Forecasting Photovoltaic Power Generation using Machine Learning LSTM Models Trained on Simulated Data

A Group Project Paper for the Digitalization Project for the Chair of Energy Network Technology at Montanuniversity Leoben

Authors:
Georg Gressl
Christoph Rinnhofer
Michael Grün

Advisors:
Dipl.-Ing. Dr.mont. Julia Vopava-Wrienz
MSc. Ahmad Fayyaz Bakhsh

Kurzfassung

In Smart Grids ist die Kurzzeit-Lastprognose (STLF) von größter Bedeutung, insbesondere bei stark fluktuierenden Energiequellen wie Photovoltaik-Anlagen, bei denen eine direkte Reduktion der Leistung wirtschaftlich nicht sinnvoll ist. Zuverlässige Lastprognosen ermöglichen ein effektives Management und die Optimierung der Energiesysteme zum Ausgleich residualer Lasten. In dieser Arbeit wird ein datenbasierte Ansatz vorgenommen, der die Simulation eines Photovoltaiksystems unter Verwendung historischer Wetterdaten mit einem Deep-Learning-Modell zur Verbesserung der STLF-Genauigkeit kombiniert. Konkret wird ein Long Short-Term Memory (LSTM) neuronales Netzwerk eingesetzt, um zukünftige Lasten basierend auf Wettervorhersagedaten, die über eine Programmierschnittstelle (API) abgerufen werden, vorherzusagen. Durch die Integration der simulierten Photovoltaik-Systemleistung mit Wettervorhersagen erfasst der vorgeschlagene Ansatz die inhärente Variabilität erneuerbarer Energiequellen und verbessert dadurch die Qualität und Zuverlässigkeit der Lastprognose. Die nahtlose Integration von datengesteuerten Simulationen und fortschrittlichen Machine-Learning-Techniken zeigt das Potenzial für eine deutliche Verbesserung der STLF-Fähigkeiten und trägt letztendlich zu einem effizienteren und nachhaltigeren Energiemanagement in intelligenten Stromnetzen bei.

Abstract

In smart grid systems, accurate short-term load forecasting (STLF) is crucial, especially for highly fluctuating energy sources like solar photovoltaic (PV) systems where directly reducing the load profile is not economically viable. Reliable load predictions enable effective management and optimization of energy resources to meet demand. This paper proposes a novel data-driven approach that combines the simulation of a PV system using historical weather data with a deep learning model to enhance STLF accuracy. Specifically, a long short-term memory (LSTM) neural network is employed to predict future loads based on weather forecast data obtained from an application programming interface (API). By integrating the simulated PV system output with weather forecasts, the proposed method captures the inherent variability of renewable energy sources, thereby improving the quality and reliability of load forecasting. The seamless integration of data-driven simulations and advanced machine learning techniques demonstrates the potential to significantly enhance STLF capabilities, ultimately contributing to more efficient and sustainable energy management in smart grid systems.

Table of content

List of Figures

List of tables

List of abbreviations

AC	Alternating Current
AOI	Angle of Incidence
API	Application Programming Interface
ENMS	Energy Management System
GW	Gigawatts
IAM	Incidence Angle Modifier
IV	Current-Voltage
kWpeak	Kilowatt Peak
LSTM	Long Short-Term Memory
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error
NIP	Network Infrastructure Plan
PV	Photovoltaic
PVGIS	Photovoltaic Geographical Information System
R^2	Coefficient of Determination
RMSE	Root Mean Squared Error
SDM	Single-Diode Model
STPF	Short-Term Power Forecasting
TSI	Total Solar Irradiance
TW	Terawatts
TWh	Terawatt-hours

1 Introduction

The increasing integration of photovoltaic (PV) energy into the power grid underscores the critical need for system optimization to maximize efficiency and reliability. As PV energy generation continues to rise, accurately forecasting the performance of PV systems becomes paramount for both financial investment and grid stability. Simulation tools, such as PVsyst, PV*SOL, HOMER, and SAM, utilize mathematical models to predict the performance of specific PV systems under various conditions, thereby enhancing the design optimization process. These predictions are vital for determining the financial viability of PV investments and ensuring the seamless integration of generated power into the electrical grid.

PV power forecasting plays a crucial role in advanced energy management systems (ENMS). By reducing or eliminating unknown factors, ENMS can operate more efficiently, effectively balancing supply and demand. Typically, PV power is forecasted using a combination of weather forecast data and historical performance data. Forecasting can be categorized into three time-based levels, each serving a specific purpose: short-term forecasts (minutes to hours) for grid stability, medium-term forecasts (hours to days) for energy trading and management, and long-term forecasts (years) for investment planning. Techniques employed in PV forecasting include statistical methods, machine learning algorithms, physical models, and hybrid approaches that integrate multiple methodologies.

This paper focuses on forecasting photovoltaic power generation using Long Short-Term Memory (LSTM) models, a type of recurrent neural network (RNN) well-suited for time series prediction. LSTM models are trained on simulated data to enhance the accuracy and reliability of PV power forecasts. By leveraging machine learning techniques, particularly LSTM models, we aim to improve the predictability of PV power output, thereby facilitating better energy management, enhancing grid stability, and optimizing financial returns on PV investments.

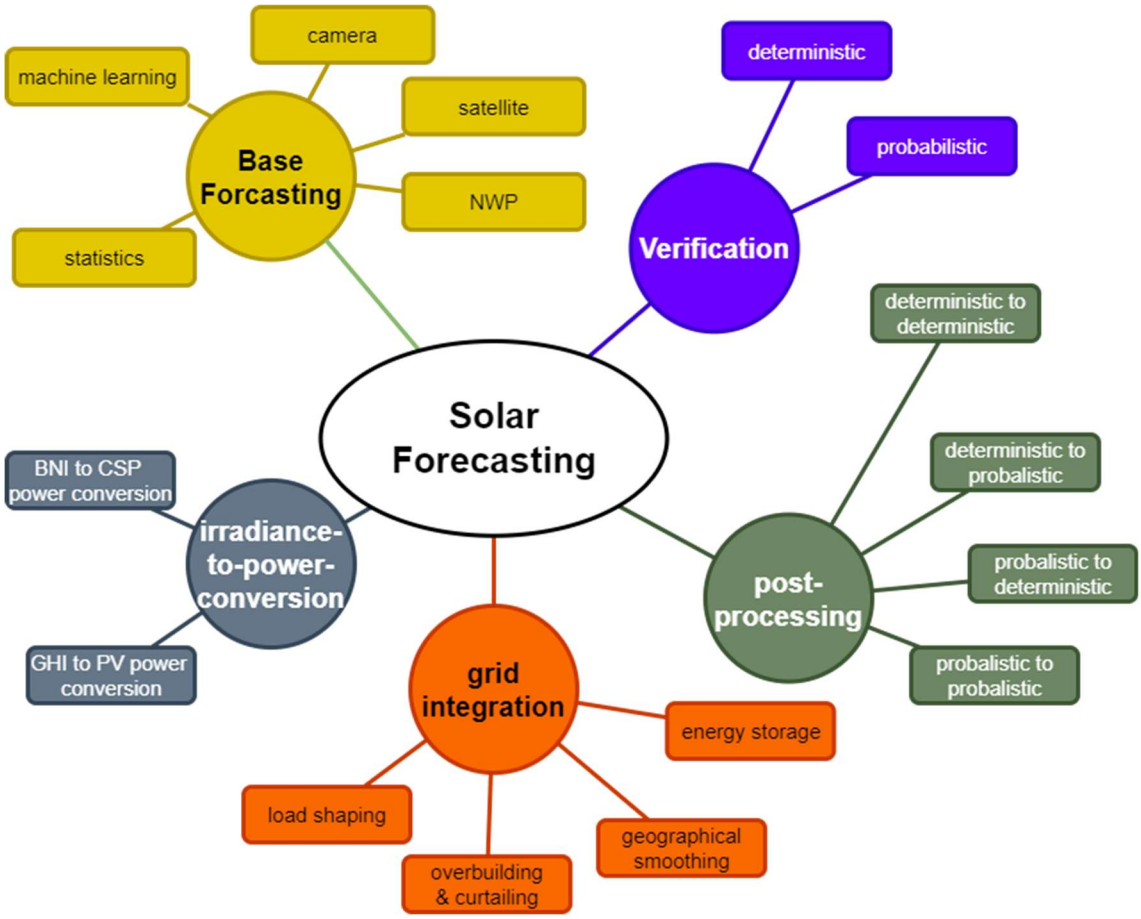


Figure 1: Five pillars of solar forecast research according to Kleissl,Yang [1]

Figure 1 illustrates the fundamental components of solar forecasting, a critical process for managing and optimizing solar energy integration into power systems. The forecasting framework is divided into five key domains: base forecasting, verification, post-processing, grid integration, and irradiance-to-power conversion. Base forecasting serves as the foundation, employing methods such as machine learning, statistical analysis, satellite imaging, ground-based cameras, and numerical weather prediction (NWP) to estimate solar energy availability based on historical and real-time meteorological data. The accuracy of these forecasts is assessed in the verification stage, where deterministic approaches compare single-value predictions against observed data, and probabilistic methods evaluate forecast uncertainty by generating prediction intervals. The post-processing stage further refines raw forecasts, transforming deterministic predictions into probabilistic outputs and vice versa, to suit specific applications and improve reliability. Grid integration focuses on leveraging these forecasts for practical energy management, addressing challenges such as load shaping, overbuilding and curtailing, energy storage, and geographical smoothing, all of which are essential for ensuring grid stability and efficiency. Lastly, the irradiance-to-

power conversion domain translates solar irradiance forecasts into actionable power output metrics for different solar technologies, including converting beam normal irradiance (BNI) for concentrated solar power (CSP) systems and global horizontal irradiance (GHI) for photovoltaic (PV) systems. Collectively, these interconnected processes demonstrate a systematic approach to enhancing solar energy utilization, supporting the transition to sustainable energy systems while addressing the variability and uncertainty inherent in renewable energy generation.

