POLITECNICO DI TORINO

College of Computer Engineering, Cinema and Mechatronics

Master's Degree Thesis

Novel Data-Driven Statistical Approach to Predict Photovoltaic Plants Production Based on Weather Forecasts

Supervisors
Prof. Bartolomeo MONTRUCCHIO
Dr. Jacopo SINI
Dr. Antonio Costantino MARCEDDU
Dr. Alessandro CIOCIA

Candidate
Gianluca CARDINALE

DICEMBRE 2024

Summary

The greatest danger to our planet is the belief that someone else will save it ... The last great exploration on earth is to survive on earth.

These are the words of Robert Swan, a renowned British explorer and environmentalist. While the exact date of this specific quote isn't clearly documented, its meaning resonates through time and should serve as an inspiration to tackle today's challenges. We live in a time of significant climate instability, witnessing our planet heat up at an accelerating pace and already starting to burn. The worst mistake is to wait for someone else to take action. We must implement measures for change and do so immediately. On the other hand, we have effective tools to deal with the problem and can leverage modern technologies to our advantage. In the field of energy transition, renewable energies are our cleanest available weapons for decarbonization and against global warming. However, to be optimally implemented, most of them must be effectively managed and scheduled. The case concerns solar energy derived from photovoltaic panels, a source now regarded as inexhaustible. However, due to the intermittent nature of its availability, its output is not constant and must come with a predictive strategy for optimal utilization.

The purpose of this master thesis is to conduct research on predictive solar energy methods and build a model utilizing artificial neural networks, based on known historical data from a plant regarding past weather conditions and forecasts and their corresponding power output production. In the best-case scenario, this model can be used to provide forecasts on upcoming energy production. The use of several sets of meteorological forecasts for the same prediction time will be useful to show and account for the improving efficiency of the model.

Chapter 1 provides an overview of renewable energy sources, emphasizing their integration into the energy sector and broader economic and political contexts. The focus then shifts to solar photovoltaic technologies, with a brief review of system components and the basic operating principles of a panel. This leads to a survey of forecasting approaches, from physical and statistical models to those based on artificial intelligence.

Chapter 2 focuses on artificial neural networks, beginning with the perceptron model, neuron equation, and activation functions. It then covers forward and backward propagation, introducing the matrix-form equation that links inputs and outputs in a feedforward neural network (FNN). Gradient Descent and error metrics for model training and performance evaluation are discussed. The chapter also includes an overview of Long Short-Term Memory (LSTM) networks and a literature review of ANN applications in solar power forecasting.

Chapter 3 details the case study and tools used for modeling, including an overview of the database, data collection methods, and SQL queries used for data handling.

Chapter 4 is the analytical core, presenting the methods and rationale. Excerpts from the MATLAB code, included as an appendix, explain functions developed for data structuring and preprocessing. FNN and LSTM models are introduced, with distinctions noted. In LSTM modeling, different sequences of past days are tested. An initial approach excludes prior forecast values, instead testing the network with forecasted temperature and irradiance; later, models are retrained with forecast inputs

Chapter 5 discusses the results from the models, comparing network outputs. Notably, when models trained on actual temperature and irradiance are tested, the results are highly accurate, suggesting that improving forecast data could enhance predictions. A new network was developed

to reduce irradiance forecast error relative to measured values. Additionally, a post-processing approach uses a secondary network to adjust power predictions, addressing both positive and negative error margins. Finally, a method for calculating underlying areas considers the impact of random external factors, such as passing clouds, on power output.

Chapter 6 summarizes the findings, focusing on analytical results and insights to clarify the approach and outcomes, and offers recommendations for future research directions and refinements.

Acknowledgements

A special thanks goes to my tutors who have guided and accompanied me throughout this work, which has indeed been a fascinating experience for me. In particular, I would like to thank Jacopo Sini for helping me find such a great and innovative application as the one addressed in this master's thesis and for the insightful discussions we have shared together with the other tutors. I also thank Antonio Marceddu for providing valuable assistance by pointing out useful resources for learning how to work with neural networks, as well as for all the tips regarding potential model development. I am grateful to Alessandro Ciocia for giving me the opportunity to be part of the broader project within Politecnico di Torino, and for providing all the tools and support to keep me up to date at all times, along with the resources related to the application and solar photovoltaic context. Lastly, I want to thank all three of them for the personal support, beyond the technical aspects, and for the time we have spent together.

Of course, I would like to thank my family for their moral and financial support throughout these years of study. Thanks to the opportunities they have given me, I have been able to combine personal experiences with my studies, and this work is a testament to that, as I carried it out remotely from Valencia, a place to which I feel connected, especially in the field of renewable energy.

Contents

List of Figures

Chapter 1

General Context

1.1 Climate Change and Historical Countermeasures

The energy transition has been a recurring phenomenon throughout history, such as the switch from wood to coal in the 19th century and from coal to oil in the 20th century. The urgency to implement effective measures likely stems from the fear of approaching a point of no return, given the increasingly rapid pace of climate-related disasters. Temperatures have particularly increased in recent years compared to the average from 1951 to 1980, as shown in Figure 1.1. This phenomenon is closely related to the greenhouse effect: the composition of the atmosphere includes some greenhouse gases (GHGs), namely gases that naturally enable life on Earth by retaining heat. These include carbon dioxide (CO_2), methane (CH_4), and water vapor (H_2O). Unfortunately, this natural balance was not designed to handle the enhanced greenhouse effect caused by human activities, which have undeniably become the primary source of CO_2 emissions and, consequently, global warming. Burning fossil fuels (coal, oil, and natural gas), deforestation, and various industrial processes release large amounts of CO_2 and other GHGs into the atmosphere daily.

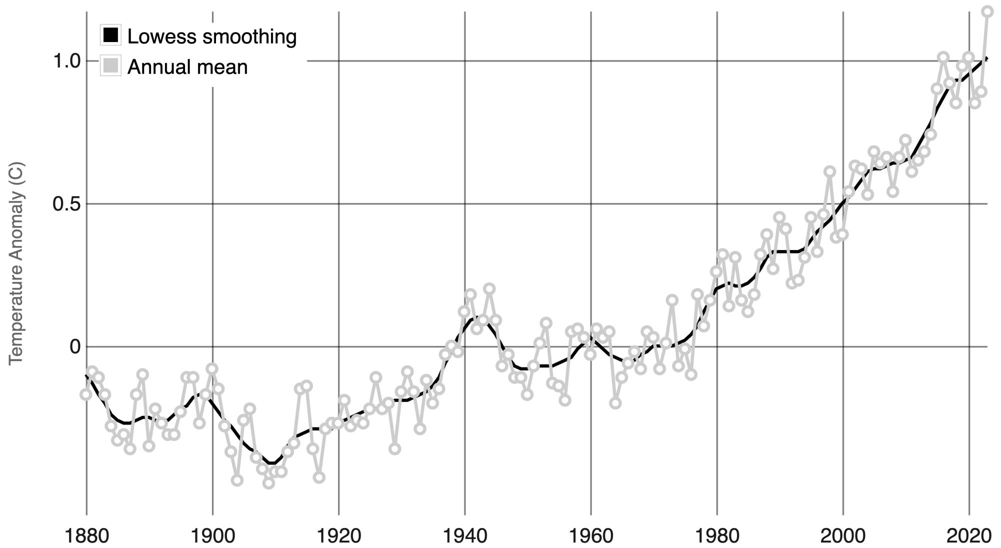


Figure 1.1. Change in global surface temperature compared to the long-term average from 1951 to 1980 [1].

This correlation between human emissions and global warming was clear already to the Intergovernmental Panel on Climate Change (IPCC) in 1990 and was the main topic on the agenda at Earth Summit held in Brazil in 1992 which led to the creation of the United Nations Framework Convention on Climate Change (UNFCCC) [2]. During the third Conference (December 1997) Kyoto Protocol was signed, under which the Parties agreed to reduce emissions by 5% between 2008 and 2012 compared to 1990 levels. The protocol entered into force in 2005. In the following years other meetings took place around the world, such as Copenhagen Conference in 2009 to discuss about counteractions, on the same line Paris Agreement was lately signed in 2015 by 196 Countries that undertook to plan countermeasures and evaluate results every 5 years thereafter. In 2020, the concentration of CO_2 in the atmosphere was 48% higher than pre-industrial levels (before 1750) [3]. The main polluting facilities in Europe are located in Germany, the United Kingdom, Poland, Spain, and Italy. Among the most recent historical events, the European Green Deal (2019) must be mentioned, a potent political instrument designed to boost the energy transition through incentives and achieve climate objectives by 2030. "Fit for 55" is a set of proposals to update EU legislation and launch new initiatives aimed at complying with climate goals, specifically to reduce net greenhouse gas emissions by at least 55% by 2030 and achieve climate neutrality by 2050 [2]. The CAT (Climate Action Tracker) thermometer in Figure 1.2 is a nice visualization tool to sum up policies and counteractions considering temperature variation from neutral average values. The goal in general is to cut off as much as possible CO_2 emissions to bring average values of temperature back to neutral ones, in particular by limiting the increase to 1.5°C compared to pre-industrial values.

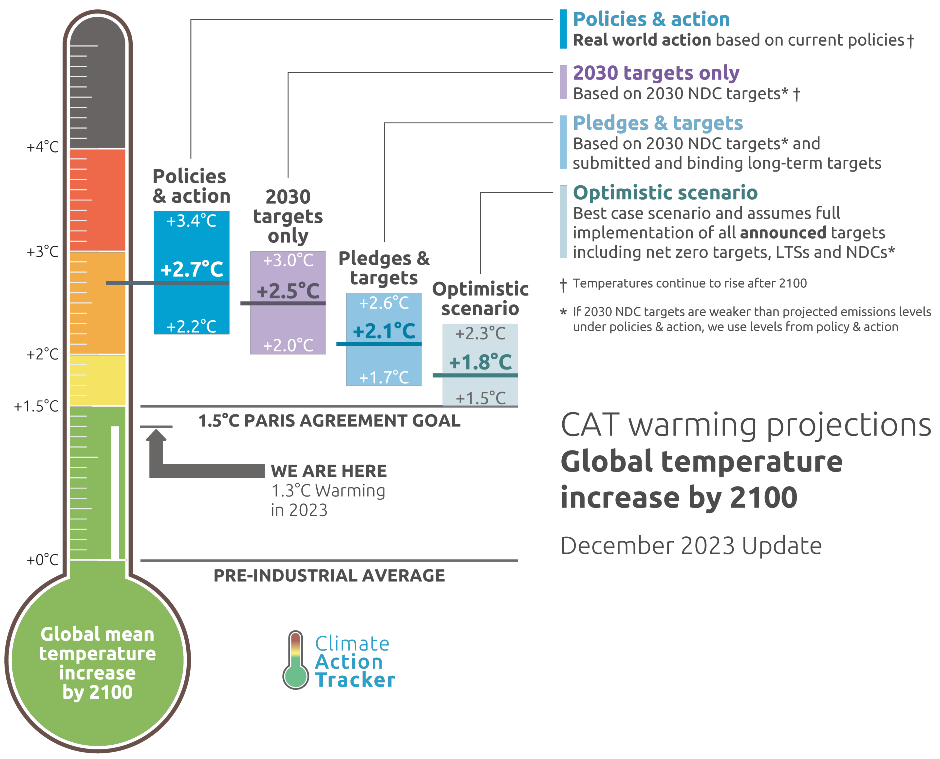


Figure 1.2. The Climate Action Tracker Thermometer [4].

1.2 Renewable Energy Sources (RES)

Energy-intensive industries require vast amounts of energy to conduct their operations and the ones in the fields of machinery manufacturing, steel production and aluminum production account for nearly 60% of the total energy demand in the industrial sector [2]. Self-consumption from renewable energy sources is becoming one of the most cost-effective solutions for industries. Furthermore, transitioning to renewable energy is key to achieving environmental and decarbonization goals to mitigate climate change. The main characteristic of RES is that they are unlimited and do not produce greenhouse gas emissions, which makes them non-polluting.

Renewable energy includes those types of energy that are obtained from natural resources [5]:

- **sun**:
  - **solar photovoltaic**: widely applicable, suitable for rooftops through solar panels installation, only available during daylight hours; subsequently, this work will be focused on solar photovoltaic production, hence a detailed description will follow;
  - **solar thermal**: suitable for large heat demands/high-temperature processes, available during daylight hours; the estimated payback period for these thermal installations in high-consumption industries is a minimum of 7 years;
- **wind**: suitable for regions with consistent and sufficient wind resources, variable availability, long lifespan, allowing companies to reduce their reliance on conventional energy sources in the long term; turbines can be installed either onshore or offshore, the latter resulting more productive as the wind in the middle of the ocean is uninterrupted and more regular turning out to hit the blades stronger;
- **water**:
  - **hydroelectric energy** is obtained from falling water which is rain water; a significant initial investment is needed, however, once in operation, they are very affordable and efficient production plants. There can be storage hydropower plants, run-of-river hydropower plants and pumped-storage hydropower plants, depending on the possibility for building dams and reservoirs;
  - **marine energy**, also known as ocean energy, in the form of either tidal energy - power is produced during the rise and fall of tides - or wave energy - power is produced by harnessing the movement of the waves -;
- **geothermal energy**: obtained by harnessing the heat from within our planet stored in rocks, soils, and groundwater. When deep-water streams from rainwater come into contact with high subsurface temperatures, they create a geothermal reservoir formed by water and steam at elevated temperatures. To transform the heat energy into electricity, a geothermal plant must be installed on the reservoir to collect the natural fluid and transform it into mechanical energy using a turbine, steam can then be redirected to thermal energy production;
- **plant or animal biomass**: applicable where biomass is abundant and sustainable, availability is ensured, CO_2 released during combustion is approximately equal to the amount absorbed by the plants during their growth; coal-powered systems can be easily adapted, either fully or partially, to be powered by biomass.

The initial investment for RES is quickly recovered – the estimated payback period for energy generation projects in industries is less than 10 years. Additionally, the costs of renewable energy and batteries are decreasing significantly year after year. RES can be categorized into two groups: programmable and non-programmable. The criterion essentially concerns the availability from the source, as a consequence programmable RES allow for scheduling and management of generation, while non-programmable RES have more variable output based on natural factors such as weather forecasts. Due to their natural intermittency, sources dependent on sun and wind as well as marine energy are considered non-programmable. Instead, geothermal, hydroelectric and biomass sources

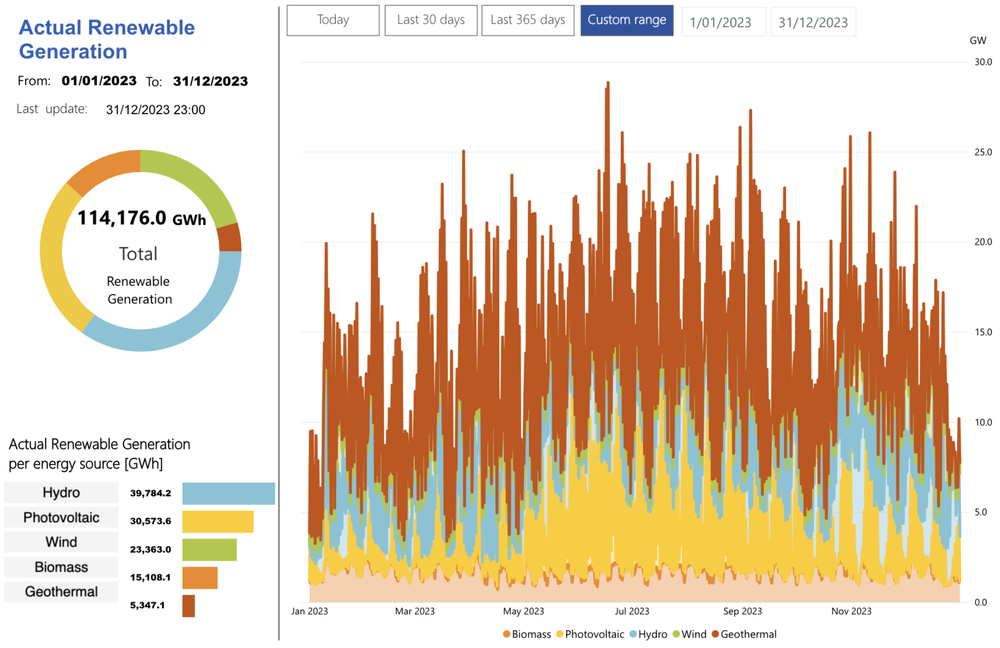


Figure 1.3. Distribution of national RES in 2023 (Italy) [6].

are considered programmable because they can be utilized whenever there is a need to activate them. An example of the distribution of renewable energies at the national level with their generation profile is shown in Figure 1.3.

1.2.1 Integration of RES into the Grid

The integration of RES into the grid (Figure 1.4) involves incorporating sources like solar, wind and hydroelectric power into the existing electricity distribution network. This process requires careful planning to manage the variability of RES output and ensure stability and reliability of the grid. Techniques such as advanced forecasting, flexible grid management, energy storage systems, and demand-side management are employed to optimize the integration of RES and maximize their contribution to the grid's overall energy mix. There are several key challenges to address [7]:

- Variability and Intermittency: RES such as solar and wind sources are intermittent and the resulting output power fluctuates depending on weather conditions. This variability can lead to challenges in matching energy supply with demand in real time. Traditional power plants provide steady, controllable power, whereas RES output is less predictable and may require additional grid flexibility measures;
- Grid Stability and Reliability: Introducing large amounts of intermittent RES can affect grid stability and reliability. Sudden changes in generation levels due to weather fluctuations can impact grid frequency and voltage stability;
- Grid Management and Operation: Advanced grid management tools and techniques are required to efficiently integrate RES while maintaining grid stability and reliability. This includes advanced forecasting, real-time monitoring, and demand response programs;
- Energy Storage: Storage technologies are crucial for smoothing out fluctuations in RES output and providing grid stability.;
- Transmission and Distribution Infrastructure: Upgrading and expanding transmission and distribution infrastructure to accommodate RES generation from remote locations to urban centers can be costly and time-consuming.

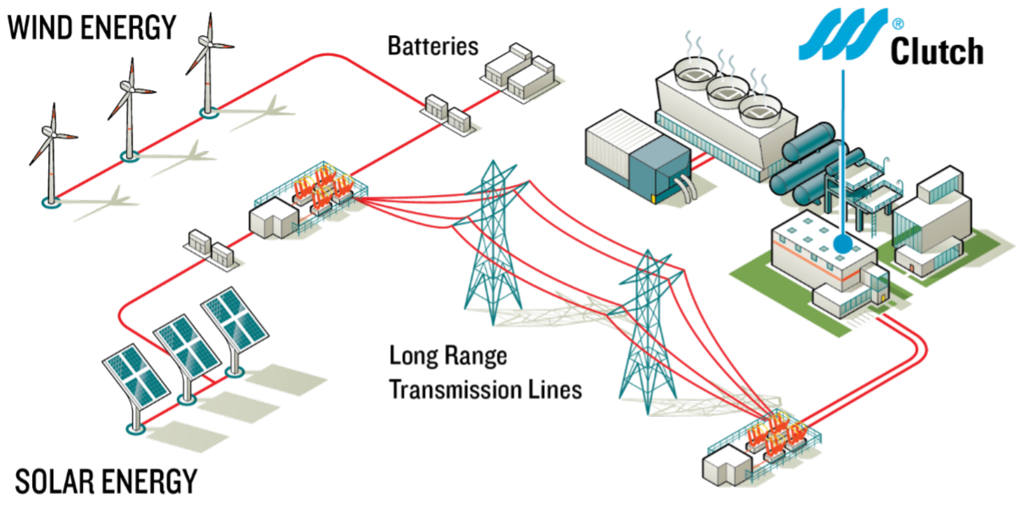


Figure 1.4. Integration of RES in the grid [8].

Addressing these challenges requires a comprehensive approach involving technological innovation, policy support, market reforms and collaboration among stakeholders including utilities, regulators, technology providers, and consumers.

1.2.2 Political and Economical Aspects

According to the International Energy Agency (IEA), accelerating the deployment of RES can mitigate the impact of high energy prices and enhance energy resilience, as well as reduce the dependence on volatile fossil fuel markets. The long-term cost savings and environmental benefits make renewables an attractive option for addressing the current energy crisis. Here are some examples of relevant national installations. in Europe that contribute to the continent's energy independence:

- Hydroelectric Power Plant **Cortes - La Muela** (Spain). Located in the Valencian Community, this plant is one of the largest renewable energy installations in Europe. It has a capacity of 1,762 MW and harnesses water energy to produce electricity. This facility is a significant step towards Spain's energy independence [9];
- **London Array** Wind Farm (United Kingdom). This is one of the largest offshore wind plants in the world, located in the North Sea. It has an installed capacity of 630 MW, sufficient to power hundreds of thousands of homes. The London Array helps the UK reduce its dependence on gas and oil imports [10];
- **Cestas** Solar Park (France). Located in the nearby of Bordeaux, this solar park is one of the largest in Europe, with an installed capacity of 300 MW. It provides energy to about 100,000 French households, significantly contributing to France's goal of increasing renewable energy production [11];
- **Hellisheiði** Geothermal Power Plant (Iceland). Although Iceland is not part of the European Union, the Hellisheiði plant is a notable example of harnessing geothermal energy. With an installed capacity of 303 MW electrical and 400 MW thermal, this plant provides clean and sustainable energy, hence being representative of the potential of geothermal energy in regions with volcanic activity [12];
- **Horns Rev 2** Wind Farm (Denmark). Located in the North Sea, this offshore wind farm has a capacity of 209 MW. It is one of Denmark's many wind power installations, a country that leads in wind energy production, significantly reducing its dependence on fossil fuels through these investments [13].

These examples illustrate how large renewable energy installations in Europe not only contribute to reducing carbon emissions but also improve the continent's energy security by reducing dependence on energy imports from abroad.

1.3 Solar Photovoltaic Energy and Technology

1.3.1 Solar Irradiance and The Tilt Angle

Solar radiation is the energy emitted by the sun that reaches any type of surface when no obstacle is in between them. From a physical point of view, solar radiation is emitted in the form of electromagnetic waves. Right after emission, it undergoes variations on its path towards the Earth, thus it can be written as [14]:

$$solar\, radiance = Dir + Dif + Ref,$$

namely *Dir* the direct radiation, *Dif* the diffuse radiation and *Ref* the albedo or reflected radiation. Direct radiation refers those rays that directly reach the Earth's surface after crossing the atmosphere free of any obstacle. Instead, diffuse radiation refers to those rays that are retained by the gasses present in the atmosphere due to Rayleigh and Mie dispersion, while reflected radiation includes those rays that are reflected after hitting a general surface or the ground. Solar panels work with direct solar radiation mainly as shown in Figure 1.5. The tilt angle of a solar panel is the angle at which the panel is inclined relative to the horizontal ground and its adjustment is the key to maximize the efficiency and energy output of the system. Of course, as the seasons change and so does the weather, tilt angle must be regulated throughout the year. As a matter of fact, a more steeply positions are characteristic of the winter season, while configurations close to the flat one are typical when panels are directly exposed to the sun, as in the summer season. The eventual choice of the tilt angle may also attempt to reduce reflection and shading.

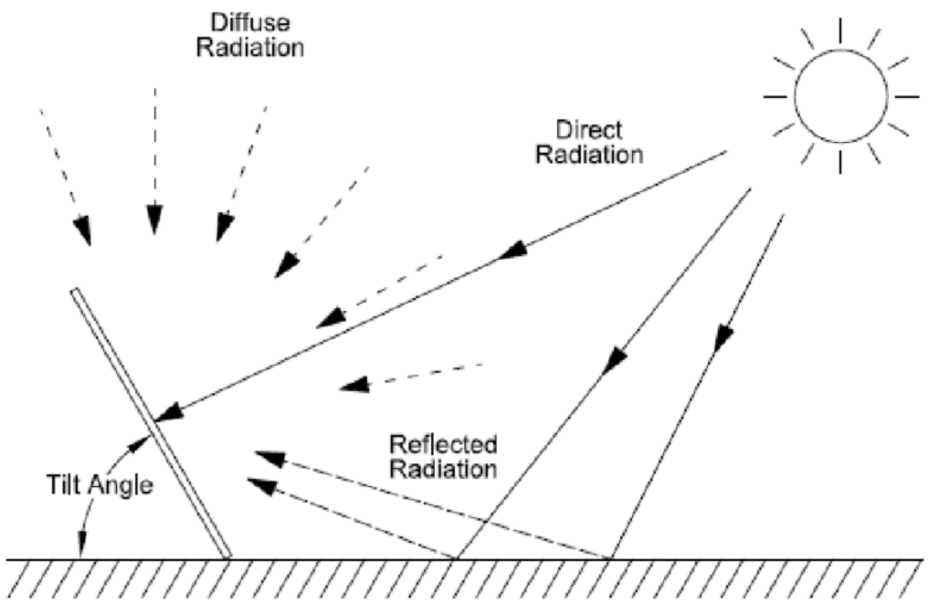


Figure 1.5. Types of Solar Radiation (Mallon et al. 2017) [15].

1.3.2 Panels Placement

Photovoltaic panels can be integrated into buildings in various ways (Figures 1.6, 1.7, and 1.8):

- Conventional roof panels. They are typically made of polycrystalline or monocrystalline silicon cells (more efficient but more costly) [16] and are placed flush with the roof if it comes with a good tilt for direct radiation or mounted on a structure to adjust the tilt;
- Transparent and semi-transparent panels [17]. These are designed to be integrated into windows or other transparent surfaces, allowing visible light to pass through while capturing solar energy. They are usually made with thin-film silicon technologies or technologies like luminescent solar concentrators (LSCs), which are still being studied to improve their efficiency;
- Facade-integrated panels [18]. Most of the time, the orientation and tilt of facades are not optimal, and the architectural function is prioritized over energy generation, making this option generally less efficient. Still, facades can provide some benefit for solar protection and energy generation.

Figure 1.6. Roof panels [19].

Figure 1.7. Transparent panels [20].

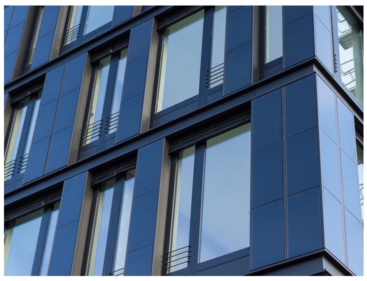


Figure 1.8. Facade panels [21]

Figure 1.9. Integration of solar PV panels into buildings.

1.3.3 Solar Photovoltaic Technology

PV Effect

Solar photovoltaic (PV) energy is obtained by converting light, or rather solar irradiance into electricity. The PV cell consists of one or two layers of a semi conducting material, usually silicon. When light hits the cell, an electric field across the layers is created, resulting in a flowing current which increases with light intensity. Nonetheless, the cells do not need direct sunlight to work, and they can still produce electricity on a cloudy day, leveraging now the effects of diffusion and reflection. The unit of measurement used to refer to the capacity of a cell is kWp, which stands for kilowatt-peak, indicating the maximum theoretical instantaneous power that the cell can produce [22].

System Implementation and Integration

Modules in a plant can be connected to each other in many ways. When connecting several cells in series, a string or array is created (Figure 1.10) and the operating voltage of the system is determined. In particular, the positive terminal of one module is connected to the negative terminal of the next one, resulting in an overall voltage which corresponds to the sum of the voltages of the single modules - the current flowing in one module is the same flowing across the whole string. Another useful configuration is the parallel one in which all the positive terminals

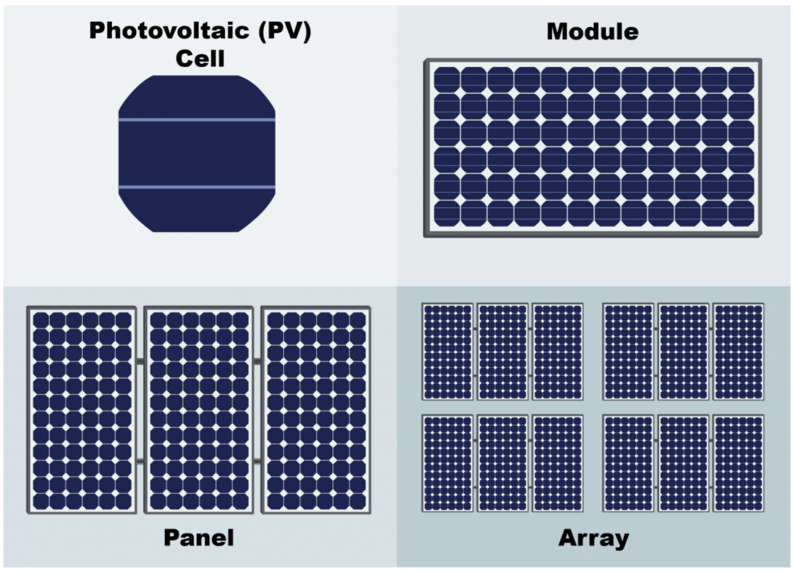


Figure 1.10. From a cell to a panel [23].

are connected together as well as all the negative ones. As a consequence, the voltage remains the same as the one of each individual module, whilst the resulting current is the sum of the currents of the single modules. Usually, a combination of series and parallel connections is preferred to optimize both the current and voltage variables considering load requirements. A reduction of the power output can be due to mismatch losses that are caused even by gently variations in the features of either the cells or the modules. These losses are more common in parallel set-ups. Once panels are installed, they must be connected to an inverter (Figure 1.11) to convert the direct current (DC) output into alternating current (AC). The inverter is then responsible for sending AC current to either the main grid in the case of integration or to a user if the system is used for self-consumption. Another option is to send the AC current to a battery when conditions allow it, such as on days of high solar availability when not all the energy produced by the system may be used.

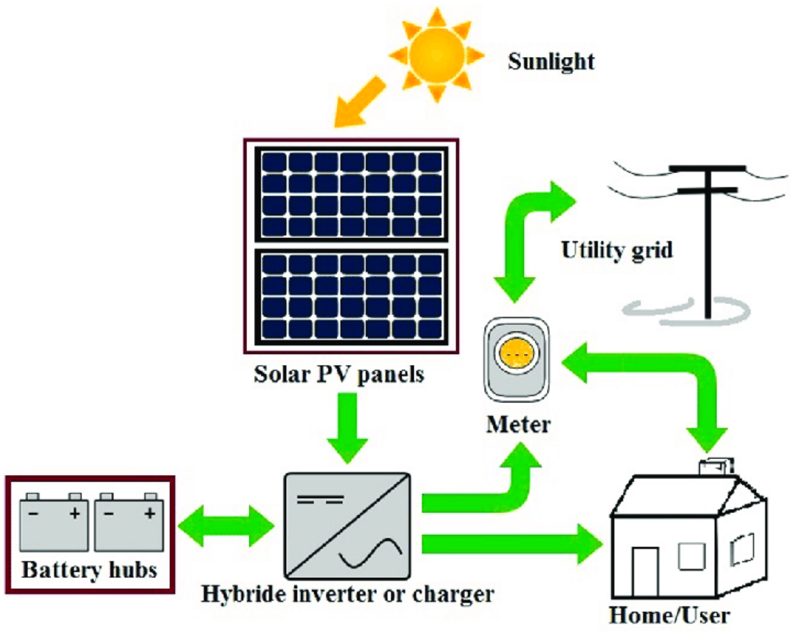


Figure 1.11. PV system composition [24].

Equivalent Circuit

Now that the working principle and system composition has been provided just in words, let's dive into some maths behind the equivalent circuit of a solar photovoltaic cell (Figure 1.12). On

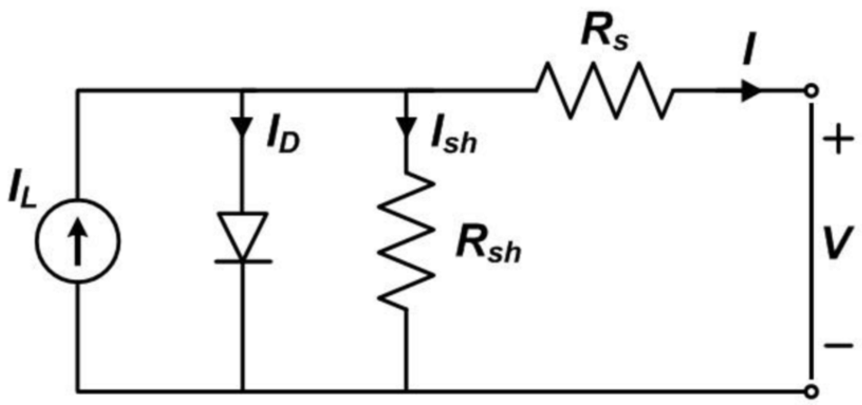


Figure 1.12. Equivalent circuit of a PV cell [25].

the left, the ideal model of a PV cell just includes a current generation parallel to a diode, though, the effects of the losses are considered on the right adding a parallel shunt resistance and a series resistance. The overall current I is then given:

$$I = I_L - I_D - I_{sh},$$

being I_L the photogenerated ideal current generated by the cell, I_D the current across the diode and I_{sh} the current across the shunt resistance R_{sh}. Analysing each term of the equivalence:

- I_L is proportional, through a coefficient k_{max} characteristic of the cell, to the area A of the cell exposed to the sun and the irradiance G:

$$I_L = k_{mat}GA$$

- I_D is the current flowing across the ideal diode, hence the Shockley low provides:

$$I_D = I_0[exp(\frac{U_D}{nV_T}) - 1]$$

  I_0 is the reverse-bias saturation current (or scale current) of the diode, U_D is is the voltage across diode (and the current generator too), and n is the ideality factor. It depends on the absolute temperature T of the cell:

$$V_T = \frac{kT}{q}$$

  , q is the elementary charge and k is the Boltzmann constant.

- I_{sh} is the current across the shunt resistance and can be simply calculated applying Omh's low:

$$I_{sh} = \frac{V + IR_s}{R_{sh}}$$

Recombining the previous equation, it gives:

$$I(V) = k_{mat}GA - I_0[exp(\frac{V + IR_s}{nV_T}) - 1] - \frac{V + IR_s}{R_{sh}}$$

and the characteristic I-V curve as well as the P-V (Power to Voltage) curve can be drawn, as shown in Figure 1.13. It's worth to notice some peculiar points of both the curves:

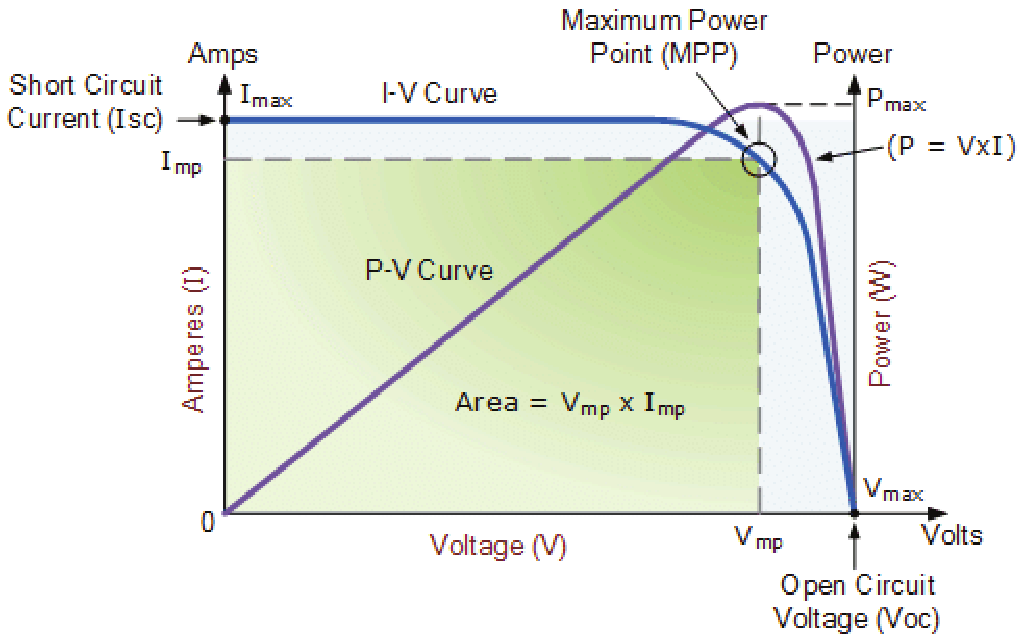


Figure 1.13. I-V and P-V curve of a solar cell [26].

- I_{sc} is the maximum value of the output current and is caused by light-generated carriers. It corresponds to a null voltage across on the diode, that is when the cell is short-circuited;
- V_{oc} is the maximum value of the voltage across the equivalent circuit, it corresponds to the open circuit condition, which is when the current across the diode is null. This voltage is caused by the overall forward-bias on the solar cell junction with the light-generated current;
- MPP is the point on the I-V curve where the voltage V_{mp} and the current I_{mp} correspond to the maximum peak in power (P-V curve). It's not trivial to adjust the working point to stay close to MPP, as a matter of fact electronic devices are used to accomplish the task.

The Effect of Temperature and Irradiance

The effects of temperature and irradiance on the characteristics of a solar panel are now widely known. It's shown the bahvior of the I-V curve in both cases independently, i.e. when the irradiance is kept constant to evaluate the effect of temperature and viceversa. As the irradiance is kept constant (Figure 1.14), an increase in the temperature has almost no effect on the current, but a very slight increase. Instead, the effect on the resulting voltage is considerable as the curve slides significantly to the left, thus the outcoming power suffers a considerable drop. On the other hand, by keeping the temperature constant (Figure 1.15) the I-V curve doesn't move much horizontally, as the V_{oc} is basically the same whether the irradiance parameter is rising or falling. Instead, the current is widely affected as it increases much with the irradiance. We can conclude that the outcome power increases with the irradiance and decreases with the temperature. The causes of these behaviors can be found studying the previous equation. For our purpose, it's enough to take notice and to account for these effects through the power thermal coefficient γ_{th} defined as follows,

$$\gamma_{th} = \frac{dP_{Max}}{dT_{PV}} \frac{1}{P_{Max}}$$

Its value changes with the material used, as for crystalline silicon it is equal to 0,5%°C^{-1} [7].

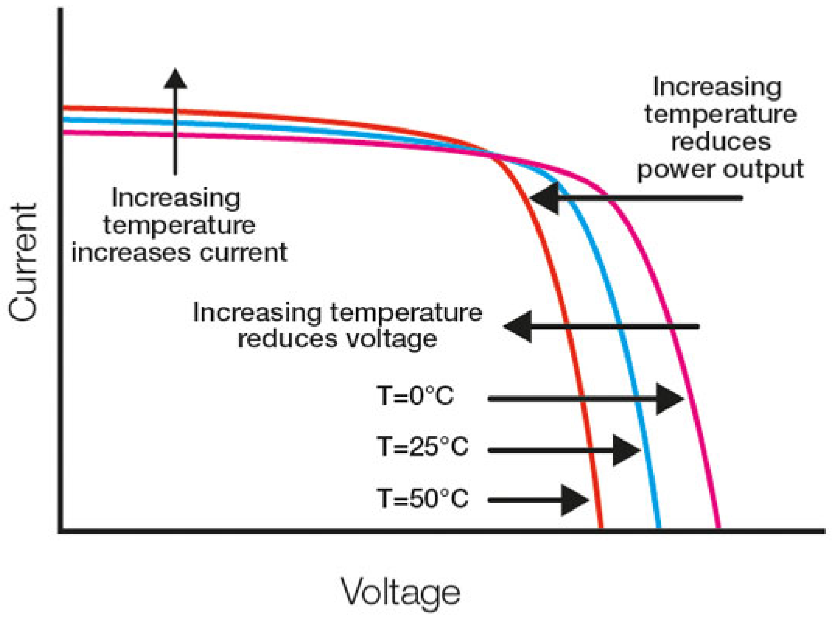


Figure 1.14. Effect of temperature [27].

