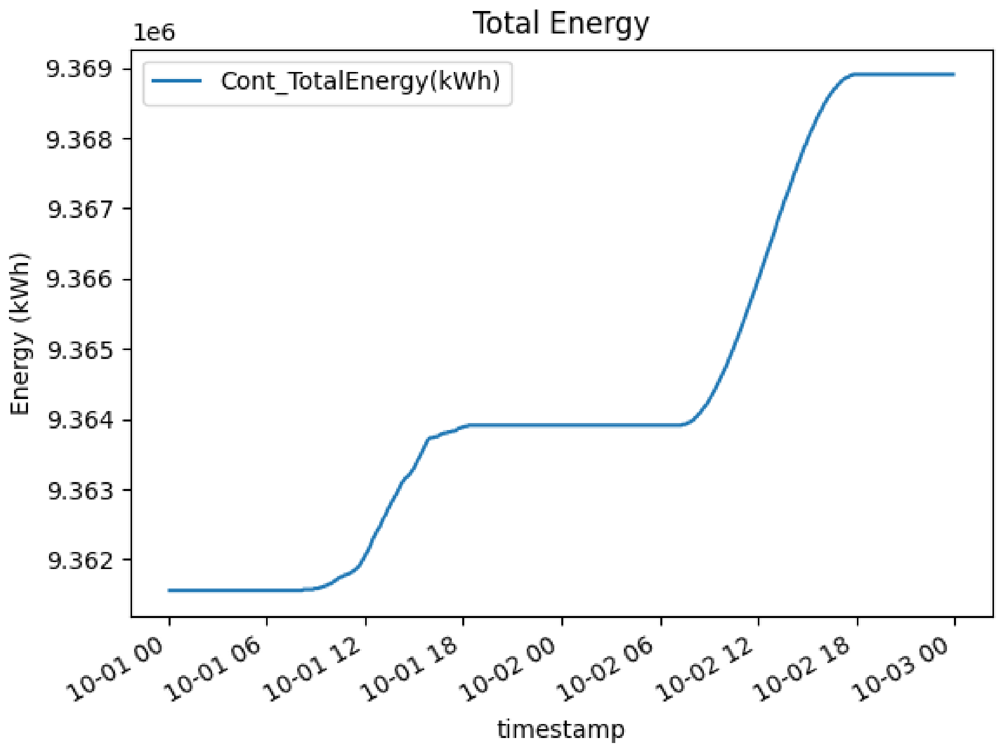


Figure 1.13: Total Energy plot, our target feature.

Name	Unit Symbol	Description
COMMUNICATION_CODE Cont	-	Communication Code
Status Cont	-	Status
Totale Energia Immessa Cont	kWh	Total Energy
Totale Energia Prelevata Cont	kWh	Total Energy Imported

Table 1.8: All available features from a `meter` file.

1.7.6 Other

In the Other type files, we can find the remaining, less relevant, features that describe the behavior and functioning of the leftover elements that make up the photovoltaic plant system.

Name	Unit Symbol	Description
COMMUNICATION_CODE NV10P	-	Communication Code
CB-State NV10P	-	CB State
Frequency NV10P	Hz	Frequency
IN1 NV10P	-	Digital IN 1
IN2 NV10P	-	Digital IN 2
Last Trip Cause NV10P	-	Last Trip Cause
NV10P - Trip BF	-	Digital IN 25
NV10P - Trip f<	-	Digital IN 24
NV10P - Trip f>	-	Digital IN 23
NV10P - Trip U<	-	Digital IN 21
NV10P - Trip U>	-	Digital IN 22
UE NV10P	V	UE
UL1 NV10P	V	Voltage AC Phase 1
UL2 NV10P	V	Voltage AC Phase 2
UL3 NV10P	V	Voltage AC Phase 3
Un NV10P	V	Un
Unp NV10P	V	Unp
COMMUNICATION_CODE NA16	-	Communication Code
CB-State NA16	-	CB State
IE NA16	A	IE
IL1 NA16	A	Current AC Phase 1
IL2 NA16	A	Current AC Phase 2
IL3 NA16	A	Current AC Phase 3
IN1 NA16	-	Digital IN 1
IN2 NA16	-	Digital IN 2
NA16 - Protection Trip	-	Digital IN 25

Table 1.9: All available features from an `other` file.

Chapter 2

Data Preprocessing

Having the dataset divided into different files, one for each day and device (see Section 1.7), with the presence of some periods, ranging from a few minutes to several days, of missing data (likely due to a data collection tool failure), results in many issues during the training phase and makes it almost impossible. In this chapter, we will see how we have addressed these problems by opting for a monolithic tabular structure for the dataset and discussing the approaches we have used for handling missing data.