2 Photovoltaic Systems

PV modules are devices, that convert sunglight directly into electricity. The advantages of PV is that once installed it does not take much effort to maintain the system and a lot can be done remote. Most devices do not have moving parts therefore the risk of malfunctioning reduces. The modules operate noiseless and can be placed on top of buildings to save area. The scaling, also upgradebility of PV is straightforward and they can also operate as standalone systems. A PV cell consists of minimum two thin layers of semiconductors, most commonly silicon. When the silicon is exposed to sunlight the electric charges are generated and a direct current as a consequence. The photoelectric effect was first observed by Heinrich Hertz in 1887 and explained by Albert Einstein in 1905 [1]. The amount of electricity produced is a function of semiconductor composition wavelength and amount of radiance available on the PV cell. [2]

2.1 Energy conversion

The solar irradiation forecast is an important predecessor of PV prediction. Solar irradiadiance is expressed as the power density (W/m^2). The amount of power available is therefore the solar irradiance multiplied by the total effective surface of the collector area ($W.m^{-2} \times m^2 = W$). As previously mentioned the energy generated from a PV system is based on the solar irradiance and many other factors of the system design. Although the solar power is fairly linear to the solar irradiance and typically changes less than 20 % in operation.

2.1.1 Solar radiation

Sunlight serves as the energy source for all concentrating solar power generation systems. The sun, with an effective temperature of around 6000 K, emits radiation across a broad spectrum of wavelengths. These wavelengths are typically categorized, from high-energy short wavelengths to low-energy long wavelengths, as gamma rays, X-rays, ultraviolet, visible, infrared, and radio waves. These divisions are referred to as spectral regions. [3, p. 2]

The total radiant energy emitted by the sun is remarkably stable. While it has traditionally been referred to as the solar constant, the preferred term now is total solar irradiance (TSI) to reflect its slight variations over time. [3, p. 3]

It is the received energy per unit area from the sun at the top of earth's atmosphere. The TSI at mean sun-earth distance is 1366 ± 7 W/m^2 [4]. The elliptical orbit of the earth

causes the TSI to vary from 1321W/m^2 at apogee to around 1425 W/m^2 at perigee. The prediction of irradiance becomes more challenging when the solar radiation enters the atmosphere.

The total hemispherical solar radiation on a horizontal surface is described in the following equation:

$$GHI = DNI \cdot \cos(SZA) + DH$$ [3, p. 6]

To explain the abbreviations used in the equation: GHI is the global horizontal radiation, DNI is the direct normal irradiance, DHI is the diffuse horizontal irradiance and SZA is the solar zenith angle (illustrated in Figure 2) computed from the date and time of measurement at a specific location. [3, p. 6]

The behavior of radiation - whether it is transmitted, absorbed or scattered - depends on the wavelength and the properties of the medium through which it passes. [3, p. 6]

Building on the effects of radiation, further effects follow, which form the basis for the mode of operation of a PV cells.

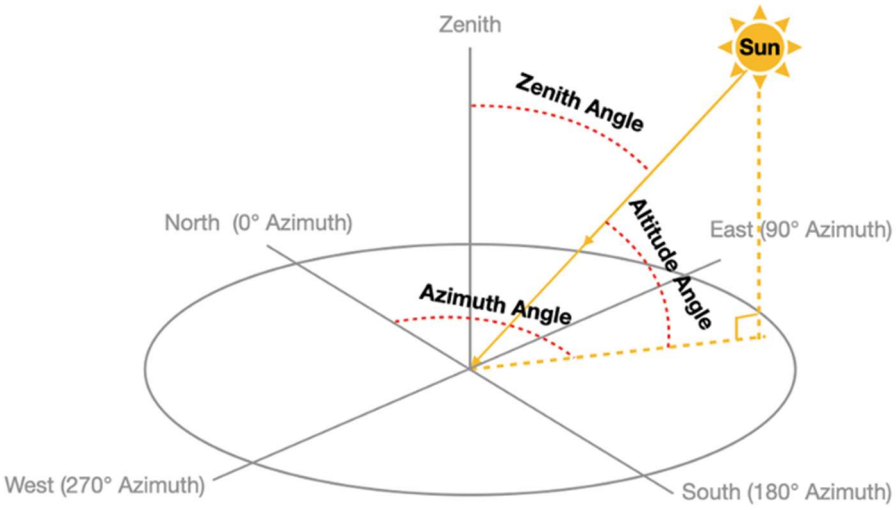


Figure 2: Demonstration of zenith and azimuth angle

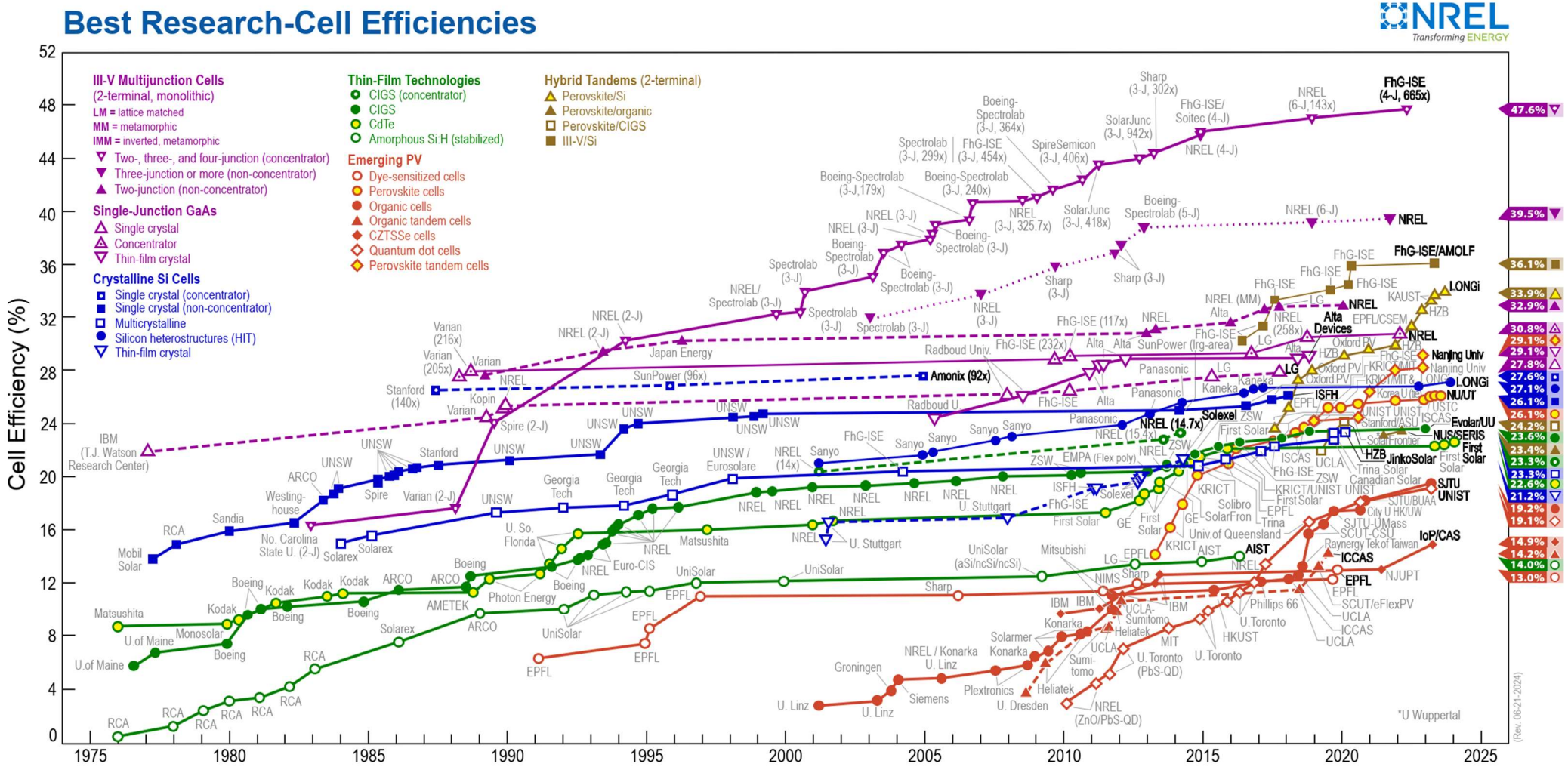


Figure 3: Chronology of improvements in PV-cell efficiciency according to technology (21.06.2024)

2.1.2 Photoelectric effect

The photoelectric effect occurs when light hits a metal surface and releases electrons from this surface. The amount of electrons depends on the frequency of the light: the higher the frequency of the light, the more electrons are released. This happens due to the quantized nature of light as particles (photons) and not as waves. The energy of these released electrons is determined by the work function - the energy required to release an electron from the material. The energy of the photon must be at least as high as the work function of the material. This means that the electrons can only be knocked out if the light has enough energy.