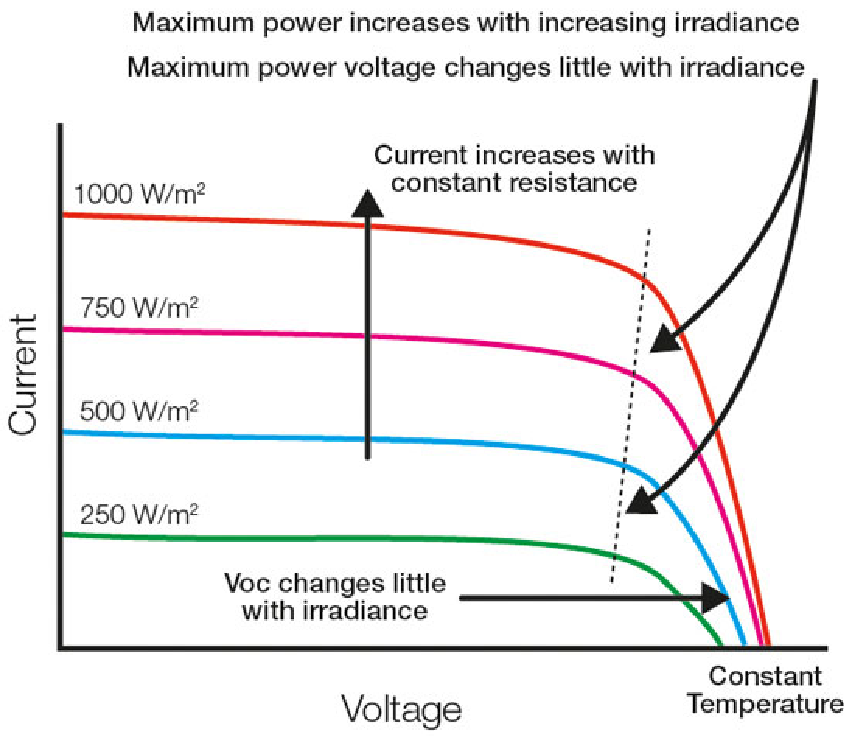
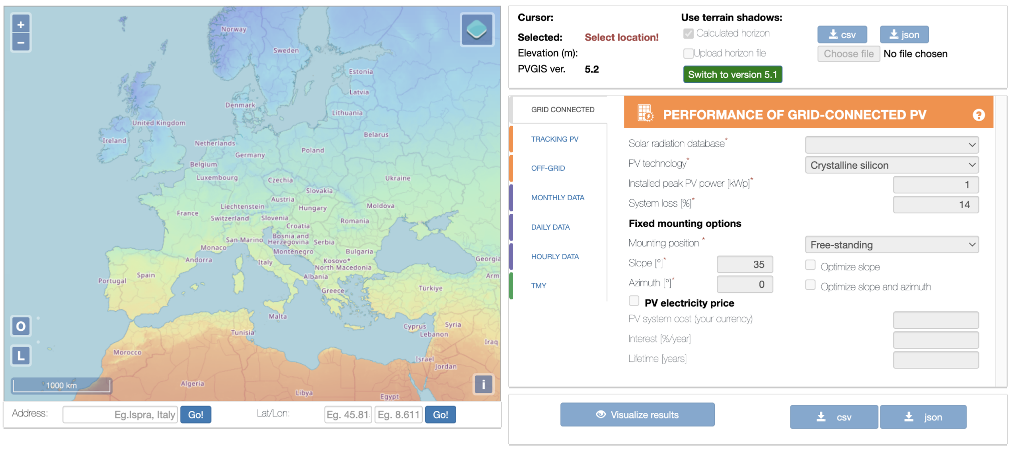


Figure 1.15. Effect of irradiance [27].



Cursor:
Selected: Select location!
Elevation (m):
PVGIS ver. 5.2

Use terrain shadows:
☑ Calculated horizon
☐ Upload horizon file
No file chosen

PERFORMANCE OF GRID-CONNECTED PV

Solar radiation database*	
PV technology*	Crystalline silicon
Installed peak PV power [kWp]*	1
System loss [%]*	14

Fixed mounting options

Mounting position *		Free-standing
Slope [°]*	35	☐ Optimize slope
Azimuth [°]*	0	☐ Optimize slope and azimuth

☐ **PV electricity price**

PV system cost (your currency)	
Interest [%/year]	
Lifetime [years]	

Figure 1.16. PVGIS platform for system performance estimation [28].

Pricing and Available Quotation Tools

The costs of photovoltaic solar energy installation and production have become considerably cheaper, by 89% in the last decade (IRENA 2020). Obviously, they represent a cost-effective solution for industries and self-consumption, too. Photovoltaic generation essentially depends on the daily solar radiation reaching the surface of the panels (thus on the geographic location, their orientation, and tilt), the efficiency of the panels or their production capability together with all the sources of power loss. Some software tools are freely available to get a quote of the costs and simulate the installation and working of a plant, as well as expectations of the power generation. One of those is PVGIS (Photovoltaic Geographical Information System) developed by the Joint Research Centre (JRC) of the European Commission. After entering all the data regarding geographic and climatic conditions and selecting "Visualize results" (Figure 1.16), it is possible to estimate the monthly and annual production per kWp installed. It then provides the optimal tilt and azimuth to maximixe the installed capacity.

1.4 PV Prediction Models

In this section, a brief description of several techniques employed to make forecasts of the output power generation is presented. It's worth to make a distinction between the models used to interpolate data in a regressive way comparing them to actual output values and the models used to make future predictions. The former procedure is very useful as besides providing a measure for the system performance through the KPIs (performance indexes), it indirectly shows the presence of anomalies thus serves a wake-up call to conduct an on-site inspection. The latter methods are the ones used for prediction aims and depending on their nature they are usually split into physical and statistical models.

1.4.1 Physical Models

These models are built considering the physical laws, known in literature, about the PV technology. Since they correlate several parameters with each other and involve additional sub-models within them, such as the temperature and wind model, a subsequent optimization phase is generally conducted to minimize the error between the estimate and the measurement. The estimate is obtained by inputting the pre-processed forecast data of the solar radiation on the POA (plane of array) in addition to a whole set of characteristic parameters of the system (geographic location, azimuth, tilt angle, yield coefficients etc). As used in the previous works [29],[7],[30] and [31], the hourly forecast of the AC power P_{AC} can be obtained theoretically as follows:

$$P_{AC} = P_{DC} \cdot \eta_{CONV} \cdot \eta_{TRANSF} = (P_{STC} \cdot \frac{G - G_0}{G_{STC}} \cdot C_T \cdot \eta_G \cdot C_A) \cdot \eta_{CONV} \cdot \eta_{TRANSF}$$

The elements of the equation are defined:

- P_{DC} is the DC power generated by the plant [kW];
- η_{CONV} is the inverter yield;
- η_{TRANSF} is the transformer yield;
- P_{STC} is the power of the plant in the Standard Test Condition (STC, cell temperature of 25°C and solar radiation of 1000 W/m2) [kW];
- G is the solar radiation on the POA (plane of array) [kW/m^2];
- G_0 is the minimum radiation required by the plant to start producing power [kW/m^2];
- G_{STC} is the radiation according to STC, it gives 1000 kW/m^2;
- C_T is the temperature coefficient and can be obtained as

$$C_T = 1 + \gamma_T \cdot (T_{cell} - T_{STC})$$

- η_G refers to the global yield, namely the product of the yields of all the technical components of the system:

$$\eta_G = \eta_{life} \cdot \eta_{dirt} \cdot \eta_{reflection} \cdot \eta_{mismatch} \cdot \eta_{cable}$$

  As easy as their names suggest, they refer respectively to the aging of the plant (how many years), the dust influence on the panels, the reflection phenomenon, the mismatch due to not perfectly balanced features of different modules and the loss caused by Joule effect in the connections.

- C_A is an adaptive coefficient that has been introduced to account for other possible inaccuracies, especially in the optimization phase.

The AHSRAE model is used to calculate the solar radiation on the POA, as the available data coming from weather stations refer to the horizontal radiation:

$$G = \frac{BHI}{\cos(\theta_Z)} \cdot \cos(\theta) + DHI \cdot F_{CS} + \rho \cdot GHI \cdot (1 - F_{CS})$$

Namely, for further information on the decomposition of the irradiance redirect to subsection 1.3.1:

- BHI is the direct or beam horizontal irradiance, it gives at most 1000 W/m^2;
- DHI is the diffuse horizontal irradiance that has been diffused in the atmosphere;
- GHI is global horizontal irradiance given by the sum:

  $$GHI = BHI + DHI + G_R,$$

  being G_R the irradiance reflected by geneuric surfaces and the ground;
- θ_Z is the Zenith angle that is the angle of the line connecting the Earth with the Sun;
- $cos(\theta)$ is the angle between the line normal to the flat plane and the line connecting the Earth with the Sun;
- F_{CS} is the Earth-sky view factor, used to quantify the fraction of the sky that is visible from a given point on the Earth's surface;
- ρ is the albedo coefficient, it indicates the fraction of incoming solar radiation that is reflected back into space.

1.4.2 Statistical Models

These kind of models rely on statistical data, basically they need a large amount of past historical events in order to make forecasts. There are several methods and a distinct approach whether the method is probabilistic or based on time series analysis.

The first case consists of searching for the closest inputs to the new ones and selecting the corresponding output, as in the case of k-nearest neighbors. In the application of PV power prediction, this model will look for the most similar weather conditions and return the value of past generated power or an adjusted estimate of it. It's trivial to understand that this method performs better and better with time, leveraging the presence of new events that are registered and added into the database. The k-nearest neighbors (k-NN) and the Nearest Centroid Classifier (NCC) models are presented below. Another approach is to consider the available dataset as a time series, indeed the solar power generation is ordered in time. In this field, the ARIMA model and the STL model are reported.

The k-NN Model and NCC Model

The k-nearest neighbors (k-NN) model consists of finding the closest k input conditions, based on the distance between points, and assign the average output [32]. Given a dataset (x, y), a correspondence between x and y is provided linking, for every single value, all the input features to the target output. New input data whose output is not known are now considered, computing their distance from all the points in the initial set. The prediction is made by averaging the k outputs corresponding to the k nearest neighbors. Among the wide range of distances considered, the ones that are mostly used for regression are listed below. To facilitate the reading, the following formula will refer to the distance between 2 generic points x_1 and x_2, reminding that they actually refer to different values from the input set and every i-th component represents an input feature among the n features (imagine in the case of PV solar power prediction, a generic input value x will contain several information about weather conditions, namely each of them is a feature, and a single output value y corresponding to the generated power).

- Manhattan distance:
$$d(x_1, x_2) = \sum_{i=1}^{n} |x_{1,i} - x_{2,i}|;$$
- Euclidean distance:
$$d(x_1, x_2) = \sqrt{\sum_{i=1}^{n} (x_{1,i} - x_{2,i})^2};$$
- Minkowski Distance:
$$d(x_1, x_2) = (\sum_{i=1}^{n} |x_{1,i} - x_{2,i}|^p)^{\frac{1}{p}},$$
it's a generalization of the Euclidean distance;
- Cosine Similarity:
$$similarity(x_1, x_2) = \frac{x_1 \cdot x_2}{||x_1|| \cdot ||x_2||},$$
and the distance is given by
$$d(x_1, x_2) = 1 - similarity(x_1, x_2).$$

Having selected the value k as the number of neighbors to be considered, for a generic new point z the k neighbors showing a minimal distance from it will be part of its output prediction, given by the average of the outputs of every single neighbor:
$$\hat{y}(z) = \frac{1}{k} \sum_{i=1}^{k} y(x_i),$$
basically, the index i now refers to the nearest neighbors and just one target value $y(x_i)$. As the initial dataset grows as new events are registered, the computational cost of the algorithm increases as well, since for new input conditions the distance from each point in the dataset must be calculated. A less computationally costly option is to adopt a classification approach, as in the case of the Nearest Centroid Classifier (NCC). Compared to the traditional k-NN algorithm, the NCC turns out to be simpler as points are not considered individually, but joined in classes. The goal of this method is to find the k nearest classes, through the minimal distance from the centroids of each class.

For each class c, the centroid μ_c is given by
$$\mu_c = \frac{1}{N_c} \sum_{1=1}^{N_c} x_i,$$
where N_c is the number of points belonging to class c and x_i is every single point containing the n input features. The corresponding output value of a class is also the average of the outputs of the N_c points of class c:
$$y_c = \frac{1}{N_c} \sum_{i=1}^{N_c} y_i$$
Let's say the Euclidean distance is chosen, then the distance of a new point X from class c is computed:
$$d(X, \mu_c) = \sqrt{\sum_{i=1}^{n} (X_i - \mu_{c,i})^2}.$$
Finally, the estimate $\hat{Y}(X)$ is given by the average of the output values of the k classes:
$$\hat{Y}(X) = \frac{1}{k} \sum_{i=1}^{k} y_{c,i}.$$
As new points are assigned to a class, the classes move and reshape because new points affect the position of the centroids. The difficulty of this method is to assign the *seeds* at the beginning, when no class is build yet, and to decide the maximum number of classes, as well as the maximum number of points in a class and the k parameter.

The ARIMA Model

In the field of time series analysis, the AutoRegressive Integrated Moving Average (ARIMA) is widely used and is composed by [33]:

- AutoRegressive (AR) component: it uses the relationship between an observation and a number of lagged observations (previous values). The autoregressive model starts from the assumption that the output variable depends linearly on its previous values and on a stochastic term;
- Integrated (I) component: it makes the time series stationary by subtracting one step to the previous one;
- Moving Average (MA) component: it uses the dependency between an observation and a residual error from a moving average model applied to lagged observations.

It is important to eliminate the non-stationarities from the time series in order to apply the ARIMA method, hence by applying the differencing component (the "integrated" part). The ARIMA model is usually denoted as $ARIMA(p, d, q)$, where p, d and q are positive integers meaning:

- p: number of lag observations included in the model (AR part);
- d: number of times that the raw observations are differenced (I part);
- q: size of the moving average window (MA part).

The objective of each of these components is to make the model fit the data as much as possible. In the case of recurrent seasonal data, the SARIMA method is used, including a seasonal component which relies on seasonal patterns, as in the case of photovoltaic solar power generation. Hence, SARIMA will be denoted as $SARIMA(p, d, q)(P, D, Q)_m$, keeping P, D and Q the same meaning of the ARIMA model, referring now to seasonal components, while m is the number of periods in each season. The general equation of an $ARIMA(p, d, q)$ model is:

$$Y(t) = c + \sum_{i=1}^{p}(\psi_i Y(t-i)) + \epsilon(t) - \sum_{i=1}^{q}(\theta_i \epsilon(t-i)).$$

Y(t) is the observed variable, namely solar power at time t, c is a constant term, ψ_i stands for the AR coefficients for lag observations up to order p, $\epsilon(t)$ is the white noise error term at time period t, θ_i accounts for the moving average coefficients for past error terms up to order q.

ARIMA is very good for short-term predictions, whilst the forecasting accuracy is not ensured for a long-term analysis. The metrics used are a wide range of defined residual errors such as MAE, RMSE, MAPE, AIC and BIC which will be later described in the next chapter, subsection 2.6. The best model is the one minimizing a trade-off among these errors.

The STL Model

At the base of the Seasonal and Trend decomposition using Loess (STL) model [34] there is the decomposition of the time series data into seasonal, trend, and remainder components:

$$Y(t) = S(t) + T(t) + R(t),$$

where Y(t) is the observed solar power production at time t, $S(t)$ represents the seasonal component, $T(t)$ represents the trend component, and R(t) represents the remainder component (irregular variations or noise), which is the last component once the former ones have been extracted.

Once the seasonality period P is determined, the first seasonal component of the series is computed:

$$S'(t) = \frac{1}{n}\sum_{i=0}^{n-1} Y(t - iP).$$

For instance, P may represent a month in the year, hence all data from month P during n years are averaged. The deseasonalized series $D(t)$ is obtained by subtracting the seasonal component from the original series

$$D(t) = Y(t) - S'(t).$$

At this point, a smoothing of $D(t)$, denoted as *Loess regression curve*, is performed [35] to remove the short-term fluctuations and extract the trend component $T(t)$. We won't dive into the details of this technique, it's just worth to mention that Loess is defined everywhere, a great feature in the presence of missing data. The detrended series $B(t)$ is obtained by subtracting the trend component from the original series

$$B(t) = Y(t) - T(t).$$

Loess smoothing is applied to the detrended series $B(t)$ to obtain the updated seasonal component $S(t)$. Finally, the residual component $R(t)$ is computed,

$$R(t) = Y(t) - T(t) - S(t).$$

The process is iterated many times until the residuals become more random and the seasonal and trend component stabilize. Once the decomposition up to time t is done, future predictions can be made by forecasting each component separately and recombining them together:

$$\hat{Y}(t + kP) = \hat{S}(t + kP) + \hat{T}(t + kP) + \hat{R}(t + kP).$$

The predictions of single components can be made in the following way:

- $\hat{S}(t + kP)$: the seasonal component is forecast by repeating the seasonal pattern for the prediction horizon;
- $\hat{T}(t + kP)$ the trend component can be extended into the future using several methods ranging from extrapolation to the ARIMA model or machine learning algorithms;
- $\hat{R}(t + kP)$: the residual component is the hardest one to predict, but some assumption on its random nature can be done, for instance considering a zero mean future residual.

1.4.3 AI Models

Finally, it's time to introduce *Artificial Intelligence* (AI) based models. The AI models include *Artificial Neural Networks* (ANNs), *Support Vector Machines* (SVMs), *Random Forest* (RF) and *Gradient Boosting* (GB). Differently from the deterministic methods where the physical relations are known and modeled, AI models are referred to as black-box models, namely we do not know what's happening inside the "box", but we see how a set of inputs is affecting the output. Unlike statistical models, AI models play a crucial role in implicitly finding complex patterns and non-linear relationships affecting the data. The method used is the same for all the above mentioned models and is structured as follows:

- data preprocessing
- feature selection
- training
- evaluation

In this work, we will focus on ANNs, a key technology in the field of *Machine Learning* (ML) (Fig 1.17) that will be described in the next chapter. A brief overview of the other AI models is now provided.

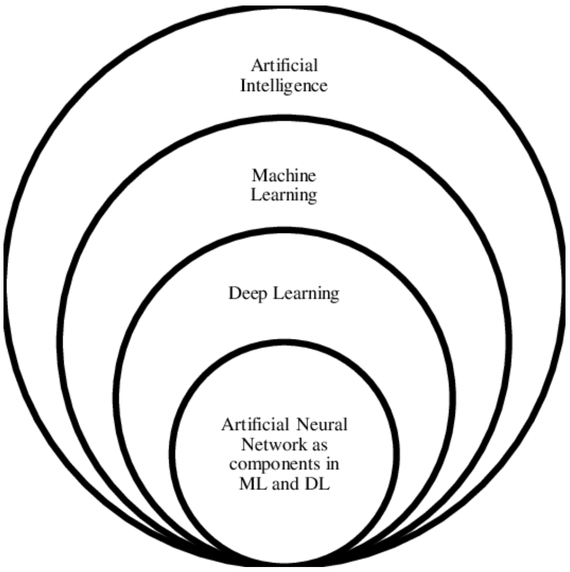


Figure 1.17. Scheme of AI, ML, DL and ANN [36].

The SVM Model

Embracing a classification/regression approach, it's worth to mention the SVMs. The goal of the *Support Vector Regression* (SVR) is to find a function that approximates the relationship between the input features and the target variable with minimal error. In solar power forecasting, SVM can be used to predict solar power generation based on past weather conditions.

The regression function is written as follows:

$$F\{f\|f(x_i = W \cdot x_i + b)\}$$

where W is the unit normal vector to the hyperplane and has dimensions K, b is the distance from the origin to the hyperplane, and x_i is the input vector. Maybe in that higher-order space, the data can be classified into linearly separeted class. The goal of SVM is to find the hyperplane that does separate the classes in the best way at minimal cost:

$$min(\frac{1}{2}||W||^2 + c\sum_{i=1}^{M}\xi_i),$$

where c is the regularization parameter of the cost function and ξ is a slack variable while M is the number of features (inputs). The regression function is subject to the boundary conditions:

$$y_i(W \cdot \phi(x_i) + b) \geq 1 - \xi_i,$$

$\phi(x_i)$ is the mapping function and can be replaced by some special kernel functions [37].

The Random Forest Model

The RF method builds a variety of decision trees during training and returns a final prediction which is the average of all individual tree predictions. A decision tree is a flowchart-like structure where each internal node represents a test on a feature, each branch represents the outcome of the test, and each leaf node represents a class label (classification) or a continuous value (regression). The tree (Figure 1.18) is built by recursively splitting the data into subsets based on the feature that results in the best separation according to some criterion (e.g., Gini impurity, entropy, mean squared error). In detail, for each of the N trees, the bootstrap sampling consists of generating a sample from the training dataset. For each node, a random subset of features is selected and

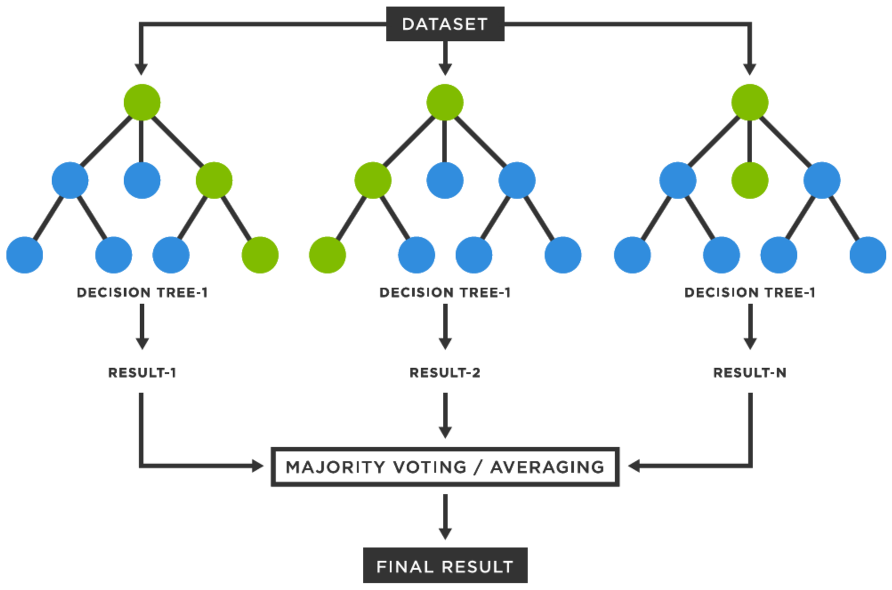


Figure 1.18. Random Forest [38].

the best feature and split point are chosen based on the selected criteria. The tree is grown to its maximum depth or until a stopping criterion is met (e.g., minimum number of samples per leaf). For classification, each tree votes for a class, and the final prediction is the class with the most votes (majority voting). For regression, each tree makes a prediction, and the final prediction is the average of all tree predictions.

The Gradient Boosting Model

When we talk about GB models, we refer to a method that builds a first algorithm for the training of a subset of the available database, followed by a second algorithm to adjust the weights and rectify the error [39].

$$\hat{y} = f(x)$$

At first, weak learners are fed with the data and an initial prediction $\hat{y_i}$ is made, then the actual output y_i is compared to the predicted output and the error is given, thus providing a metric to be minimized through the weights. The errors can be basically of 2 types, bias error and variance error. Several loss functions can be used depending on the nature of the problem, whether it is a classification or regression problem. Generally, the MSE error is considered for regression and the *loss function* is defined:

$$L_{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y_i})^2,$$

being n the number of available samples. The GB model is composed by a sequence of models where the current model tries to reduce the bias error of the previous one, hence making the prediction stronger as the number of iterations increases:

$$f_{m+1}(x_i) = f_m(x_i) + h_m(x_i),$$

where $h_m(x_i)$ is the new predictor, m refers to the current iteration

$$h_m(x_i) = y_i - f_m(x_i).$$

To compute $h_m(x_i)$, the gradient descent is used, thus providing the direction towards which the loss function will decrease,

$$-\frac{\partial L_{MSE}}{\partial f(x_i} = \frac{2}{n}(y_i - f(x_i)) = \frac{2}{n}h_m(x_i).$$

The whole sequence of weak learners makes the final predictor a strong one, resulting in a final forecast which is the sum of previous estimates

$$\hat{F}(x) = \sum_{i=1}^{m} \gamma_m h_m(x) + c.$$

Now, each single prediction can be written as

$$f_0(x) = arg\min{}_{\gamma} \sum_{i=1}^{n} L(y_i, \gamma),$$

and the m estimate is given by

$$f_m(x) = f_{m-1}(x) + (arg\min{}_{h_m} \sum_{i=1}^{n} L(y_i, f_{m-1}(x_i) + h_m(x_i))(x).$$

Since the optimization problem is computationally infeasible, instead of searching for the best h_m that minimizes the loss function, a local minimum is found by moving by small quantities in the negative direction of the gradient of the loss function.

Several techniques starting from the GB algorithm have been developed to improve it, giving rise to a wide range of models such as the *Extreme Gradient Boosting* (XGBoost), *Light Gradient Boosting Machine* (LightGBM), CatBoost *Histogram-based Gradient Boosting* (HistGB), *Natural Gradient Boosting* (NGBoost), *Stochastic Gradient Boosting* (SGB) and so on.

Chapter 2

Artificial Neural Networks

2.1 Historical Notes on Neural Networks

In general, ANNs have been conceived as an inspiration by the human brain's neural networks, which they are called after. Thus, new information and any kind of knowledge coming from neurology has been inspiring the birth and progress of ANNs. Neurologists study how neurons in the brain process information, learn, and adapt. Prompted by the challenge to simulate and reproduce biological neurons, the first neuron models were born in the early '40s by a collaboration that would involve neuroscientists and logicians too, such as Warren McCulloch and Walter Pitts [40]. In 1958 Frank Rosenblatt, starting from the McCulloch–Pitts' model of a neuron, introduced the *perceptron*, the first artificial neural network that could learn binary classification tasks [41]. Later in the '70s and the '80s no great interest was turned to the subject, still some remarkable books were published as for the case of "Perceptrons" based on Rosenblatt's previous research. The development of *Multi-Layer Perceptrons* (MLPs) idea was taking place and so the backpropagation algorithm. It was just in the '90s that first commercial applications based on neural nets were used, ranging from speech recognition to early forms of data mining. In 2006, Geoffrey Hinton and his colleagues introduced *Deep Belief Networks* (DBNs). Their work showed that training deep networks was feasible and effective. From the early theoretical models to the modern deep learning techniques, ANNs have transformed into powerful tools driving advancements in AI and ML across numerous domains. Today the applications are countless, as the adaptability of network architectures to real cases is high and generally efficient.

2.2 The Perceptron

The perceptron is the simplest model of a neural network. It was firstly conceived as an easy algorithm able to change its parameters in order to adapt the inputs to a desired output. In other words, the perceptron was among the first examples of ML and cleared the way for future technologies based on artificial intelligence.

The perceptron was born as the model of a non-linear neuron able to classify patterns, known to be linearly separable, by adjusting some weights and bias [42]. Given an input vector $\mathbf{x}$ containing n features $x_1, x_2, ..., x_n$, the perceptron equation linking the input features to a single binary output y is the following:

$$y(\mathbf{x}) = \theta(\mathbf{x} \cdot \mathbf{w} + b), \tag{2.1}$$

where $\mathbf{w}$ is the weights vector, b is the bias and $\theta(\dot{})$ is the Heaviside step-function used as a predictor for the classification, defined as follows

$$\theta(z) = \begin{cases} 1, & \text{if } z \geq 0 \\ 0, & \text{if } z < 0 \end{cases}$$

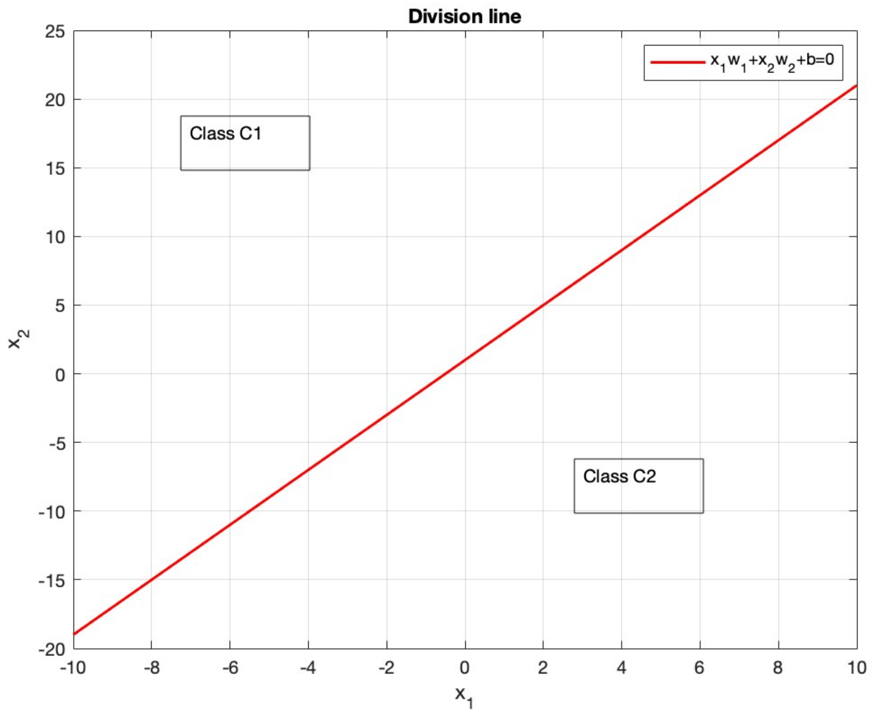


Figure 2.1. Division line.

The goal of this predictor is to classify whether a generic input belongs to class C_1 or C_2, namely the associated output is 1 or 0. In the simplest form of the perceptron, there are two decision regions corresponding to the classes separated by a hyperplane; it actually corresponds to a division line in the case of 2 input features, and to a hyperplane in a higher dimension space when the number of features is greater than 2. Its equation is simply defined by the argument of the predictor:

$$\mathbf{x} \cdot \mathbf{w} + b = 0. \tag{2.2}$$

Let us consider the easiest case when there are just 2 input features x_1, x_2, hence the division line is defined as:

$$x_1w_1 + x_2w_2 + b = 0.$$

Figure 2.1 shows the case of a 2 order space with $w_1 = 2$, $w_2 = -1$ and $b = 1$. We can name the region above the line as class C_1 and the one below the line as class C_2. To train the perceptron to properly classify, some data must be filled in so the perceptron learns the association of labeled outputs with corresponding inputs. In other words, by receiving some labeled data, the perceptron will adjust the weights w_1,w_2,...w_n of the n features and the bias b in a way that minimizes some error metrics used as a reference through an error-correction approach. Once the weights and bias that separate classes in the "best way" are found, new input data can be fed into the perceptron for classification. Namely, the classifier $\theta()$ gets the hyperplane equation as an input and returns values associated with distinct classes. In the upper example, an input value $z_1 = z_{1,1}w_1 + z_{1,2}w_2 + b$ greater than 0 corresponds to a point which is located above the division line, hence it will belong to class C_1 as $\theta(z_1 \geq 0) = 1$, while an other point $z_2 = z_{2,1}w_1 + z_{2,2}w_2 + b$ smaller than 0 will return $\theta(z_2 < 0) = 0$, which corresponds to class C_2.

So far we have focused on the implementation of the perceptron for a binary classification, starting from the assumption that the classes are linearly separable and that the weights and bias could be obtained somehow. It's important to mark that due to these initial conditions, the single perceptron shows limitations, but it's the first brick for the construction of an ANN.

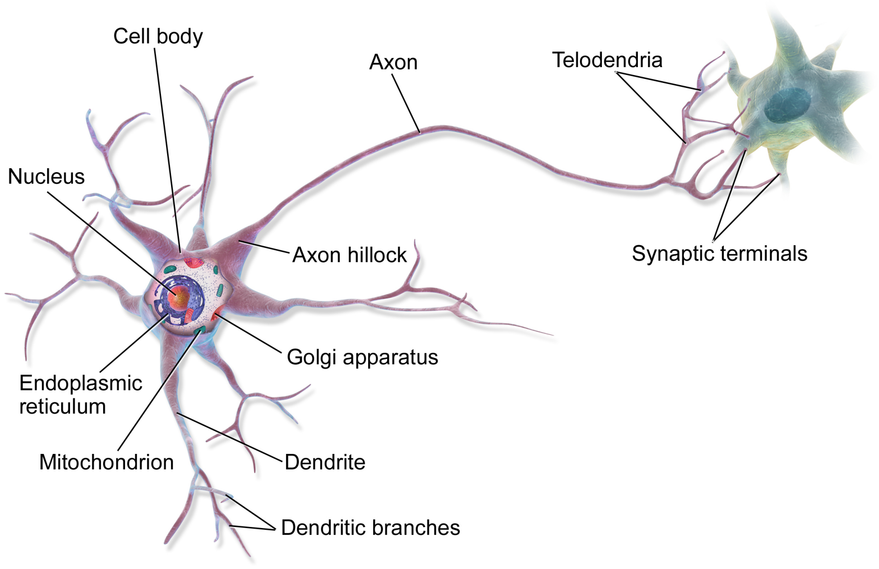


Figure 2.2. Activation of a neuron [44].

2.3 Artificial Neural Network Structure

ANNs can be used not only for classification, but for regression and prediction, too. This means that the output of the net won't be just a numeric label standing for a class, but a real value returned by a real function. That is to say, the purpose is not anymore to generalize the classification problem by finding the weights and bias to get a classification rule, but to approximate a non-linear function able to fit the available data for training and return an accurate estimate when new input are fed into the net.

To better understand how to build the network, let's start by substituting the earlier Heaviside step-function (section 2.2) with a generic activation function $f(\cdot)$. It is called an *activation function* because it resembles the phenomenon happening at the synapses of a neuron (Figure 2.2) where electrical impulses through the dendrites provoke an action potential stimulating the share of information with the other neurons [43].

The activation function is indeed what carries non-linearity into the network. Basically, it can be any real function and its choice depends on the nature of the problem[1], hence it should be selected accordingly. Among the most used activation functions [45], there are:

- the Sigmoid function, or logistic function (Figure 2.3):

  $$\sigma(x) = \frac{1}{1+e^{-x}},$$

  the exponential term in the denominator ensures that the output is always between 0 and 1, which is perfect for binary classification problems or to represent probabilities;

- the Rectified Linear Unit (ReLU) function (Figure 2.4):

  $$f(x) = max(0, x),$$

  it returns 0 for negative input and the same input value otherwise, simply put it cuts out the negative values;

[1]For instance, the use of the ReLU function for the prediction of solar power is recommended in the output layer, as the output value is never negative in the actual generation of PV energy.

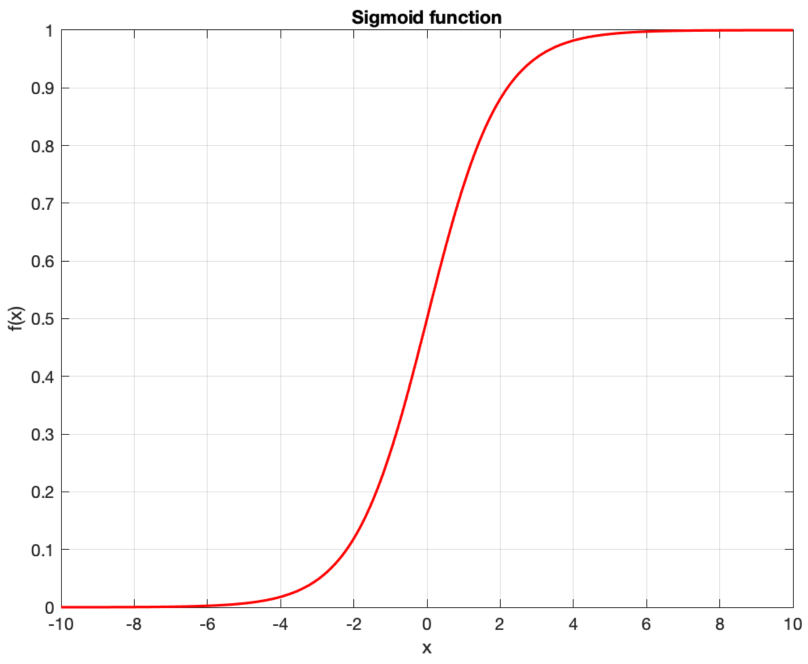


Figure 2.3. Sigmoid function.

- the Hyperbolic tangent (Figure 2.5):

$$f(x) = tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}},$$

similar to the sigmoid function, it returns values in the range between -1 and 1, a characteristic that might be exploited whether the model benefits itself from negative and positive contributes.

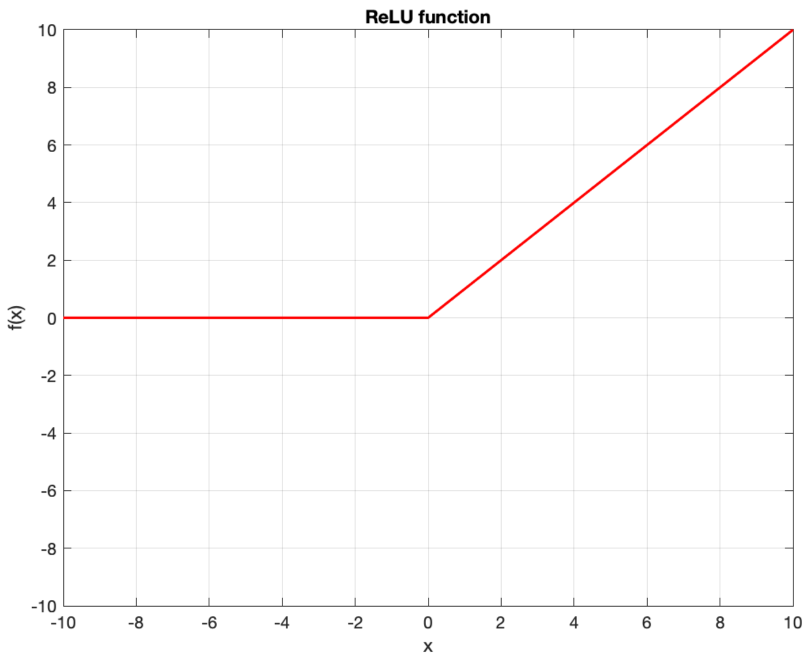


Figure 2.4. ReLU function.

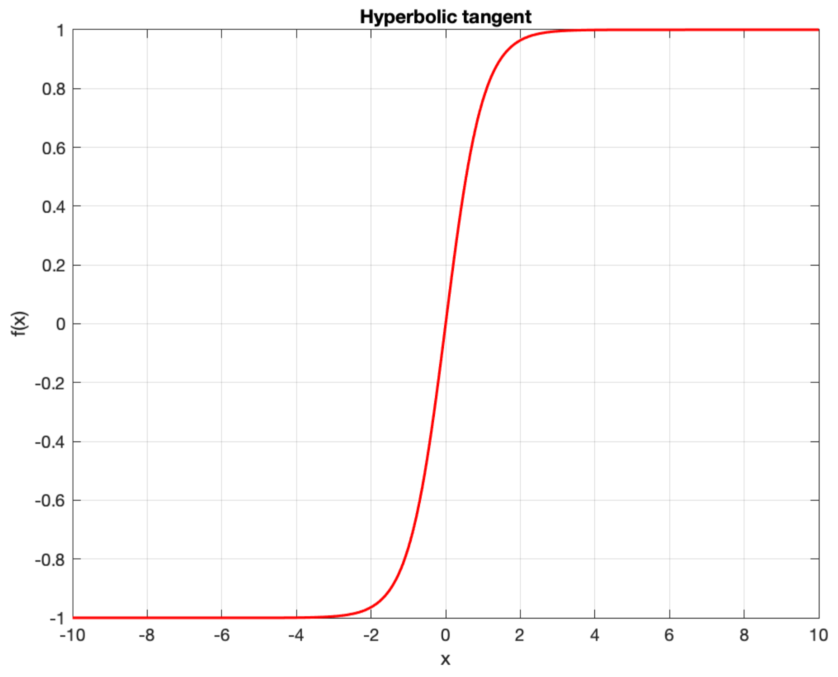


Figure 2.5. Hyperbolic tangent function.

The commonly implemented diagram for visualization of a simple neuron is showed in Figure 2.6, while the detailed diagram showing the math behind, namely passing the first member of equation (2.2) as the argument of the activation function $f(\cdot)$, is shown in Figure 2.7 and in particular:

$$y = f(w_1 x_1 + w_2 x_2 + b). \tag{2.3}$$

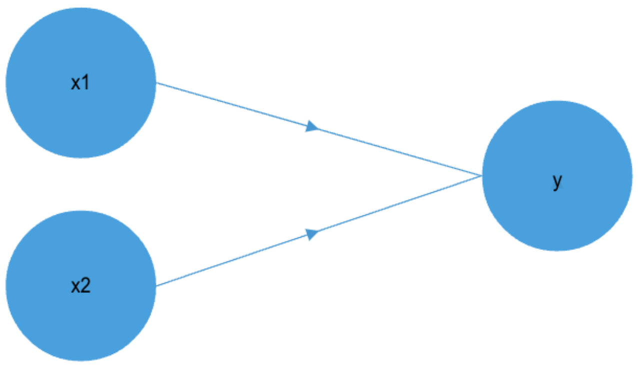


Figure 2.6. Simplified diagram of a neuron.