2.1 Dataset Realization

For the creation of our dataset, we devised a procedure that allowed us to obtain a single file, in CSV format, where for each timestamp, we have all the data for the entire system at that exact moment. Below is the final structure of the dataset and the algorithm for generating it.

timestamp	**DEV.NAME$_1$__FEAT$_1$**	...	**DEV.NAME$_n$__FEAT$_n$**
01/10/2022 10:00	...	...	...
01/10/2022 10:05	...	...	...
01/10/2022 10:10	...	...	...

Table 2.1: Final dataset feature structure.

Algorithm 1 Dataset aggragation algorithm

```
Require: data_folder
Ensure: data_folder exists
  dev_types ← find all file types inside data_folder (e.g. meter, inverter, ...)
  dev_content ← a dictionary with dev_types types as keys and empty values
  for each key in dev_conent.keys do
      files ← find all file matching type key inside data_folder
      sort files by date (asc.)
      temp_type_aggregate ← and empty csv table
      for each file in files do
          append all file lines to temp_type_aggregate table
      end for
      dev_content[key] ← temp_type_aggregate
  end for
    ▷ At this time we have a dictionary mapping a file type with all its available
  data

  dataset ← an empty csv table
  for each type, data in dev_content do                ▷ type is key, data is value
      rename all data columns to data.deviceID_column.name
      except for 'timestamp' column
      dataset ← merge dataset and data tables using 'timestamp' column
  end for
  save dataset table to file
```

timestamp	INV01_PowerAC	...	Cont_TotalEnergy
2022-02-02 00:05:00	NaN	...	NaN
2022-02-02 00:10:00	NaN	...	NaN
...	...	...	...
2022-06-22 10:20:00	175.66	...	8900941.5
2022-06-22 10:25:00	178.29	...	8900995.5
2022-06-22 10:30:00	180.82	...	8901036.0
...	...	...	...
2023-06-16 18:00:00	NaN	...	NaN
2023-06-16 18:05:00	NaN	...	NaN

Table 2.2: Some data from dataset after running Algorithm 1

2.1.1 Timestamp cyclical encoding

To enable the models to learn the alternation of minutes, days, and months as effectively as possible during the training phase, we applied a procedure to transform each individual timestamp into a pair of sine and cosine values, thus performing a cyclic encoding of various seasonalities.

Algorithm 2 Cyclical Encoding Algorithm

```
Require: dataset table
Ensure: dataset is not empty
  dataset['minute_sin'] $\leftarrow \sin(2\pi(\frac{\texttt{dataset.timestamp.minute}}{60}))$
  dataset['minute_cos'] $\leftarrow \cos(2\pi(\frac{\texttt{dataset.timestamp.minute}}{60}))$
  dataset['hour_sin'] $\leftarrow \sin(2\pi(\frac{\texttt{dataset.timestamp.hour}}{24}))$
  dataset['hour_cos'] $\leftarrow \cos(2\pi(\frac{\texttt{dataset.timestamp.hour}}{24}))$
  dataset['day_sin'] $\leftarrow \sin(2\pi(\frac{\texttt{dataset.timestamp.day}}{31}))$
  dataset['day_cos'] $\leftarrow \cos(2\pi(\frac{\texttt{dataset.timestamp.day}}{31}))$
  dataset['month_sin'] $\leftarrow \sin(2\pi(\frac{\texttt{dataset.timestamp.month}}{12}))$
  dataset['month_cos'] $\leftarrow \cos(2\pi(\frac{\texttt{dataset.timestamp.month}}{12}))$
```

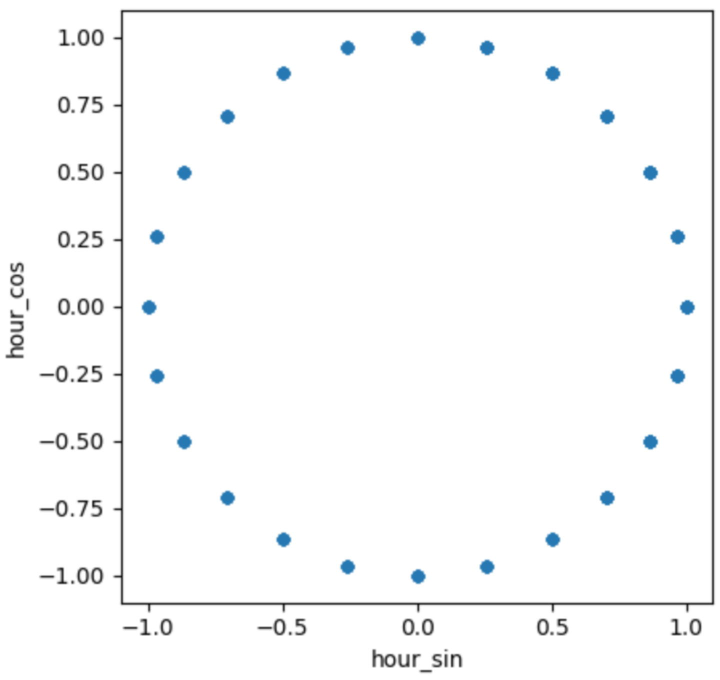


Figure 2.1: Hour cyclical encoding plot.