If the photon has enough energy to overcome this work function, it can release an electron from the material. The remaining photon then carries less energy than before. This effect can be used to convert solar energy into electricity. PV cells are used for this purpose. These are made of special materials such as silicon or other semiconductors and are designed in such a way that they can capture photons. When these photons hit the solar cell, they knock electrons out of the material, which are then directed in a certain direction by an electric field. This generates a current flow and thus electrical energy. [5]

2.1.3 PV cells structure

PV cells consist mainly of semiconductor materials, typically silicon. Silicon is widely used due to its availability and good semiconductor properties and has four valence electrons, in other words it is 4-valent. The cells consist of two layers of semiconductor materials with different doping, on the one hand the p-layer (positively doped), which has an excess of holes (missing electrons), for example the trivalent boron, and on the other hand the n-layer (negatively doped), which has an overload of free electrons, for example with 5-valent phosporus atoms. These are so-called electron-hole-pairs. Bringing an n- and a p-doped semiconductor together creates a so-called depletion region: At the boundary layer between the n- and p-layer, the holes diffuse into the n-layer and the electrons into the p-layer, creating a depletion region of a certain width that is depleted of free charge carriers. As a result of this shift, the n-layer now has a positive space charge and the p-layer a negative space charge. This creates an electric field in the space charge zone.

In Figure 3, which is on the previouos landscape page, shows the development of the efficiency of PV cells over the last 5 decades. There has been a continuous improvement

in the different materials. Monocrystalline cells, for example, have an efficiency of 23,3 % under laboratory conditions.

The structure of a PV cell is illustrated in Figure 4.

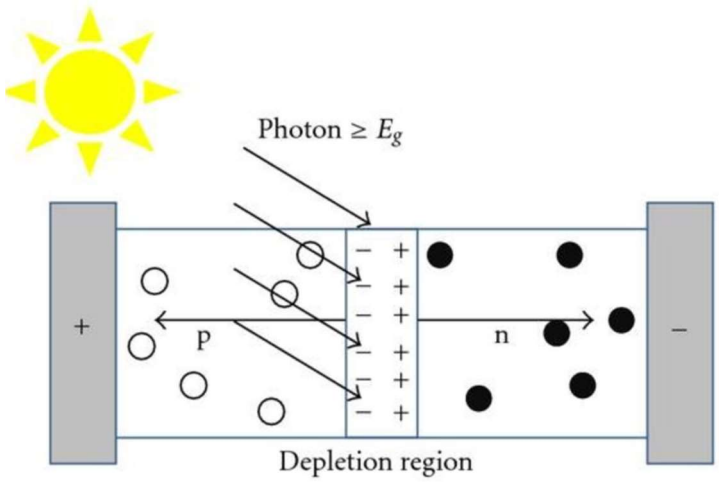


Figure 4: Structure of a PV cell [6, p. 5]

The effect of light in the space charge zone is now the actual motor of the cell. When light (i.e. photons) hits the space charge zone, it can detach an electron from the atom. The remaining atom is then positively charged and has an electron deficiency, i.e. a hole. This process is known as the internal photoelectric effect. The photoelectric effect can lead to a permanent detachment of the electron if it takes place in the space charge zone or its immediate vicinity. The negative electron and the positive hole move apart according to the electric field generated by the space charge zone, which exerts a force on the moving charge carriers. The electron moves in the direction of the stationary positive space charge in the n-doped region, the positive hole moves in the direction of the negative space charge in the p-doped region. This leads to charge separation and thus to a voltage that can be tapped at the metal contacts. If the metal contacts of the solar cell are connected to a load, current can flow. The electrons move upwards through the n-layer and enter the outer circuit. The holes flow down through the p-layer and are canceled out again by the electrons that flow to the p-layer via the outer circuit (recombination). [7]

2.1.4 Single-diode model

In the following subchapter, the single-diode model (SDM), which is used in the rest of the work, is described for the initial time.

Physical models, such as the SDM or the two-diode model, are used to simulate the behavior of a PV system, to forecast their electrical power output, and to assess and compare the performance and characteristics of newly developed devices. [8, p. 707]

PV cells are modeled using diode models in order to digitally represent the expected behavior of PV systems and to take a number of dependencies into account (real irradiance, temperature conditions). PV cells exhibit a non-linear current-voltage (IV) characteristic, where the current and power output of the PV generator are influenced by the operating voltage at the terminals. Additionally, the maximum power point changes with ambient conditions, such as solar radiation and both ambient and module temperatures. Various models are used to describe the real electrical behavior of PV cells, primarily categorized into one-diode (or single-diode) and two-diode models. The SDM represents a PV cell as a current source connected in series with a diode. Using the SDM, the output current of a PV cell can be represented by accounting for the generated photocurrent (I_{PH}) and the Shockley equation, which describes the exponential current-voltage characteristic I(U) of a p-n junction. [9, p. 4]

The equivalent circuit of a SDM, consisting of a photosensitive current source, a diode, as well as a parallel and a series resistance, is illustrated in Figure 5.

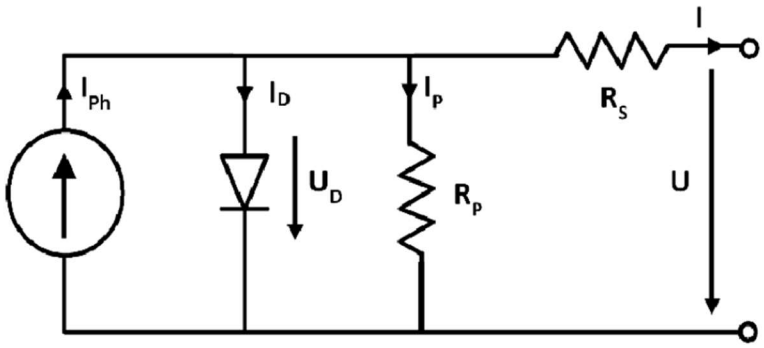


Figure 5: Equivalent circuit of a SDM [9, p. 4]

The following equation can be used to describe the output current of the PV cell of the SDM. An iterative calculation with a feedback loop is necessary for the exact calculation of the output current. [9, p. 5]

$$I(U) = I_{PH} - I_D - I_P = I_{PH} - I_0 \cdot \left(e^{\frac{U+I*R_S}{m*U_T}} - 1 \right) - \frac{U + I \cdot R_S}{R_P}$$

In this equation, I_{PH} represents the photocurrent, I_D the diode current, I_P the current through the parallel resistance and I_0 the diode reverse saturation current. Furthermore, U_T represents the thermal voltage, m the ideality factor, R_S the series resistance and finally R_P the parallel resistance. [9, p. 5] By adjusting the described parameters of the SDM, the behavior of the PV cell can be simulated and optimized under different conditions.

The exact procedure for implementing the SDM in the code can be found in chapter 4.2.5, Step 4.

2.2 Current situation and outlook

Most of the global energy supply still relies on fossil fuels, which contribute significantly to global warming through their emissions. Over the decades, PV have evolved from niche applications - where connecting to electrical grids was impractical or uneconomical - to a mainstream technology for renewable electricity. This transition has resulted in unprecedented cost reductions and efficiency improvements in energy technologies. Today, in many parts of the world, PV offer the lowest levelized cost of electricity production.

Continuous optimization of processes to achieve higher efficiency and reduce material usage has significantly decreased the materials needed per installed capacity, thereby reducing the CO_2-footprint. Consequently, the energy payback time—the period required for a PV system to generate the amount of energy used in its production—now ranges from a few months to less than two years, depending on the technology and location.