Now that the structure and functioning of a single neuron has been shown, the next step to delve into ANN is to join several neurons together. For lexical purposes, the definition of a *layer* is needed: whereas it's trivial to understand that the *input layer* corresponds to the aggregation of all the input features that will sum up through the weights and bias to be passed into the activation function and that the *output layer* is the one corresponding to the output(s), the

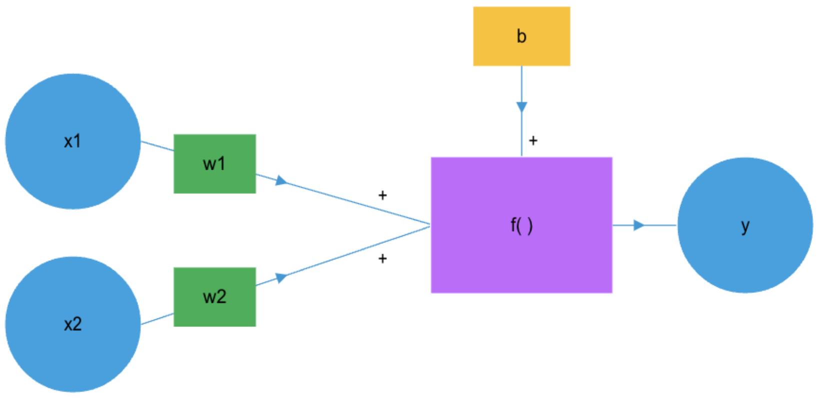


Figure 2.7. Detailed diagram of a neuron.

Figure 2.8 might result useful to visualize a *hidden layer*. In this case, there is only one hidden layer with 3 neurons h_1, h_2 and h_3 between the 4 features input layer, composed by neurons x_1, x_2, x_3 and x_4, and the output layer composed by 2 neurons y_1 and y_2. When moving from the hidden to the output layer, the values produced by neurons h_1, h_2 and h_3 are treated the same way as the input features. Indeed, they are the result of extrapolating rather infrequently trivial characteristics from the input data and model them through non linearity to increase the chance of extracting even more not evident features to pass to the next layers. Depending on the number of hidden layers, the *depth*[2] of the model is determined [46], while its *width* depends on the dimensionality of the hidden layers (in this case 3 neurons).

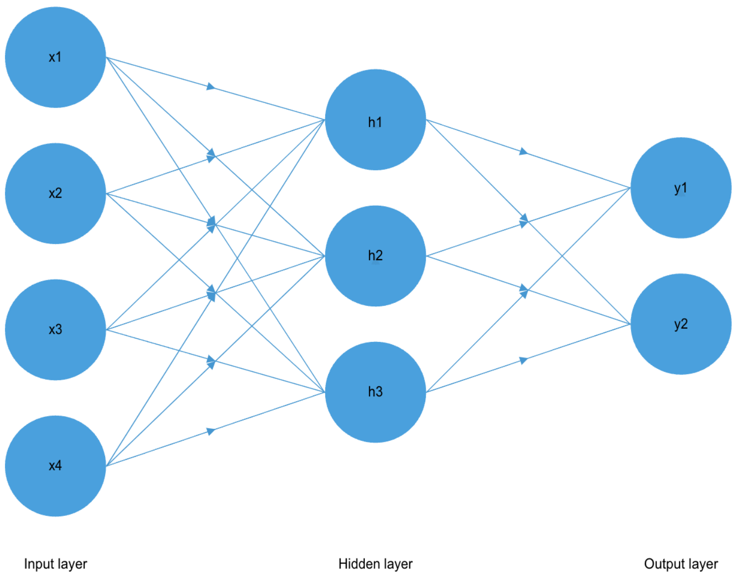


Figure 2.8. Layers of an ANN.

[2]The term used for *Deep learning* arose from this terminology

It's important to notice that, in general, there is a connection line between every neuron of one layer with each neuron of the next layer. From the perspective of a neuron there is a connection line coming from all the neurons of the previous layer. With only these types of connections, the net is called a *Feedforward Neural Network*(FNN), or MLPs, as it is only traversable from the left to the right. There are other scenarios where some feedback connections are introduced, linking some information to go back to the left; in this case, the proper name is *Recurrent Neural Network* (RNN).

2.4 Forward Propagation and Backward Propagation

With reference to Figure 2.8, we can expect that some inputs will be introduced in the neural network to eventually produce the outputs. The way information is carried from the left to the right is called *forward propagation* [47]. It should be meant as a chain in which at each layer, except the input layer, and for each neuron, the equation (2.3) of a single neuron is computed. A new notation is needed for both the weights and bias, as well as the activation functions and the neurons of the net:

- x_1,x_2,...,x_n or in general all the elements of the input layer represent the n input features;
- the name of the neuron is put on top of it, remarking that each one of them should be seen as the as the blue circle plus the connections that lead to it:
  - circles containing h_1,h_2,...,h_m (one hidden layer) or in general $h_{i,j}$ - where i and j respectively represent the neuron number $i \in m_1$ belonging to the hidden layer number $j \in m_2$ (multiple hidden layers) - are the neurons of the hidden layers,
  - circles containing y_1,y_2,...,y_k are the neurons of the output layer;
- weight $w_{i,j}$ refers to the weight of feature x_j to be sent to h_i, for instance weight $w_{1,2}$ will be multiplied by x_2 and added to the other contributions ($w_{1,1}x_1$, $w_{1,3}x_3$ and $w_{1,4}x_4$) plus the bias $b_{1,1}$ as the argument of the activation function $f_{1,1}(\cdot)$ which will return the value h_1 (Figure 2.9);
- weight $W_{i,j}$ refers to the weight of feature h_j to be sent to y_i, for instance weight $W_{2,3}$ will be multiplied by h_3 and added to the other contributions ($W_{2,1}h_1$ and $W_{2,2}h_2$) plus the bias $b_{2,2}$ as the argument of the activation function $f_{2,2}(\cdot)$ which will return the value y_2. In the case of multiple hidden layers, new letters should be used to refer to the weights, excluding w and W which are already taken;
- bias $b_{i,j}$ refers to the bias associated to neuron number i belonging to the layer j (excluding the input layer), in fact in the previous examples $b_{1,1}$ and $b_{2,2}$ are associated respectively to neurons h_1 and y_2, while bias $b_{3,1}$ is associated to neuron h_3;
- activation function $f_{i,j}(\cdot)$ refers to the activation function of neuron number i belonging to the layer j (excluding the input layer), in fact in the previous examples $f_{1,1}(\cdot)$ and $f_{2,2}(\cdot)$ are associated respectively to neurons h_1 and y_2, while $f_{1,2}(\cdot)$ is associated to neuron y_1;
- y_1,y_2,...,y_k represent the result of applying the neuron equation to each neuron of the output layer and get the final k outputs.

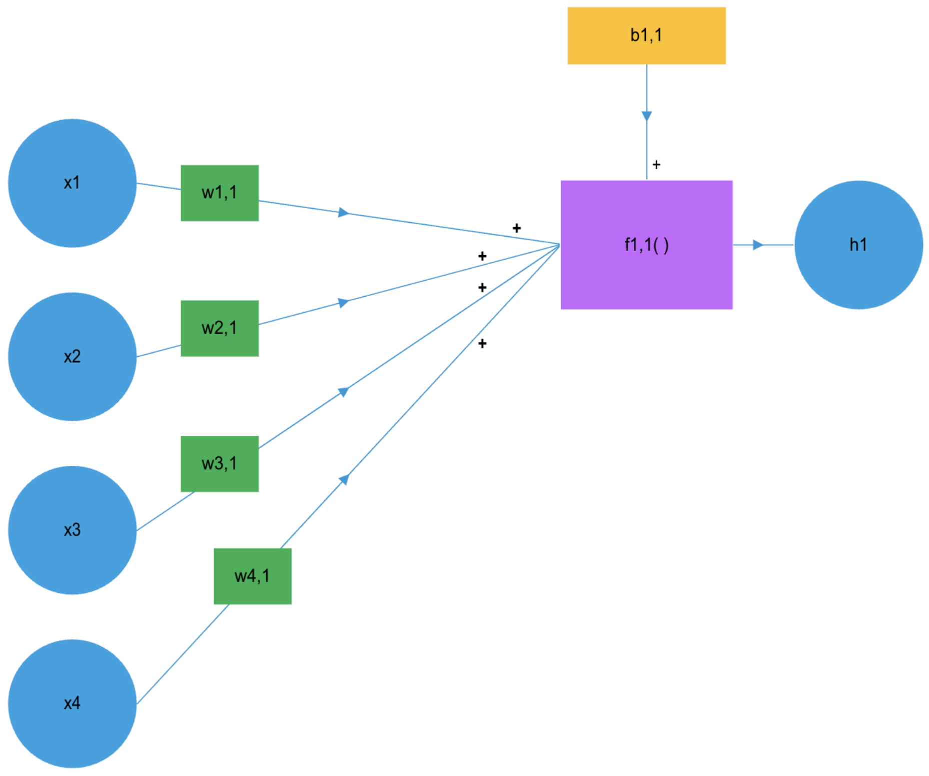


Figure 2.9. Weights and bias of neuron h_1.

It's interesting to notice that an actual mathematical chain can be obtained, as the outputs can be seen as a composition of functions and make it possible to get back to the inputs. This approach is what is known as *backward propagation* or just *backpropagation*, its meaning will get clearer very soon as the subject of minimizing the Loss function is addressed (section 2.5). For now, let's try to find some relation between the inputs and the outputs which can be written in a compact format. Trivially, the output of neuron h_1 is computed as:

$$h_1 = f_{1,1}(w_{1,1}x_1 + w_{1,1}x_2 + w_{1,1}x_3 + w_{1,1}x_4 + b_{1,1}).$$

Accordingly, the outputs of neurons h_2 and h_3 will be:

$$h_2 = f_{2,1}(w_{2,1}x_1 + w_{2,2}x_2 + w_{2,3}x_3 + w_{2,4}x_4 + b_{2,1}),$$
$$h_3 = f_{3,1}(w_{3,1}x_1 + w_{3,2}x_2 + w_{3,3}x_3 + w_{4,3}x_4 + b_{3,1}).$$

In the same way, we can now compute the final outputs y_1 and y_2:

$$y_1 = f_{2,1}(W_{1,1}h_1 + W_{2,1}h_2 + W_{3,1}h_3 + b_{2,1}),$$
$$y_2 = f_{2,2}(W_{2,1}h_1 + W_{2,2}h_2 + W_{2,3}h_3 + b_{2,2}).$$

Once these equations have been written, they can be compacted in a matrix notation, which looks easier to read. Let's express the input features, the outputs and the intermediate values in the vector form:

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix}, \quad \mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}, \quad \mathbf{h} = \begin{pmatrix} h_1 \\ h_2 \\ h_3 \end{pmatrix}.$$

Let's now write the corresponding weights and bias matrices:

$$\mathbf{w} = \begin{pmatrix} w_{1,1} & w_{1,2} & w_{1,3} & w_{1,4} \\ w_{2,1} & w_{2,2} & w_{2,3} & w_{2,4} \\ w_{3,1} & w_{3,2} & w_{3,3} & w_{3,4} \\ w_{4,1} & w_{4,2} & w_{4,3} & w_{4,4} \end{pmatrix}, \quad \mathbf{b_1} = \begin{pmatrix} b_{1,1} \\ b_{2,1} \\ b_{3,1} \end{pmatrix}, \quad \mathbf{W} = \begin{pmatrix} W_{1,1} & W_{1,2} & W_{1,3} \\ W_{2,1} & W_{2,2} & W_{2,3} \end{pmatrix}, \quad \mathbf{b_2} = \begin{pmatrix} b_{1,2} \\ b_{2,2} \\ b_{3,2} \end{pmatrix}.$$

The last element to design are the activation functions pseudo-vectors:

$$\mathbf{f_1}(\cdot) = \begin{pmatrix} f_{1,1}(\cdot) \\ f_{2,1}(\cdot) \\ f_{3,1}(\cdot) \end{pmatrix}, \quad \mathbf{f_2}(\cdot) = \begin{pmatrix} f_{2,1}(\cdot) \\ f_{2,2}(\cdot) \\ f_{2,3}(\cdot) \end{pmatrix}$$

By obtaining the resulting vector of the row-column product between the weight matrix $\mathbf{w}$ and the features vector $\mathbf{x}$ and adding the bias contribution through the bias vector $\mathbf{b_1}$, the activation functions vector $\mathbf{f_1}(\cdot)$ can be applied to obtain the hidden layer vector $\mathbf{h}$:

$$\mathbf{h} = \mathbf{f_1}(\mathbf{w} \cdot \mathbf{x} + \mathbf{b_1}).$$

Similarly, the output vector $\mathbf{y}$ is obtained:

$$\mathbf{y} = \mathbf{f_2}(\mathbf{W} \cdot \mathbf{h} + \mathbf{b_2}).$$

Finally, we can see a direct relation between the inputs and the outputs, which results in a composition of functions:

$$\mathbf{y} = \mathbf{f_2}(\mathbf{W} \cdot \mathbf{f_1}(\mathbf{w} \cdot \mathbf{x} + \mathbf{b_1}) + \mathbf{b_2}).$$

2.5 The Training Phase and The Loss Function

In this section, the method used to get to the weights and biases is presented. To this regard, it's important to introduce the concept of *fitting*, which is basically how much the final relation between inputs and outputs, obtained by deriving the values of the weights and biases of the neural net and choosing the activation functions, fits the labeled outputs associated with the inputs. In general, a database $D(\mathbf{x}, \mathbf{y})$ is always needed and depending on an arbitrary decision, it is split [48] in varying percentages[3] into:

- the *training set*, in turn composed of:
  - a subset used to train the neural network. Labeled outputs $\mathbf{y}$ are fed into the neural net as well as the inputs $\mathbf{x}$ they are associated to. At this stage, the network isn't trained yet, which means that the values for the weights and bias are unknown;
  - a subset called the *validation set* used to tune the model and validate its performance during training. At this stage, the weights and bias are being computed and updated according to some error metrics, namely the Loss function, to be minimized through backpropagation. Eventually, some final values will be kept to step to the test phase;
- the *test set* used evaluate the final model's performance. It is made of new data not used during training which are fed into the net as input to compare, once again, their corresponding produced output $\hat{\mathbf{y}}$ with the actual known outputs $\mathbf{y}$.

This approach is both implemented for classification and regression/prediction. In the former case the outputs will be generally integer numbers standing for distinct classes, while in the latter case a real value is generated. To delve into the mathematics behind it, let's think about the *Loss function* as the Loss that is obtained by classifying or predicting a generic output $\hat{y}$ in comparison with the actual output y. Hence the aim of the training phase is to minimize the Loss function, that is to say the task of finding a proper set of characteristic parameters of an ANN is reduced to an optimization problem.

[3]There are numerous available methods for partitioning the initial dataset. The splitting should be a good compromise with the goal of avoiding a bad fitting in the test stage generally related to an overfitting of the training subset.

2.5.1 Binary Classification

Section 2.2 discussed the case of binary classification using a single neuron, which is done by selecting the Heaviside step-function. The sign function could be also used, which is defined as:

$$\gamma(z) = sign(z) = \begin{cases} 1, & \text{if } z \geq 0 \\ -1, & \text{if } z < 0 \end{cases}$$

Let's define the *score* of a binary classification problem as the vector product $\mathbf{w}\cdot\mathbf{x}$, which represents how confident we are predicting $+1$ [49]. The *margin* is defined as $(\mathbf{w}\cdot\mathbf{x})y$, which tells us how correct we are. The *zero-one Loss* is then defined as:

$$Loss_{0-1}(\mathbf{x}, y, \mathbf{w}) = \mathbf{1}[\gamma(x) \neq y] = \mathbf{1}[(\mathbf{w}\cdot\mathbf{x})y \leq 0],$$

where the operator $\mathbf{1}[k]$ returns $+1$ when k gets a TRUE logic value $(+1)$ and returns 0 otherwise, namely k gets a FALSE logic value (-1). In fact, we expect the Loss function to be max $(+1)$ when the classified output $\hat{y} = \gamma(\mathbf{w}\cdot\mathbf{x})$ differs from the actual labeled output y and to be 0 when they coincide. Indeed, a value of the zero-one Loss equal to $+1$ corresponds to a negative margin: reminding that the vector product $\mathbf{w}\cdot\mathbf{x}$ can be equaled to 0 to get the division line equation $(\mathbf{w}\cdot\mathbf{x} = 0)$, positive values will correspond to points located in the region pointed by the weight vector (graph 2.1), while negative values will correspond to points located in the opposite region. As the sign function is applied, positive values will turn into a $+1$ and negative values into a -1. It's trivial to draw conclusions: when the score and the actual output differ in sign, namely their product (the margin) is negative, the zero-one Loss function is max, which can be easily seen in figure 2.10.

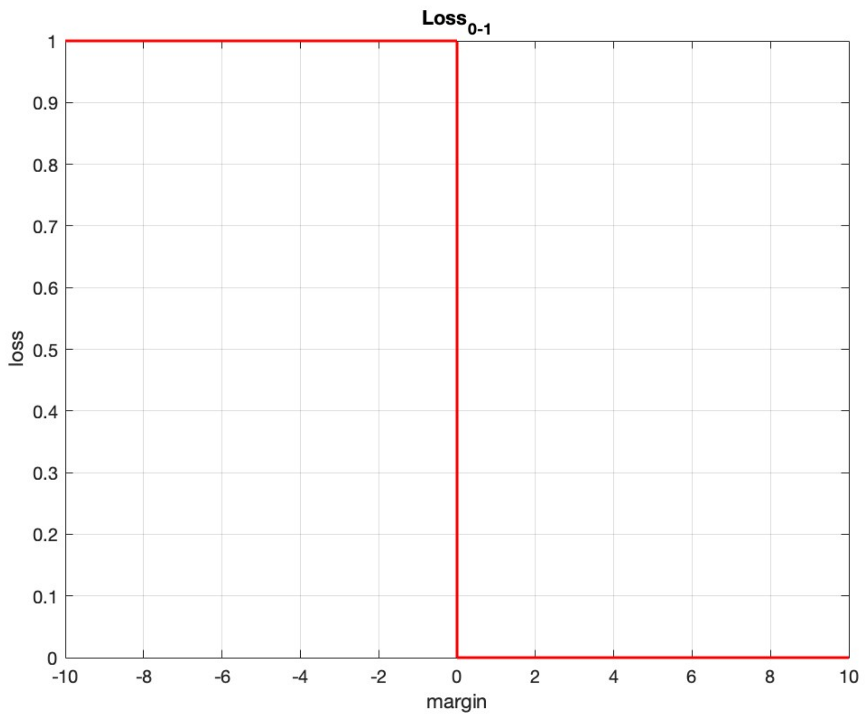


Figure 2.10. Relationship between the margin and the Loss function.

Let's consider the following example, where the weights vector $\mathbf{w}$ and a triplet of feature vectors $\mathbf{input(x)}$ are given:

$$\mathbf{w} = [2, -1], \quad \mathbf{input(x)} = \{[2,0], [0,2], [2,4]\}.$$

The vector **output**($\hat{y}$) corresponding to the triplet of outputs is computed and returns

$$\begin{aligned} output_1 &= sign(\mathbf{w} \cdot input_1) = sign([2,-1] \cdot [2,0]) = sign(+4) = +1, \\ output_2 &= sign(\mathbf{w} \cdot input_2) = sign([2,-1] \cdot [0,2]) = sign(-2) = -1, \\ output_3 &= sign(\mathbf{w} \cdot input_3) = sign([2,-1] \cdot [2,4]) = sign(0) = +1, \end{aligned}$$

hence,

$$\mathbf{output}(\hat{y}) = \{+1, -1, +1\}.$$

Imagine we dispose of the labeled outputs **output**(y) = $\{+1, +1, -1\}$, namely it's possible to see and inspect whether the classifier returns a value which may coincide with the actual output. The Loss function is used as a measure and returns 0 for $output_1$ as the predicted and actual output coincide and have value $+1$. Likewise, the Loss will return value $+1$ for $output_2$ and $output_3$ as the classifier returns values that differ from the actual ones. The very important ability of artificial neural networks is to exploit information given by the Loss function in order to adjust the weights and bias and make the predicted outputs more similar to the actual ones, that is in the case of binary classification to get to a scenario where only a few or no outputs will correspond to a positive Loss. The way it is capable of doing so is later described in the optimization problem in section 2.5.3.

2.5.2 Regression and Prediction

Besides classification purposes, ANNs are commonly implemented for regression and prediction. Instead of returning a class, here the outputs are real numbers returned by real functions. It is no longer reasonable to use the zero-one Loss function to quantify how close is the prediction with respect to true known values. In the case of linear regression, the activation function is simply the identity function $f(x) = x$, i.e. it returns the same values is receives. The prediction is then simply:

$$\hat{y} = \mathbf{w} \cdot \mathbf{x}.$$

In the literature, it is common to refer to the above product as the *score*, while the *residual* is defined as the difference $(\mathbf{w} \cdot \mathbf{x}) - y$ or simply $score - y$ and quantifies the amount by which the prediction $\hat{y}$ overshoots the target y. The *squared Loss* is introduced:

$$Loss_{squared} = (\hat{y} - y)^2 = ((\mathbf{w} \cdot \mathbf{x}) - y)^2 = residual^2.$$

The Loss is a quadratic function of the residual, hence it has an absolute minimum value centered at 0 residual (Figure 2.11). Another measure is the absolute deviation Loss function which is defined as:

$$Loss_{absdev} = |(\mathbf{w} \cdot \mathbf{x}) - y| = |residual|.$$

Let us consider an example: a train of inputs **input**($\mathbf{x}$) is given, as well as known values for the outputs **output**($\mathbf{y}$). The weight vector $\mathbf{w}$ has not been computed yet, hence we gather the available data:

$$\mathbf{w} = [w_1, w_2], \quad \mathbf{input}(\mathbf{x}) = \{[1,0], [1,0], [0,1]\}, \quad \mathbf{output}(\mathbf{y}) = \{2, 4, -1\}.$$

The quadratic Loss is used for each pair that links an input with its output and returns:

$$\begin{aligned} Loss(input_1, output_1, \mathbf{w}) &= ([w_1, w_2] \cdot input_1 - output_1)^2 = ([w_1, w_2] \cdot [1,0] - 2)^2 = (w_1 - 2)^2, \\ Loss(input_2, output_2, \mathbf{w}) &= ([w_1, w_2] \cdot input_2 - output_2)^2 = ([w_1, w_2] \cdot [1,0] - 4)^2 = (w_1 - 4)^2, \\ Loss(input_3, output_3, \mathbf{w}) &= ([w_1, w_2] \cdot input_3 - output_3)^2 = ([w_1, w_2] \cdot [0,1] - (-1))^2 = (w_2 + 1)^2. \end{aligned}$$

This example is good to introduce the subject of the next section, that is how to find the weight components. Remembering that the goal is to minimize the distance between the predictions and the true actual values, the elements w_1 and w_2 should be assigned in such a way as to reduce the

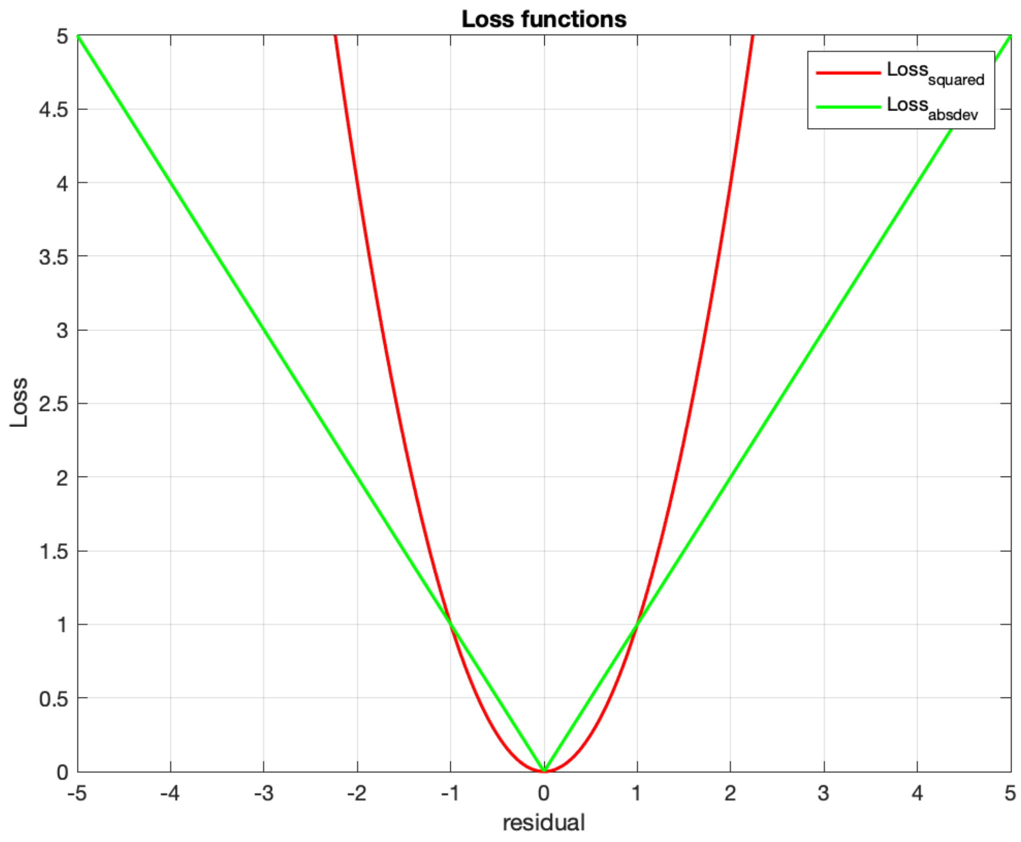


Figure 2.11. Relationship between the residual and the Loss functions.

losses. The most intuitive way to accomplish the task is to add together all the quadratic losses in a single metric, which we define as the $TrainLoss(\mathbf{w})$:

$$TrainLoss(\mathbf{w}) = \frac{1}{D_{train}} \sum_{(x,y)\in D_{train}} (\mathbf{w}\cdot\mathbf{x} - y)^2,$$

where D_{train} stands for the dimension of the losses train, namely the number of available pairs input-output. In our example, we will get:

$$TrainLoss(\mathbf{w}) = \frac{1}{3}((w_1 - 2)^2 + (w_1 - 4)^2 + (w_2 + 1)^2)$$

. Being the $TrainLoss$ only made of quadratic functions of w_1 and w_2, we know for sure that the minimums values for w_1 and w_2 are absolute in the real domain. They can be easily obtained by computing the partial derivatives of the $TrainLoss$ with respect to each component:

$$\frac{\partial TrainLoss}{\partial w_1} = 2(w_1 - 2) + 2(w_1 - 4) = 4w_1 - 12,$$
$$\frac{\partial TrainLoss}{\partial w_2} = 2(w_2 + 1) = 2w_2 + 2.$$

The points where the partial derivatives equal 0 correspond to the absolute minimums of the $TrainLoss$:

$$w_1 = 3, \quad w_2 = -1.$$

For this 2-dimensional example - notice that the dimension is given by the number of input features, that is the length of each input vector - a 3D graph showing the $TrainLoss$ can be obtained (Figure 2.12). The analysis by a quick glance is not trivial, indeed some tools which allow the user to move around the function and inspect it make it easier to work with. For us,

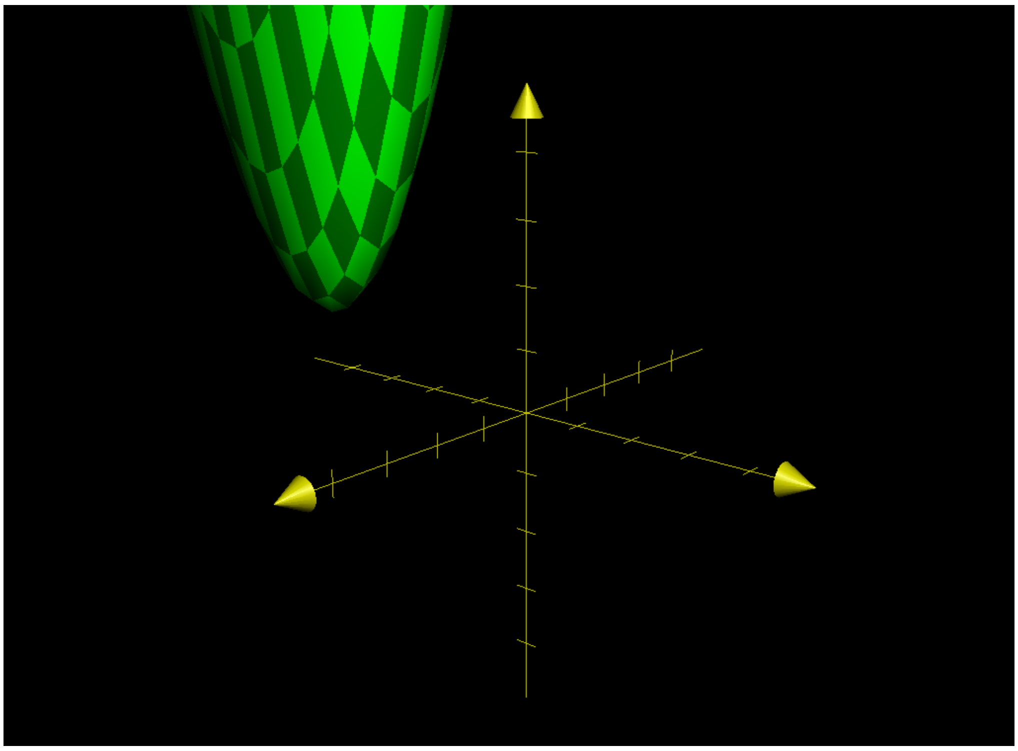


Figure 2.12. 3D graph of the $TrainLoss$.

it is sufficient to confirm that the vector $\mathbf{w} = [3, -1]$ is the one that minimizes the $TrainLoss$ at the absolute minimal value:

$$TrainLoss([3, -1]) = (3-2)^2 + (3-4)^2 + (-1+1)^2 = 2.$$

With respect to Figure 2.12, the arrow pointing to the left is positioned on the axis w_1, where the minimum is located just above the third tick mark and aligned with the first negative notch on the axis w_2, where the arrow points to the right. The value of the $TrainLoss$ at this minimum is 2, which corresponds to the second tick mark on the vertical axis.

2.5.3 The Optimization Problem: GD and SGD

The previous subsections have presented the case of binary classification and the regression problem with a focus on the Loss function. It is clear that the final goal of the training phase is to minimize the Loss, which is a function of known inputs and outputs and the unknown weight vector $\mathbf{w}$. In other words, the objective of the training is to look for $\mathbf{w}$ that corresponds to a minimum value of the $TrainLoss$, which is now defined as:

$$TrainLoss(\mathbf{w}) = \frac{1}{D_{train}} \sum_{(x,y) \in D_{train}} Loss(x, y, \mathbf{w}).$$

The $TrainLoss$ has been generalized with respect to the one used in the example in subsection 2.5.2 and it is now referred to a generic Loss function chosen for the application; D_{train} is the dimension of the weight vector. Hence, the optimization problem to solve is as follows:

$$\min_{\mathbf{w} \in R^d} TrainLoss(\mathbf{w}).$$

One of the most efficient and powerful way to find the minimum (or maximum) of a function is the use of its gradient: the gradient of a function points in the direction of the steepest increase in the function's value. By moving in the opposite direction, we can decrease the function's value most rapidly, leading us toward a local minimum. We will use the notation $\nabla(\cdot)$ to refer to the gradient of a function, i.e. $\nabla_{\mathbf{w}} TrainLoss(\mathbf{w})$ is the direction that increases the Loss the most. The negative sign will allow us to "walk" in the negative direction and the iterative algorithm for *Gradient Descent* (GD) is as follows:

```
w = [0,...,0]
for t=1,...,T:
  w = w - eta*Gradient(TrainLoss)
```

At each iteration t up to the final iteration T, the weight vector w which has been initialized as a null vector is updated by moving in the direction of the negative gradient by a quantity "eta" η:

$$\mathbf{w} = \mathbf{w} - \eta \nabla_{\mathbf{w}} TrainLoss(\mathbf{w}).$$

The meaning of η is to provide a metric on the rate of descent along the gradient, indeed it is called the *step size.* A value of η close to 1 may result aggressive as $\mathbf{w}$ would change drastically, potentially overshooting the minimum point. On the other hand, a value of η close to 0 may result in a more stable but slower descent. To this regard, one strategy is to keep the value of η constant at low values, another option is make its value variable. After all, it is reasonable to expect that as the algorithm iterates more, it should get closer to a solution, hence a rough descent may be allowed initially, with the process becoming slower and more stable afterward:

$$\eta = \frac{1}{\sqrt{t}},$$

being t the current iteration - the value of η will decrease along to the series 1, $\frac{1}{\sqrt{2}}$,..., $\frac{1}{\sqrt{T}}$.

To make the algorithm even less computationally expensive, the gradient could be estimated using a randomly selected subset of the data instead of the entire training dataset. GD uses the entire dataset and is also called *Batch Gradient Descent* (Batch GD); a Mini-Batch GD only uses a small, randomly selected subset of the data to compute the gradient; *Stochastic Gradient Descent* (SGD) uses only one training example at a time. This means that each iteration of SGD updates the parameters based on different data points or subsets, leading to more frequent updates compared to Batch GD. The noisy updates caused by random sampling can help the algorithm escape local minima; however, this can sometimes slow down convergence. To pass from GD to SGD, the iterative algorithm is modified as follows:

```
w = [0,...,0]
for t=1,...,T:
   for (x,y) in Dtrain:
      w = w - eta*Gradient(TrainLoss)
```

It is important to point out that it is not always easy to compute $\mathbf{w}$ as it should be a good tradeoff to reduce the Loss of all the elements of the training set. Moreover, the optimization of linear functions is usually simple and fast to solve, but the same cannot be said about ANNs. So far we have dealt with basic academic examples of a simple neuron with no bias and activation function equal to the sign function for classification and the identity function for regression. Let us consider the ANN in Figure 2.13, with the activation functions h_1 and h_2 the step function $\theta(z) = \mathbf{1}[z \geq 0]$.

We would like to get a solution of the optimization problem as to reduce the squared residual Loss between the predicted output $\hat{y}$ and the actual output y. With reference to the notation used in section 2.4, we can compute the output $\hat{y}$ as the sum of the contributions coming from the hidden layers h_1 and h_2:

$$\begin{aligned}
&h_1 = \mathbf{1}[w_{1,1}x_1 + w_{2,1}x_2 + w_{3,1}x_3 + b_{1,1} \geq 0],\\
&h_2 = \mathbf{1}[w_{1,2}x_1 + w_{2,2}x_2 + w_{3,2}x_3 + b_{2,1} \geq 0],\\
&\hat{y} = W_1 h_1 + W_2 h_2 + b_{1,2}.
\end{aligned}$$

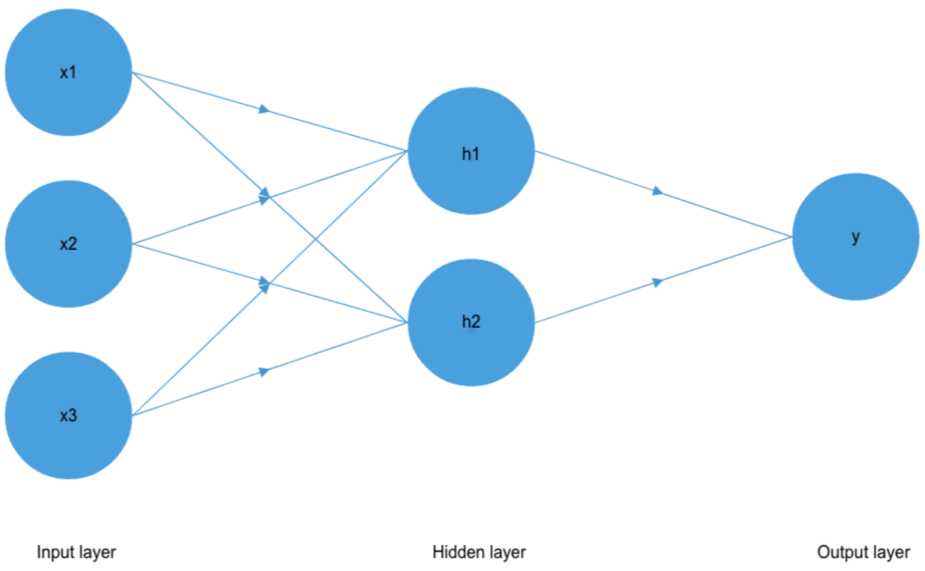


Figure 2.13. ANN with logistic activation functions.

The squared Loss is used, so that:

$$Loss(x, y, \mathbf{w}, \mathbf{W}, \mathbf{b}) = (y - \hat{y})^2 = (y - f_{\mathbf{w},\mathbf{W},\mathbf{b}}(x))^2,$$

namely,

$$f_{\mathbf{w},\mathbf{W},\mathbf{b}}(x) = \sum_{j=1}^{2} W_j \theta(\sum_{i=1}^{3} w_{i,j} x_i + b_{j,1}) + b_{1,2},$$

the *TrainLoss* is computed as:

$$TrainLoss(\mathbf{w}, \mathbf{W}, \mathbf{b}) = \frac{1}{D_{train}} \sum_{(x,y)\in D_{train}} Loss(x, y, \mathbf{w}, \mathbf{W}, \mathbf{b}).$$

The goal is once again to solve the optimization problem:

$$\min_{\mathbf{w},\mathbf{W},\mathbf{b}} TrainLoss(\mathbf{w}, \mathbf{W}, \mathbf{b}),$$

which is done by computing the gradient:

$$\nabla_{\mathbf{w},\mathbf{W},\mathbf{b}} TrainLoss(\mathbf{w}, \mathbf{W}, \mathbf{b}).$$

To compute the gradient, we must differentiate the *TrainLoss* with respect to $\mathbf{w}$,$\mathbf{W}$ and $\mathbf{b}$, i.e. compute the partial derivatives along each component. The chain rule must be used, which means to differentiate by steps linking each component to the final quantity to optimize. For example, the partial derivative of the *TrainLoss* with respect to W_2 is:

$$\frac{\partial TrainLoss}{\partial W_2} = \frac{\partial TrainLoss}{\partial Loss} \frac{\partial Loss}{\partial f_{\mathbf{w},\mathbf{W},\mathbf{b}}} \frac{\partial f_{\mathbf{w},\mathbf{W},\mathbf{b}}}{\partial W_2}.$$

Though, when we compute the partial derivative of the *TrainLoss* with respect to $\mathbf{w}$, we get in trouble with the step function, in fact it is non-differentiable at $x = 0$, due to the step itself. A common solution is to replace it with the sigmoid function:

$$\sigma(z) = (1 + e^{-z})^{-1},$$

whose derivative with respect to z is simply:

$$\sigma'(z) = \sigma(z)(1 - \sigma(z)).$$

Applying the substitution to the previous example, we are now able to compute the partial derivative of $f_{\mathbf{w},\mathbf{W},\mathbf{b}(x)}$ with respect to each weight and bias component, for instance:

$$\frac{\partial f_{\mathbf{w},\mathbf{W},\mathbf{b}}}{\partial w_{1,2}} = \frac{\partial f_{\mathbf{w},\mathbf{W},\mathbf{b}}}{\partial \sigma}\frac{\partial \sigma}{\partial w_{1,2}},$$

having set:

$$f_{\mathbf{w},\mathbf{W},\mathbf{b}} = \sum_{j=1}^{2} W_j \sigma(\sum_{i=1}^{3} w_{i,j}x_i + b_{j,1}) + b_{1,2},$$

which gives:

$$\begin{aligned}
&\frac{\partial f_{\mathbf{w},\mathbf{W},\mathbf{b}}}{\partial \sigma} = W_1 + W_2,\\
&\frac{\partial \sigma}{\partial w_{1,2}} = \frac{\partial \sigma}{\partial arg}\frac{\partial arg}{\partial w_{1,2}}, \quad arg = \sum_{i=1}^{3} w_{i,j}x_i + b_{j,1},\\
&\frac{\partial \sigma}{\partial arg} = \sigma(1-\sigma), \quad \frac{\partial arg}{\partial w_{1,2}} = x_1,
\end{aligned}$$

eventually we get:

$$\frac{\partial f_{\mathbf{w},\mathbf{W},\mathbf{b}}}{\partial w_{1,2}} = (W_1 + W_2)\sigma(\sum_{i=1}^{3} w_{i,j}x_i + b_{j,1})(1 - \sigma(\sum_{i=1}^{3} w_{i,j}x_i + b_{j,1}))x_1.$$

It is worth to note the contribution of the bias also, as for the case of $b_{1,2}$:

$$\frac{\partial f_{\mathbf{w},\mathbf{W},\mathbf{b}}}{\partial b_{1,2}} = 1,$$

which means that $b_{1,2}$ is just seen as a constant from the network, while for the case of $b_{1,1}$:

$$\frac{\partial f_{\mathbf{w},\mathbf{W},\mathbf{b}}}{\partial b_{1,1}} = (W_1 + W_2)\sigma(\sum_{i=1}^{3} w_{i,j}x_i + b_{j,1})(1 - \sigma(\sum_{i=1}^{3} w_{i,j}x_i + b_{j,1})).$$

Now the meaning of backpropagation resonates more than ever. The following summary might result useful for getting an overview of the whole method used in the training phase:

- in the forward propagation, the input data is passed through the network layer by layer, applying the weights and bias and activation functions, until it gets to the output layer. The network generates a prediction based on the current weights, which are generally initially set to small random values;
- the predicted output is compared to the actual known output using a loss function which quantifies the error in the prediction;
- in the backward propagation, the algorithm calculates the gradient of the loss function with respect to each weight in the network. This is done by applying the chain rule, which propagates the error backward from the output layer to the input layer;
- the gradients indicate how much a change in each weight would affect the Loss. These gradients are then used to update the weights in the opposite direction of the gradient.