2.1.2 Dealing with Holes

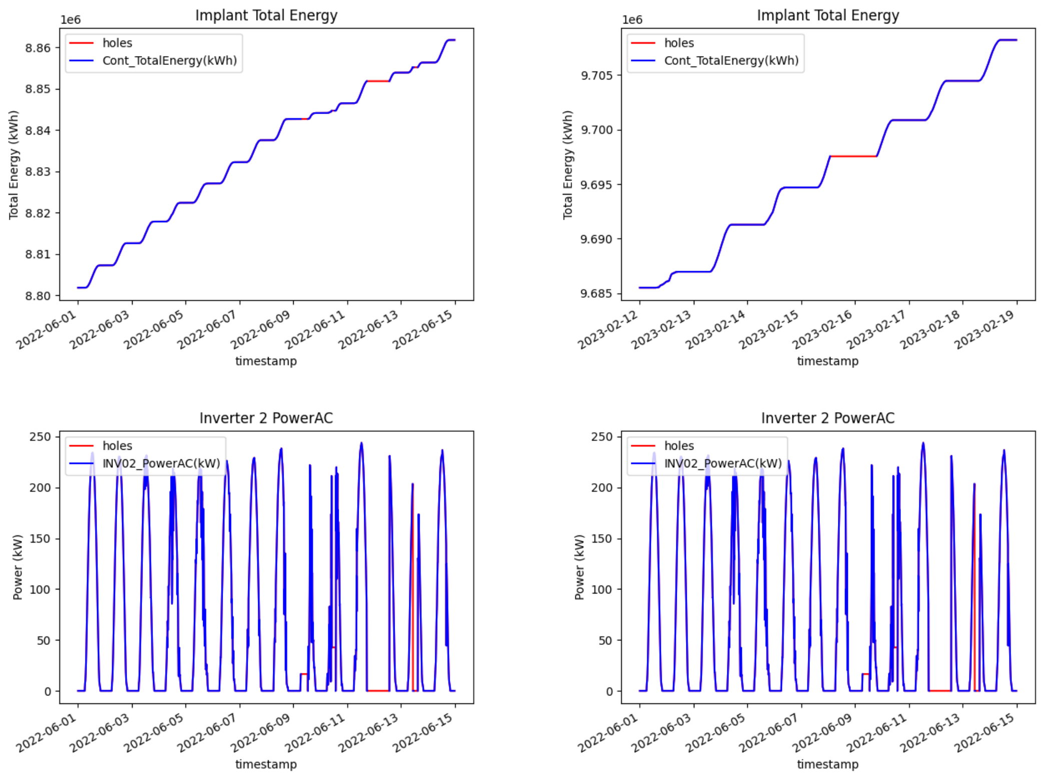


Figure 2.2: Some dataset “holes”. The charts at the top refer to the Implant’s Total Energy, while those at the bottom refer to the Inverter 2’s Power. The charts on the left range from 01-06-2022 to 15-06-2022, while those on the right range from 12-02-2023 to 19-02-2023.

As we can see from Figure 2.2, there are certain periods within the dataset (highlighted in red) where data is missing, which we refer to as “holes”. Leaving these gaps in the dataset causes problems during model training (hindering the correct calculation of the loss function), and therefore, they need to be removed. One possible approach for holes removal is to perform a *fill* operation: filling the gap with the first available non-null value. This tactic may be considered acceptable only if the time span involves just a few timestamps. However, if we are talking about several hours or even days, it significantly distorts the overall production and instantaneous power trends, resulting, especially in very unfortunate cases, with extremely abnormal production curves.

The solution we have adopted to address this problem is the removal of the entire day when a hole occurs. For example, if we have a data gap from 12-02-2023

23:00 to 13-02-2023 10:40, the days 12-02-2023 and 13-02-2023 will be completely removed. With this method, we lose some data, but as we will see later, the number of gaps is not extremely high, and this data loss is not debilitating. The following algorithm summarizes what has just been described.

Algorithm 3 Holes Removal Algorithm.

```
Require: dataset table
Ensure: dataset is not empty
  holes ← find all timestamp from dataset table, where there are some Nans inside the columns
  for each row in dataset.rows do
      if row.timestamp is in holes then
          drop row from dataset table
      end if
  end for
```

Timestamp
2022-06-09
2022-06-10
2022-06-11
2022-06-12
2022-06-13
2022-06-28
2022-06-29
2022-06-30
2022-08-26
2022-09-23
2022-10-06
2023-02-03
2023-02-15
2023-02-16
2023-03-26

Table 2.3: Timestamps deleted after running the Algorithm 3

2.1.3 Historical weather

2.2 Feature Selection