With annual growth rates of around 30 %, the PV market is highly dynamic, leading to the constant construction of new, high-output production facilities. However, as PV systems are increasingly installed in diverse applications and climate conditions, ensuring the installation, operation, and maintenance of high-quality products to generate high yields over the long term remains a significant challenge. [10, pp. 7-8]

2.2.1 Austrian situation

Austria wants to be climate-neutral by 2040 and cover 100 % of its electricity needs from renewable energies by 2030. PV will play a key role in the future energy mix. [11, p. 1]

With the *Renewable Energy Expansion Act (Erneuerbaren-Ausbau-Gesetz: EAG)*, the Austrian federal government has set itself the goal of increasing annual electricity generation from renewable sources by 27 TWh by 2030. PV accounts for 11 TWh of this. Electricity demand will continue to rise in the future due to the electrification of various sectors. PV expansion of 41 TWh is feasible by 2040. [11, p. 1, 12, p. 11]

For the first time since the initial phase of innovators and standalone systems in the 1980s, the Austrian photovoltaic market experienced a surge in 2003 following the passage of the *Green Electricity Act (Ökostromgesetz)*. However, this growth was short-lived as the market collapsed in 2004 due to the capping of feed-in tariffs. The market reached its peak diffusion in 2013, driven by an additional funding process, and then stabilized from 2014 to 2018 with annual installed PV capacities between 150 and 190 MW_{peak}.

After a period of continuous growth in subsequent years, the Austrian photovoltaic market achieved new record values in 2021 and 2022, with installations reaching 740 MW_{peak} and 1,009 MW_{peak}, respectively. This upward trend continued, with even higher values recorded in 2023. [13, p. 11]

In 2023, 2,603 MW_{peak} of PV was added for the first time - an absolute record. This value corresponds to an increase of 158 % with around 134,000 newly installed PV systems, meaning that there are now just under 390,000 PV systems with a grid connection in operation, of which around 89 % have an installed capacity of less than 20 kWPeak. These have a cumulative total output of 6395 MW peak. As a consequence, the total electricity produced by PV plants in operation amounted to at least 6,395 GWh in 2023, leading to a reduction in CO_2 equivalent emissions by 1.996 million tons. [13, p. 11, 14, p. 1]

These numbers are illustrated in Figure 6.

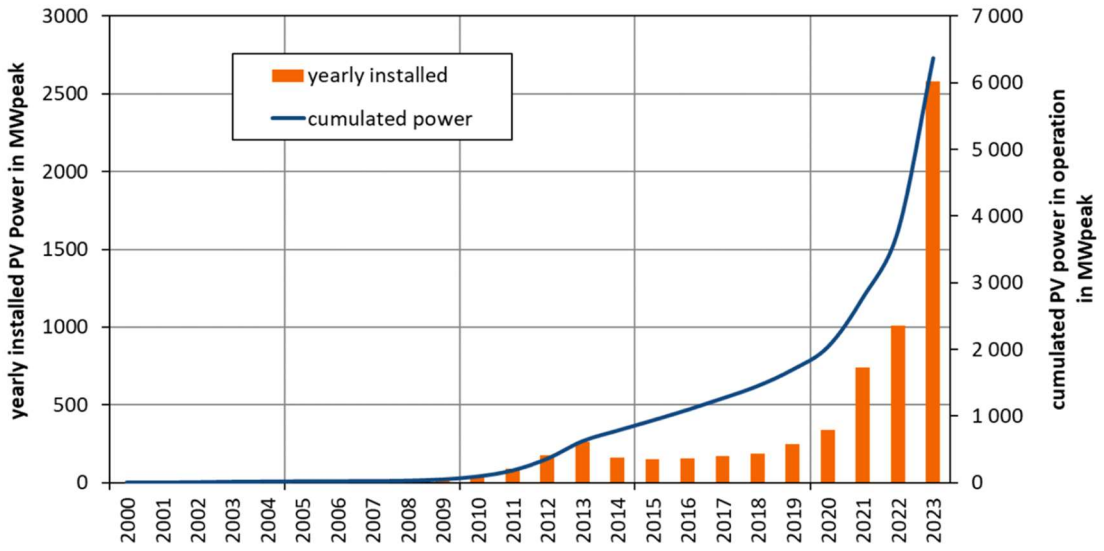


Figure 6: Development of PV systems in Austria [15, p. 44]

Out of the nine federal states of Austria, Upper Austria, Lower Austria and Styria stand out in particular, each with more than 500 MW_{peak} of new PV capacity in 2023. These three also have a total capacity of more than 1,200 MW_{peak} by the end of 2023. [14, pp. 4-5]

Next, the types of solar cells installed in Austria over the last 13 years are examined: After monocrystalline cells still accounted for the largest share in 2010 at 53 %, their share decreased increasingly in the following years and stood at 6 % in 2015. In the years that followed, the share of monocrystalline cells rose again, reaching 100 % of the total newly installed capacity in Austria in 2022 for the first time. This value remained unchanged in 2023, which means that monocrystalline cells were almost exclusively installed in Austria in 2023 as well. Polycrystalline cells and thin-film cells therefore continued to play no role in the Austrian PV market in 2023. [13, p. 21]

These numbers are illustrated in Figure 7.

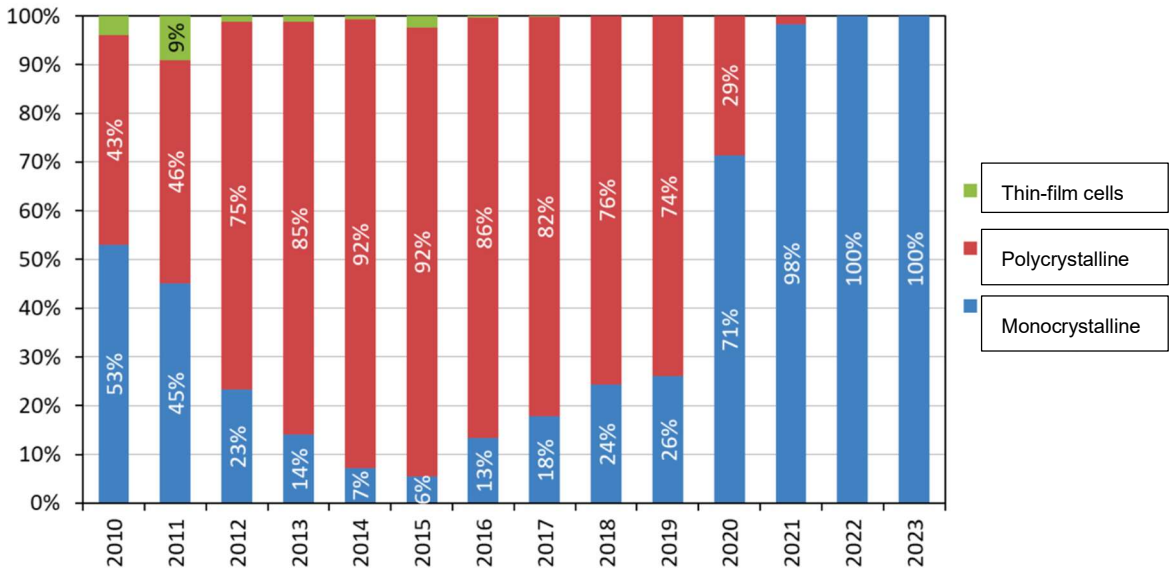


Figure 7: Percentage share of installed solar cells types in Austria [13, p. 21]

In this regard, the system and installation types of the last 10 years in Austria are examined. After a slight increase to 95.9 % in 2020, the share of rooftop installation in 2021 in relation to the newly installed PV capacity in that year fell back to around the 2019 level and amounted to 84.8 %. With a share of 83.7 % (2022) and 85.65 % (2023), the share of rooftop installation remained almost unchanged in the following years, but rose slightly again in 2023 for the first time since 2020. In comparison, the share of freestanding PV systems in total newly installed capacity fell for the first time since 2020, from 14.9 % in 2022 to 11.81 % in 2023. While the share of facade-integrated PV systems fell slightly (2022: 0.53 %; 2023: 0.33 %), the share of roof-integrated systems rose significantly from 0.71 % in 2022 to 2.21 % in 2023. However, with a total share of newly installed capacity of around 2.5 %, facade and roof-integrated systems still only play a subordinate role in 2023. [13, p. 21]

To conclude on the Austrian situation, here is a look at the financial aspects of PV.

At the end of 2023, there were just under 13,000 full-time jobs in the PV industry in Austria, with total sales of €4.3 billion, which in turn contributed €2.1 billion to Austrian value creation. [14, p. 1]

2023 in the PV industry waFs characterized by a massive oversupply of Chinese modules, which led to a collapse in module sales prices in Europe and culminated in sales prices that were in some cases below the manufacturing costs. This led to a decline in European PV module production by around a third to around 4 GW_{peak} across Europe, which also severely affected Austrian module production. Insolvencies, bankruptcies and

closures were the result. The development of the sales prices of Austrian module producers and the average module purchase prices of Austrian PV planners and installers reflect this development. [13, p. 22]

For 2023, a price of around EUR 1,669/kW_{peak} was calculated for ready-to-use 5 kW_{peak} systems. This corresponds exactly to the value that was also surveyed in 2022. In comparison, the average price for systems with an output of 10 kW_{peak} fell to EUR 1,347/kW_{peak} compared to 2022 (2022: EUR 1,448/kW_{peak}), meaning that the average system prices in 2023 were also more than 13 % higher than the previous lows in 2019 and 2020. As in the previous year, system prices for systems with an output of 30 to 50 kW_{peak} were also surveyed in 2023, which also fell significantly.