Once the first part of the training phase has been concluded[4], the next step is to validate the model, meaning how well the model performs on unseen data. The validation set D_{val} is distinct from the training set D_{train}, though, we still know the actual desired outputs that are used for a comparison through the error metrics (next section 2.6). This is a crucial part of the model development process as it allows to fine-tune the model and make decisions about its architecture, hyperparameters and to prevent overfitting.

2.6 The Error Metrics Used for Validation and Performance Evaluation

The goal of performance evaluation is to minimize error on unseen future examples. This is achieved using a test set D_{test} that contains only inputs. Evaluation can be performed later, once the actual outputs are available, such as in time-series predictions where we must wait for the true values to be revealed. For both validation and final evaluation, some error metrics must be introduced as a reference of the accuracy of the ANN. To be computed, they usually requires a target set y_1,...,y_N used for comparison with the set of N predicted outputs $\hat{y}_1$,...,$\hat{y}_N$. In the case of data forecasting, some commonly used characteristic metrics[5] are:

- *Mean Bias Error* (MBE), which measures the average value of the errors considered keeping their sign, it is defined as:
$$MBE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i);$$
- *Mean Absolute Error* (MAE), which measures the average magnitude of the errors in a set of predictions, without considering their direction and is defined as:
$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|;$$
- *Median Absolute Error* (MAD), it is the median of the absolute errors, giving them the same weights:
$$MAD = median(|y_i - \hat{y}_i|);$$
- R^2 (Coefficient of Determination), it is a measure for the goodness of fit; higher values indicate better performance. While R-squared is a useful metric, relying on it alone can indeed lead to overfitting. Using other criteria can help create a more robust and generalizable model. This coefficient is defined as:
$$R^2 = 1 - \frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{N}(y_i - \overline{y})^2},$$
being $\overline{y}$ the average of actual values y_1,...,y_N;
- *Mean Squared Error* (MSE), it gives more weight to larger errors with respect to MAE, making it sensitive to outliers. It is defined as:
$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2;$$

[4]It is not necessarily the case that the training phase is completed only once. In general, for applications like ours in solar power forecasting, new data is continuously added to the database, which means that the training set could either grow significantly or be shifted over time. Therefore, when we say 'concluded,' we are referring to having obtained the values of the weights and biases.

[5]These metrics were collected and provided by the project tutor coordinator, beyond the scope of this work, and were shared by other students working on the same subject.

- *Root Mean Squared Error* (RMSE), it keeps the same unit as the original data, hence it results easier to be read or seen on a graph. It is defined as:

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}.$$

Depending on the nature of the problem, the choice of metrics may vary. Only those metrics that best reflect our objectives and expectations, and that are appropriate for the data and problem domain, should be selected.

2.7 Hyperparameters

Until now, we have considered the potential structures of ANNs without delving into the challenges associated with determining the number of hidden layers required, the number of neurons per layer, or the choice of activation functions. These aspects fall under the category of hyperparameters - parameters selected by the user that are not part of the mathematical optimization process of the model [50]. Unlike model parameters (such as weights and biases), which are learned during training, hyperparameters are defined before the training and are not directly learned from the data. Instead, they are tuned to optimize the performance of the network. In other words, they are crucial for reducing performance errors and mitigating the risk of overfitting. Some of them pertain to the training process itself, such as:

- the learning rate, which is the rate at which the optimization algorithm updates the weights and biases based on the error metrics;
- the batch size, which is the size of the subset used for training as well as the division of the overall set into D_{train}, D_{val} and D_{test};
- the number of epochs, where an epoch refers to one iteration, i.e., an update of weights and biases through the entire training dataset.

These parameters are usually characteristic of a learning method; hence, they are applied by default in the automated versions of optimization algorithms available in software such as MATLAB. Finding the optimal set of hyperparameters often involves experimentation through a trial-and-error approach and, depending on the complexity of the problem, is typically performed using established techniques. In the case of multiple hidden layers, grid search or random search may be considered when determining the optimal number of neurons per layer with the aim of reducing error metrics. Chapter 4 addresses the challenge of hyperparameters in the context of our specific application.

2.8 RNN and CNN

So far in Chapter 2, we have described the structure and working principle of ANNs, focusing on the simplest architecture, which is a FNN. Before proceeding with the applications, it is worth taking a brief look at *Recurrent Neural Networks* (RNNs) and *Convolutional Neural Networks* (CNNs). Both RNNs and CNNs are built upon the fundamental principles of FNNs, such as neurons, weights, and activation functions.

An RNN is a type of neural network designed to handle sequential data with significant order, such as time series, text, or audio. RNNs introduce recurrent connections to model temporal dependencies, unlike FNNs, which process inputs statically and independently [51]. RNNs have connections that loop back to previous neurons in the same layer, allowing information to persist over time steps. In an RNN, the same set of weights is applied at each time step, making them well-suited for variable-length sequences. Similar to a closed-loop control architecture, the hidden state in RNNs acts as a memory, retaining information about previous inputs. This is essential for tasks like language modeling or time-series prediction. In the application of solar power prediction, RNNs are very useful for modeling and keeping track of updating weather forecasts—imagine having not just a single forecast but a sequence of forecasts that updates as we approach the actual time of prediction.

A CNN is a type of neural network designed to process images and videos[6] [52]. CNNs consist of three distinct layers (this sequential structure may be repeated many times): the convolutional layer, the pooling layer, and the fully connected layer. In the convolutional layer, a mask filter or kernel is passed through the input frame to extract relevant features from the image. Depending on the shape of the mask (essentially a matrix of numbers representing digital levels in the color reference system, i.e., RGB), several features can be extracted, such as undulation, gradient, roughness, shape, and so on. The objective of the pooling layer is to reduce the dimension of the image analyzed in the convolutional layer. This is achieved by applying a mask to extract the maximum value (max pooling) or the average value (average pooling) from the grid covered by the kernel. The image is then flattened into a column vector (for instance, a 5x4 matrix becomes a 20x1 vector) so that it is ready to enter a feedforward neural network, where backpropagation is applied during training. In CNNs, the weights, which are tuned to detect structural features in images, are reused across different parts of the input, thus reducing the risk of overfitting.

2.9 LSTM

When dealing with RNNs, it is not uncommon to encounter issues like *gradient exploding* or *gradient vanishing* [53]. Due to the inherent nature of an RNN, which uses the same weights in the recurrent branches, when numerous recursions are made, the first input x_1 will be multiplied by the weight w_1 a number of times equal to the number of recursions in the model. Therefore, for a value of w_1 greater than 1, the gradient can easily 'explode'. On the other hand, for a value less than 1 and not even a very high number of recursions, the gradient tends to 'vanish'. For this reason, special care must be taken when using RNNs, and for our application, we will use models based on *Long-Short Term Memory* (LSTM). Its use eliminates the gradient problems just mentioned. The structure of an LSTM cell is shown in Figure 2.14..

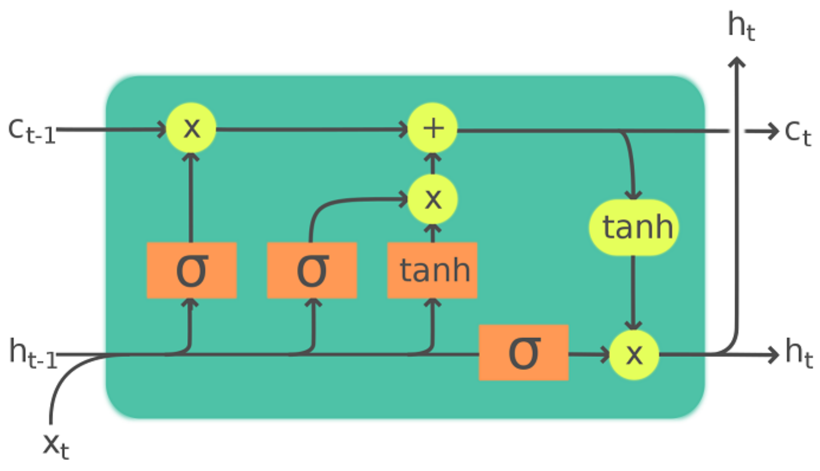


Figure 2.14. LSTM cell structure [54].

[6]Initially designed to deal with images and videos containing a significant amount of information, which underscores the necessity of reducing the dimension and complexity of the analysis, CNNs have been successfully applied to other types of data that have spatial or local structures. This is the case with Text Classification, Named Entity Recognition (NER), and Sentence Modeling, as well as Speech Recognition.

The cell is composed of two paths: the long-term memory (LTM, from c_{t-1} to c_t) and the short-term memory (STM, from h_{t-1} to h_t). Several percentiles are computed to determine how much information to retain for both paths. From left to right, the input x_t and the STM (h_{t-1}) from the previous stage are passed together. After applying the general perceptron equation through the first sigmoid function, the percentage of LTM to retain is computed. This is then multiplied by the LTM from the previous stage (c_{t-1}) and sent to a summation operator to obtain the current LTM (c_t). The perceptron equation, with new weights and biases, is also used for the second sigmoid function, which determines the percentage of potential LTM to retain. This potential LTM is computed from the output of the hyperbolic tangent function. By multiplying the potential LTM by its respective percentage, we obtain the second term for the calculation of the current LTM. As for the calculation of the current STM (h_t), it is obtained by multiplying the current LTM, passed through the hyperbolic tangent function, by the percentage of STM to retain. This percentage is in turn calculated by the final perceptron equation for the third sigmoid function.

It is important to note that an LSTM cell requires a vector input, as we provide data in sequences to our models in Chapter 4. Thanks to this characteristic, and depending on the length of the input sequence, LSTM networks are also capable of capturing time and seasonal dependencies.

2.10 Application of ANNs to Real Problems

Modern applications relying on ANNs range in a wide set of sectors, thus leading to tailored structures for specific domains. Here below just a few are listed[7]:

- Image and Video Processing, using CNNs as discussed before in section 2.8;
- Natural Language Processing (NLP), involves understanding, interpreting, and responding to human language. The application may vary from text generation to sentiment analysis, text summarization and chatbots;
- Speech Recognition and Audio Processing, is the science of analyzing and synthesizing audio signals to extract meaningful information with several applications such as enhancing the quality of sound, noise reduction up to interpreting purposing like Cross-Language Recognition and audio synthesis;
- Forecasting, as in our case, may range from weather to financial, from economic to healthcare domain;
- Autonomous Systems and Robotics, the applications range from object recognition using CNNs to Control and Motion Planning, as well as Adaptive Control;
- Medicine, including cancer detection, drug response prediction, clinical decision support, robotic surgery and disease modeling;
- Commerce, ANNs are widely used to enhance decision-making, optimize operations, and improve customer experiences.

2.11 Hybrid Models for PV Power Forecast Using ANNs

This last section is intended to presenting a series of applications of ANNs for AI models drawn from the state of the art in the field of solar power forecasting.

In [55] the data collected was the 24-h mean temperature, 24-h mean relative humidity, 24-h mean wind speed and global radiation from the year 1985 to 2012 in the region of Kuala

[7]To compile this list, results found online by searching for applications of ANNs have been collected.

Terengganu, Malaysia. The best models were selected using statistical analysis and compared with different models developed in the literature. First of all, data was preprocessed and normalized. Four models were conceived varying the combination of the input features, while the optimal number of neurons in the hidden layer was determined by increasing it and looking at the MSE as well as the correlation coefficient R between the actual known and predicted values. Three different algorithms were used for the training, thus comparing the correlation R for both the training and validation sets among the 4 available models. Eventually, the performance results were plotted together for a clear visualization. It turned out that the Bayesian Regularization back-propagation algorithm was the best-suited algorithm in the study to develop a solar radiation predicting ANN model. It was also concluded that temperature and relative humidity are closely related to solar radiation whereas wind speed had little influence.

In [56], a case study has been done in the Peer Panjal region. Data was collected over several months and an ANN architecture with 1 hidden layer of 14 neurons was built. Both the GD and Levenberg Marquardt's[8] (LM) backpropagation algorithms were used for the training, adopting the sigmoid activation function for all neurons in the hidden layer and the linear function in the output layer. The ambient temperature, solar irradiance, and relative humidity were chosen as input features, while the output to be forecast was the voltage generated by the considered plant. To reach the optimal number of neurons in the layer, which was found to be 14, several structures were compared, resulting in the best tradeoff model when using the LM approach. The results showed the least Mean Square Error and the maximum R-value through both the training and testing.

In [57], data from a plant located in central Malaysia were collected. The time spans ranged from August 2020 to June 2021 and from September 2022 to October 2022, at a 5-minute frequency. A hybrid model was built, starting from an ANN architecture that was enhanced through the implementation of the Pearson Correlation Coefficient (PCC), which proved to be very useful for the feature selection task. PCC is based on the correlation between the available input parameters and the generated power output. The results were later compared with the Seasonal ARIMA (SARIMA) model for short-term power forecasts. Due to their common statistical nature, the error metrics were good for both models, although performance improved when PCC was applied. In particular, RMSE and R^2 were significantly improved. As a result, while the total global horizontal irradiance, global irradiance on the module plane, ambient temperature, and PV module temperature showed a high correlation with power generation, only a slight correlation was found for wind speed, and almost no correlation was found for horizontal and total slope irradiance. Applying feature selection resulted in a 45% improvement in forecast accuracy, and through hyperparameter optimization techniques, the accuracy increased further, enhancing the robustness of the prediction.

[8]Concerning the GD approach, the LM exploits information coming not only from the gradient but also from the second derivative, which is grouped in an approximation of the Hessian matrix. More in detail, depending on the nature of the error and whether it goes through a significant or slight increase/decrease, the LM algorithm will perform as a hybrid method between GD and the Gauss-Newton approach. Depending on the value of a damping coefficient, GD will be applied to take smaller steps towards an optimal solution (the damping factor is large); conversely, the Gauss-Netwon is preferable when it is needed to take larger steps (the damping factor is small). LM approach often converges faster than simple GD algorithms and results are more robust.

Chapter 3

The Case Study and The Available Data

In the context of growing interest in alternative solutions for environmental sustainability and the use of renewable energy sources, key institutions with significant public profiles, such as universities and hospitals, have promptly taken action. This is the case for Politecnico di Torino, which has seven photovoltaic plants that will be the focus of this master's thesis. These plants have been thoroughly described in previous thesis work, such as in [29], where all the technical characteristics of each installation - including location, orientation, number of panels, technical specifications, inverter power, photovoltaic modules, and a description of the electrical panels - were reported. We will not analyze the aforementioned data, given the different approach to the problem of predicting generated power. In the case of ANN models, not all that information is required. Instead, we will use a subset of that data, assuming its accuracy, as input for our models. The relevant information about the plants is summarized in Table 3.1, which actually shows six plants: the "Sede Centrale" plant will be considered as two separate plants due to the physical division of the panels into two general groups[1].

Plant	n°panels	Tilt	Azimuth	Panel power [W]	Plant power [kW]
"Sede Centrale"	400 88	26°	33°	360 430	144 38
"Ex Tornerie"	1849	26°	26°	327	605
"Aule R"	117	0°	28°	400	47
"Ex Fucine"	108	26°	113° -67°	280	30.4
"Energy Center"	210	10°	28°	327	46
"Aule P"	144	30°	28°	345	49

Table 3.1. PV plants of Polytechnic of Turin.

The total power generated by all the plants amounts to 960 kW, sufficient to sustain the university's basic load. We aim to provide a final estimate of the overall production, thus it is very useful to drive the inspection of several plants, build a model for each one of those, and sum the generation results (Table 3.1).

These data represent the initial technical characteristics available to us regarding the installations. We aim to identify the other essential information necessary for constructing the predictive model. Undoubtedly, the presence of known meteorological conditions, including atmospheric data or weather forecasts, is crucial.

[1]Indeed, the panels within the "Sede Centrale" installation are distributed across various roofs. However, for generalization purposes, they can be categorized into two groups, as detailed in the table.

3.1 Atmospheric Data from Politecnico di Torino

Politecnico di Torino is equipped with a weather station for measuring meteorological variables such as temperature, relative humidity, wind speed and direction, solar irradiance, as well as an air quality indicator [58]. It is possible to access the database through the "LivingLAB@Politecnico di Torino.it" online portal after logging in with the credentials of Politecnico's staff or students. The data have been acquired[2] at a frequency of fifteen minutes, which makes it possible to draw an exact analysis. The meteorological station at Politecnico di Torino is equipped with a series of instruments and sensors designed to monitor various atmospheric parameters. Typically, a meteorological station like the one at Politecnico includes the following elements:

- Thermometer: measures air temperature.;
- Hygrometer: measures relative humidity;
- Barometer: measures atmospheric pressure;
- Anemometer: measures wind speed;
- Wind Vane: measures wind direction;
- Rain Gauge: measures precipitation (rain, snow, etc.);
- Radiometer: measures solar radiation intensity;
- Wet Bulb Thermometer: determines dew point temperature and relative humidity;
- Air Quality Sensors (if present): monitors atmospheric pollutants like CO2, NO2, PM10, etc.

These instruments work together to provide a comprehensive view of local weather conditions. The station may also have data recording and transmission systems, allowing remote access to the collected information. Additionally, Politecnico di Torino, specifically for monitoring data related to photovoltaic panels, has both a pyranometer[3] and cell sensors for the irradiance measurement.

3.2 Weather Forecasts

In addition to the meteorological conditions associated with solar power production, we need to consider weather forecasts. As will be clarified in the next Chapter 4, training is performed using historical weather conditions; however, the goal is to provide a power estimate based on forecasts, i.e., an estimate derived from weather predictions. Furthermore, production curves will be presented, highlighting how a more accurate weather forecast (closer in time to the power prediction moment) leads to a power estimate that is closer to the subsequently measured value. The first step involves collecting atmospheric data through sensors and local ground stations, as well as weather balloons or images from meteorological satellites. Radars, on the other hand, are used to monitor cloud cover [60]. Depending on the type of forecast desired, particularly the relationship between spatial resolution and temporal frequency[4], different techniques are employed. Whether

[2]Depending on the plant, the data may have been made available years apart from one another, which might be related to the date of the installation's initial operation as well as potential sensor malfunctions, or delays in monitoring.

[3]A pyranometer is a device which measures total solar radiation from the planetarium (scattered as well as direct), usually in the horizontal plane. This means that it must give an unbiased response to radiation from all directions. It consists of a horizontally oriented thermal sensor and a glass dome that limits the wavelength range[59].

[4]In the field of Image Processing, spatial resolution refers to the smallest object that can be distinguished in an image. In a photographic system, it is usually measured as the minimum separation at which objects appear distinct and separate in the photograph. Temporal resolution, on the other hand, refers to the frequency with which an image can be obtained over the same area.

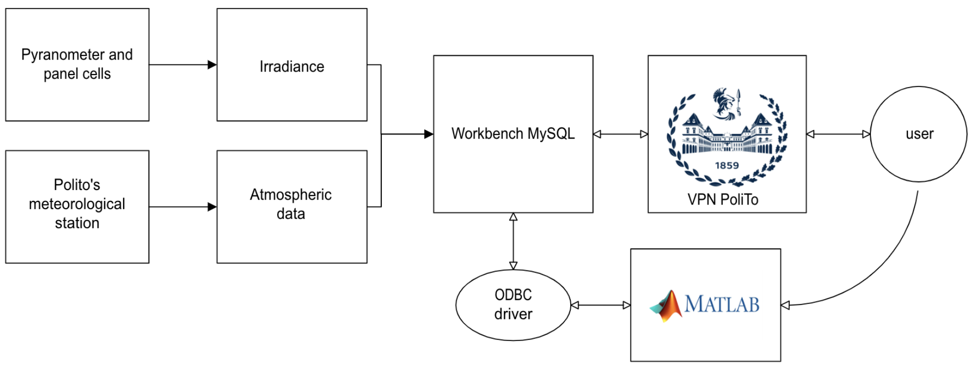


Figure 3.1. User-Dataset interaction diagram.

dealing with numerical data or images, they are transmitted to collection centers in an encoded format, where they are then forwarded to meteorological forecasting centers. Here, the data are decoded, and graphs are created for visualization (e.g., bar or line graphs for temperature, wind, pressure, or synoptic and satellite maps). Specifically, surface isobaric maps connect geographical points with the same atmospheric pressure, considering wind direction; this allows for the prediction of high and low-pressure areas, as well as cyclones. To depict wind flow, upper air charts are created using streamline analysis. For pressure distribution, tide monitoring, and precipitation forecasting, weather charts are produced. The use of computers with numerical analysis is another crucial tool, as it highlights the movement traced in these charts and extends it over time for the coming hours or days.

3.3 The Dataset for Our Case Study

This thesis work is part of a larger project involving several thesis students, each conducting research on the topic of forecasting the generation of PV installations at the Politecnico di Torino and proposing a model. In the past, both deterministic and statistical models have been developed, showing very positive results. In any case, the available data are the same for everyone, and thanks to the work of students in previous years, and the professors who supervised and assisted them, we can now start with much more reliable data. The data have been uploaded into a private MySQL workbench, requiring credentials to connect via VPN to the Politecnico di Torino network and access the database (Figure 3.1). By using simple queries, it is possible to extract the relevant data, such as by selecting a known time range or a subset - choosing only the columns corresponding to the necessary data rather than the entire set of information for the installations. In particular, the available data regards solar irradiance, the corresponding atmospheric conditions[5], and the related power that has been generated by each plant. It is important to emphasize that the data used in this work have undergone extensive cleaning, and they have been carefully checked and refined. Specifically, at the beginning of the project, a thorough review of the data was conducted, and errors of various kinds were identified, ranging from sensor malfunctions to acquisition errors. Also measurement errors, like in the case of excessively negative temperature or too high irradiance, were detected. In this thesis, as well as in previous ones, the queries, along with all the computational work - and in this case, automation - were conducted in the MATLAB environment. MATLAB is notably user-friendly for data organization, processing, and visualization. An ODBC driver, available in the MATLAB Database Toolbox, is used to communicate with the Workbench.

[5]In this work, we will consider only irradiance and temperature both for historical regression model and the predictive one.

Chapter 4

Design

In this chapter, we will present our models, considering only the data from the *Cittadella* plant, as this dataset is the most complete and free from missing information. Two different data sets were used: the first (Oct 2022 - Jun 2024), which includes historical measurements of weather conditions and the corresponding power output, and the second (Apr 2024 - Oct 2024), which contains the same information along with forecast weather conditions. Although the data appeared to be mostly correct, some preliminary checks were necessary, along with an evaluation of the accuracy of the forecasts. To determine the relevance of each variable and assess whether it should be included in the analysis, we reviewed the relevant theory, identifying irradiance and temperature as the most influential factors on power output. These variables will be used as inputs for our models, and we will focus on them in this study. Additionally, a preprocessing phase is required to prepare the data before training the models.

4.1 Preliminary Checks

The dataset has been inspected to identify anomalies, missing data, or mismatches. Since the data available were expected to have already undergone validation and corrections, only minimal adjustments were required in this phase. Specifically, to mitigate the negative influence that unrealistic scenarios could have on the training of the models - and consequently on predictions for unseen data - the following inconsistencies in the dataset were addressed:

- Predicted irradiance value $G_{\text{for}} = 0$, while the measured irradiance value $G \neq 0$;
- Predicted irradiance value $G_{\text{for}} = 0$, while the measured power value $Power \neq 0$.

In both cases, the values of G and $Power$ were set to 0 to ensure coherence between forecast and measured values, at least for null entries. Otherwise, the models could learn unrealistic relationships, such as linking a null prediction to a non-null power output. Such mismatches clearly indicate sensor malfunctions or related issues. Instead of removing entire days from the database, this correction strategy preserves the essential condition that both irradiance and power should simultaneously be zero under such circumstances.

4.2 Accuracy of the Weather Forecasts

This section serves as a point of reflection on the fact that future power prediction models inherently depend on the quality of the meteorological data forecasts.

As mentioned above, the dataset starting from April 2024 includes past weather forecasts. Specifically, forecasts are available at different times throughout the day, grouped into four intervals starting at midnight, with updates every six hours. These forecasts cover a three-day period;

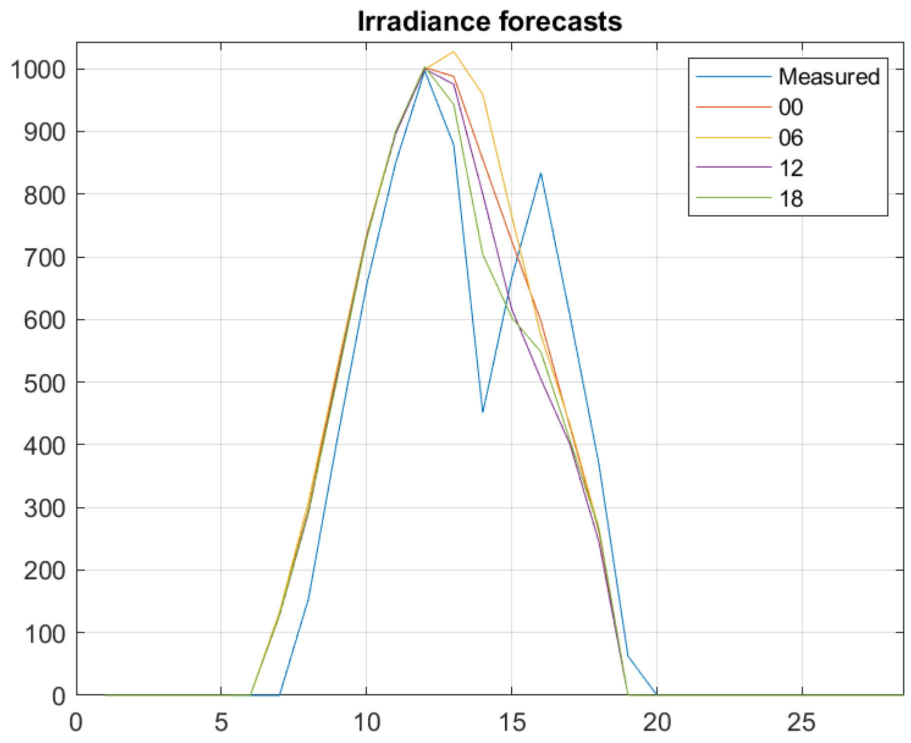


Figure 4.1. Irradiance forecasts at different times in a clear day.

however, the objective of our study is to provide power predictions for a single day. In general, the target day is any day for which weather forecasts are available, preceded by one or more days with known measured conditions.

Figure 4.1 illustrates distinct forecasts for a clear day in April. All forecasts appear accurate and tend to smooth the peaks in the measured irradiance, likely caused by the passage of clouds at specific timestamps or delays in sensor acquisition. On a cloudy day, the irradiance reaches a lower range, making it more challenging for the forecasts to accurately predict the behavior of the irradiance curve (Figure 4.2). Nevertheless, we expect that forecasts made closer to the actual time they pertain to would perform better than those made several hours in advance, as forecast accuracy generally improves over time. This is evident in the green line, which is the closest to the blue one, as it represents forecasts made at 18:00 for the following day (00:00–23:00). In general, forecasts made at 12:00 are also good and are the ones considered in this work.

When it comes to temperature, considering that it does not influence the final power output as strongly as irradiance and that the forecasts are quite similar to each other (Figure 4.3), we did not focus extensively on this variable.

4.3 Preprocessing

The data must be preprocessed to ensure compatibility with our networks. The first step involves normalizing the inputs and, in some cases, the targets as well. Next, seasonality within the training data must be accounted for to ensure a balanced distribution across different periods. Finally, the data will be organized into sequences suitable for LSTM models.

4.3.1 Time Filtering

The data available in the dataset have a quarterly cadence, while the temperature and irradiance forecasts are hourly. We would expect better accuracy if we could train with a timestamp closer

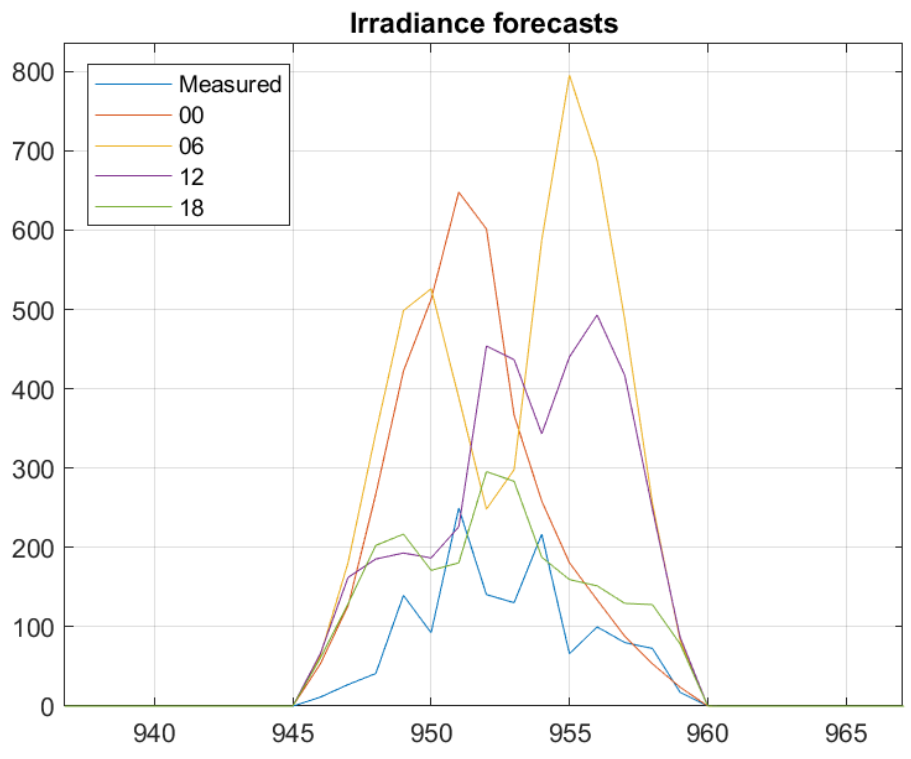


Figure 4.2. Irradiance forecasts at different times in a cloudy day.

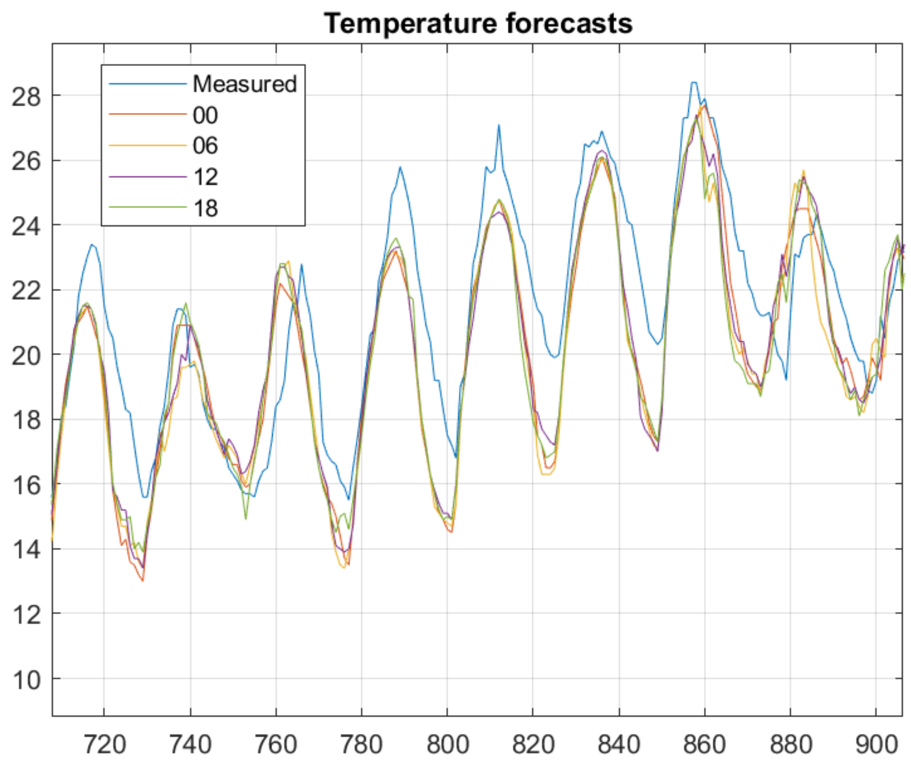


Figure 4.3. Temperature forecasts for several days.

to 15 minutes, but we do not have forecast data at that frequency, and there is no certainty that interpolation methods would lead to actual improvements. For this reason, this characteristic implies that the training data must be filtered to an hourly frequency, starting at 00:00 and proceeding hour by hour (01:00, 02:00, ...) until 23:00.

To begin with, the forecasts were compared through graphical observation of their similarity to actual conditions, alongside the KPIs computed over the entire dataset for the four types of atmospheric forecasts. The results are presented in Tables 4.1 and 4.2 for the cases of temperature and irradiance, respectively.

Forecasts at	MBE	MAE	MAD	R^2	RMSE
12:00 AM	1.4698	2.2074	1.5	0.47671	4.0843
6:00 AM	1.2489	2.0782	1.5	0.62036	3.4788
12:00 PM	1.2244	2.0075	1.5	0.64475	3.3652
18:00 PM	1.393	2.0802	1.5	0.54912	3.7912

Table 4.1. Temperature forecasts: KPIs at different times in a day on the whole dataset (2).

Forecasts at	MBE	MAE	MAD	R^2	RMSE
12:00 AM	-41.275	84.005	15.677	0.68021	160.24
6:00 AM	-44.148	82.78	16.218	0.68937	157.93
12:00 PM	-43.141	79.042	13.696	0.7258	148.38
18:00 PM	-40.399	83.363	16.501	0.67708	161.03

Table 4.2. Irradiance forecasts: KPIs at different times in a day on the whole dataset (2).

Based on these KPIs, the forecasts at 12:00 PM were selected and will be used for this analysis.

4.3.2 Normalization

Normalization is a common practice in neural networks due to its advantages in making training smoother and more stable, thereby improving the model's performance. Additionally, it prevents the dominance of one feature over the others by bringing all input features to a similar scale.

Both in the case of FNN and LSTM models, the inputs have been normalized in the [0,1] scale, which is:

$$x_{\text{norm}} = \frac{x - \min(x)}{\max(x) - \min(x)}.$$

In the case of LSTM models, due to the variability of the power, which is used as the target in the training, it could be[1] more stable and efficient if also the targets are normalized, which leads to the need of denormalizing afterwards. The denormalization must be the reverse of the normalization, hence the outputs will be treated as follows:

$$y = y_{\text{norm}} \cdot (\max(y) - \min(y)) + \min(y).$$

The minimum and maximum values used in the denormalization are retrieved from the training, hence they are computed from the targets.

4.3.3 Seasonality

To ensure that the models are trained considering the entire time span in a balanced way[2], a function named *seasonality*_24 has been implemented. This function, for every 30 days, randomly

[1] Indeed, in our models the targets are not normalized, but it can turn useful to account for this option, too.

[2] We expect that by adding new data and once at least one year of data is available, the models will improve by having the system's behavior across all seasons and with different weather conditions.

selects 6 numbers (20%) between 1 and 30, which correspond to the days allocated for validation (10%) and testing (10%). The function runs once before training and calculates the indices of these days for the validation and test sets.

4.3.4 Sequence Extraction

Another important action to prepare data to work with LSTM models is to group them into sequences. In the case of just a day, as we are trying to train a model which provides hourly power predictions, several sequences of 24 data are obtained from the database. When several days are considered, more sequences are joined together. This operation is done in MATLAB by the function *sequence_extraction_var*, which has been built for this purpose. It takes as inputs the sets that we want to sequence and returns the 24-data sequences, i.e., for the 24 hours of a day. Different types of sequences have been extracted, and for simplicity, we will refer to dataset 1, which contains the actual data, and dataset 2, which contains the forecast data:

- *seq_dates* (24×1), the sequences of datetime data containing information about the date and time of each row from dataset 1;
- *seq_dates_for* (24×1), same as the previous sequences, but they now refer to dataset 2;
- *seq_act_in* (24×2), containing the temperature and irradiance values corresponding to dataset 1;
- *seq_act_out* (24×1), referring to the corresponding produced power values in dataset 1;
- *seq_for_in* (24×2), the sequences of forecast values for irradiance and temperature from dataset 2;
- *seq_for_out* (24×2), the sequences of measured values for temperature and irradiance from dataset 2;
- *seq_for_pow* (24×1), containing the corresponding power output from dataset 2;
- *seq_for_irr* (24×1) and *seq_act_irr* (24×1), the sequences of the forecast and measured values of irradiance from dataset 2[3]

4.4 FNN and LSTM Models

After the checks and preprocessing, we are now ready to present the FNN and LSTM models used in this work. For each model, the structure will be outlined, specifying which and how many layers have been used, as well as figures illustrating the training phase, showing the Training and Validation Loss. The results in terms of KPIs will be presented for the validation phase and later compared in the next chapter on the whole dataset.

Regarding the nomenclature, the models are named according to the following criteria:

- The first word refers to the type of model (FNN or LSTM), followed by an underscore ("_");
- Next comes an abbreviation indicating the type of training, specifically:
  - **a2p:** from actual (measured) values to power, meaning the model learns a direct correlation between the sensor-measured values and the produced power during training;
  - **f2c:** from forecast to corrected values, referring to models developed to correct forecasts of irradiance by learning the relationship between forecast and subsequently measured real values;

[3]These data are actually already present in the sequences *seq_for_in* and *seq_for_out*, but it is relevant to separate irradiance from temperature in the subsequent analysis for the irradiance correction model 4.4.3.

  - **f2p:** from forecast values to power, where the model learns a direct correlation between forecast temperature and irradiance values and the produced power.

- After another underscore ("_"), the number of days (in hours) considered is indicated:

  - **24h:** when only a single day is considered. The data is organized into sequences of size (24×1);
  - **96h:** when multiple days (4) are considered. In this case and when providing a power prediction, only the last day includes forecast data, while the previous three days refer to measured values. The data is organized into sequences of size (96×1). For example, to calculate the power forecast for November 19, an input sequence must be created by concatenating three subsequences, each 24 long, of measured temperature and irradiance values corresponding to November 16, 17, and 18, along with a 24-long sequence of forecast temperature and irradiance values for November 19.

4.4.1 FNN_f2p

The *FNN_f2p* model can be considered the simplest both logically and in its structure. Regarding the reasoning behind its development, this model follows the idea that would occur to anyone in the task of power prediction based on weather forecasts: training the model on forecast data using measured power data as the target. From the structural point of view, the *FNN_f2p* model can be considered the simplest approach overall, as it simply includes an input layer with n number of features (in our case, two: temperature and irradiance), a customizable number of hidden layers with m neurons, and an output layer that returns the single power variable. The structure is as follows:

- 1 featureInputLayer (input layer);
- 1 fullyConnectedLayer[4] with m neurons;
- 1 fullyConnectedLayer (output layer).

This same structure will be employed later in the upcoming LSTM-based models, always starting with the substitution of a single LSTM layer in place of the current hidden layers. For the initial training, we start with low values of m, ranging from 1 to 5, and report the results in terms of validation KPIs in Table 4.3. The number of max epochs was set to 30 since the model, with the minibatch size currently set to 32, stabilizes the Loss functions at their minimum within this limit (see Figure 4.4 for $m = 5$).

m	MBE	MAE	MAD	R^2	RMSE
1	0.49067	37.402	6.4395	0.73569	65.441
2	0.43426	37.008	6.7075	0.73648	65.344
3	-0.42721	37.182	6.8539	0.7363	65.365
4	-0.11061	36.944	6.8347	0.73655	65.335
5	-0.31269	37.029	6.7317	0.73648	65.343

Table 4.3. KPIs during validation for $m = 1 : 5$.

[4]Due to the nature of the data, which are always positive and normalized in the range 0-1 for the inputs, and since the targets have been kept denormalized, in this work linear activation functions have always been used, specifically the ReLU function.

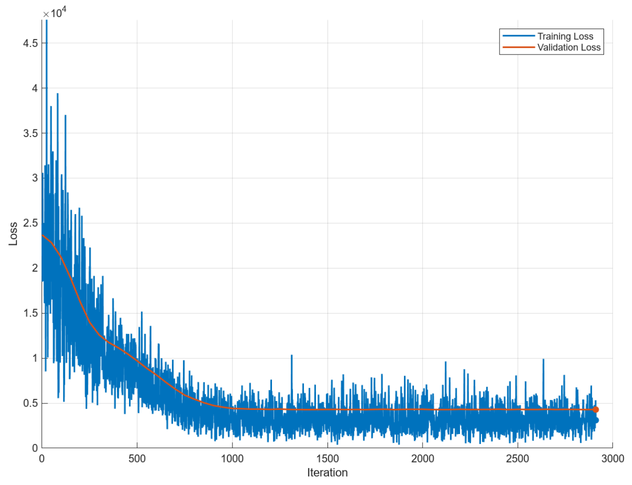


Figure 4.4. Training with $m = 5$.

Table 4.4 reports the validation KPIs obtained by selecting multiples of m in increments of 5. It should be noted that every time a new simulation starts, the training and validation datasets change, as the indices used to select certain days over others are randomized. This ensures a fair rotation of data and introduces seasonality into the analysis. For this reason, it is not surprising that the KPIs for $m = 5$ in Tables 4.3 and 4.4 differ. The central row with $m = 15$ seems to

m	MBE	MAE	MAD	R^2	RMSE
5	-1.7527	30.659	5.2579	0.7416	57.896
10	-3.5317	30.64	4.7635	0.73995	58.08
15	**-1.898**	**31.07**	**5.0929**	**0.74196**	**57.856**
20	-3.7087	32.514	6.6075	0.74049	58.019
25	-0.9158	31.025	6.0461	0.74049	58.02

Table 4.4. KPIs during validation for m = 5,10,15,20,25.

perform best in this simulation; therefore, let us try adding more hidden layers starting from this configuration:

- 1 featureInputLayer (input layer);
- 1 fullyConnectedLayer with n neurons;
- 1 fullyConnectedLayer with 15 neurons;
- 1 fullyConnectedLayer with k neurons;
- 1 fullyConnectedLayer (output layer).

Several combinations have been tested, and the validation KPIs are presented in Table 4.5. The horizontal line at the center of the table indicates where the number of max epochs was reduced to 15, since this new value is more than sufficient to allow the Loss functions to stabilize around their minima (Figure 4.5). As we can observe from the table, it is not necessarily true that increasing complexity leads to better performance. In fact, for this type of analysis, the best approach might be to save the best model up to a certain point and update it when the performance reaches a new optimization. However, in general, we cannot expect to obtain the same results with new simulations, and especially not to see the same performance outside of the validation set.

The two marked rows appear to correspond to the models with the smallest RMSE error and the highest R^2 correlation index. These models will be examined in detail in the next chapter to compare their results with respect to the type of day and the power curve.

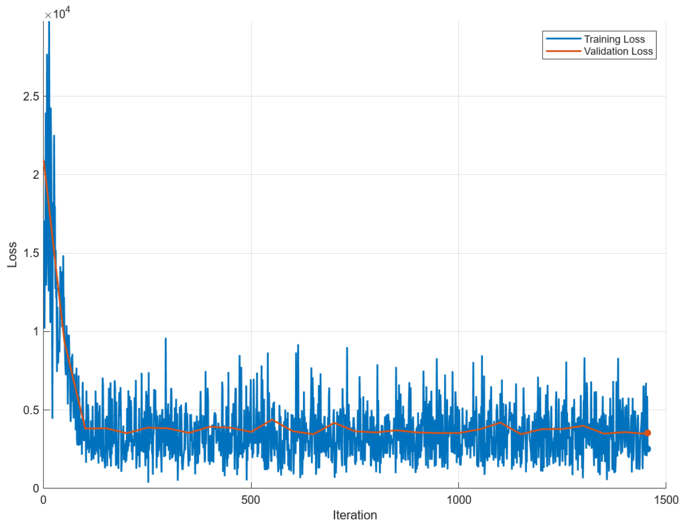


Figure 4.5. Training with $n, m, k = 15$.

n	k	MBE	MAE	MAD	R^2	RMSE
1	0	0.32689	33.957	5.6111	0.71766	62.282
2	0	0.10143	33.932	5.7717	0.71786	62.261
5	0	1.4674	34.041	6.3216	0.71781	62.266
10	0	1.5437	35.636	6.1605	0.69223	63.502
15	0	-0.20655	35.581	6.5927	0.6927	63.454
0	1	-3.1092	28.956	6.783	0.81574	53.547
0	**2**	**-1.1016**	**28.521**	**5.6532**	**0.81564**	**53.562**
0	5	-1.798	28.357	5.4591	0.81593	53.519
0	10	-1.2863	31.107	5.6	0.80055	55.618
0	15	-1.3133	31.368	6.4	0.80018	55.669
5	**5**	**-1.5066**	**31.698**	**7.371**	**0.82473**	**53.855**
5	10	-1.0956	31.929	7.6021	0.82331	54.074
5	15	-3.2834	31.908	8.8152	0.8231	54.106
10	5	-1.9351	36.623	8.2681	0.74508	65.96
10	10	0.13185	36.555	7.6463	0.74545	65.912
10	15	-0.8222	36.918	8.6661	0.74406	66.092
15	5	-0.90308	32.665	5.9141	0.76712	58.38
15	10	-1.1253	32.788	6.629	0.76626	58.488
15	15	-4.1492	33.216	6.5404	0.76382	58.792

Table 4.5. KPIs during validation for $m = 15$ and distinct combinations of n, k.

4.4.2 LSTM_a2p_24h

This model was created to evaluate the ability of a network containing LSTM cells to characterize the system. Specifically, it is trained and tested in regression on the measured data. This model has been trained on dataset 1, the larger dataset containing only measured values. The input sequences have a dimension of 24×2 and contain past values corresponding to the measured temperature and irradiance. The target sequences are of dimension 24×1, corresponding to the observed power output. Validation and testing are initially performed on unseen data corresponding to past measured values of temperature and irradiance (without forecasts). As a first trial, a simple structure has been adopted:

- 1 sequenceInputLayer (input layer);
- 1 LSTM layer with m cells, from 1 to 5;

- 1 fullyConnectedLayer (output layer).

The following Figures show the training process and the Loss functions for $m = 1$ and $m = 5$.

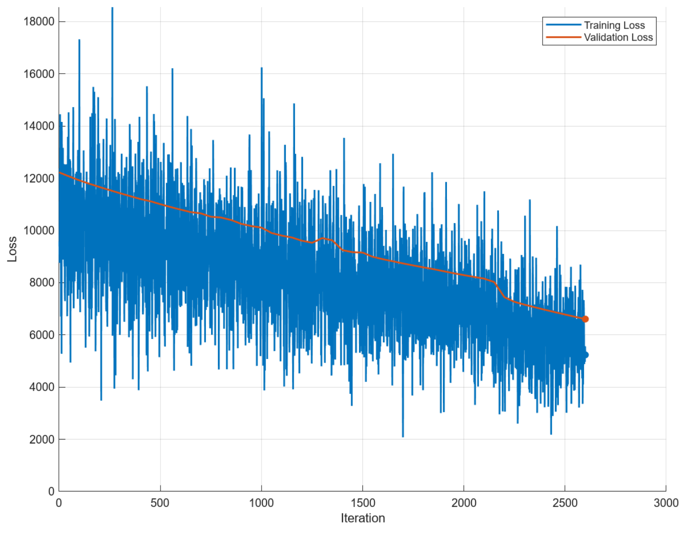


Figure 4.6. Training with $m = 1$.

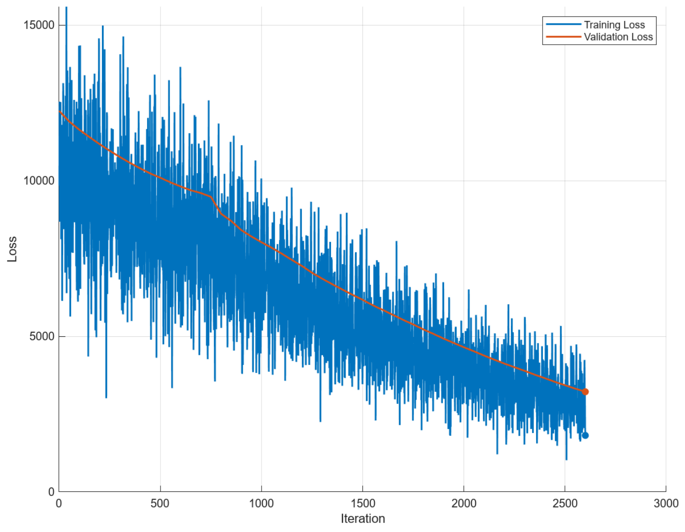


Figure 4.7. Training with $m = 5$.

The KPIs during validation appear to improve as the number of LSTM cells increases (Table 4.6), which is why we continued increasing the value of m. For m=10, the KPIs show improvement and continue to enhance as m increases (Figure 4.8 for m=15 and Table 4.7 for m = 10,15,50), albeit with diminishing returns and at the expense of higher training and future computational costs for running the model.

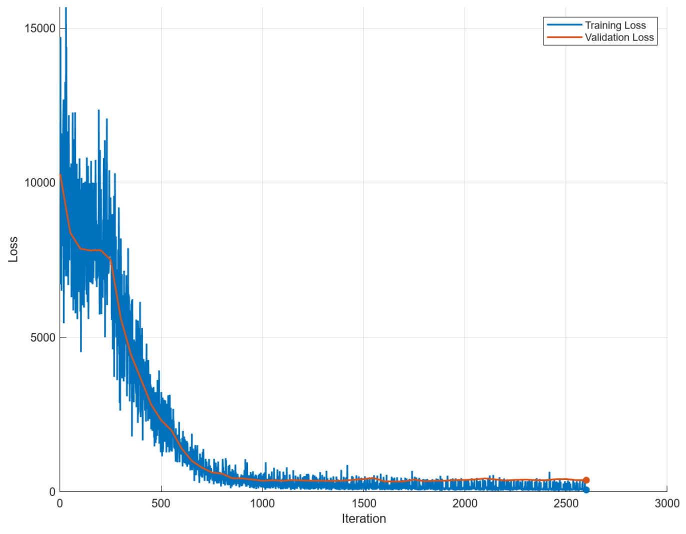


Figure 4.8. Training with $m = 15$.

m	MBE	MAE	MAD	R^2	RMSE
1	29.288	39.859	7.0989	0.18562	91.106
2	26.991	31.434	1.6737	0.35195	81.272
3	22.751	27.373	1.6712	0.48639	72.352
4	19.727	22.615	0.017691	0.59739	64.059
5	15.324	21.005	1.7618	0.68438	56.717

Table 4.6. KPIs during validation for $m = 1 : 5$.

m	MBE	MAE	MAD	R^2	RMSE
10	3.8774	8.9511	1.7606	0.92891	25.729
15	-0.40446	5.7432	0.25123	0.94786	21.659
50	-1.5462	5.3736	0.66831	0.96131	18.33

Table 4.7. KPIs during validation for m = 10,15,50.

4.4.3 LSTM_f2c_24h

This model was designed to improve irradiance forecasts, i.e., to perform a correction directly based on the relationship between weather forecasts and the measured values of irradiance and temperature for the specific day of interest for the forecast. The input sequences have a dimension of 24×1 and contain past values corresponding to the forecast irradiance. The target sequences are of dimension 24×1, corresponding to the measured irradiance. As with the *LSTM_a2p_24h model*, the search for a good model began with a basic structure of the following type:

- 1 sequenceInputLayer (input layer);
- 1 LSTM layer with m cells;
- 1 fullyConnectedLayer (output layer).

We started with low values of m, observing that as m increased, the KPIs improved (although R^2 values remained negative and RMSE was on the order of 300 W/m^2). However, with $m = 5$, we were still far from the expected results, as evidenced by the training plot (Figure 4.9). First, we attempt to increase m further and try with $m = 15$. The training plot continues to decrease until the last epoch, so we increase the maximum number of epochs to 1000 (Figure 4.10). It seems there is still room for improvement, and in particular, we note that with an increasing number of LSTM cells, the model is able to minimize the loss functions more quickly, while with lower values of m, the model requires a longer training period. Therefore, we can consider the value of max epochs to compare different models (Table 4.8).

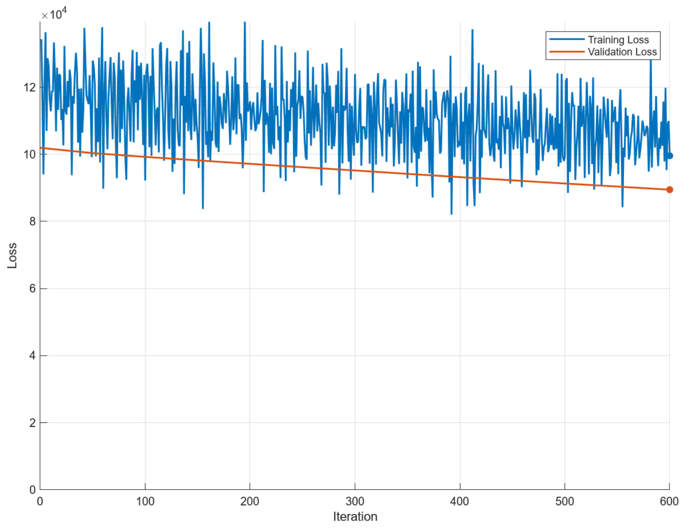


Figure 4.9. Training with $m = 5$.

m	MBE	MAE	MAD	R^2	RMSE	max epochs
15	1.2017	67.255	14.625	0.762	126.76	1500
20	0.45032	78.368	18.626	0.78307	143.99	1200
25	17.015	76.775	15.416	0.71641	144.94	1000
50	0.034517	70.762	16.71	0.78642	128.97	600

Table 4.8. KPIs during validation for distinct combinations of m and max epochs.

The training plot for $m = 50$ is shown in Figure 4.11.

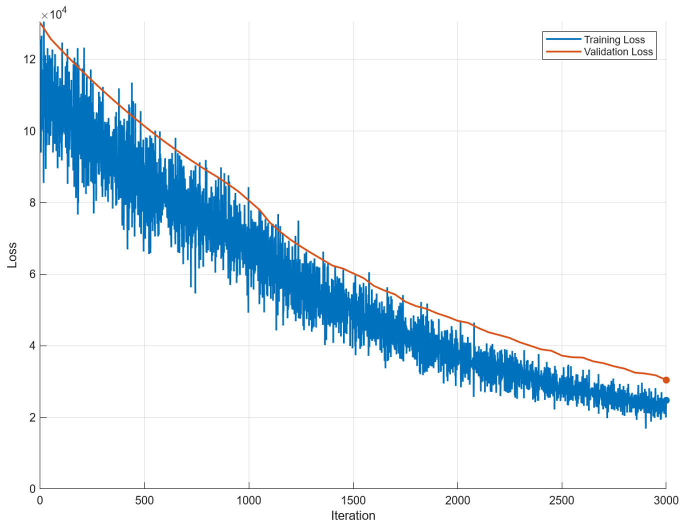


Figure 4.10. Training with $m = 15$, max epochs: 1000.

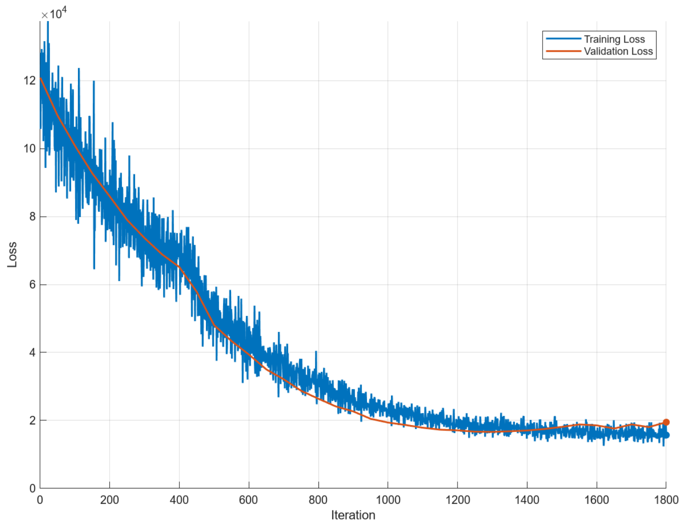


Figure 4.11. Training with $m = 50$, max epochs: 600.

The next step, aimed at further improving the performance during validation by reducing the $RMSE$, involved adding additional fully connected layers both before and after the LSTM layer:

- 1 sequenceInputLayer (input layer);
- 1 fullyConnected layer with n neurons;
- 1 LSTM layer with m cells;
- 1 fullyConnected layer with k neurons;
- 1 fullyConnectedLayer (output layer).

Keeping the value of m equal to or greater than 15, a maximum number of 200 epochs is more than sufficient. Several combinations were tested, as presented in Table 4.9. The training plot for the row with $n = 10$, $m = 20$, $k = 10$, considered the best configuration, is shown in Figure 4.12. It should also be noted that these KPIs refer only to a small portion of data, specifically the validation set, which for each new training session is randomly extracted from the general dataset using the *seasonality_24* function. Therefore, the same performance may not necessarily be replicated on the test set or the entire dataset (next Chapter).

n	m	k	MBE	MAE	MAD	R^2	RMSE
2	15	2	-4.455	74.072	15.842	0.73743	135.17
5	15	2	5.5453	70.659	12.244	0.76358	135.72
5	15	5	-6.3294	66.09	18.384	0.80235	120.58
10	15	5	-2.5825	71.83	17.796	0.79824	129.39
10	20	5	6.9595	66.714	14.661	0.84932	125.05
10	**20**	**10**	**-5.8216**	**67.653**	**20.663**	**0.81577**	**119.63**
10	20	15	6.1395	67.345	16.219	0.8137	125.85
15	20	15	2.8361	68.907	9.1264	0.8036	126.39
10	25	10	-8.1667	66.886	14.167	0.76123	123.93
10	25	15	15.106	76.425	19.426	0.79487	134.61
15	25	10	3.4882	69.891	14.288	0.82813	126.64
15	25	15	12.64	73.451	16.119	0.81047	133.46

Table 4.9. KPIs during validation for distinct combinations of n, m, k.

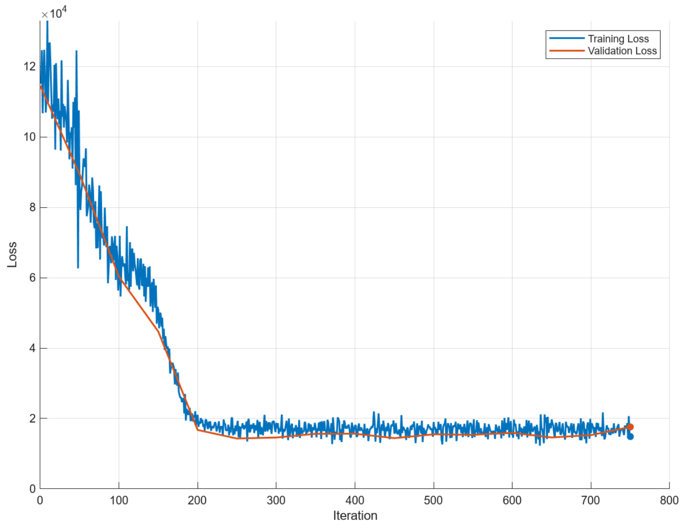


Figure 4.12. Training with $n = 10$, $m = 20$, $k = 10$.

4.4.4 LSTM_f2c_96h

Similar to the previous model, this one also aims to provide a correction of the irradiance forecast. However, in this case, the time window used for training is broader and includes the three days preceding the forecast. The sequences are constructed by concatenating four sequences with dimensions 24×1, corresponding to four consecutive days with forecast irradiance, from which only the last one is the target day to be predicted. The output is a sequence with dimensions 96×1, corresponding to the power output. We must trim the last sequence[5] to retain only the part that represents the prediction for the day of interest.

In this case, we also start with a very simple structure, such as:

- 1 sequenceInputLayer (input layer);
- 1 LSTM layer with m cells;
- 1 fullyConnectedLayer (output layer).

We expect from the outset that we will need to complicate the model, as we are no longer dealing with sequences of 24 data points, but with sequences of 96 data points. However, it is true that the model should produce a similar irradiance profile to the one from the *LSTM_f2c_24h* model if we use a structure similar to the one implemented at that time. It is therefore clear that it becomes even more difficult to determine how to complicate this model so that it performs well in forecasting, as it now also needs to consider the previous days leading up to the target day. We try using values of m that are multiples of 5 and obtain the validation KPIs reported in Table 4.10, setting the maximum epochs to 500.

m	MBE	MAE	MAD	R^2	RMSE
5	127.32	191.65	63.832	-0.12494	309.03
10	111.4	154.99	37.393	0.13642	270.77
15	86.395	133.37	31.391	0.33752	237.15
20	63.093	117.08	30.088	0.50156	205.71
25	37.317	110.03	39.454	0.61367	181.1
30	22.658	86.3	17.528	0.66657	157.27
35	14.193	80.897	19.095	0.70529	147.86
40	2.7199	75.58	23.051	0.74043	138.76
45	-0.82293	70.86	15.901	0.7628	132.65
50	-8.1584	68.027	22.561	0.77728	128.54

Table 4.10. KPIs during validation for distinct values of m.

It seems that as m increases, the validation KPIs improve. The training plot for $m = 50$ is shown in Figure 4.13. Since these KPIs, with a single LSTM layer consisting of numerous cells and no additional fully connected layers, are already comparable in terms of validation performance to the *LSTM_f2c_24h* model, we will not complicate it further for now. Instead, we will observe the results and compare it with the other irradiance correction model in the next chapter.

[5]To achieve this, a *for* loop in the code keeps track of the sequence number, even in cases where the sequences at the beginning or end of the database contain fewer than 24 data points (e.g., if the first data point does not correspond to 00:00). A variable named *lengths_seq_for* tracks the length of all sequences, which is guaranteed to be 24 from the second to the penultimate sequence. This ensures the last sequence is correctly trimmed from the 96-data output power sequence by referencing the start and end indices of the sequence in the database.

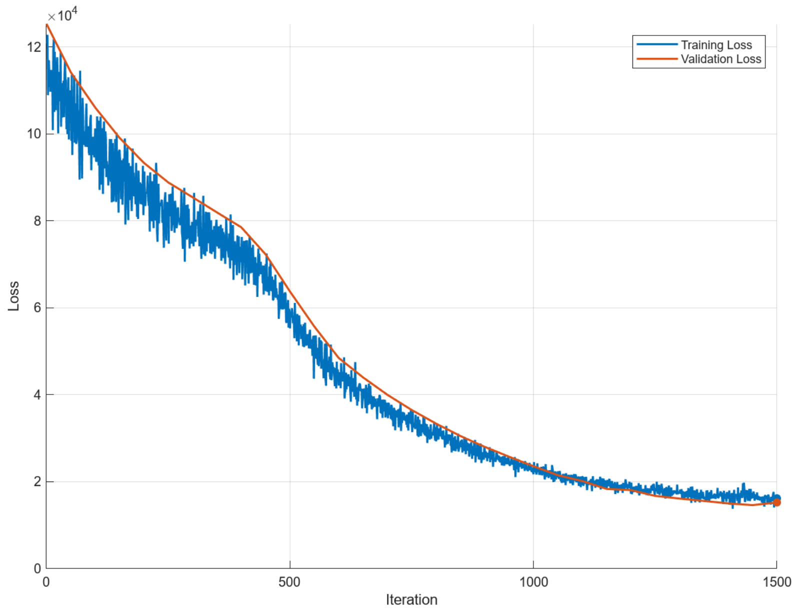


Figure 4.13. Training with $m = 50$.

4.4.5 LSTM_f2p_96h

This model is built for power forecasting, based on three days of measured temperature and irradiance data combined with one day of weather forecast data. It shares the same structure as the *LSTM_f2c_96h* model; however, it directly relates the forecast conditions to the output power without any intermediate correction step. This means that the input sequences, with dimensions 96×2, consist of 3 days of known conditions followed by 1 day of forecast temperature and irradiance[6]. The output is a sequence with dimensions 96×1, corresponding to the power output. As in the previous model, only the last sequence, corresponding to the power prediction for the day of interest, will be retained. As for the *LSTM_f2c_96h* model, given the number of inputs and outputs, the minibatch size was set to 100 to expedite the training process. The initial implementation, as usual, begins with a basic structure:

- 1 sequenceInputLayer (input layer);
- 1 LSTM layer with m cells;
- 1 fullyConnectedLayer (output layer).

We start by testing multiples of 5 for m, setting an initial max epochs value around 400. This configuration does not seem adequate (Figure 4.14, $m = 25$) to allow the Loss functions to stabilize around their minimum. Consequently, max epochs is incrementally increased to 1000, then 1500, and finally 2000 (Figure 4.15).The KPIs for this last case are shown in Table 4.11.

m	MBE	MAE	MAD	R^2	RMSE
10	1.6132	31.309	5.0509	0.7196	64.572

Table 4.11. KPIs during validation for $m = 25$, max epochs set to 2000.

The results of the power profile are presented in the next chapter, where the possibility of adding fully connected layers is also evaluated. For now, the network complexity remains lower compared to the other models, leaving room for potential improvements in performance.

[6]This scenario is different from the one seen in the *LSTM_f2c_96h* model, where all four considered days are taken as forecasts, effectively expanding the input window of the *LSTM_f2c_24h* model to 4 days.

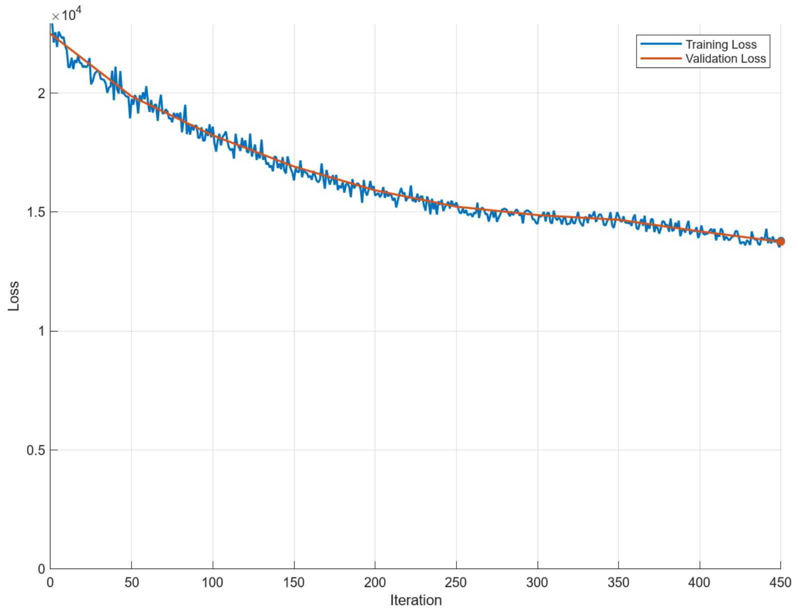


Figure 4.14. Training with $m = 25$, max epochs $= 400$.

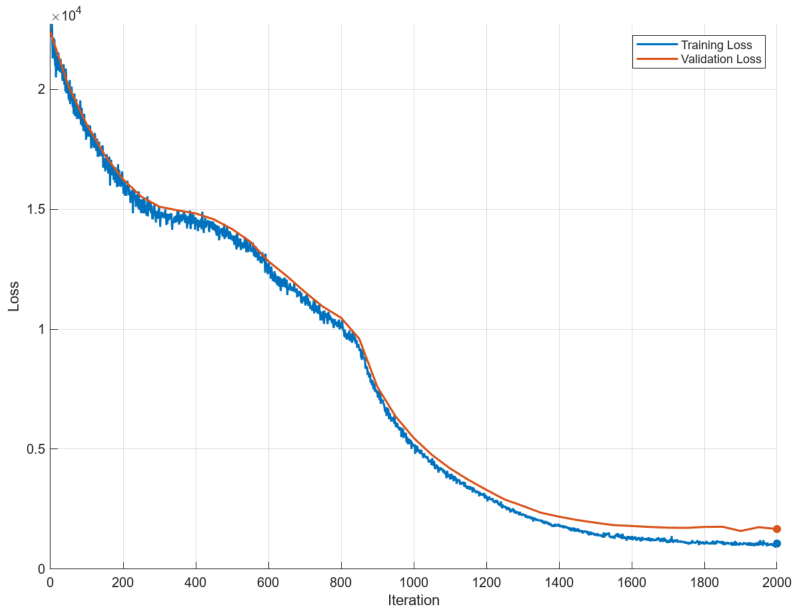


Figure 4.15. Training with $m = 25$, max epochs $= 2000$.

4.5 Post-Processing Techniques

By post-processing, we refer to a variety of techniques and approaches that can be implemented after obtaining the simulation results from our models, with the goal of further improving them. The simplest case involves applying a filter to the simulation outputs to limit the power predictions to non-negative values only. All models feature this filter, referred to as *low_power_filter_level*, which sets any value below a specified threshold to 0. Fortunately, for all models, it was sufficient to use a threshold set exactly to 0, with no need to remove potential noise in the lower power range. However, it is worth noting that this option remains available if needed.

In this work, inspired by the application type and visual observations of the predicted power profiles, we present two post-processing techniques. The first involves the use of a secondary network, acting as a refinement model. The second is a more "statistical" approach based on calculating the enclosed areas under the power profiles, aimed at evaluating which model performs better and, more importantly, under which conditions. For a more coherent reading flow, these two techniques will be described here, while their application and the final power output result are presented in the following chapter. This approach was chosen because these methods are applied to the models' results rather than being part of their original structure.

4.5.1 The Power Residual-Based Refinement

The idea behind the first approach is to develop a network capable of adjusting a simulated power profile by relying once again on the actual measured power profile, which typically includes less predictable peaks. These peaks are often caused by passing clouds during an otherwise sunny day or by generally low-light conditions, which do not follow a clear irradiance and power profile. Less frequently, such peaks might be attributable to sensor malfunctions. Therefore, we do not expect to predict these peaks accurately. However, we aim to correct frequent overestimations or underestimations characteristic of specific types of days—features that we could attempt to extrapolate from the irradiance profile. Our goal is to adjust the output power predicted by a model. By defining P_{meas} as the measured power and P_{sim} as the simulated power from one of the previous models, we define res as the residual error:

$$res = P_{meas} - P_{sim},$$

which consists of both positive and negative components. If we were able to estimate the residual, denoted as $pred\_res$, we could add it to the simulated power output to better approximate the actual value. This relationship can be expressed as:

$$P_{meas} = P_{sim} + res,$$

which leads to:

$$P_{post-processed} = P_{sim} + pred\_res.$$

Without questioning the nature of the simulation results (later described and justified in Chapter 5), we consider the results of the *LSTM_f2p_96h* model and observe in Figure 4.16 the trend of the residual error.

It is important to exclude the first two daily profiles, which fully represent the recorded power curve for days 1 and 2 in the dataset, as the model (described in Section 4.4.5) starts providing predictions only from day 3 onwards. Therefore, to establish a correlation between the simulated power and the residual, it is necessary to consider only the subsequent days, starting from the third day in the dataset. Let us recall that our available input consists of the simulated power data from the *LSTM_f2p_96h* model, i.e., power profiles ranging from 0 to around 400 kW (once normalized between 0 and 1) that will be coupled in sequence with the irradiance forecast (input sequence size: 24×2). This should be transformed into the residual profile function, which oscillates between positive and negative values, making it a challenging task for a network. Thus,

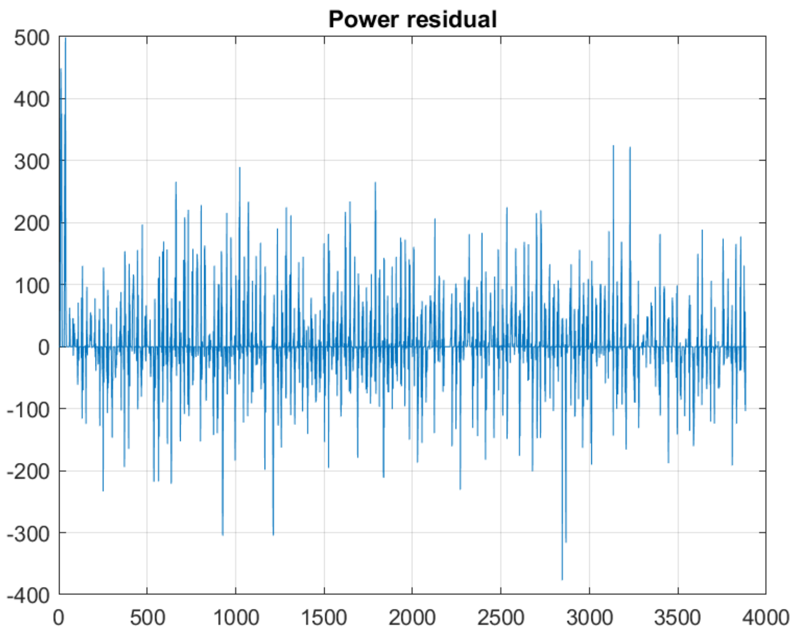


Figure 4.16. Residual error for Power prediction using the *LSTM_f2p_96h* model.

we propose breaking the problem into two sub-models corresponding to positive and negative residuals:

$$res = res^{+} + res^{-},$$

by decomposing the residual shown in Figure 4.16 into a positive and a negative component, as illustrated in Figure 4.19.

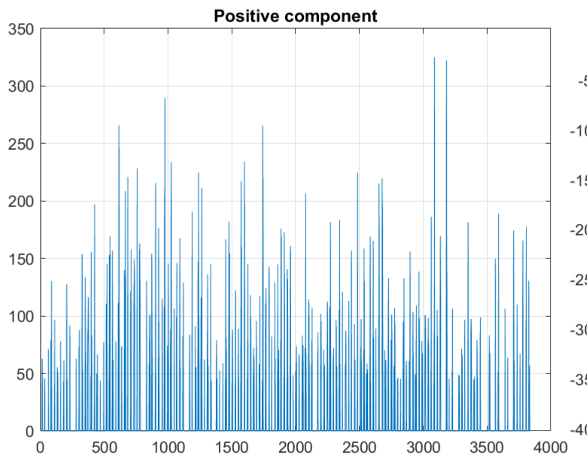


Figure 4.17. Positive component.

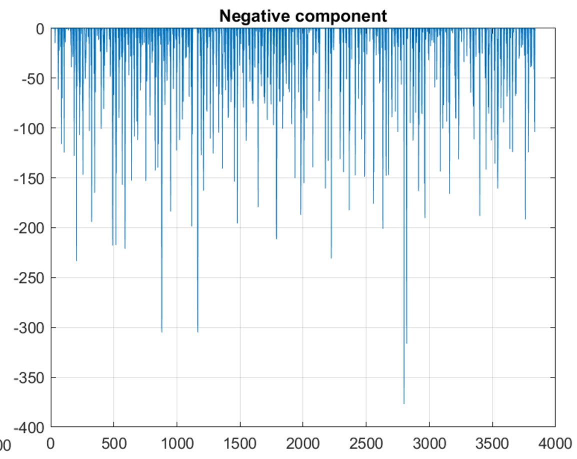


Figure 4.18. Negative component.

Figure 4.19. Decomposition of the residual into a positive and negative component.

We start with the positive component, which will be our target, and we begin with a basic structure of this type:

- 1 featureInputLayer (input layer);
- 1 lstmLayer with m cells;
- 1 fullyConnectedLayer (output layer).

We immediately notice that in this case, we are not considering different days, as logically there is no correlation between different days: while the general trend of the irradiance or power curve may be similar over several days, for peak corrections we do not expect the same mutual influence, as each day is affected differently by the passage of clouds. We test with multiples of 5 for m and look at the validation results in Table 4.12.

m	MBE	MAE	MAD	R^2	RMSE
5	5.448	17.771	1.8922	0.28473	36.736
10	5.1867	17.862	1.5601	0.288	36.652
15	4.6349	18.295	2.1624	0.28913	36.623
20	4.7903	18.17	1.8047	0.2916	36.559
25	5.2174	17.922	1.1728	0.29291	36.525

Table 4.12. KPIs during validation for multiples of 5 for m.

The results are all similar to each other, and even for large values of m, they do not improve significantly, as shown in Table 4.13.

m	MBE	MAE	MAD	R^2	RMSE
50	0.21608	15.453	1.8358	0.34356	30.244
100	-0.52696	15.497	1.6503	0.34627	30.182
150	-0.29237	15.753	1.7113	0.3348	30.445

Table 4.13. KPIs during validation for m = 50,100,150.

For $m = 150$, the residual prediction doesn't change much. It is clear that the LSTM layer alone is not able to reach the peaks of the residual function. Therefore, we add a fully connected layer with a variable number n of neurons. After numerous trials, we find that we will not achieve high peaks like those of the residual function, because this model also tries to average the solution by minimizing the mean squared error. Therefore, we keep a simple model ($n = 5, m = 15, k = 10$), whose response is shown in Figure 4.20. Similarly, the model for estimating the negative residual error with combination $n = 5, m = 15, k = 5$ was chosen, and its result is shown in Figure 4.21.

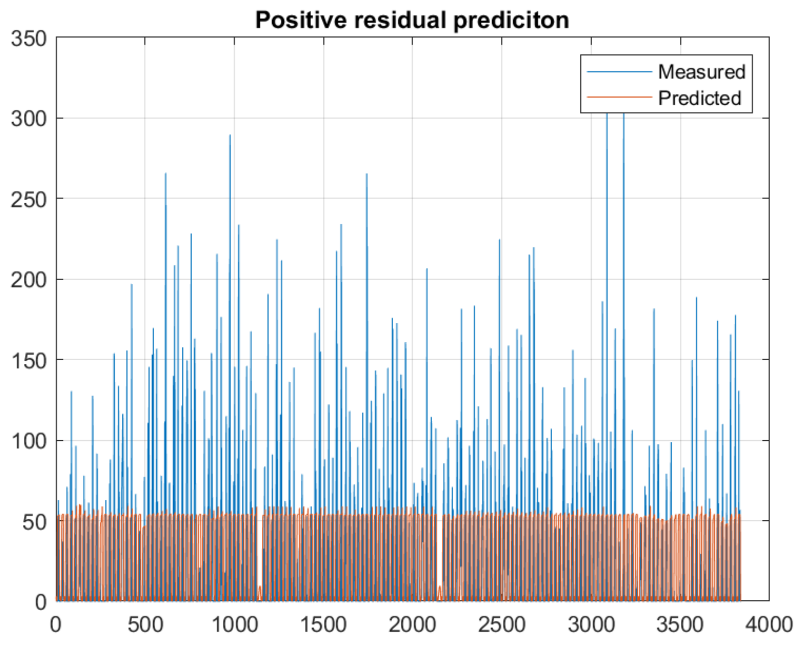


Figure 4.20. Positive residual error prediction, final model

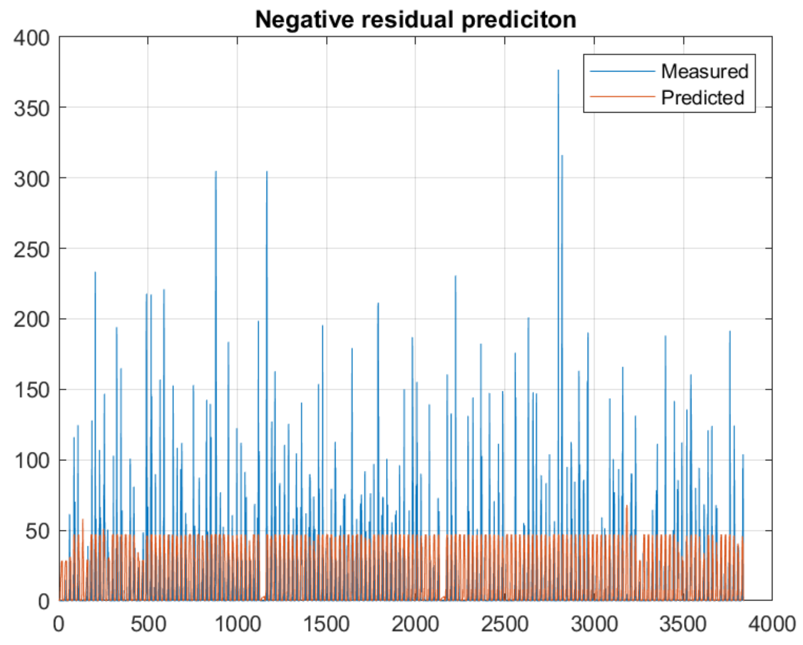


Figure 4.21. Negative residual error prediction, final model

All that remains is to sum the two components, noting that the negative one was "flipped upside down" to facilitate training, so now it needs to be flipped back downwards:

$$pred\_res = (pred\_res^{+}) + (-pred\_res^{-}).$$

The result of this operation, along with the inclusion of the residual in the power profile, is presented in Chapter 5.