It can be seen that specific system prices fall with increasing system size (in relation to installed capacity). With a system size of 30 to 50 kW_{peak}, the costs per kW_{peak} are almost 51.03 % lower than for a 5 kW_{peak} system. [13, p. 25]

Finally, a list of the various potential funding instruments that were available in the federal states and at federal level in 2023 follows.

There were investment subsidies from the federal states, investment subsidies from the *Climate and Energy Fund (Klima- und Energiefonds: KLIEN)*, the *EAG investment subsidy for photovoltaics and electricity storage (EAG Investitionszuschuss Photovoltaik und Stromspeicher)*, as well as the *EAG market bonus subsidy (EAG Marktprämienförderung)*, the *Green Electricity Act feed-in subsidy (ÖSG 2012) / tariff subsidy (Ökostromeinspeiseförderung (ÖSG 2012) / Tarifförderung)* and the *Investment subsidy in accordance with §27a ÖSG 2012 (Investitionsförderung gemäß §27a ÖSG 2012)*.

In addition, in the federal states of Carinthia, Lower Austria, Upper Austria, Salzburg, Styria and Tyrol, PV systems were subsidized via *Housing subsidies (Wohnbauförderung)*. [13, p. 27]

2.2.2 International situation

The next subchapter is dedicated to the international PV situation.

The global cumulative PV capacity grew from 1.2 TW in 2022 to 1.6 TW in 2023, with around 450 GW of new PV installations commissioned - and an estimated 150 GW of modules in global stocks. After several years of material and transportation cost pressures, module prices fell in a massively oversupplied market, keeping PV competitive even as electricity prices fell from historic highs in 2022.

Thanks to an active energy policy, PV installations in China rose to a record level and accounted for more than 60 % of new global capacity. Overall, China reached 662 GW of PV capacity by the end of 2023 (an increase of 235 to 277 GW, depending on reporting [IEA reports; Official China reports]). Europe continued to see strong growth and installed 61 GW (of which 55.8 GW in the EU), led by a resurgence in Germany (14.3 GW) and increases in Poland (6.0 GW), Italy (5.3 GW) and the Netherlands (4.2 GW), while Spain declined slightly (7.7 GW). In North and South America, both major markets grew - the USA installed 33.2 GW again in 2022 after a weak year, and Brazil continued its dynamic development in 2022 and installed 11.9 GW, putting its cumulative capacity in the top ten worldwide.India had a slightly slower year with 16.6 GW, again mainly in centralized systems. Other Asia-Pacific markets also slowed down, including Australia (3.8 GW), while Korea (3.3 GW) and Japan (6.3 GW) remained constant. Market growth outside China reached around 30 %, while China's growth was over 120 %, which explains the enormous global development of the PV market. [13, p. 41, 16, p. 7]

These numbers are illustrated in Figure 8.

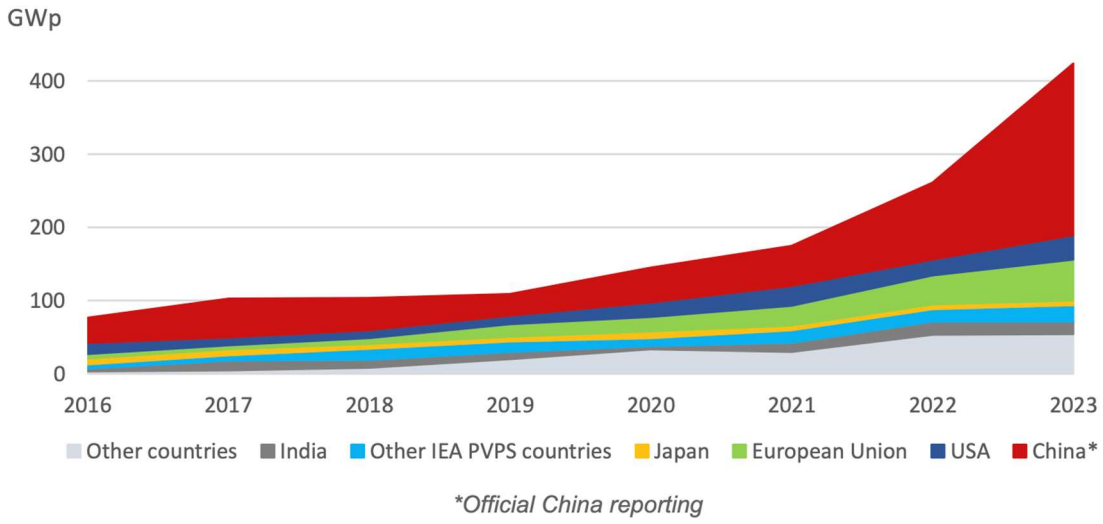


Figure 8: New annual capacity growth in key markets [16, p. 8]

In addition to the figure, Table 1 of the top 10 countries with installed PV capacity follows:

Table 1: Top 10 countries with installed PV capacity [16, p. 10]

Rank	Country	Cumulative installed capacity [GW]
1	China	662.0
(2)[1]	European Union (EU)	268.1
2	United States (USA)	169.5
3	India	95.3
4	Japan	91.4
5	Germany	81.6
6	Spain	37.6
7	Brazil	35.5
8	Australia	34.6
9	Italy	30.3
10	South Korea	27.8

The top five countries, starting with China down to India and then Germany, have at least 40 GW more capacity than the next countries. They have also widened the gap by more than 10 GW since 2022. [16, p. 11]

The Asia-Pacific region holds the majority share of global PV installations, which increased again in 2023, rising to over 60 % with a cumulative installed capacity of at least 947 GW. A closer examination reveals that it is China, rather than the entire Asia-Pacific region, that has driven this increase. Excluding China, the three other major regional markets—the remainder of Asia-Pacific, Europe, and the Americas—are roughly equivalent, with the first two holding 19 % and the latter 15 % of global cumulative PV installations. [16, p. 12]

In 2023, all others Asia-Pacific markets experienced slight contractions or remained steady compared to 2022. The Japanese market, which has fluctuated between 6 GW and 10 GW annually over the past decade, seems to be stabilizing at around 6.3 GW, reaching a cumulative capacity of 91.4 GW. India, with a similar cumulative capacity, has shown more volatility, ranging from under 1 GW in 2014 to 18.6 GW in 2022. Challenges

[1] In 2023, the European Union comprised 27 countries. Among them, Germany, Spain, and Italy ranked among the top ten in terms of cumulative installed capacity. [16, p. 10]

to steady growth in India include administrative procedures, grid access, and financing, but the country is still expected to play a leading role due to its ambitious local manufacturing and development targets. Australia's market, which had been steadily promising at 4 GW to 5 GW for five years, dropped below 4 GW this year. Meanwhile, South Korea's market fluctuated between 2.6 GW and 5 GW, stabilizing at just over 3 GW in the past two years. Both countries now have cumulative capacities between 25 GW and 35 GW. Other significant markets in the Asia-Pacific region include Vietnam, whose fragile grid halted new capacity after two multi-GW years in 2019/2020, and Taiwan, which might reach 3 GW in 2023. Despite specific reasons cited for sluggish markets in each country, a more fundamental underlying problem persists. [16, p. 12]

In the European regional market, almost all countries saw growth in annual PV capacity in 2023. This increase was driven by the rising competitiveness of PV due to high electricity prices resulting from the Ukraine conflict, as well as urgent political mandates to boost energy autonomy and renewable energy production. Germany remains the leading contributor, adding 14.3 GW to reach a total of 81.6 GW, driven by strong political motivation. Spain, with a cumulative capacity exceeding 30 GW, saw a decrease from 8.5 GW in 2022 to 7.7 GW in 2023. Italy, also with over 30 GW in cumulative capacity, experienced significant growth, adding 5.3 GW - more than doubling its previous capacity. France followed with 23.6 GW (+3.9 GW), the Netherlands with 22.4 GW (+4.2 GW), and Poland, which made notable gains, reaching 18.5 GW (+6 GW). Additionally, nearly a dozen other European countries installed more than 1 GW of PV capacity in 2023, including Austria. [16, p. 12]

In the American market, the USA remains the most important driver and, after a weaker year in 2022, has returned to the volume of 2021. The US added 33.2 GW in 2023, bringing its total capacity to 169.5 GW, which accounts for over 70 % of the region's cumulative PV capacity. Meanwhile, Brazil continued its dynamic growth and increased its capacity by 11.9 GW (from 9.9 GW in 2022) to 35.5 GW. Chile also recorded growth, increasing its cumulative capacity by 1.3 GW to 9.2 GW and is expected to reach a double-digit figure by 2024. Canada, with a cumulative capacity of 7.3 GW, is expected to reach the same milestone within the next three years. [16, p. 13]

In the Middle East and Africa, South Africa was the primary contributor, installing nearly 3 GW of new capacity, while other countries in the region added less than 700 MW combined. This significant disparity highlights a gap between the number of announced projects and actual construction progress. For instance, Egypt added just over 300 MW of new capacity in 2023, despite having 700 MW under construction and an announced

pipeline exceeding 11 GW. This situation underscores the region's ongoing challenge of translating project announcements into completed installations. [16, p. 14]

These numbers are illustrated in Figure 9.