4.5.2 Integrating Power Profiles for Energy Estimation and Model Performance Trends

This second post-processing technique is more statistical in nature rather than AI-based. In fact, no new model will be developed here. Instead, we will focus on reorganizing all the models discussed so far and attempt to combine them to leverage the strengths of each at the right moment, using the area-based strategy as our guiding principle. Essentially, as we know, the area under a curve corresponds to its integral. In the case of power, integrating the curve yields a quantity known as energy—in this case, solar energy. Upon reflection, while the power output and its profile are undoubtedly important, it is equally meaningful to consider energy in order to evaluate how much we are overestimating or underestimating in energetic terms and to determine if any trends can be identified from this perspective.

The method involves calculating the area under the measured power curve, which, as always, serves as our reference, on a daily basis. We then repeat this process for the power curves generated by all our models, calculating the respective areas, and compare the differences between them. In general, we will use the following nomenclature to refer to the residual energy:

$$delta\_model = A\_model - A\_meas,$$

where A_meas represents the daily energy computed from the measured power, and A_model represents the daily energy computed from the model outputs. Once we have calculated the areas for all the days in the dataset and for all models, we can analyze the relationship between A_meas and $delta\_model$. An example of this relationship is shown in Figure 4.22 for the $LSTM\_a2p\_24h$ model. To better highlight any relation or bias between the two quantities, we employ two types

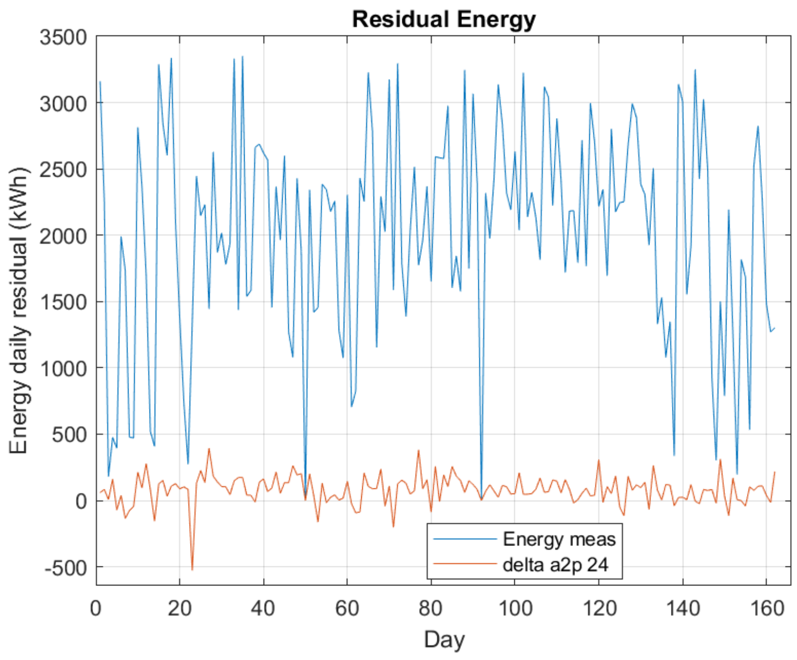


Figure 4.22. Energy residual for *LSTM_ a2p_ 24h* model.

of plots: the scatter plot and the boxchart. A description of these plots is provided below, while their implementation for all the models is detailed in Chapter 5.

The scatter plot is constructed by placing *A_meas* on the x-axis and *delta_model* on the y-axis. This allows us to visually detect any patterns. For instance, if the plot appears dispersed, there is likely no correlation, and the residual error is mostly random. Conversely, if points concentrate in a specific region, a correlation may exist, indicating a systematic error for certain energy values. In other words, we could determine that, within a specific range, the model's error tends to fall within a predictable band. An example for *LSTM_ a2p_ 24h* is shown in Figure 4.23.

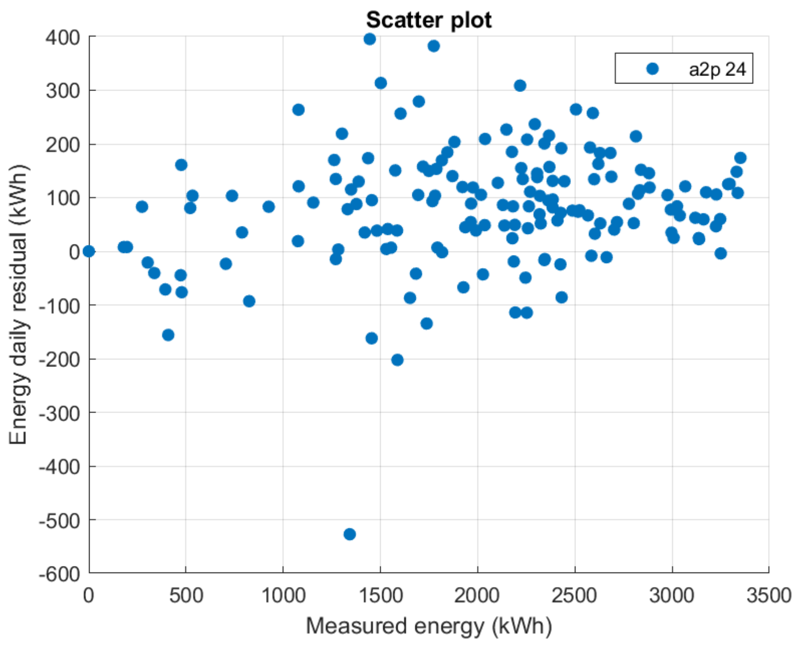


Figure 4.23. Scatter plot for *LSTM_ a2p_ 24h* model.

The boxchart provides a similar analysis but discretizes the energy ranges into predefined intervals. By grouping the data points (days) into these intervals based on measured energy *A_meas*, the boxchart illustrates how the residual error is distributed within each range. It displays the minimum, maximum, median, and any outliers (exceptional points deviating from the trend). This visualization can help identify systematic biases or outlier behavior within specific energy ranges. An example for *LSTM_ a2p_ 24h* is shown in Figure 4.24. The blue boxes represent the densest concentration of data, bounded by the lower and upper quartiles, which correspond to the thresholds below and above which 25% and 75% of the data lie, respectively. The entire vertical line represents the total data distribution and is bounded at the top and bottom by the whiskers, depicted as small horizontal lines in the figure. These whiskers indicate the true upper and lower limits, excluding outliers, which are represented as individual circles.

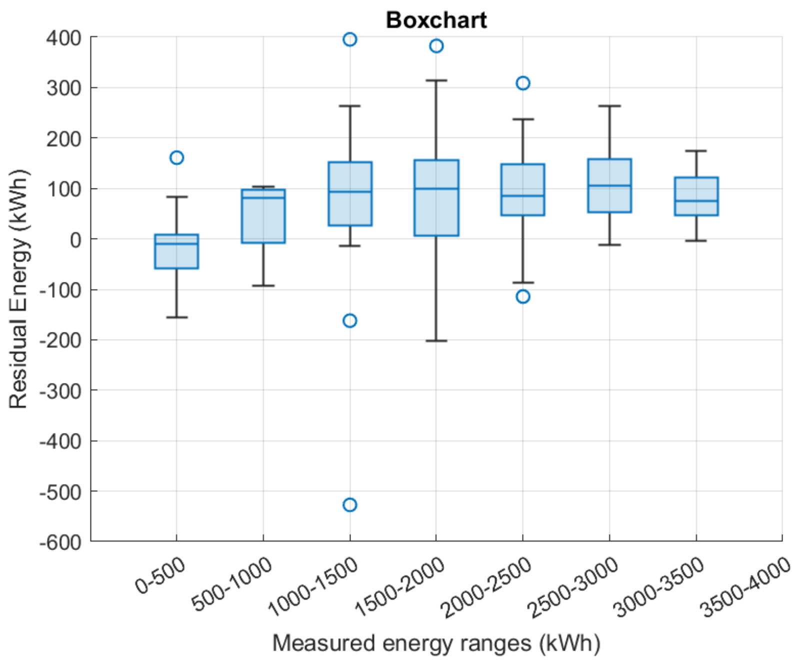


Figure 4.24. Distribution of model energy residual error across measured energy ranges for*LSTM_ a2p_ 24h* model.

Chapter 5

Results and Further Refinements

The search for the best model should not rely solely on the best performance in terms of KPIs, especially since during training these are calculated on the entire dataset. A more accurate analysis should instead investigate KPIs on a monthly and daily basis, which could be a future improvement for this work. Moreover, it is also important to consider graphical results, as visual inspection can immediately reveal potential issues.

We begin with the *FNN_f2p* model, which was constructed using only fully connected layers for different combinations of the number of neurons, up to a maximum of 3 hidden layers. During training and observing the validation KPIs, we arrived at the following configuration:

- 1 featureInputLayer (input layer);
- 1 fullyConnectedLayer with n neurons;
- 1 fullyConnectedLayer with 15 neurons;
- 1 fullyConnectedLayer with k neurons;
- 1 fullyConnectedLayer (output layer).

We identified good performance in the models $n = 0, m = 15, k = 2$ and $n = 5, m = 15, k = 15$.

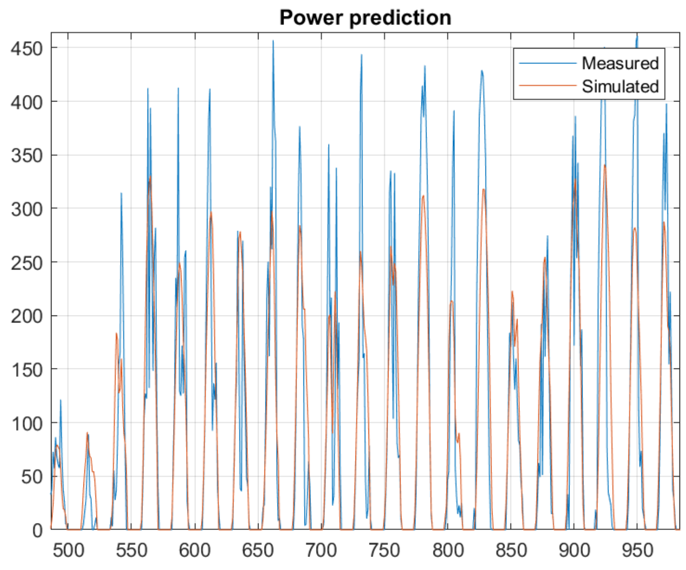


Figure 5.1. Power output with $n = 0, m = 15, k = 2$.

Let us examine the behavior of the first combination ($n = 0, m = 15, k = 2$) in Figure[1] 5.1. At first glance, it seems that the model is underestimating the target power curve. However, what it is actually doing is following the forecasts and attempting to return values similar to the target ones. Thus, a clearer and more intrinsic correlation exists, aiming more at smoothing and mediating the power curve rather than precisely replicating the real profile, which often exhibits a less smooth line. To ensure the reliability of the model, let us analyze three days (Figure 5.2) and their KPIs in Table 5.1.

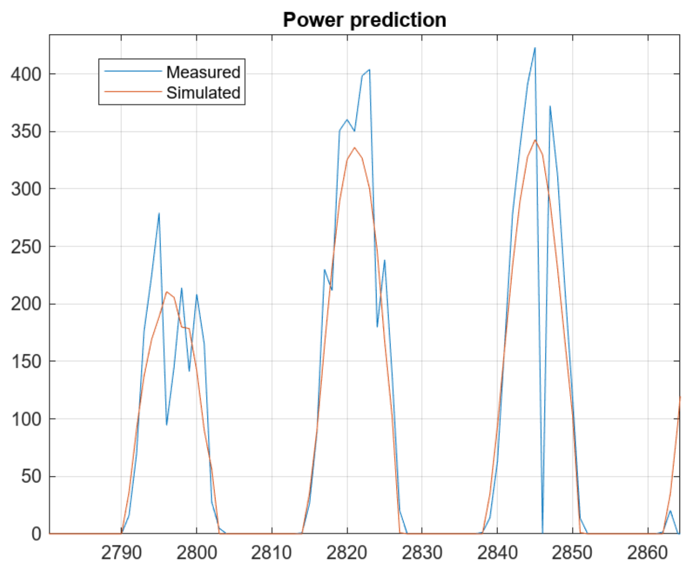


Figure 5.2. Power output for 3 days with $n = 0, m = 15, k = 2$.

Error	Day 1	Day 2	Day 3
MBE	3.4092	15.9039	4.0714
MAE	27.0102	23.9954	35.8341
MAD	12.4661	5.3152	8.3165
R^2	0.7815	0.9310	0.7414
RMSE	42.8674	39.3220	76.6323

Table 5.1. Error Table for 3 days with $n = 0, m = 15, k = 2$.

Let us make some general considerations applicable to all models. Regarding the R^2 KPI, it is expected not to achieve values within a high range (above 0.8) for this type of application. This is primarily due to the fact that the power profile often exhibits sudden drops, likely caused by passing clouds. Consequently, this indicator will tend to be higher in the absence of such events, as observed in the central day shown in Figure 5.2. Concerning the MBE, MAE, and MAD values, we aim for these deviation thresholds and the mean error to be as low as possible; the same applies to the root mean square error (RMSE). Nevertheless, as we will see for other models, these results, given the influence of variability in the power curve relative to the irradiance forecast, are already satisfactory.

Let us now test the model with $n = 5, m = 15, k = 5$, whose power output is shown in Figure 5.3. As in the previous case, it appears that the model's output profile underestimates the power curve, a tendency confirmed by the MBE. To further analyze its performance, we examine four specific days (Figure 5.4) and their corresponding KPIs, as summarized in Table 5.4.

[1]In these figures, the x-axis represents the data according to their row position in the dataset, while the y-axis indicates the quantity being shown, which is typically either the power produced (expressed in kW) or the irradiance (W/m^2).

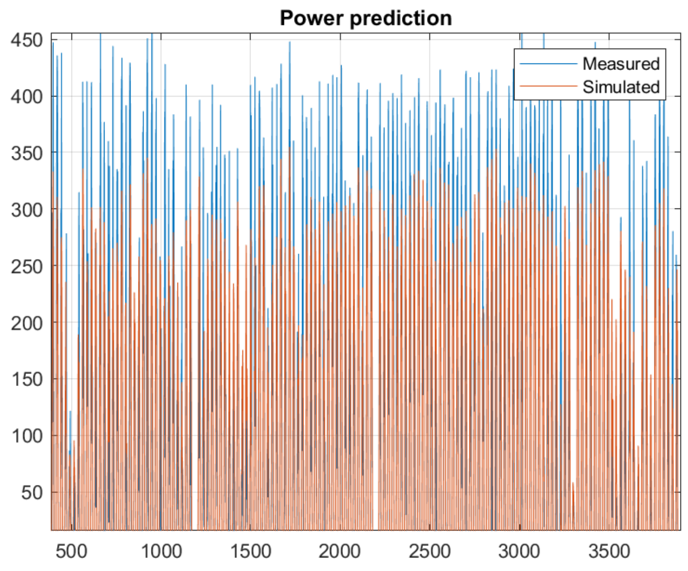


Figure 5.3. Power output with $n = 5, m = 15, k = 5$.

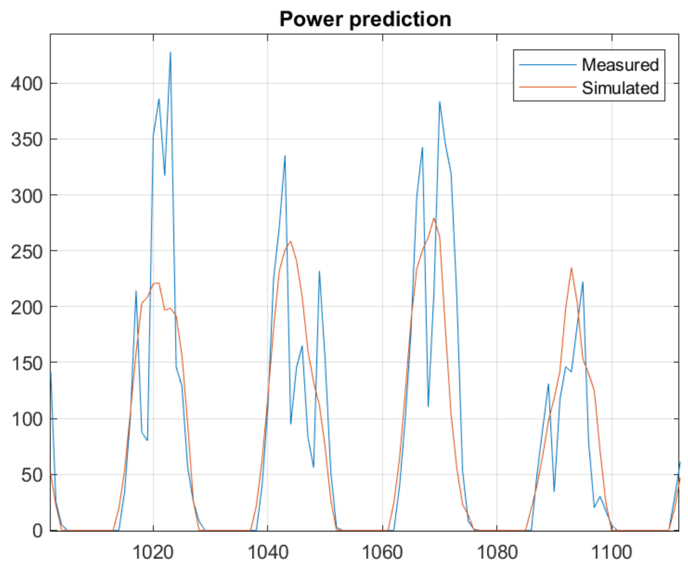


Figure 5.4. Power output for 4 days with $n = 5, m = 15, k = 5$.

Error	Day 1	Day 2	Day 3	Day 4
MBE	14.3833	-3.7835	23.1723	-14.5359
MAE	45.9371	37.3932	47.5048	26.8080
MAD	11.8047	19.3277	14.1899	14.5499
R^2	0.6538	0.6535	0.6485	0.6124
RMSE	79.4144	58.4484	80.2508	41.7803

Table 5.2. Error Table for 4 days with $n = 5, m = 15, k = 5$.

It seems that the KPIs are worse compared to the previous ones, so we will keep the earlier combination. However, let us remember that each day should be considered individually, and for now, our focus is on obtaining results that are faithful to the irradiance forecasts, meaning that the models should be able to smooth the power profiles. Later, some post-processing methods will be presented to refine the resulting curves from the models.

When running the *LSTM_ a2p_ 24h* model with $m = 15$, the performance seems good in terms of KPIs, but looking at the graphical representation (Figure 5.5) over the entire dataset[2] 1, it becomes evident that the model fails to reach the height of the power peaks, indicating that the model needs further refinement.

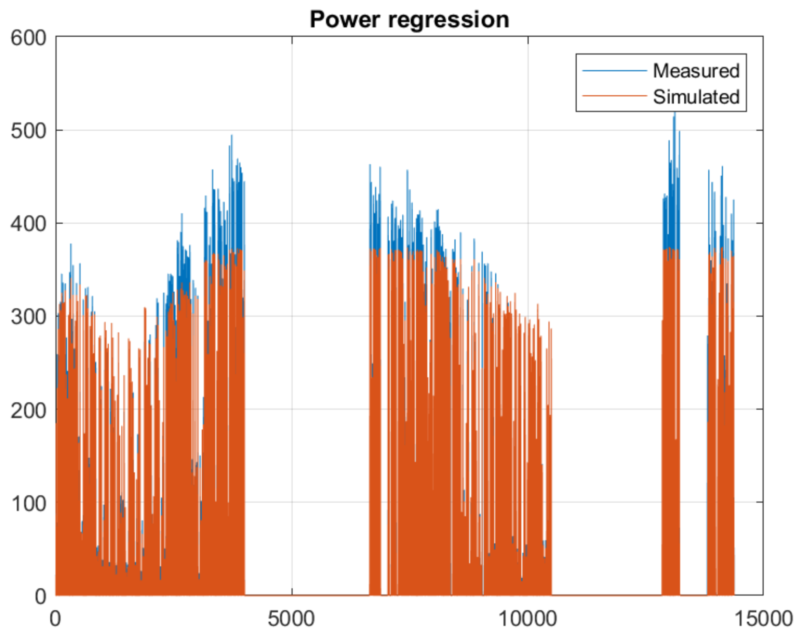


Figure 5.5. Power output on the entire dataset 1 (m=15).

By further increasing the value of m, the model becomes increasingly sensitive to variations in the irradiance profile, which are reflected in the power curve. However, a clear scaling issue persists, requiring the introduction of additional fullyConnected layers. For this reason, the final structure of the model is as follows:

- 1 sequenceInputLayer (input layer);
- 1 LSTM layer with 15 cells;
- 1 fullyConnectedLayer with 5 neurons;
- 1 fullyConnectedLayer (output layer).

With the addition of the hidden layer containing 5 neurons, the model is now able to reach the peaks (Figure 5.6), while maintaining the initial accuracy provided by the LSTM cells (Figure 5.7). The model appears to be very accurate, so we tested it on new data from dataset 2. By providing the irradiance and temperature measurements as input, the excellent performance of the model is confirmed even on the second dataset (Figure 5.8), which demonstrates that the model has successfully characterized the system.

[2]It is worth noting that entire empty periods have been removed from the dataset for the reasons mentioned in Section 4.1.

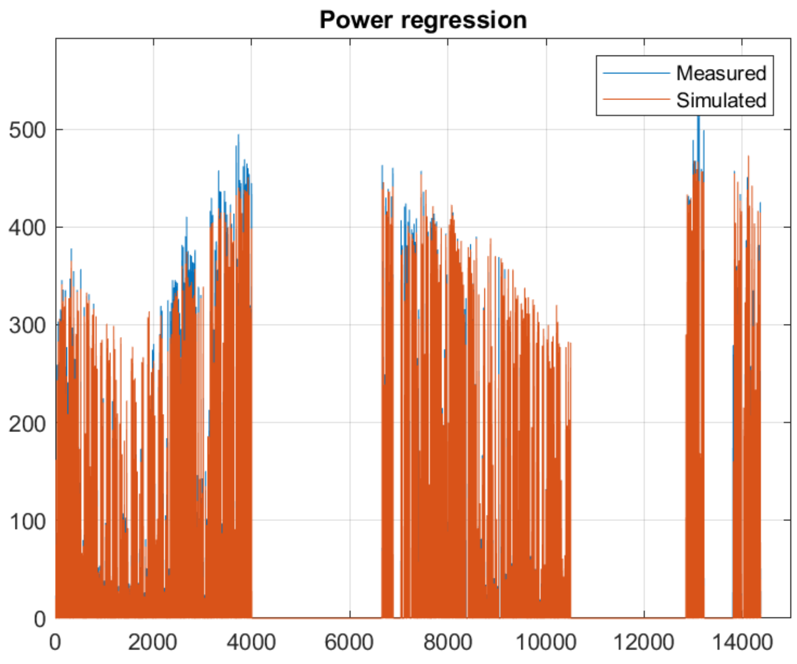


Figure 5.6. Power output on the entire dataset 1 (final model).

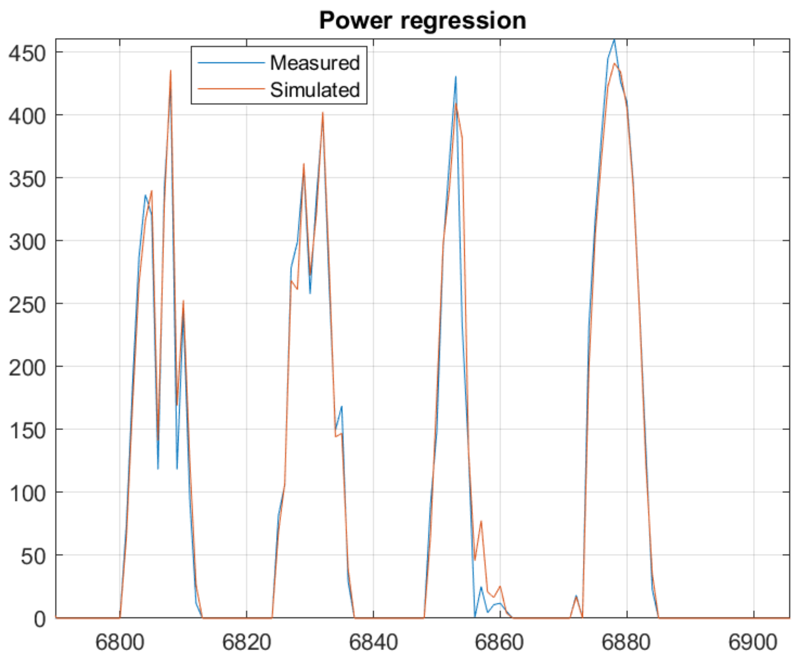


Figure 5.7. Power output on measured inputs from database 1 (final model).

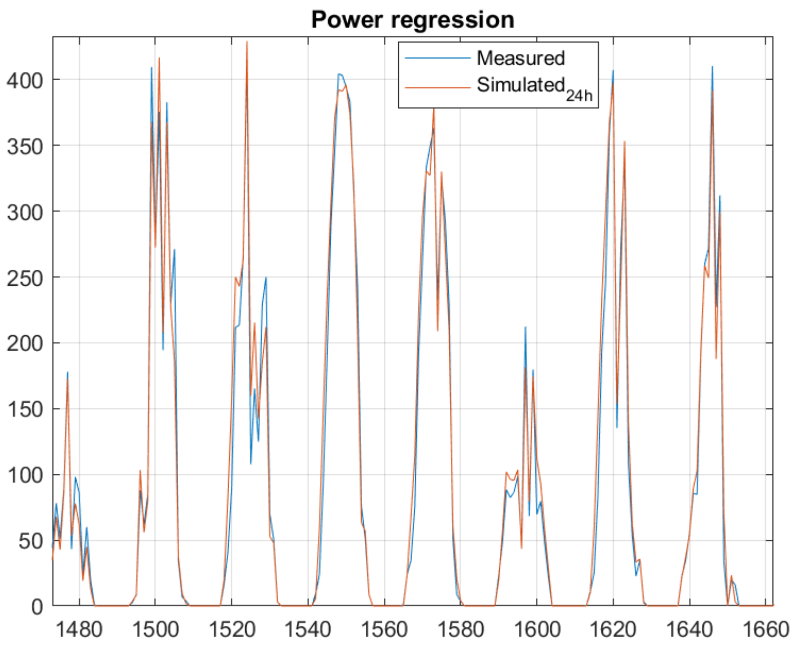


Figure 5.8. Power output on measured inputs from database 2 (final model).

However, it is important to remember that our main goal is to make predictions, which means we must account for the error introduced by the forecasts of meteorological conditions, as they are essentially estimates. This uncertainty is reflected in the resulting power output, as shown in Figure 5.9, where the model was fed with forecast sequences.

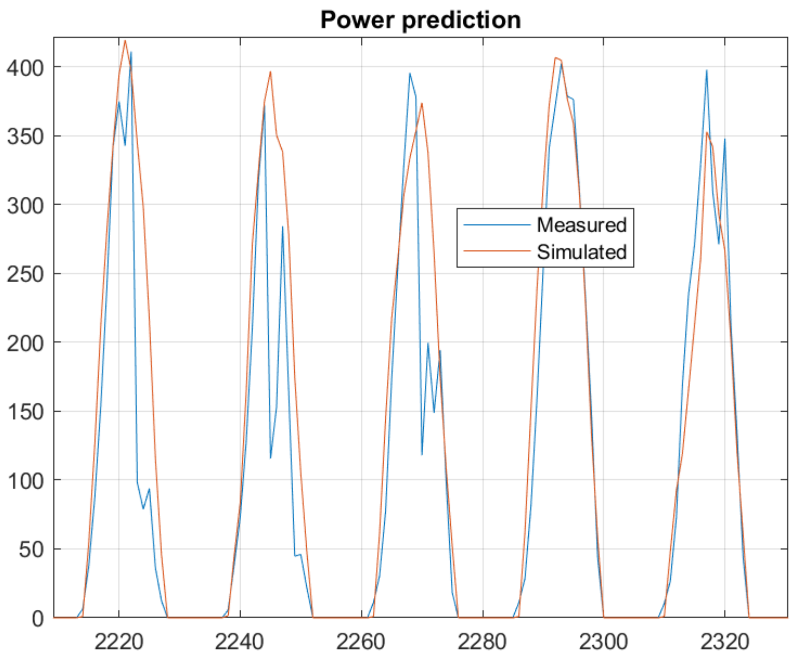


Figure 5.9. Power output on forecasts inputs from database 2 (final model).

The model's performance degrades, although it still manages to capture the overall trend and approximate or average the power curve. To analyze this further, we examine some samples from a clear and a cloudy day, comparing the model results when measured and forecast input sequences are provided. Figure 5.10 illustrates this comparison[3] for a clear day, while Figure 5.11 does so for a cloudy day. A significant improvement in the daily KPIs can be observed when real measurements are used as input (Tables 5.3 and 5.4).

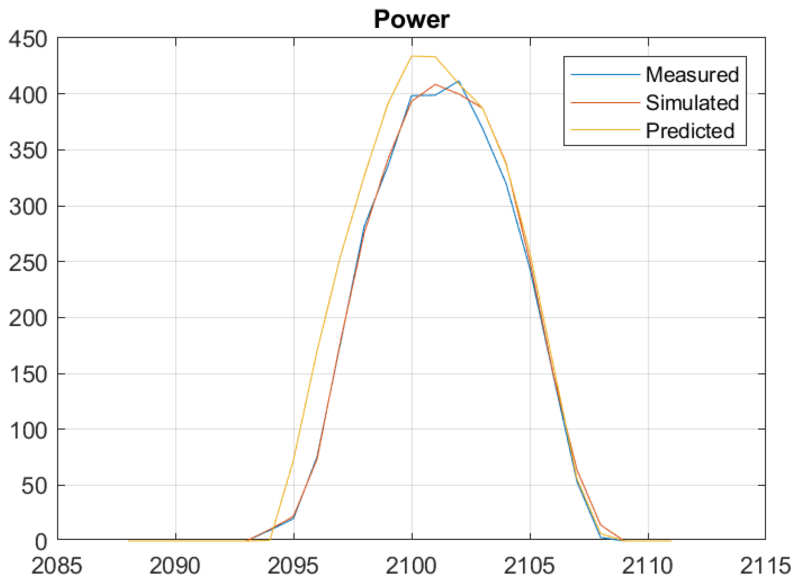


Figure 5.10. Clear day comparison (final model).

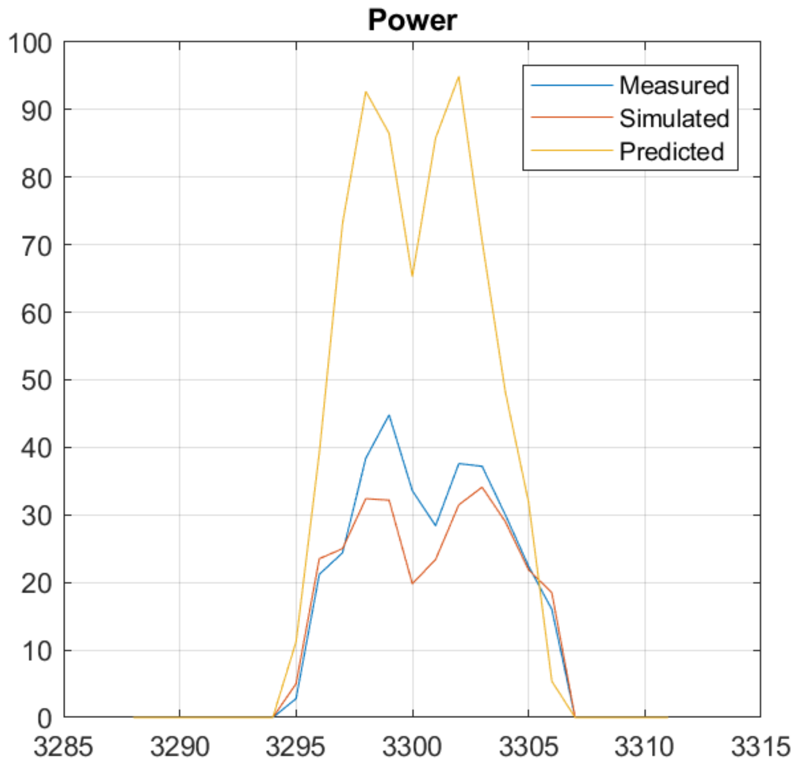


Figure 5.11. Cloudy day comparison (final model).

[3]Here, we refer to the results obtained by using measurements as the "Simulated" output, while the "Predicted" output corresponds to the case where forecasts are provided as input.

Error	Predicted	Simulated
MBE	-18.6555	-2.5013
MAE	19.6217	4.5409
MAD	6.2497	1.7052
R^2	0.9563	0.9979
RMSE	33.0951	7.2462

Table 5.3. Error Table for a clear day.

Error	Predicted	Simulated
MBE	-15.3335	1.6902
MAE	16.2197	2.3209
MAD	4.1968	0.2653
R^2	-1.6936	0.9234
RMSE	26.3762	4.4479

Table 5.4. Error Table for a cloudy day.

Nonetheless, the fact that the results are excellent when using measured data as input inspired the analysis to take the following approach: to attempt a correction of the forecasts with a dedicated correction model, aiming to make the meteorological forecasts as close as possible to the measured values of irradiance and temperature. As suggested by the theoretical framework, we aimed to ensure that the irradiance value was the primary factor influencing the power curve. To validate this, we simulated a hybrid model combining the prediction based solely on temperature with the actual measured irradiance values. Figure 5.12 shows that the model performs exceptionally well when using the measured irradiance and the forecast temperature.

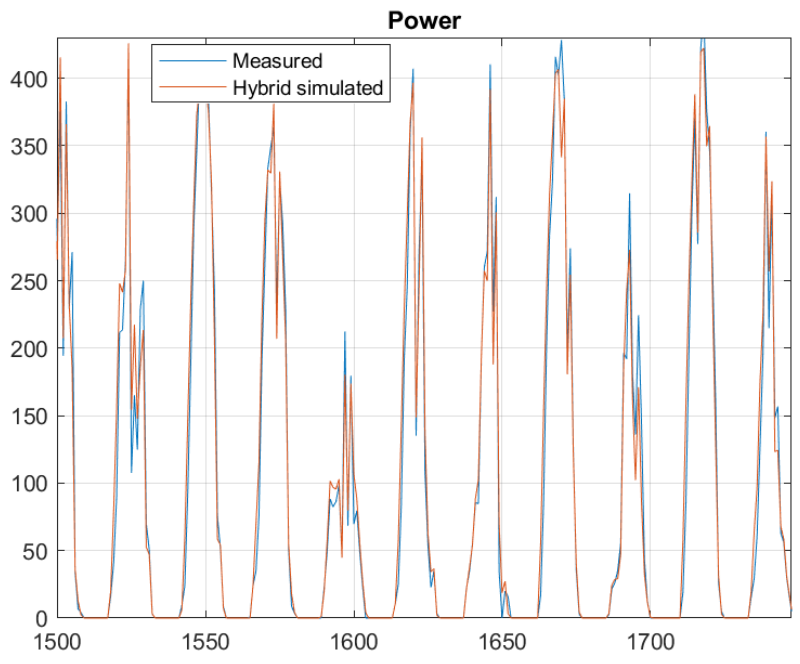


Figure 5.12. The impact of irradiance.

This may indicate two things: the temperature forecasts are highly accurate and/or temperature does not have a significant impact on the calculation of the produced power value. Referring back to the previous case of a cloudy day (Figure 5.11), we now add the newly obtained power curve (Figure 5.13) and compare the KPIs in Table 5.5.

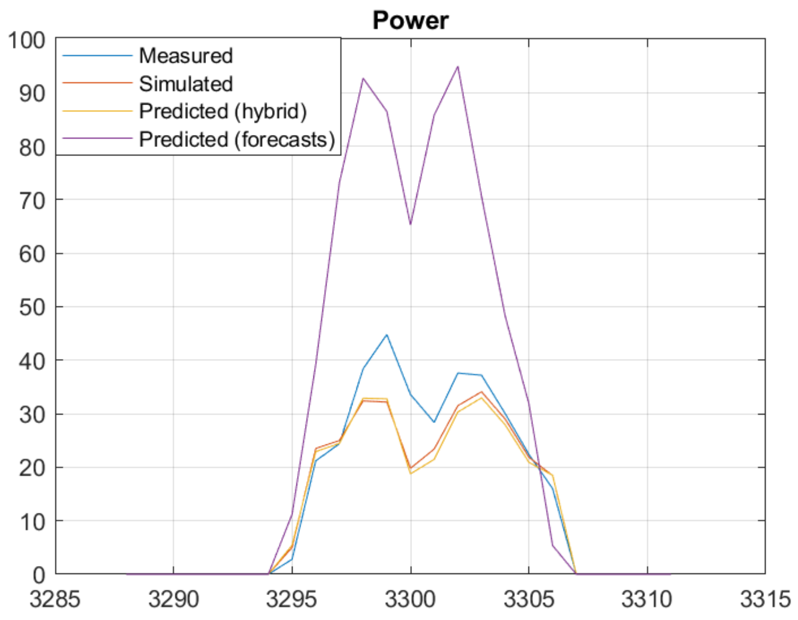


Figure 5.13. Cloudy day, the impact of irradiance.

Error	Predicted (forecasts)	Predicted (hybrid)	Simulated
MBE	-15.3335	1.9752	1.6902
MAE	16.2197	2.5524	2.3209
MAD	4.1968	0.0342	0.2653
R^2	-1.6936	0.9136	0.9234
RMSE	26.3762	4.7243	4.4479

Table 5.5. Error Table for a cloudy day, the impact of irradiance.

It is evident that irradiance has the greatest weight on the power value, as in this case the model shows almost no variations when using only the temperature forecasts compared to the fully observed case. Indeed, some KPIs (more thoroughly described in terms of forecast accuracy in Section 4.2) from the irradiance forecasts already exhibited a worse performance in terms of magnitude, yet outperformed temperature forecasts in capturing the overall variance of the data. Conversely, the temperature forecasts demonstrated excellent accuracy in minimizing individual point-wise errors. Therefore, we now shift our focus to correcting the irradiance predictions.

At this point, the *LSTM_f2c_24h* model has been developed with the aim of improving irradiance forecasts, bringing them closer to the values later recorded through measurements. As discussed in the previous chapter (Section 4.4.3), finding a good combination of hyperparameters is certainly not an easy task. Moreover, as previously mentioned, the model with the best performance during validation does not necessarily perform well on the test set or the entire irradiance forecast dataset. Using the best validation model from Section 4.4.3, we examine in Figure 5.14 the type of relationship the model has learned to perform the correction. It is immediately evident that the model tends to lower the forecast curve, which is usually very smooth, in an attempt to mediate the alternation of unexpected peaks that are instead observed in the measured data. This type of correction improves forecasts on cloudy days (Figure 5.15) but penalizes accuracy on clearer days. Table 5.6 presents this in terms of daily KPIs, corresponding to a clear day around the x-axis value of 2100 in Figure 5.14 and a cloudy day around the x-axis value of 85 in Figure 5.15. The challenge for the model lies in discerning when the actual irradiance curve will deviate in shape from the forecast and when it will not. On the other hand, is it truly possible to estimate such behavior? It is important to note that for high irradiance profiles, a clear day often exhibits sudden peaks caused by transient clouds at the time of sensor measurement, which is entirely unpredictable with current tools. This is actually the main reason why, in Section 4.4.3, a model that reduced the mean squared error (RMSE) was preferred over one that, by chance, exhibited a good correlation index (R^2).

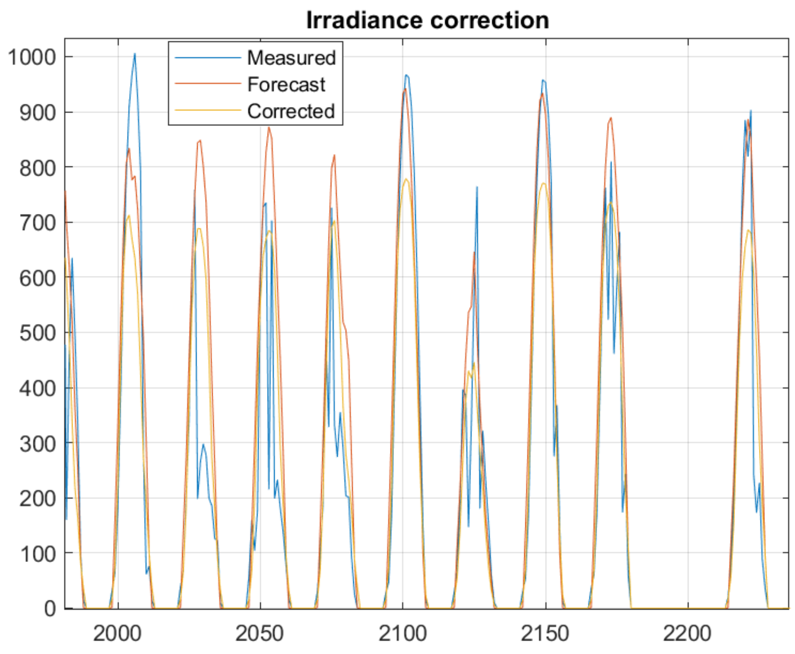


Figure 5.14. Irradiance correction.

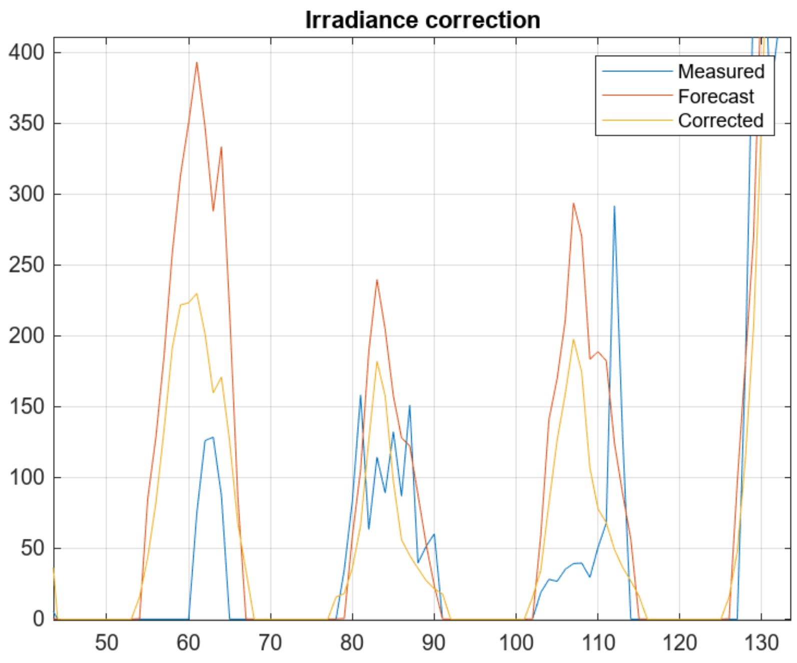


Figure 5.15. Irradiance correction on cloudy days.

The next step is to see how such improvements or degradations affect the power curve. We consequently expect that the power curve improves on days that, according to the forecast, will already have low profiles, while showing a penalized behavior on the clearest days[4]. It remains to

[4]Indeed, completely clear days, which are the ones when the power output is penalized the most, are probably

Error	Forecast	Corrected	Forecast	Corrected
MBE	4.6644	47.1724	-12.7309	6.7756
MAE	52.4341	59.6848	27.5347	26.1605
MAD	26.7164	20.7736	0.8230	16.0895
R^2	0.9586	0.9363	0.1561	0.3938
RMSE	76.0037	94.2858	48.4439	41.0601
	Clear day		Cloudy day	

Table 5.6. Error Table: Forecast and Correction comparison for a clear and a cloudy day.

be seen whether, on clear days affected by passing clouds, the 'averaging' correction is beneficial. Let us use the resulting irradiance correction in the *LSTM_a2p_24h* model[5] to examine the power curve. Figure 5.18 and Figure 5.21 show the translation of the irradiance correction into the power profile for a cloudy day and a clear day with passing clouds, respectively.

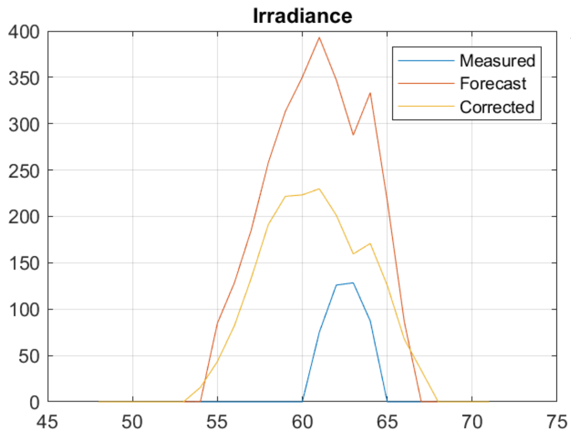


Figure 5.16. Irradiance correction.

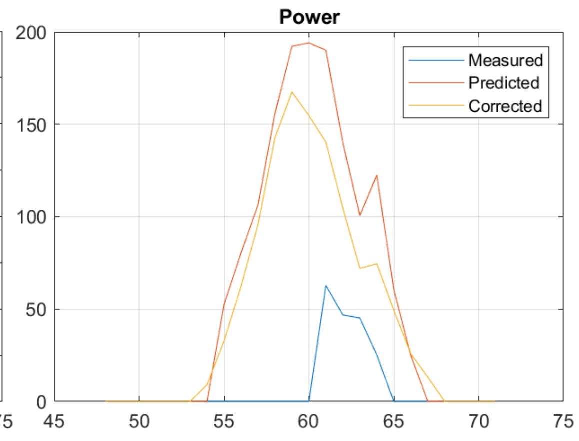


Figure 5.17. Power.

Figure 5.18. Irradiance Correction and corresponding Power for a cloudy day.

Let us refer to Table 5.7 to compute the KPIs for Figure 5.21 and compare those associated with the irradiance correction to the ones corresponding to the related power.

Error	Forecast	Corrected	Predicted	Corrected
MBE	-96.0206	-47.7629	-32.7360	-24.2396
MAE	103.6515	74.4785	36.5819	30.5125
MAD	3.0581	0.6336	2.8845	15.4655
R^2	-0.2057	0.4542	0.7652	0.8537
RMSE	193.0736	129.9089	66.5697	52.5546
	Irradiance		Power	

Table 5.7. Error Table: Irradiance and Power KPIs after correction for a clear day with passing clouds.

It seems that the *LSTM_f2c_24h* model is effective in improving the final power prediction for both cloudy days and sunny days with passing clouds. What we have wondered at this point is

the least frequent in the database. In contrast, clear days disturbed by passing clouds are much more common.

[5]If interested in the more technical aspects related to the code, note that the irradiance correction model *LSTM_f2c_24h* outputs a sequence of 24 data points corresponding to one day. To use this correction in the *LSTM_a2p_24h* model, the input sequences (24×2) must be reconstructed by placing the temperature forecast in the first column and the irradiance correction in the second column, using the *sequence_extraction_var* function (Section 4.3.4).

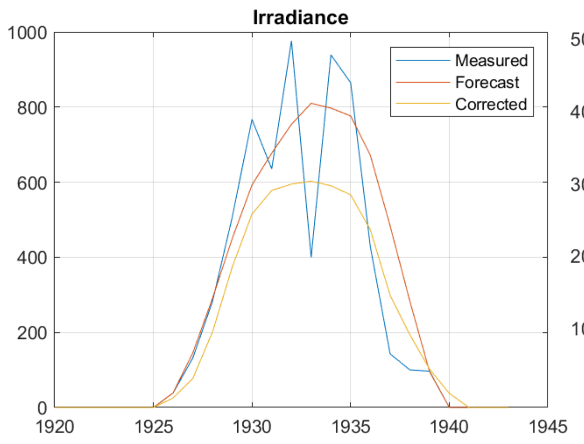


Figure 5.19. Irradiance

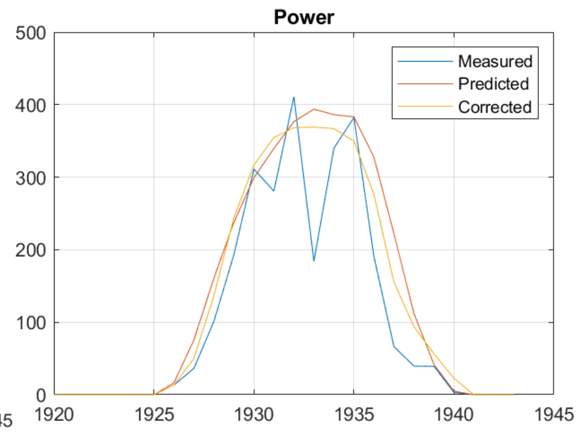


Figure 5.20. Power.

Figure 5.21. Irradiance Correction and corresponding Power for a clear day with passing clouds.

whether a new irradiance correction model, expanding the input data window beyond just a single day's forecast, could provide a more accurate correction. For example, considering the three days preceding the forecast day. It's as if we are asking the model to look at what happened in those three days, because we reasonably expect the power curve behavior to be similar for consecutive days. In other words, we aim to emphasize the seasonality even more and verify if there is truly a trend over a few consecutive days. This is how the *LSTM_f2c_96h* model came to be.

Once we have understood the reasoning beyond its working principle in Section 4.4.4, we are ready to look at some results and comparison with the *LSTM_f2c_24h* model. A value of $m = 50$ is chosen, results are shown in Figure 5.22 for a couple of days, while the daily KPIs for the two days are reported in Table 5.8.

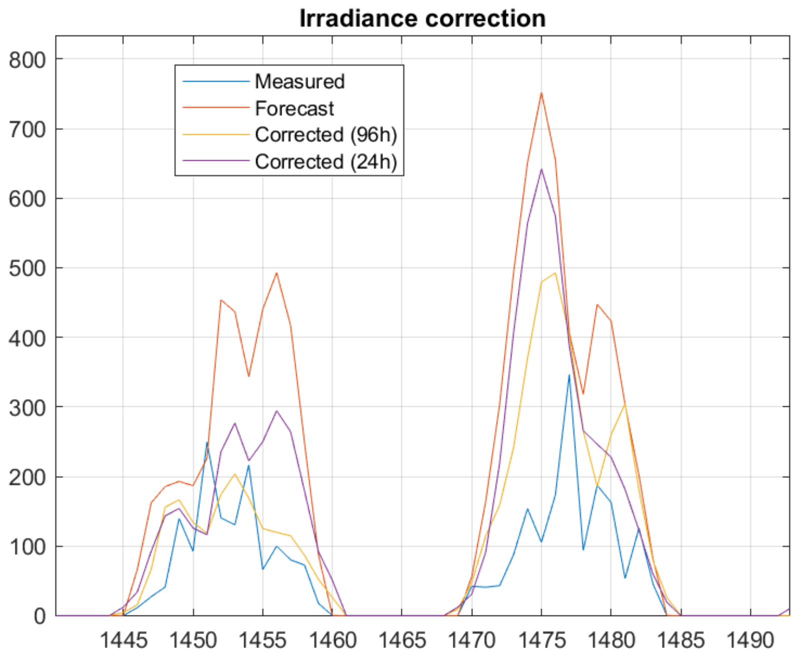


Figure 5.22. Irradiance correction: models comparison.

Error	Forecast	Corrected 24h	Corrected 96h	Forecast	Corrected 24h	Corrected 96h
MBE	-106.4189	-48.2410	-14.4528	-149.8338	-99.4624	-81.5164
MAE	108.3839	59.3610	29.3598	149.8338	100.7077	81.6929
MAD	54.1278	27.9363	22.8339	48.0129	16.5335	29.7932
R^2	-4.7424	-0.5452	0.5909	-7.0693	-3.6435	-1.5364
RMSE	171.5207	88.9722	45.7799	241.8879	183.4932	135.6136
		Day 1			Day 2	

Table 5.8. Error Table: Irradiance correction comparison between 24h and 96h models.

As with the previous model, we now report the irradiance correction in the *LSTM_ a2p_ 24h* model for power calculation. What immediately stands out is that now the power profile tends to be overestimated on the best days, as is evident in Figure 5.23. We therefore need to refine the model and improve it, especially since we have not yet included any fully connected layers. Let us consider the following structure, for which several combinations of the values n, m, and k are reported in Table 5.9 in terms of KPIs:

- 1 sequenceInputLayer (input layer);
- 1 fullyConnected layer with n neurons;
- 1 LSTM layer with m cells;
- 1 fullyConnected layer with k neurons;
- 1 fullyConnectedLayer (output layer).

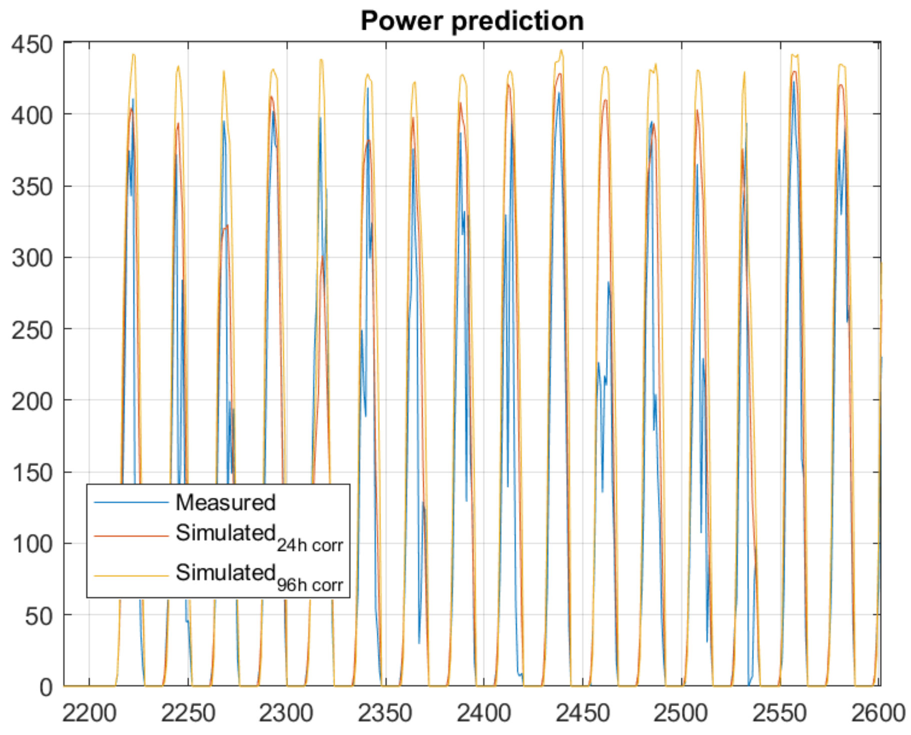


Figure 5.23. Power after irradiance correction based on 96h.

When the value of n is set to 15 and m = 25, a maximum of 200 epochs is sufficient to reach a minimum for the Loss function. With respect to Table 5.9, the horizontal line marks the point where the minibatch size was changed. Until this point, it was set to 32; however, considering the intrinsic complexity of this model and the relatively high amount of data (Dataset

n	m	k	MBE	MAE	MAD	R^2	RMSE
5	25	0	69.465	108.6	20.614	0.60014	199.43
10	25	0	52.454	93.418	11.699	0.61967	177.55
15	25	0	24.116	92.754	15.844	0.6187	169.01
15	25	5	0.4817	68.245	18.819	0.79534	121.17
15	25	10	-12.12	63.372	12.664	0.76991	122.42
15	25	15	5.2048	59.52	13.251	0.85481	112.25
20	25	15	-8.0805	73.871	14.085	0.75268	142.41
20	30	15	8.0609	72.29	15.421	0.8075	132.98
20	25	15	2.2667	57.848	12.064	0.86826	110.11
20	**25**	**20**	**7.4842**	**76.732**	**10.652**	**0.71908**	**154.46**

Table 5.9. KPIs during validation for distinct values of n, m, k.

2 contains 3887 rows), it could be a good choice to increase this parameter to speed up the training process. The last row, highlighted in bold, does not stand out in terms of performance compared to other combinations of n, m, and k in the validation set. However, when performance is evaluated on the test set instead of the validation set, and by examining the irradiance correction profile (Figure 5.24), it appears to perform better than the other scenarios. This type of analysis highlights once again that relying solely on validation KPIs is neither the only nor the most effective tool for identifying the best model. This is because validation sets are never identical and are randomly extracted from the general dataset. Moreover, in this case, there is an unknown phenomenon of error amplification between models, which sometimes exhibits counterintuitive behavior with respect to the KPIs.

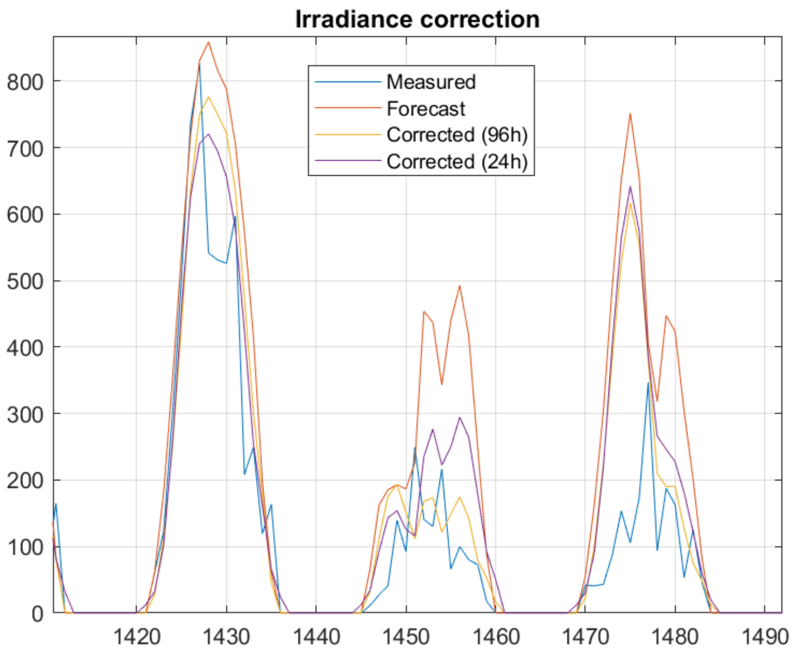


Figure 5.24. Irradiance correction with 96h.

Figures 5.27 and 5.30 show the translation of the irradiance correction onto the power profile, comparing it with both the forecasts and the corrections made by the *LSTM_f2c_24h* model for a cloudy day and a clear day with passing clouds, respectively. The improvements in the KPIs are presented in Table 5.10 and Table 5.11 for the two scenarios, respectively.

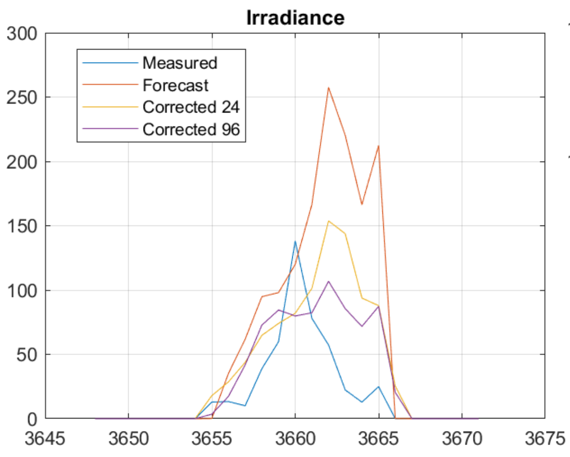


Figure 5.25. Irradiance correction.

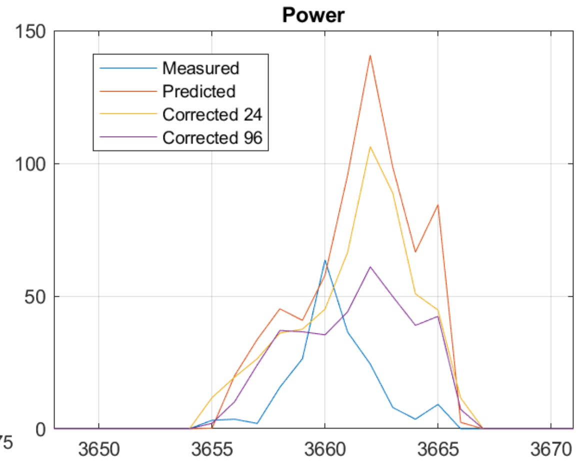


Figure 5.26. Power.

Figure 5.27. Irradiance Correction and corresponding Power for a cloudy day.

Error	Forecast	Corr (24h)	Corr (96h)	Predicted	Corr (24h)	Corr (96h)
MBE	-40.1853	-18.6844	-11.8860	-20.4020	-14.5323	-8.0256
MAE	42.7670	23.3713	17.5299	21.1327	16.0764	10.4717
MAD	0.0845	2.4842	2.0418	1.2024	4.2688	0.5701
R^2	-4.8955	-0.5769	0.2153	-5.7722	-2.5575	-0.3527
RMSE	80.0962	41.4242	29.2208	39.3726	28.5366	17.5966
	Irradiance			Power		

Table 5.10. Error Table: Irradiance and Power KPIs after correction for a cloudy day.

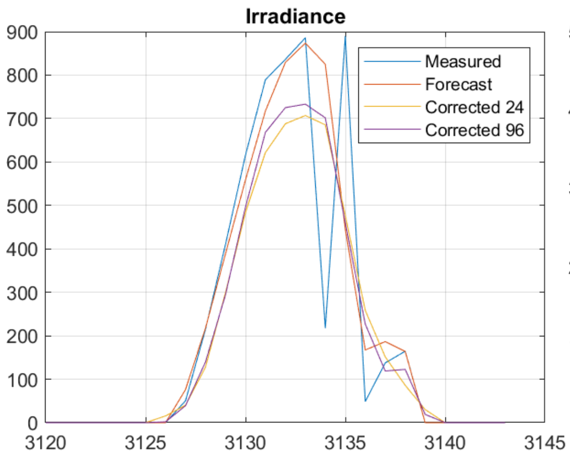


Figure 5.28. Irradiance

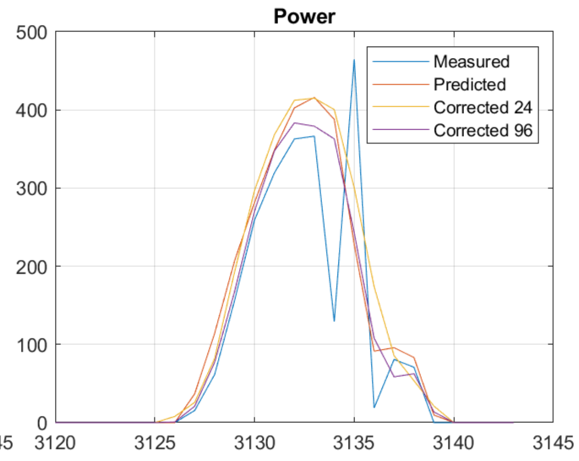


Figure 5.29. Power.

Figure 5.30. Irradiance Correction and corresponding Power for a clear day with passing clouds.

Error	Forecast	Corr (24h)	Corr (96h)	Predicted	Corr (24h)	Corr (96h)
MBE	-7.8405	24.5469	21.4159	-16.5541	-22.1104	-7.9187
MAE	59.1263	85.9739	78.3610	36.2913	37.2395	28.9583
MAD	0.2717	14.7252	14.9264	11.0697	9.0874	6.8364
R^2	0.7566	0.7702	0.7796	0.7126	0.7208	0.7647
RMSE	157.2352	152.7836	149.6269	76.3520	75.2529	69.0752
	Irradiance			Power		

Table 5.11. Error Table: Irradiance and Power KPIs after correction for a clear day with passing clouds.

Finally, the *LSTM_f2p_96h* model was developed with a different approach compared to the correction-based methods. In this model, the training directly targets the relationship between the forecasts of both temperature and irradiance and the measured power values. The considered time window spans three days of measured data plus one day of forecasts. Specifically, if the code were to be updated and executed up to day d, it would require temperature and irradiance measurements up to day d and temperature and irradiance forecasts for day $d + 1$[6].

After reviewing the logic behind the construction of the *LSTM_f2p_96h* model in Section 4.4.5, we present in Figure 5.31 the first power results corresponding to the basic structure with a single LSTM layer containing 25 cells, a minibatch size set to 100, and a maximum of 2000 epochs.

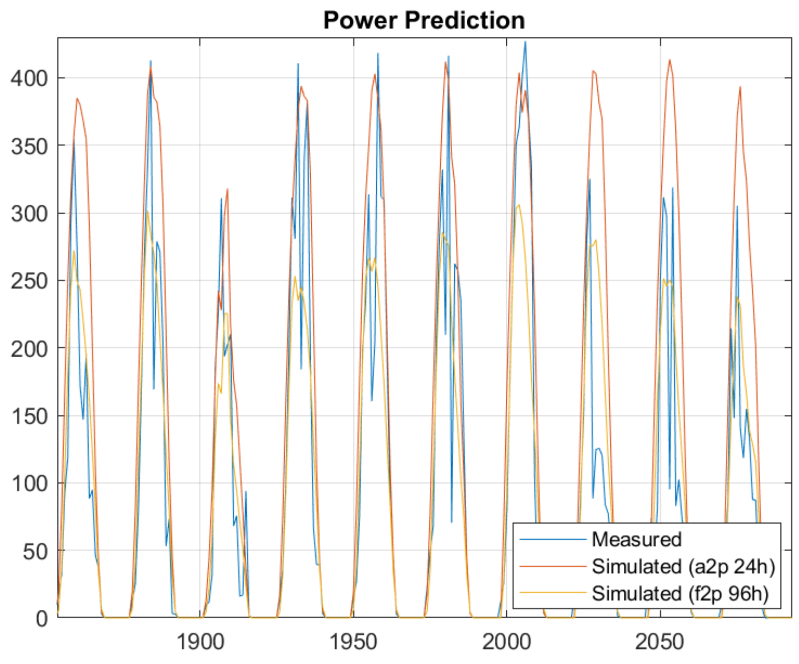


Figure 5.31. Power prediction with $m = 25$.

[6]In other words, to meet these two conditions, the code could be executed around 10:00 PM on day d to obtain the power forecast for day $d + 1$. At this time, the meteorological conditions for day d would be fully known: irradiance at 10:00 PM will certainly be zero, as sunset has already occurred, while temperature has minimal influence and could be replaced with symbolic values that would not compromise the power estimation. The forecasts for day $d + 1$, according to the methodology used in all models discussed in this work, would already be available from noon on day d

What stands out compared to the *LSTM_ a2p_ 24h* model is that, unlike the latter which is trained on past data and used for forecasting, tightly linked to the irradiance prediction as if it were the true value, this model is able to account for how the forecast changes relative to the measured power. It observes the actual conditions over the previous three days and then estimates the power for the fourth day, which is the forecast day. It appears that the yellow curve is able to effectively smooth the measured power curve, whereas the red curve tends to overestimate it. We calculate KPIs to compare the two models for a cloudy day (Figure 5.32) and a clear day with passing clouds (Figure 5.33). The technical results are presented in Table 5.12 and Table 5.13, respectively.

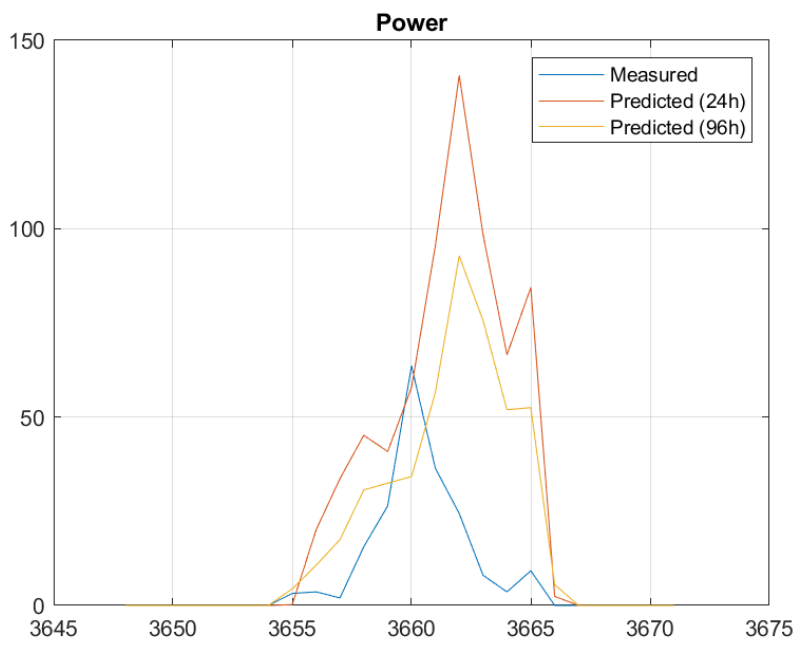


Figure 5.32. Power prediction comparison for a cloudy day.

Error	Predicted (24h)	Predicted (96h)
MBE	-20.4020	-11.2010
MAE	21.1327	13.6501
MAD	1.2024	0.6057
R^2	-5.7722	-1.7877
RMSE	39.3726	25.2611

Table 5.12. Error Table: Power KPIs for a cloudy day.

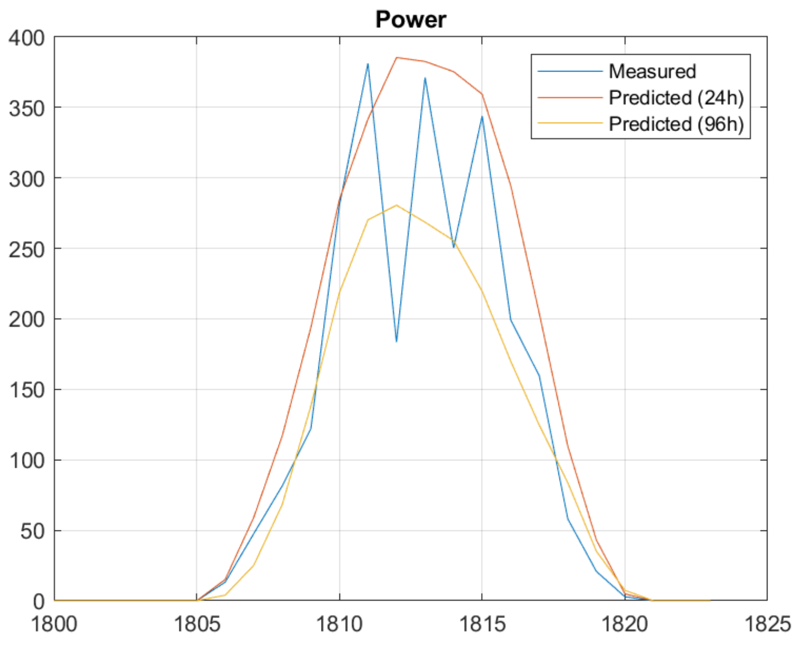


Figure 5.33. Power prediction comparison for a clear day with passing clouds.

Error	Predicted (24h)	Predicted (96h)
MBE	-27.2872	14.4236
MAE	30.5718	27.9991
MAD	7.8279	11.2392
R^2	0.8051	0.8630
RMSE	57.4038	48.1221

Table 5.13. Error Table: Power KPIs for a clear day with passing clouds.

We then attempted to complicate the model by adding fully connected layers, but the performance remains roughly the same in validation. Meanwhile, this simple model remains the best on the entire power set, as shown by the KPIs in Table 5.14.

Model	MBE	MAE	MAD	R^2	RMSE
FNN f2p	1.6206	32.179	5.6	0.77868	58.285
LSTM a2p 24	-32.157	40.868	7.1348	0.62178	76.194
LSTM c2p 24	-30.323	40.057	9.5482	0.633	75.055
LSTM c2p 96	-14.963	32.881	5.3835	0.71927	65.644
LSTM f2p 96	4.9977	29.558	4.9193	0.78078	58.009

Table 5.14. Error Table: KPIs on the whole dataset (2).

The models that seem to be the best candidates are the *FNN_f2p* and the *LSTM_f2p_96h* models, both of which directly leverage the relationship between input forecasts and power output measurements. Let us take a closer look at the two models and make some immediate observations. For instance, on a relatively clear day, both models tend to underestimate the power output (Figure 5.34), but the *LSTM_f2p_96h* model aligns more closely with the actual curve. On the other hand, for low-light days with flat power profiles, both models tend to overestimate the curve (Figure 5.35), and once again, the *LSTM_f2p_96h* model exhibits a smaller degree of overestimation.

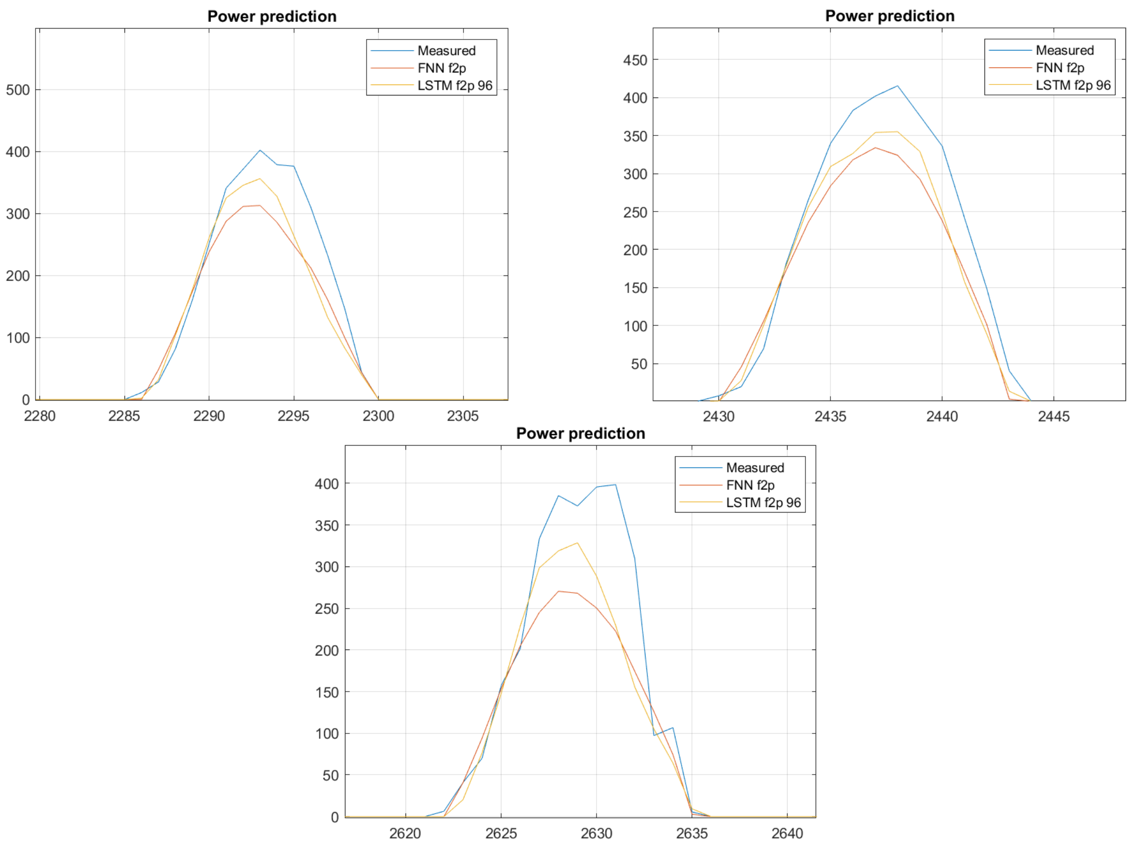


Figure 5.34. Underestimation of Power prediction in clear days.

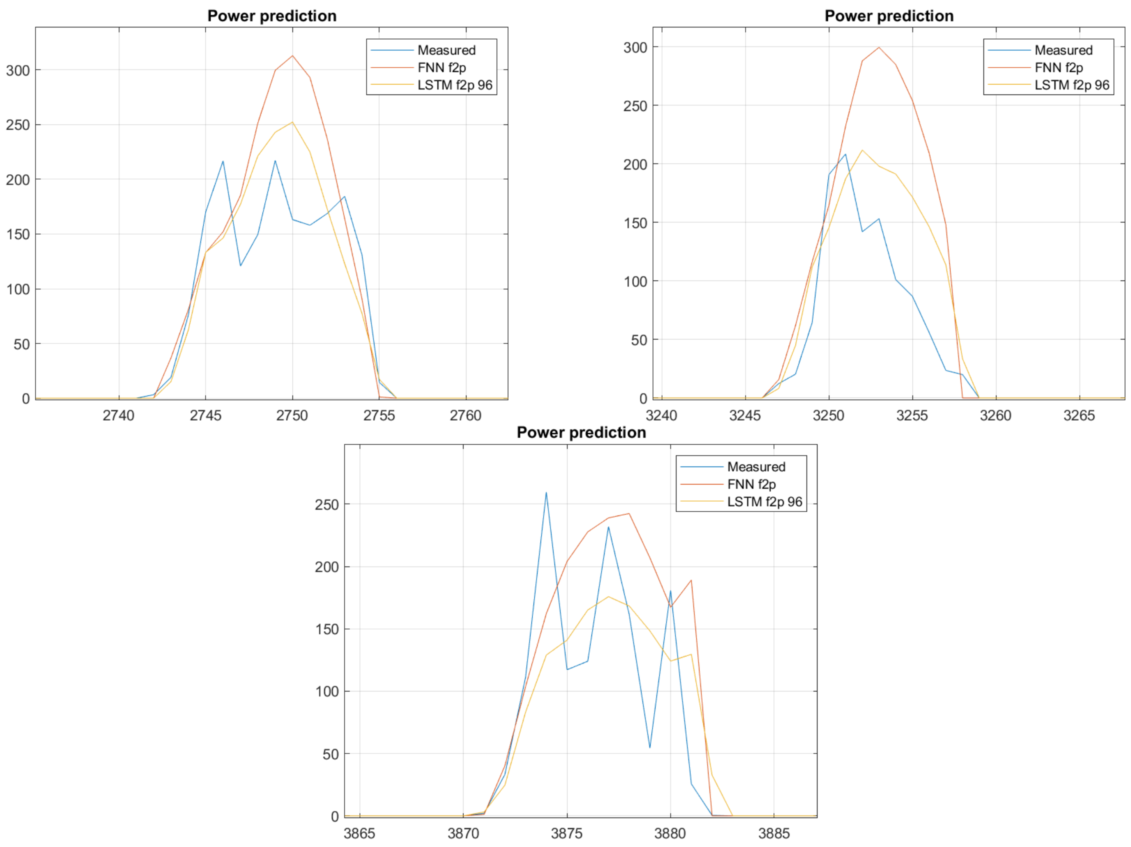


Figure 5.35. Overestimation of Power prediction in cloudy days.

To conclude, let us observe the power curve obtained by applying the post-processing methods. As for the reconstruction of the residual error, something quite curious happens now: in Figure 5.36 we notice that the negative component is practically absorbed by the positive one, and only positive peaks remain. This means that we can at most correct some underestimations, slightly pushing the power curve upward. The final post-processing power curve with the output data of the *LSTM_f2p_96h* model is shown in Figure 5.37. The KPIs of this refinement, referring to the entire dataset, are shown in Table 5.15. At first glance, it seems that there is no improvement, but we have seen in the figure that, on sunny days, the post-processed curve is slightly better than the one without refinement. A characteristic of all models is that they do not perform well in every scenario, but sometimes they perform well only for certain types of days. This is why, even following a technique like the one just shown, the result on the dataset may not seem to improve. However, if after applying a post-processing action to the results, we continue to maintain the same KPIs, it means that somewhere we have improved and somewhere else we have worsened. With this reasoning, we believe that we can no longer rely on a model, technique, or approach that tries to solve everything at once. Instead, we should aim to understand which model works best under which circumstances, and we will use the area integration method to do so.

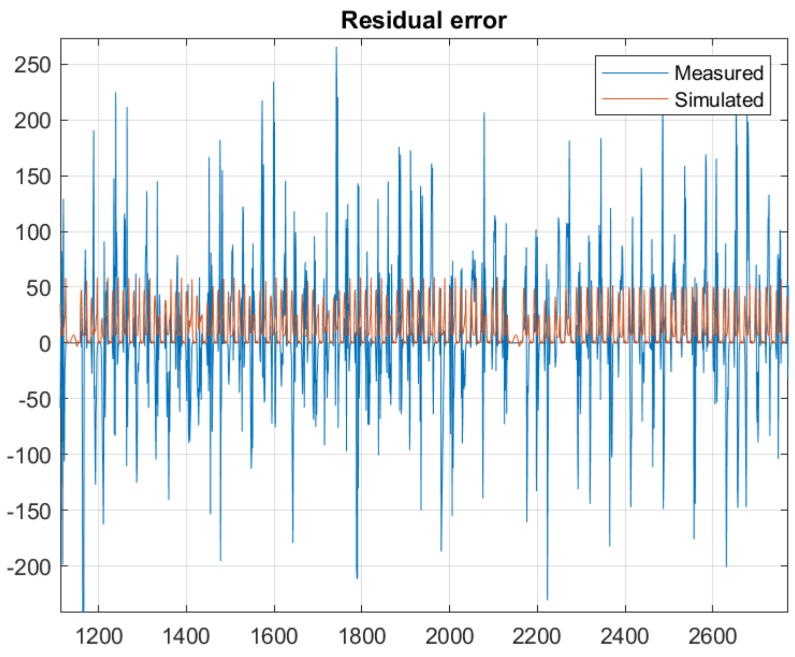


Figure 5.36. Summing Positive and Negative components of the residual error, prediction and measurement.