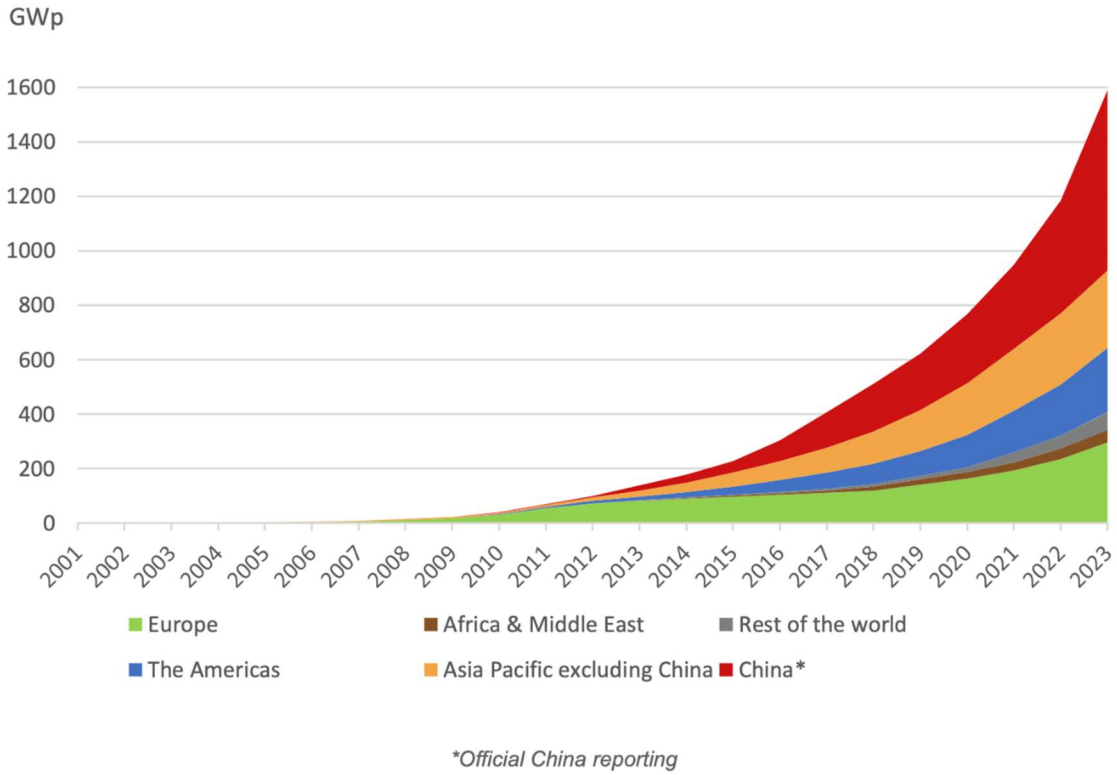


Figure 9: Development of PV installations in key markets [16, p. 13]

In summary, it can be said that the number of countries with a theoretical market penetration of more than 10 % of total electricity demand has doubled to 18 since 2022.

The growing installed capacity of PV systems is significantly impacting global electricity consumption. The two major markets China and the European Union each account for about 10 % of their electricity from PV. Overall, PV now meets more than 8 % of global electricity demand. [16, p. 177]

Individual markets remain sensitive to policy support and domestic electricity prices, despite competition in most market segments in many countries. PV played an important role in reducing CO_2 emissions from electricity generation with 75 % of new renewable capacity installed in 2023; around 60 % of electricity generation from new renewable capacity came from PV. [13, p. 41]

2.2.3 Outlook

As already mentioned, Austria wants to be climate-neutral by 2040 and cover 100 % of its electricity needs from renewable energies by 2030. Therefore, PV will play a key role in the future energy mix.

Following on from the record year 2023 in the PV industry, the aim is to continue on this path in a coordinated manner with the PV strategy so that the expansion of PV not only contributes to achieving climate targets but also ensures a sustainable and flexible power supply. The focus should now shift from the *2030 electricity targets (Stromzielen)* to the *2040 climate neutrality goal (Klimaneutralitätsziel)*, as outlined in the *Integrated Austrian Network Infrastructure Plan (Integrierten österreichischen Netzinfrastrukturplan: NIP)*, which sets a target of 41 TWh for photovoltaics. This translates to an annual expansion of approximately 2.5 GWpeak. Although this scale was reached for the first time in 2023, it must be viewed in context, considering various external factors. Notably, the significantly increased electricity prices on the stock exchange in 2023 and public concern over rising energy costs, coupled with supply and personnel shortages, likely created a unique situation where economic considerations took a back seat. This also accelerated planned investments in PV and electricity storage. It remains uncertain whether similarly high expansion rates can be sustained over the next 1 to 2 years.

The advancement and expansion of innovative PV module and cell production, as well as further development across the entire PV value chain, should be expedited and streamlined to prevent lagging behind the substantial global growth trends in the PV sector. In this regard, sustaining or enhancing domestic and European value creation is imperative to secure the long-term availability of components and supply chains for this technology, which currently supplies over 10 % of the national electricity demand. Opportunities for the Austrian market beyond mere installation will predominantly arise if research and development efforts are intensified. This would facilitate the introduction of new and innovative PV components and applications, thereby diminishing reliance on Asian markets. [13, p. 10]

As PV distribution increases, greater importance should be attached to on-site management of the energy generated by promoting energy management systems in order to best coordinate local PV generation with local energy requirements such as e-mobility, heat pumps and cooling loads. Due to the high simultaneity factor of photovoltaics, there are already increasingly low or negative prices on the European electricity market on sunny days around midday, which are clearly attributable to PV. These times will increase with the further spread of PV and lead to local management of

energy and/or direct purchase agreements becoming significantly more important. A great deal of attention must therefore be paid to the management of excess PV capacity. The further expansion of PV towards the targets outlined in the NIP (41 TWh annual PV production) will also make it possible for significant shares of electricity demand to be covered by PV even at times of low solar irradiation, thus reducing the need for storage and the use of flexibility. It is also essential to adapt applications to demand: in Austria, too, the value of the energy generated during off-peak times (daytime or winter) should increasingly take center stage. Vertical systems (facades etc.) will thus become significantly more important for the electricity generated.

More ambitious requirements for PV obligations in new construction and renovation in national building codes, parking canopies and other applications are essential. Subsidies should be increasingly redirected to applications where the market is still underdeveloped (building integration with a focus on facades, roofing in the traffic sector, agri-PV, floating PV, etc.).

The Austrian Photovoltaic Technology Platform (Österreichische Technoologieplattform Phootovoltaik: TPPV) sees building-integrated photovoltaics in particular as a promising path for Austria, whereby integration is seen as aesthetic/architectural as well as optimal systemic integration into the local energy system. This brings energy management and local load management, which can extend to supplying local mobility needs, into focus. [13, p. 52]

2022 and 2023 were characterized by particularly strong growth in installed PV capacity compared to previous years. This is due on the one hand to the briefly high energy prices in the wake of the Ukrainian conflict and the discussion about the availability of natural gas, and on the other hand to the fall in the cost of PV modules. The desire for greater independence from external price effects was also a main reason for this development. Simplifications in approvals and support processes are another reason for market growth. It cannot be assumed that the market will continue to grow at a rate of around 2 GW per year, especially as energy prices have now fallen again significantly and the strong expansion has led to greater restrictions on feed-in to the public grids.

Low and negative prices on the European energy market occur more frequently at times of high PV irradiation and give developers of large-scale projects reason to think about the long-term marketing of PV electricity, especially as these times will expand rapidly due to the further expansion of PV in Europe. The management of PV surpluses will therefore have to be given a great deal of attention very quickly; dynamic consumption and feed-in tariffs will benefit the storage market as well as other technological developments for the comprehensive use of flexible consumers (bidirectional e-charging,

tariff-controlled heating and cooling applications, commercial and industrial processes, etc.). Dynamic feed-in management in the distribution grid sector, which requires real-time monitoring of grid conditions, is necessary in order to be able to lift blanket feed-in limits and make full use of valuable photovoltaic energy during off-peak and winter periods.

Maintaining a high level of acceptance, a constant funding landscape, the rapid expansion of the electricity grids, the expansion of decentralized energy management including the comprehensive use of decentralized flexibility and storage are general cornerstones that will determine whether the domestic PV market develops into a constant market in the order of over 2 GW per year, which would be necessary for the planned climate neutrality in 2040. [13, p. 53]

Finally, there is an international outlook.

Most countries expect the volume of PV installations to remain stable or increase slightly in 2024. However, there are exceptions where political changes (Belgium), rising inflation and interest rates (Sweden) as well as problems such as grid bottlenecks and new fees (Denmark) could slow down the market. The Chinese market is expected to also see a decline in volume from 2023 onwards to between 190 GW and 220 GW, which is still significantly higher than the figures for 2022.