Model	MBE	MAE	MAD	R^2	RMSE
LSTM f2p 96	4.9977	29.558	4.9193	0.78078	58.009
post-processed	-9.5506	34.819	16.681	0.77918	58.008

Table 5.15. Error Table: KPIs on the whole dataset (2).

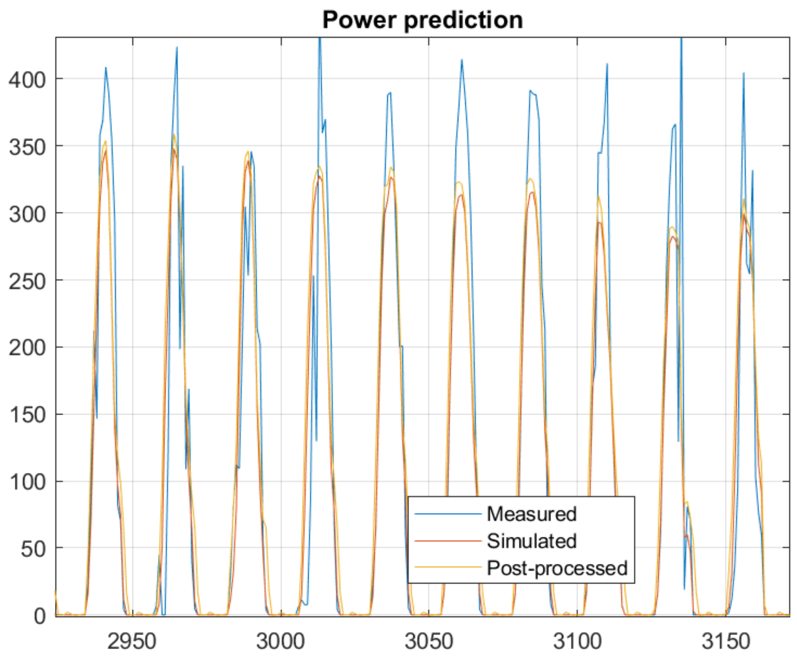


Figure 5.37. Power prediction after post-processing with *LSTM_ f2p_ 96h* data.

We now focus on the area method and present the scatter plots and boxcharts for the various models. The plots previously introduced in Chapter 4 for the *LSTM_ a2p_ 24h* model are shown again in Figure 5.40.

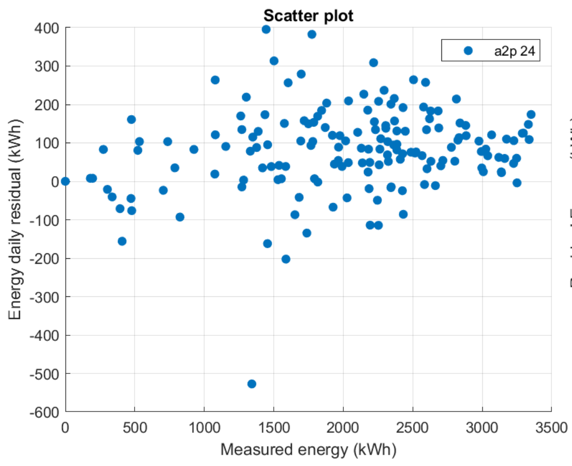


Figure 5.38. Scatter plot.

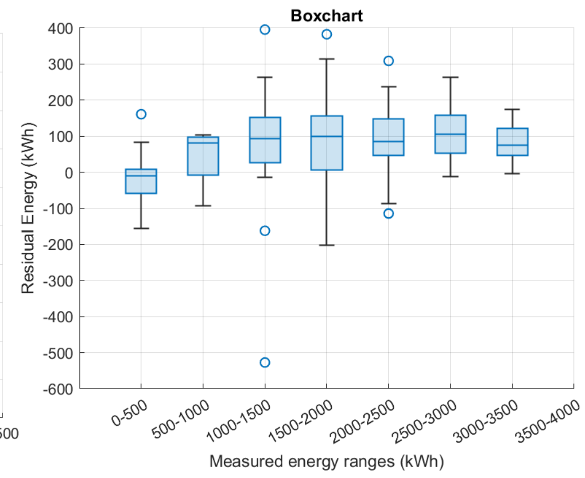


Figure 5.39. Boxchart.

Figure 5.40. Daily energy residual distribution for *LSTM_ a2p_ 24h* model.

In general, it appears that the concentration tends to be positive across all energy bands. Given that the residual energy was calculated as the difference between the energy estimated from the model's simulated power and the measured energy, this suggests that the model tends to overestimate the output power. However, in the lower range (0-500 kWh), the residual seems to be very small and close to 0.

We now move on to the *FNN_f2p* model and present the plots in Figure 5.43. In this case, there seems to be a clear tendency to underestimate energy (and therefore power) as it increases, that is, for increasingly sunny and productive days for photovoltaic generation. Therefore, we may prefer another model for the best days. However, this model seems to show a small underestimation error, with a dense concentration centered around 0 in the 2000-2500 kWh range.

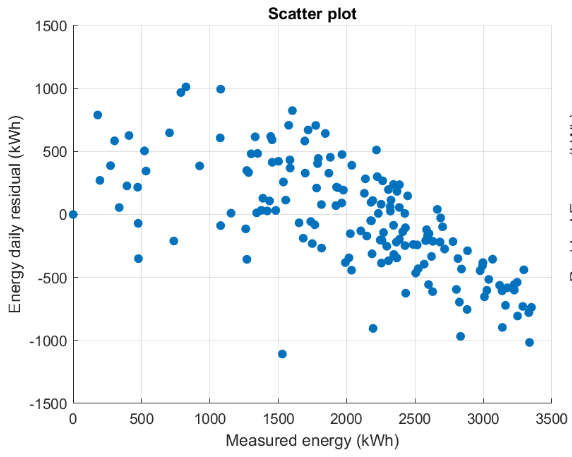


Figure 5.41. Scatter plot.

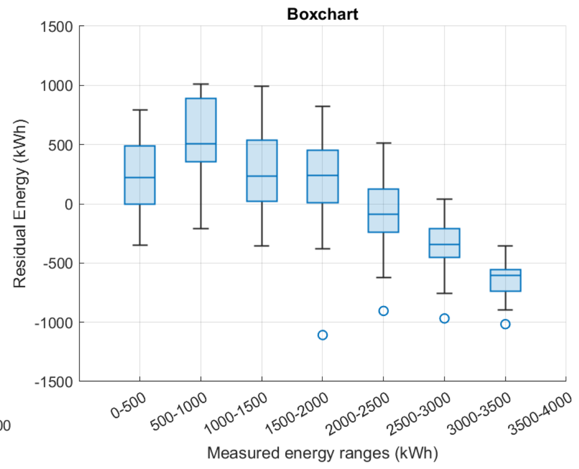


Figure 5.42. Boxchart.

Figure 5.43. Daily energy residual distribution for *FNN_f2p* model.

We now proceed with the *LSTM_c2p_24h* model, the plots of which are shown in Figure 5.46. Once again, there is a clear tendency to overestimate the actual power output. This indicates that the irradiance correction, when observing the total KPIs (Table 5.14) of the dataset, provides better performance but increases the overestimation of energy that was already present, albeit to a lesser extent, in the *LSTM_a2p_24h* model.

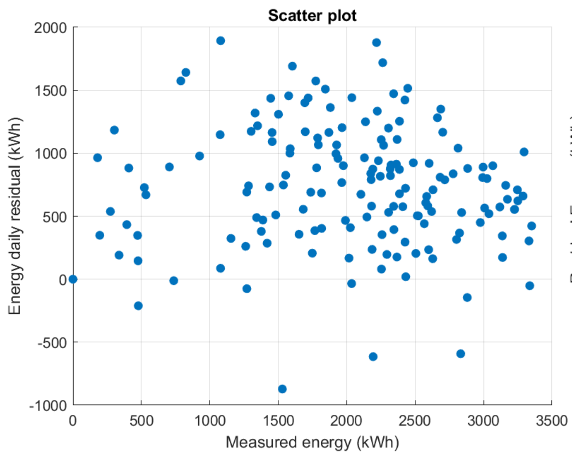


Figure 5.44. Scatter plot.

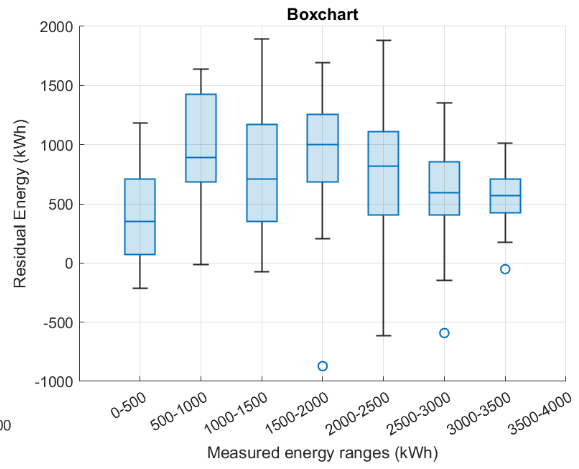


Figure 5.45. Boxchart.

Figure 5.46. Daily energy residual distribution for *LSTM_c2p_24h* model.

We now move on to the other irradiance correction model, the *LSTM_ c2p_ 96h* model, and examine the error distribution shown in Figure 5.49. A general overestimation of energy is observed, but with a small residual in the ranges of 0–500, 2500–3000, and 3000–3500 kWh.

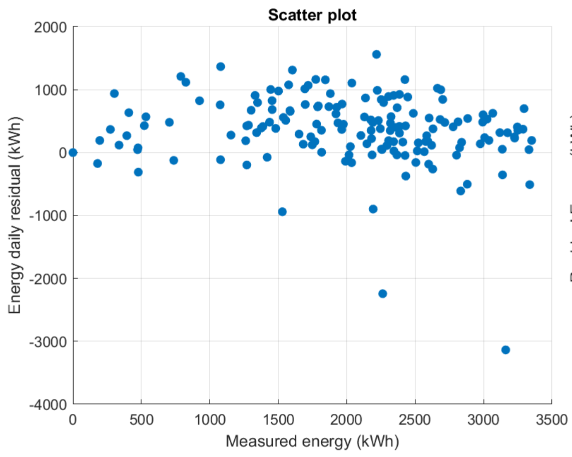


Figure 5.47. Scatter plot.

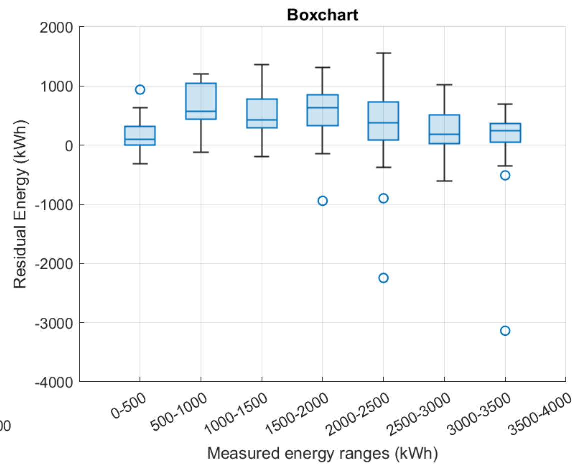


Figure 5.48. Boxchart.

Figure 5.49. Daily energy residual distribution for *LSTM_ c2p_ 96h* model.

We proceed with the *LSTM_f2p_ 96* model, whose plots are shown in Figure 5.52. This model clearly reflects the performance already observed in terms of KPIs. For numerous ranges, the residual errors are concentrated around 0, or the dense areas within each range remain narrower compared to the other models. Specifically, the 0–500, 1000–1500, 1500–2000, and 2000–2500 kWh ranges are the ones closest to a negligible error.

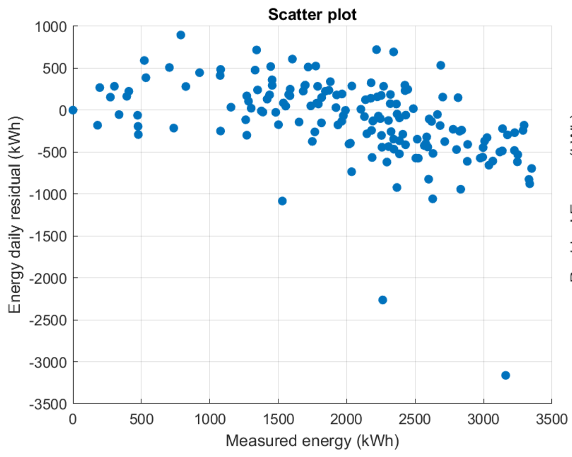


Figure 5.50. Scatter plot.

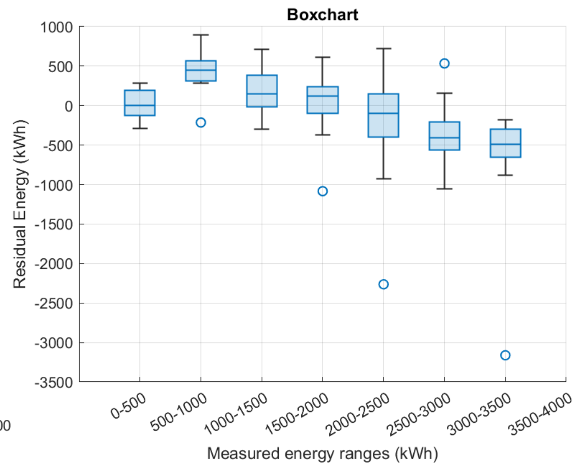


Figure 5.51. Boxchart.

Figure 5.52. Daily energy residual distribution for *LSTM_f2p_ 96h* model.

Finally, we also examine in Figure 5.55 the plots corresponding to the *LSTM_f2p_96* model after the post-processing step for residual error prediction has been applied. In general, the error distribution has shifted upward, thereby increasing overestimation where it was already present, while compensating for underestimation, particularly in the higher energy ranges where the error has been significantly reduced. This explains why the resulting KPIs remained nearly unchanged.

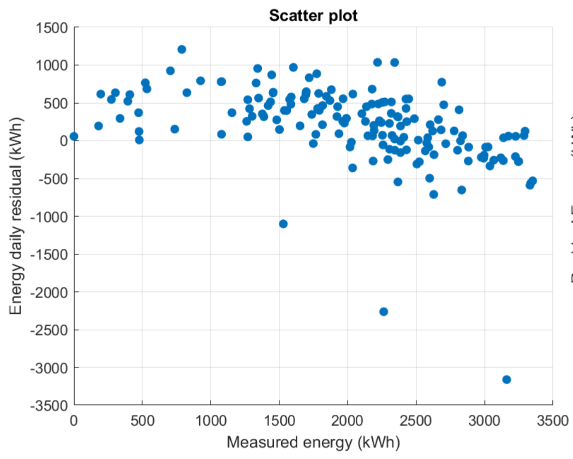


Figure 5.53. Scatter plot.

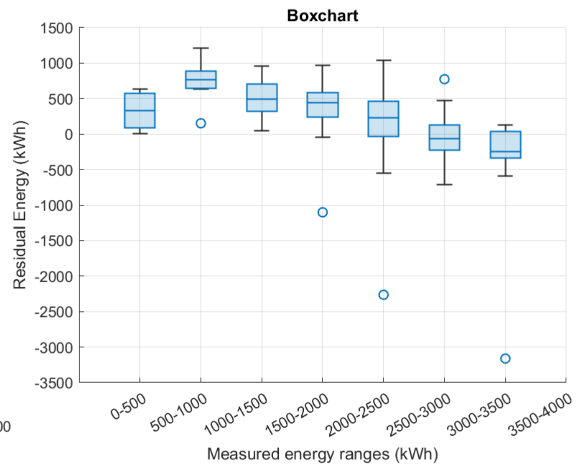


Figure 5.54. Boxchart.

Figure 5.55. Daily energy residual distribution for *LSTM_f2p_96h* model after post-processing.

The type of analysis based on the observation of the areas under the curve, conducted so far, would be ideal for a model combination, if the correspondence between measured and model-observed energy were certain. However, it is important to note that this relationship is not always guaranteed, and some of our models, particularly those directly based on forecasts, were trained using irradiance predictions. In Figure 5.58, we show the variability of irradiance data with respect to the measured power profile through a pair of scatter plots: one using the area[7] calculated daily using the measured irradiance value and the other using the forecast value.

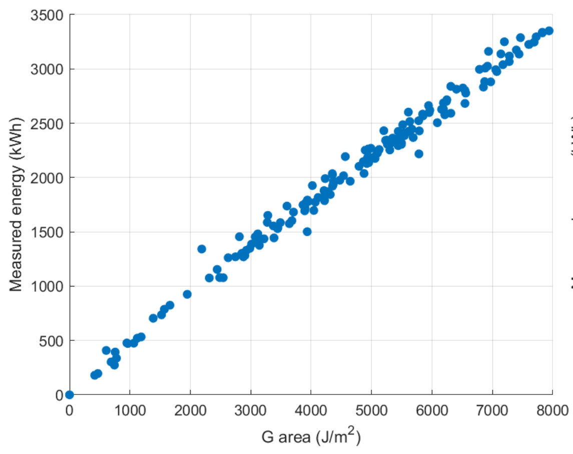


Figure 5.56. Irradiance.

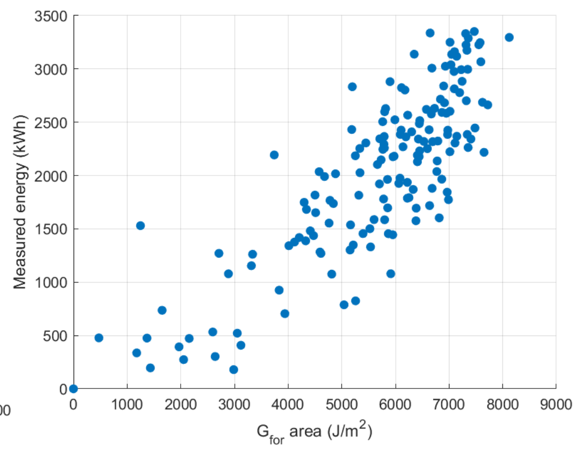


Figure 5.57. Forecast Irradiance.

Figure 5.58. Scatter plots for irradiance area across solar energy.

The final step involves determining how to combine the models effectively. On one hand, we know which models perform better and for which energy ranges, but only when relying on

[7]By calculating the area under the irradiance curve, we are essentially computing its integral, which physically corresponds to the distributed energy arriving on the surface per square meter, expressed in J/m^2.

the measured power values connected to energy. In this sense, for a new and unknown day, we could calculate the power predictions using all the models and determine the expected energy range. However, this approach might be computationally inefficient. A more practical alternative would be to predict the energy range in advance and compute results only for the best candidate model. However, the only data available for such predictions are the atmospheric forecasts. To address this, we analyze a boxchart that relates irradiance predictions to measured power (in terms of areas) and compare it with the corresponding boxchart that represents the true correlation between measured irradiance and power as shown in Figure 5.61.

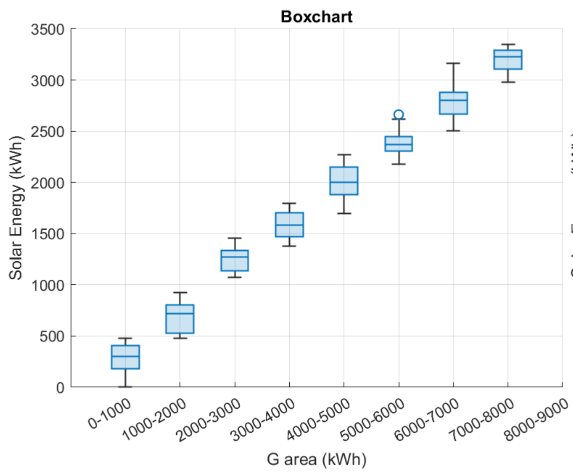


Figure 5.59. Irradiance.

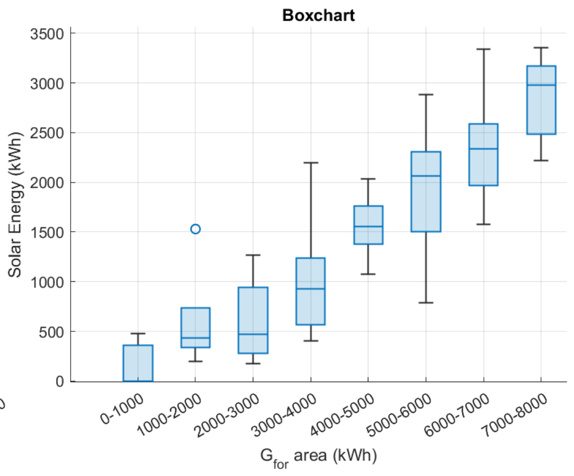


Figure 5.60. Forecast Irradiance.

Figure 5.61. Solar energy distribution across ranges of the area under irradiance: measured and forecast.

It is immediately apparent that the dense regions for predicted irradiance are much more spread out compared to those of measured irradiance. Additionally, we observe discrepancies between the medians, as well as overlapping ranges. For instance, it seems that for predicted irradiance areas in the 1000–2000 and 2000–3000 intervals, the solar energy target assumes average daily values around 500 kWh. Nevertheless, we expect that higher predicted irradiance should generally correspond to higher power output. Next, we identify the best-performing models for each solar energy (power) range, as summarized in the following table:

Solar Energy Range (kWh)	best model
0-500	LSTM f2p 96
500-1000	LSTM a2p 24
1000-1500	LSTM f2p 96
1500-2000	LSTM f2p 96
2000-2500	FNN f2p
2500-3000	post-processed
3000-3500	LSTM c2p 96

Table 5.16. Best model belonging to an energy range.

We then establish the correspondence between solar energy ranges (measured power) and the ranges of the area under the curve for predicted irradiance, as shown in Table 5.17.For the energy range 5000–6000 kWh, it was decided to divide it between 5000–5500 kWh with the *LSTM_f2p_96* model, and 5500–6000 kWh with the *FNN_f2p* model.

We now attempt to recalculate the KPIs for the entire dataset by selecting the models based on the area under the predicted irradiance curve, as indicated in Table 5.17. The steps to follow are as follows:

- calculate the daily area under the predicted irradiance curve;

Forecast Irradiance area (J/m^2)	Solar Energy range (kWh)	range median	best model
0-1000	0-350	0	LSTM f2p 96
1000-2000	350-750	430	LSTM f2p 96
2000-3000	280-940	470	LSTM f2p 96
3000-4000	570-1240	920	LSTM a2p 24
4000-5000	1380-1760	1550	LSTM f2p 96
5000-6000	1500-2300	2070	LSTM f2p 96 FNN f2p
6000-7000	1970-2580	2330	FNN f2p
7000-8000	2490-3170	2980	post-processed

Table 5.17. Forecast irradiance area ranges vs solar energy ranges.

- identify the best-performing model;
- compute the KPIs for that day using the selected model;
- iterate over all days in the dataset.

The results are presented in Table 5.18 under the label "area-based" and compared with the KPIs obtained using individual methods. It is noteworthy that while the R^2 correlation index is approximately an average of the individual models' correlation indices, the metric that shows a significant improvement is the $RMSE$. This is because the objective was to minimize the residual area between the actual and predicted curves, focusing less on their overall trend or relative position (which primarily affects R^2), and more on reducing the discrepancy between the curves.

Model	MBE	MAE	MAD	R^2	RMSE
FNN f2p	1.6206	32.179	5.6	0.77868	58.285
LSTM a2p 24	-32.157	40.868	7.1348	0.62178	76.194
LSTM c2p 24	-30.323	40.057	9.5482	0.633	75.055
LSTM c2p 96	-14.963	32.881	5.3835	0.71927	65.644
LSTM f2p 96	4.9977	29.558	4.9193	0.78078	58.009
area-based	-13.485	28.766	13.97	0.67904	45.614

Table 5.18. KPIs Comparison: Models vs. Areas-Based Post-Process.

Chapter 6

Conclusions and Possible Future Works

In this thesis, we explored whether the estimation of irradiance, which most significantly influences the power profile, plays a critical role in the uncertainty of power forecasting—namely, whether the error in irradiance estimation affects power prediction. Indeed, when the initial model, *LSTM_a2p_24h*, was trained on historical data, it successfully characterized the photovoltaic plant under analysis. When using actual irradiance values and temperature forecasts, the model continued to perform well, but its accuracy decreased significantly when using irradiance forecasts. This highlighted not only the importance of irradiance but also the need to correct its forecasts to bring them closer to actual values, thus producing a power curve more consistent with reality. This approach, along with all the models, was presented in Chapter 4 for *LSTM_c2p_24h* and *LSTM_c2p_96h*, with results detailed in Chapter 5. This correction resulted in notable improvements. We also experimented with new models that extended the input window and removed auxiliary correction, opting instead for a single model that simultaneously processed the last three days of measured data along with the irradiance and temperature forecasts for the prediction day. This resulted in the *LSTM_f2p_96h* model, which exhibited the best performance. As a cautionary note for potential model development, it is important to emphasize that irradiance correction cannot be applied in this case. This is because the correction aims to reduce the error in the irradiance profile, aligning with models based on measured data for both inputs and outputs. Applying corrections to models relying on the direct relationship between forecasts and measured power would, therefore, be inappropriate. Additionally, we developed a feedforward neural network (*FNN_f2p*) that demonstrated performance comparable to *LSTM_f2p_96h*, albeit processing individual data points rather than sequences like the LSTM models.

Next, shifting our focus from KPI-based performance evaluations, we visually inspected the results. All the models closely follow the irradiance forecast, demonstrating their ability to perform at their best given the provided forecast data. However, it is this discrepancy between the forecast and the actual recorded values that makes achieving better results challenging. We understand that these variations are primarily caused by the unpredictable passage of clouds, but estimating their behavior remains nearly impossible without more advanced additional tools. Often, sudden positive and negative spikes in the power profile rendered its shape less smooth. Consequently, we developed a final refinement for the *LSTM_f2p_96h* model's results by training a new model to estimate the residual error between the predicted and actual power curves, accounting for both positive and negative deviations. Repeatedly, the LSTM refinement model failed to predict abrupt spikes, instead learning to balance dense areas of positive and negative spikes by smoothing them into broader arches. The result was that the refinement mainly corrected underestimation moments while amplifying overestimations in the initial power curve.

At last, we adopted a statistical approach inspired by the fact that no single model performs optimally under all scenarios. It is important to highlight that the KPIs computed across the entire dataset are merely numerical indicators and may not fully reflect the actual model performance under specific conditions, such as cloudy days or clear days with passing clouds. Through the synergy of multiple models, we might identify patterns that dictate which model to activate under given circumstances. To explore this, we presented scatter plots and boxcharts in Chapter 5 to highlight any potential relationships between residual errors and the power curve. Specifically, we examined the area under the power curve (solar energy) and the residual energy deviation (the difference between the areas under the real and predicted curves for each model). This allowed us to investigate solar energy ranges, identify systematic overestimations or underestimations, and determine the best-performing models for each range based on minimal area discrepancies. However, to leverage this information effectively, we would need to predict the energy range of a power profile for an unknown day using only weather forecasts. Thus, we also explored possible correlations between irradiance forecasts and measured power, creating a correspondence between ranges that differed from those derived from actual irradiance. Finally, we hypothesized the application of the best models identified for each range across the entire dataset, discretized by range. The results showed a slightly worse R^2 correlation coefficient but a significantly lower RMSE compared to individual models.

Numerous approaches and techniques could be applied for further study and analysis of this plant, with potential innovative implications for renewable energy forecasting in general. For instance, in this specific case, four different forecasts for temperature and irradiance were available for each time step in the dataset. In this work, we consistently used the forecasts with the highest correlation and lowest error compared to real data across the entire dataset. However, discretizing by day type might reveal that other forecasts are more accurate for specific conditions. Similarly, statistical discretization based on day types (e.g., clear, cloudy, clear with intermittent clouds) could help develop scenario-specific models and a more robust statistical analysis could better compare them by considering daily KPIs and monthly averages. Another potential technique could involve using a separate shadow simulation model applied post-power forecasting as a parallel post-processing method rather than integrating it directly into the model data. Moreover, the area-based method successfully reduced RMSE by minimizing errors within energy ranges, which could be further refined by introducing finer ranges. The selection criterion could also be modified, for example, by choosing the model that minimizes a specific KPI for each range. Finally, new networks and irradiance correction models can always be explored.

Bibliography

[1] Nasa. «Global temperature». (), [Online]. Available: https://climate.nasa.gov/vital-signs/global-temperature/?intent=121.

[2] EnTRAINER, *Energetic transition*, Training course program on ETA for future energy professionals, attended in Polytechnic University of Valencia, associated with EnTRAINER, 2024.

[3] EU. «Le cause dei cambiamenti climatici». (), [Online]. Available: https://climate.ec.europa.eu/climate-change/causes-climate-change_it#:~:text=il%20riscaldamento%20globale.-,Riscaldamento%20globale,di%200%2C2%C2%BAC%20per%20decennio..

[4] C. A. Tracker. «The cat thermometer». (), [Online]. Available: https://climateactiontracker.org/global/cat-thermometer/.

[5] Repsol. «All about renewable energy sources». (), [Online]. Available: https://www.repsol.com/en/about-us/what-we-do/developing-renewable-energies/types-of-renewable-energy/index.cshtml.

[6] Terna. «Renewable generation». (), [Online]. Available: https://www.terna.it/it/sistema-elettrico/transparency-report/renewable-generation.

[7] A. Gasperoni, «Accuracy validation of a model, based on weather forecasts, for the hourly power calculation of photovoltaic systems», Polytechnic of Turin, XXX, 2023.

[8] R. E. World. «Balancing a renewable grid: What are the options?» (), [Online]. Available: https://www.renewableenergyworld.com/solar/balancing-a-renewable-grid-what-are-the-options/%5C#gref.

[9] European Association for Storage of Energy (EASE), *Cortes-la muela: The largest pumped-hydro storage plant in europe*, 2024. [Online]. Available: https://ease-storage.eu/.

[10] L. A. W. Farm, *London array offshore wind farm*, 2024. [Online]. Available: https://www.colcomms.com.

[11] P. Tech, *Europe's largest solar park: Squeezing out maximum energy from the minimum land space*, 2024. [Online]. Available: https://www.pv-tech.org.

[12] I. G. (ÍSOR), *Geothermal energy in iceland*, 2024. [Online]. Available: https://www.isor.is.

[13] Ørsted, *Horns rev 2 offshore wind farm*, 2008. [Online]. Available: https://orsted.com/en/what-we-do/offshore-wind/horns-rev-2.

[14] ScienceDirect. «Reflected irradiance». (2024), [Online]. Available: https://www.sciencedirect.com/topics/engineering/reflected-irradiance#:~:text=Diffuse%20irradiance%3A%20Received%20from%20the,radiation%20on%20clear%20sunny%20days..

[15] M. et al., *Environmental impacts of integrating photovoltaic modules on electric light utility vehicles - scientific figure on researchgate*, Accessed 28 Nov 2024. [Online]. Available: https://www.researchgate.net/figure/Types-of-Solar-Radiation-Mallon-et-al-2017_fig5_339176950.

[16] EnergySage, *Monocrystalline vs. polycrystalline solar panels*, 2024. [Online]. Available: https://www.energysage.com/solar/solar-panels/monocrystalline-vs-polycrystalline/.

[17] ScienceDirect, *Transparent photovoltaic panels and luminescent solar concentrators*, 2024. [Online]. Available: https://www.sciencedirect.com/topics/engineering/transparent-photovoltaic-panels.

[18] P. Magazine, *Building-integrated photovoltaics (bipv): Facade panels*, 2024. [Online]. Available: https://www.pv-magazine.com/.

[19] Enact Solar, *Eco-friendly house with a large solar panel roof*, Accessed: 28 Nov 2024. [Online]. Available: https://enact.solar/wp-content/uploads/2024/01/eco-friendly-house-with-large-solar-panel-roof.png.

[20] W. Solar, *Future solar tech: What are transparent solar panels, how do they work, and when will they be available?*, Accessed: 28 Nov 2024. [Online]. Available: https://wysesolar.ie/future-solar-tech-what-are-transparent-solar-panels-how-do-they-work-and-when-will-they-be-available/.

[21] M. Powalla, S. Paetel, D. Hariskos, *et al.*, *Image from "advances in cost-efficient thin-film photovoltaics based on cu(in,ga)se2"*, Accessed: 28 Nov 2024. [Online]. Available: https://ars.els-cdn.com/content/image/1-s2.0-S2095809917306033-gr8.jpg.

[22] ScienceDirect, *Reflected irradiance*, 2024. [Online]. Available: https://www.sciencedirect.com/topics/engineering/reflected-irradiance#:~:text=Diffuse%20irradiance%3A%20Received%20from%20the,radiation%20on%20clear%20sunny%20days.

[23] U. of Central Florida. «Solar electricity basics». (), [Online]. Available: https://energyresearch.ucf.edu/consumer/solar-technologies/solar-electricity-basics/.

[24] A. K. Behura, A. Kumar, D. K. Rajak, C. I. Pruncu, and L. Lamberti, *Image from "towards better performances for a novel rooftop solar pv system"*. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2095809917306033.

[25] H. Room, *Modelo de celda solar en atpdraw*, https://hansroom17.wordpress.com/2017/02/02/modelo-de-celda-solar-en-atpdraw/.

[26] A. of the article, *Current-voltage characteristic of a typical solar panel*, https://www.researchgate.net/figure/Current-voltage-characteristic-of-a-typical-solar-panel-The-above-curves-shows-the_fig2_326878893.

[27] SEAWARD. «How does temperature and irradiance affect i-v curves?» (), [Online]. Available: https://www.seaward.com/gb/support/solar/faqs/00797-how-does-temperature-and-irradiance-affect-i-v-curves/#:~:text=The%20open%20circuit%20voltage%20of,turn%20reduces%20the%20power%20output..

[28] E. Commission, *Photovoltaic geographical information system*, Developed by the European Commission, 2024. [Online]. Available: https://re.jrc.ec.europa.eu/pvg_tools/en/.

[29] P. Velardocchia, «Indicatori di prestazione specifici dall'elaborazione dei dati di monitoraggio per gli impianti fotovoltaici del politecnico», Polytechnic of Turin, XXX, 2024.

[30] G. Alba, «Modello previsionale di produzione di impianti fotovoltaici e applicazione statistica su larga scala», Polytechnic of Turin, XXX, 2020.

[31] D. Giuliana, «Perfezionamento di un modello di calcolo previsionale della potenza oraria per impianti fotovoltaici», Polytechnic of Turin, XXX, 2021.

[32] N. A. Ramli, M. F. A. Hamid, N. H. Azhan, and M. A. A.-s. Ishak, «Solar power generation prediction by using k-nearest neighbor method», in *AIP Conference Proceedings*, 2019. [Online]. Available: file:///Users/gianlucacardinale/Downloads/1.5118124.pdf.

[33] A. Hayes. «Autoregressive integrated moving average (arima) prediction model». (), [Online]. Available: https://www.investopedia.com/terms/a/autoregressive-integrated-moving-average-arima.asp.

[34] E. R. M. Shouman, *Solar Power Prediction with Artificial Intelligence*, A. Y. Abdelaziz, Ed. Intechopen, 2024.

[35] R. B. Cleveland, W. S. Cleveland, J. E. McRae, and I. Terpenning, «Stl: A seasonal-trend decomposition procedure based on loess», *Journal of Official Statistics*, vol. 6, no. 1, pp. 3–73, 1990. [Online]. Available: https://www.wessa.net/download/stl.pdf.

[36] A. of the article, *Scheme of ai, dl, ml, and ann.* [Online]. Available: https://www.researchgate.net/figure/Scheme-of-AI-DL-ML-and-ANN_fig1_371205057.

[37] C.-F. Yen, H.-Y. Hsieh, K.-W. Su, M.-C. Yu, and J.-S. Leu, «Solar power prediction via support vector machine and random forest», *E3S Web Conf*, vol. 69, no. 01004, p. 6, 27 November 2018. [Online]. Available: https://www.e3s-conferences.org/articles/e3sconf/abs/2018/44/e3sconf_gesg2018_01004/e3sconf_gesg2018_01004.html.

[38] PredikData, *¿qué son y para qué se usan los modelos predictivos?* [Online]. Available: https://predikdata.com/es/que-son-y-para-que-se-usan-los-modelos-predictivos/.

[39] Wikipedia. «Gradient boosting». (), [Online]. Available: https://en.wikipedia.org/wiki/Gradient_boosting.

[40] Wikipedia. «Artificial neural networks». (), [Online]. Available: https://en.wikipedia.org/wiki/Artificial_neural_network.

[41] Wikipedia. «Deep learning». (), [Online]. Available: https://en.wikipedia.org/wiki/Deep_learning.

[42] «Rosenblatt's perceptron». (), [Online]. Available: https://www.pearsonhighered.com/assets/samplechapter/0/1/3/1/0131471392.pdf.

[43] T. U. of Queensland. «Action potentials and synapses». (), [Online]. Available: https://qbi.uq.edu.au/brain-basics/brain/brain-physiology/action-potentials-and-synapses.

[44] Wikiwand, *Axolema.* [Online]. Available: https://www.wikiwand.com/es/articles/Axolema.

[45] N. McCullum. «Deep learning activation functions». (), [Online]. Available: https://www.nickmccullum.com/python-deep-learning/deep-learning-activation-functions/.

[46] «Deep feedforward networks». (), [Online]. Available: https://www.deeplearningbook.org/contents/mlp.html.

[47] N. Sardana, «Neural networks: Forward and backpropagation», 2017. [Online]. Available: https://tjmachinelearning.com/lectures/1718/nn2/nn2.pdf.

[48] J. Wilber and B. Werness. «The importance of data splitting». (), [Online]. Available: https://mlu-explain.github.io/train-test-validation/.

[49] S. Online. «Stanford cs221: Artificial intelligence». (), [Online]. Available: https://www.youtube.com/playlist?list=PLoROMvodv4rO1NB9TD4iUZ3qghGEGtqNX.

[50] K. Nyuytiymbiy. «Parameters and hyperparameters in machine learning and deep learning». (), [Online]. Available: https://towardsdatascience.com/parameters-and-hyperparameters-aa609601a9ac.

[51] C. Stryker. «What is a recurrent neural network (rnn)?» (), [Online]. Available: https://www.ibm.com/topics/recurrent-neural-networks#:~:text=A%20recurrent%20neural%20network%20or%20RNN%20is%20a,sequential%20predictions%20or%20conclusions%20based%20on%20sequential%20inputs..

[52] A. Rosebrock. «Convolutional neural networks (cnns) and layer types». (), [Online]. Available: https://pyimagesearch.com/2021/05/14/convolutional-neural-networks-cnns-and-layer-types/.

[53] J. Starmer. «Recurrent neural networks (rnns), clearly explained». (), [Online]. Available: https://www.youtube.com/watch?v=AsNTP8Kwu80&t=683s.

[54] W. Commons, *Lstm cell*, 2024. [Online]. Available: https://commons.wikimedia.org/wiki/File:LSTM_cell.svg.

[55] S. Y. Heng, W. M. Ridwan, P. Kumar, *et al.*, «Artificial neural network model with different backpropagation algorithms and meteorological data for solar radiation prediction», *Scientific Reports*, 2022. [Online]. Available: https://doi.org/10.1038/s41598-022-13532-3.

[56] N. Sharma, V. Puri, S. Mahajan, L. Abualigah, R. A. Zitar10, and A. H. Gandomi, «Solar power forecasting beneath diverse weather conditions using gd and lm-artificial neural networks», *Scientific Reports*, 2023. [Online]. Available: https://doi.org/10.1038/s41598-023-35457-1.

[57] H. N. Amer, N. Y. Dahlan, A. M. Azmi, M. F. A. Latip, M. S. Onn, and A. Tumian, «Solar power prediction based on artificial neural network guided by feature selection for large-scale solar photovoltaic plant», *ELSEVIER*, vol. Energy Reports, no. 9, pp. 262–266, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352484723013896?ref=pdf_download&fr=RR-2&rr=8a5a11284d2c5d4e.

[58] Polito. (), [Online]. Available: https://smartgreenbuilding.polito.it/monitoraggio/.

[59] Even. «What is a pyranometer and how it measures the irradiance?» (), [Online]. Available: https://www.sevensensor.com/what-is-pyranometer.

[60] PAGASA, *How a weather forecast is made.* [Online]. Available: https://www.pagasa.dost.gov.ph/learning-tools/how-weather-forecast-made#:~:text=Land%2Dbased%20weather%20stations%20around,anemometer%2C%20thermometer%2C%20psychrometer%20or%20hygrometer.