Conversely, other markets are likely to experience growth or stable volumes, driven by policy changes, new incentives (Switzerland), and simplifications for self-consumption, such as updated building codes promoting solar energy (South Korea). Many countries have a robust pipeline of projects currently in development, including Spain with 30 GW pending approval by 2023, France with 26 GW, and the USA with 30 to 40 GW awaiting grid connections and regulatory approvals.

High interest rates in several countries have diminished the competitiveness of PV systems, while extremely low module prices have hindered production projects, affecting the willingness to invest in PV development in key markets like the US and India. [16, p. 20]

The PV sector will certainly face many problems and challenges in the future. Nevertheless, the numbers just presented show that a lot has happened, especially in the record year 2023, and that a number of major projects in the key markets can also be expected in the coming years.

3 Methodology and Computational Framework

In this chapter, an overview of th projects workflow should be given. Also it should give an insight over the used methods and tools which where necessary to accomplish the project goals.

3.1 Overview

In the following, a brief summary of the most important tasks should be shown. Therefore the single tasks were assigned to five sections. For an easier understanding, the performed operations of the python-script are illustrated in the flowchart in Figure 10.

Preparation:

- Understanding of the project goal and definition of the desired output
- Seperation of the main task in sub tasks
- Project organisation (project documentary, online storage place for files, etc.)
- Literature review and examination of the previous work related to this project

PV-System Modeling:

- Investigation of existing python libraries/packages which fit the demands of our project – especially used open source library for our purposes: *pvlib python*
- Getting started with pvlib python:
  - Reading of documentation and studying similar best practice excamples on the pvlib-homepage [17]
  - Watching of tutorial videos (YouTube – Channel: Sasha Birk) [18]
- Usage of "*pvlib python"* for modeling of the PV-system with the specific parameters of the existing system at EVT Leoben
- Accessing and processing historical weather data from the *PVGIS* – data base inside the code unsing *pvlib python*

Deep Learning:

- Applying of existing code which uses a LSTM (Long Short-Term Memory) approach in order to develop a model which is able to predict a specific power output based on temperature and irradiance data

Forecasting:

- Investigation of tools/websites/weather-services which provide desired forecast parameters
- Access of data from datasets provided by *„GeoSphere Austria“* via API-calls within the python script
- Handling the format of the received data in a suitable way for further processing.

Output:

- Using the LSTM model and the formatted forecast-data to generate the desired load forecast of the PV-system

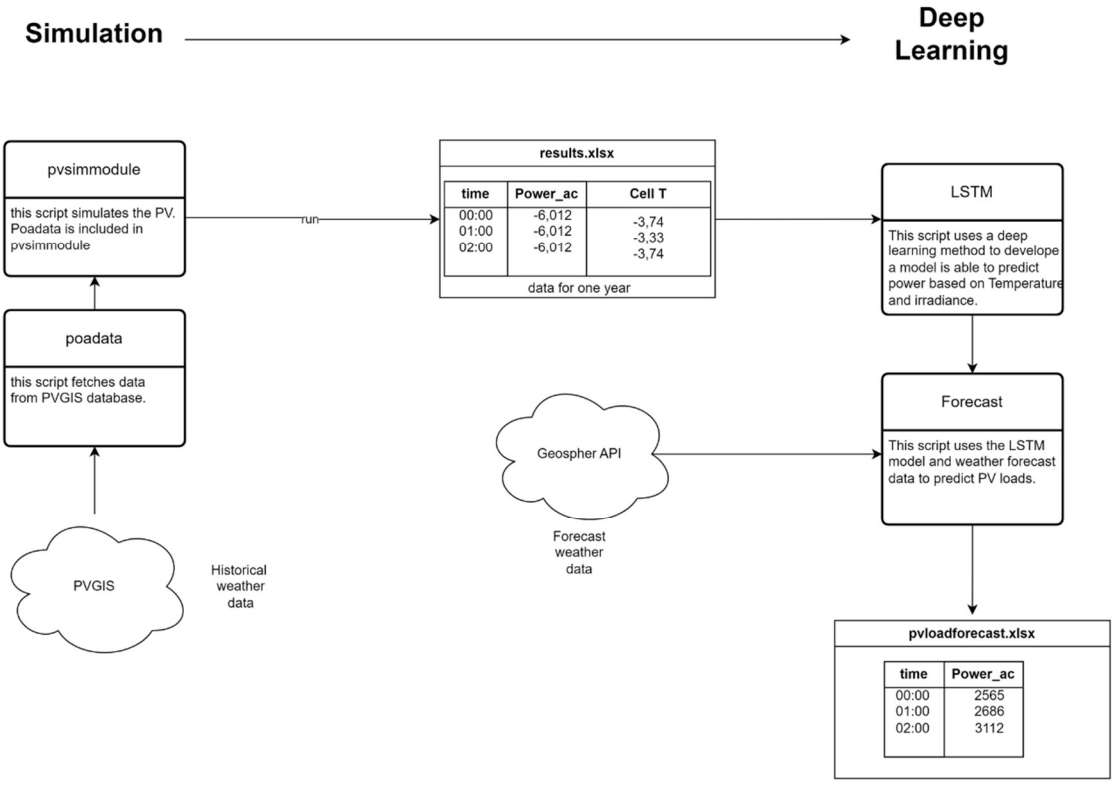


Figure 10: Illustration of the project results in form of a flowchart

3.2 Pvlib and used methods

Figure 11: pvlib logo

Modelling and simulating PV-systems in order to describe the behaviour of real life applications can be achieved in many different ways. One approach, the one we used for this project, is using the open-source library "*pvlib python*" for these purposes. [19, 20] Pvlib python is a toolbox developed by a community of scientists, researchers and many other contributors. It offers a set of functions and classes for simulating the performance of photovoltaic energy systems and accomplishing associated tasks. Pvlib started in 2013 when it was translated from the "*PVLIB for Matlab"* toolbox by *Sandia National Laboratories.* Today it contains many contributions from more than a hundred people which have been working on the development of pvlib python, causing the consistent growth of the toolbox until today. Pvlib python is maintained ba a core group of PV modelers from many different institutions. [17]

Regarding the mission of pvlib, one can read the following on the website:

"*The core mission of pvlib python is to provide open, reliable, interoperable, and benchmark implementations of PV system models*." [17]

3.2.1 Pvlib python – example: Calculating IV curves of a pv module

In the following example one possible use case for pvlib python should be shown.To investigate the performance of a PV module, IV- curves (current-voltage-curves) are often examinated. The resulting IV-curve of a PV-module depends on several parameters, e.g. temperature and irradiance.

However in this demonstrated case, the IV-curves for a specific PV module's (*Canadian Solar CS5P-220M*) base characteristics at reference conditions have been determined. To show the above mentioned dependency on temperature (T_{cell}) and irradiance (G_{eff}), five different IV-curves for five value pairs have been created.

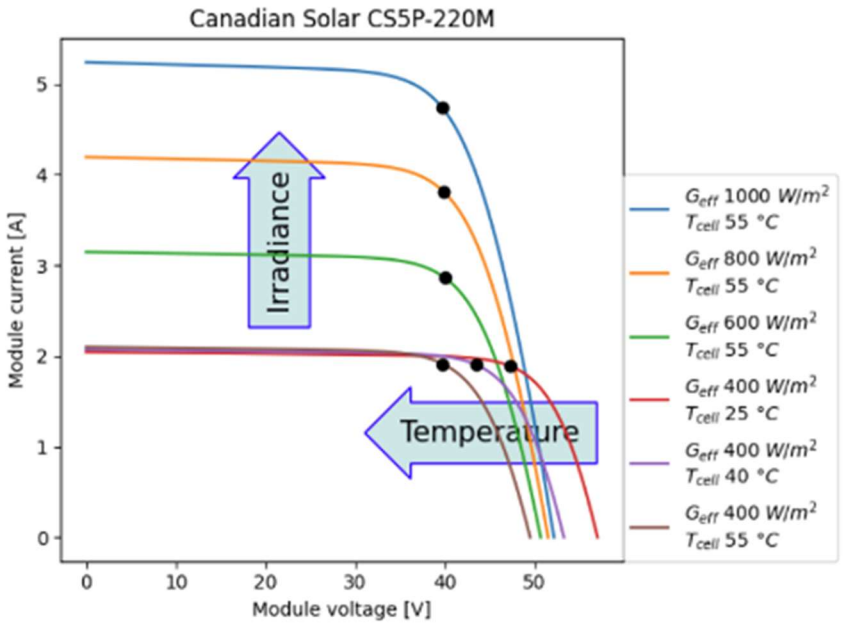
The used approach can be divided into two steps:

1. First step was to calculate five needed electrical parameters for an IV-curve at certain T_{cell} and G_{eff}. Therefore the De Soto model [21] was used.
2. In the second step the five received parameters were used to solve the so called *"single diode equation*", in order to get the desired IV-curves. The single diode equation is a circuit-equivalent model of a PV cell and is further discussed in chapter 2.1.4. [22]

Pv-lib python uses the function *pvlib.pvsystem.calcparams_desoto()* to calculate the needed electrical parameters to solve the single-diode equation.Then the functions *pvlib.pvsystem.singlediode()* and *pvlib.pvsystem.i_from_v()* can be used to generate the IV curves and receive the corresponding electrical output parameters (i_sc, v_oc, i_mp, v_mp, p_mp). [22]

In Figure 12 the resulting plot and console output can be observed. The code of the used python-script can be found in the

Appendix.



Out:

	i_sc	v_oc	i_mp	v_mp	p_mp
0	5.235561	52.129783	4.742475	39.614016	187.868473
1	4.190781	51.483033	3.805721	39.867812	151.725757
2	3.144837	50.649228	2.861983	39.956701	114.355380
3	2.043319	56.987478	1.886789	47.278408	89.204377
4	2.070523	53.238567	1.901044	43.490203	82.676791
5	2.097727	49.474044	1.912108	39.735026	75.977656

Figure 12: Results of IV-curves-modeling for a certain pv-module using ppvlib python [9]

3.3 Neural Networks and LSTM

Neural networks are computational models inspired by the human brain's structure, designed to recognize patterns in data. They consist of interconnected layers of nodes (neurons), where each node represents a mathematical function that processes input signals to produce outputs. Typically, neural networks are organized into three main layers: the input layer, hidden layers, and the output layer. The input layer receives raw data, the hidden layers extract features and learn patterns, and the output layer provides the final prediction or classification.

The learning process in neural networks involves adjusting the weights of the connections between nodes using a method called backpropagation. This iterative optimization process minimizes a predefined loss function to ensure that the model's predictions align closely with the true values. Common neural network architectures

include feedforward networks, convolutional neural networks (CNNs), and recurrent neural networks (RNNs).

Recurrent Neural Networks (RNNs) are specifically designed to handle sequential data by incorporating loops that allow information to persist across different time steps. However, traditional RNNs suffer from limitations like vanishing or exploding gradients, which hinder their ability to learn long-term dependencies in sequences. To address these issues, a more advanced variant called the Long Short-Term Memory (LSTM) network was introduced.

Long Short-Term Memory (LSTM) Architecture

LSTM networks are a specialized type of recurrent neural network (RNN) that overcome the limitations of traditional RNNs by introducing a memory cell that can maintain information over long periods. An LSTM unit consists of several interacting components, including a cell state, input gate, forget gate, and output gate. These components enable the network to regulate the flow of information, selectively remembering or forgetting information at different time steps.

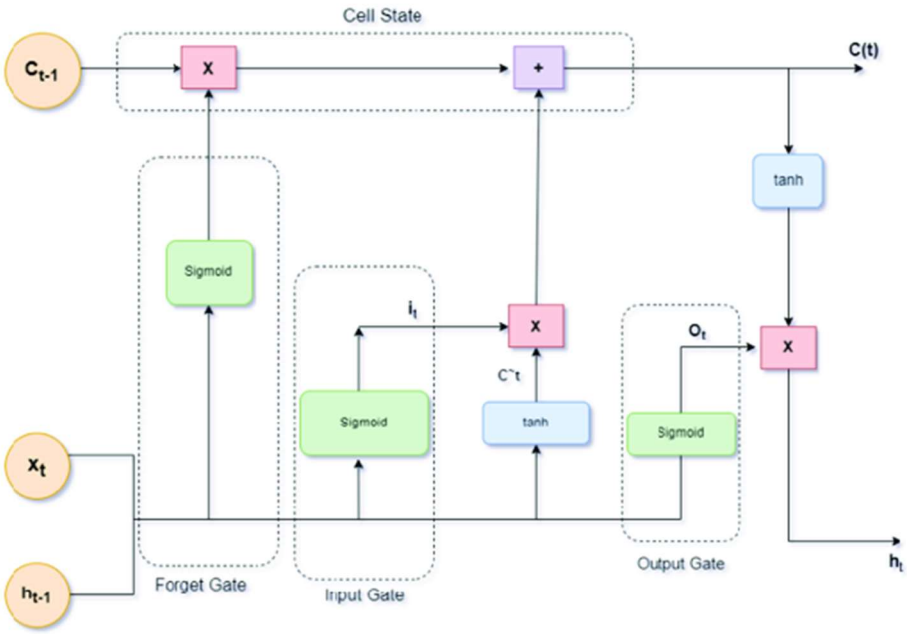


Figure 13: Illustration of the connections between the different gates in a LSTM unit

Cell State: The cell state is a crucial element in LSTMs that acts as a conveyor belt, allowing information to flow relatively unchanged across time steps. It can be modified by gates to maintain relevant information and discard irrelevant details.

Gates: LSTM networks use three gates to control the flow of information:

Forget Gate: Decides which information from the previous cell state should be discarded or kept, based on the current input and previous hidden state.

Input Gate: Determines what new information should be added to the cell state. It uses a sigmoid function to filter values and a tanh function to create new candidate values that can be added to the cell state.

Output Gate: Controls what part of the cell state should be output as the hidden state for the current time step, influencing subsequent predictions or inputs.

The mathematical operations governing the LSTM are as follows:

Forget Gate:

$$f_t = \sigma\left(W_f \cdot \left[h_{\{t-1\}}, x_t\right] + b_f\right)$$

Where f_t is the forget gate's activation, W_f and b_f are the weights and bias for the forget gate, $h_{\{t-1\}}$ is the previous hidden state, x_t is the current input, and σ is the sigmoid activation function.

Input Gate:

$$i_t = \sigma(W_i \cdot [h_{\{t-1\}}, x_t] + b_i)$$

$$\widetilde{C_t} = tanh\left(W_C \cdot \left[h_{\{t-1\}}, x_t\right] + b_C\right)$$

Where $i_t$is the input gate's activation, and $\widetilde{C_t}$ represents new candidate values.

Cell State Update:

$$C_t = f_t * C_{\{t-1\}} + i_t * \widetilde{C_t}$$

Where C_t is the updated cell state, and $*$ denotes element-wise multiplication.

Output Gate:

$$o_t = \sigma\left(W_o \cdot \left[h_{\{t-1\}}, x_t\right] + b_o\right)$$

$$h_t = o_t * tanh(C_t)$$

Where o_t is the output gate's activation, and h_t the hidden state.

The inclusion of these gates enables LSTM networks to effectively learn both short-term and long-term dependencies in sequential data, making them highly effective for time series forecasting tasks, such as predicting photovoltaic power based on historical weather conditions. This architecture's ability to capture temporal dependencies and handle varying sequence lengths has made LSTMs the go-to choice for many applications in sequence modeling, including speech recognition, natural language processing, and energy forecasting.

4 Implementation of simulation, deep learning and forecasting

The following chapter should give an insight on the abstraction of the already existing PV-system in Leoben to the simulated model, in which deep learning- and forecasting mechanisms are included.

In the first sub-chapter the relevant technical and geographical data of the PV-systems site in Leoben will be presented (). Afterwards a detailed description of the simulation and forecasting approaches should be given. This includes the descriptions of:

- the data retrieving methods (using PVGIS) – `poadata.py`
- the applied functions of the pvlib python library[2] - `pvsimmodule.py`
- the LSTM approach – `lstm.py`
- and the forecasting approach – `forecast.py`

4.1 Description of the real system

Table 2 shows the technical and geographical specifications of the existing PV-system on the top of the EVT-building in Leoben at Parkstraße 31, 8700 Leoben. The original spec sheet of the manufacturer of the solar panel can be observed in the

[2] In the following, not every used pvlib function will be discussed in detail. For further informations see pvlibs API [23].

Appendix in 8.1.

Table 2: Specifications of the existing PV-system (PV Leoben)

Parameter	Variable name in *pvsimmodule.py*	Value	Unit
Latitude	latitude	47.38	degrees
Longitude	longitude	15.10	degrees
Tilt angle	tilt	30	degrees
Azimuth angle	azimuth	149.72	degrees
Type of PV cell	celltype	polycristalline	/
Nominal max. power (Pmp) = Maximum power point (= MPP)	pdc0	240	W
Voltage at MPP (Vmp)	v_mp	29.87	V
Current at MPP (Imp)	i_mp	8.04	A
Open-circuit voltage (Voc)	v_oc	37.33	V
Short-circuit current (Isc)	i_sc	8.78	A
Temp. coefficient for Isc	alpha_sc	0.0041	A/K
Temp. coefficient for Voc	beta_voc	- 0.114	V/K
Temp. coefficient for Pmp	gamma_pdc	- 0.405	%/K
Number of cells in series	cells_in_series	69	/

An illustrated Google Maps view of the site is shown in Figure 14.

Figure 14: Google-Maps view of the PV-system location (EVT Leoben)

4.2 Simulation of the PV-system – *pvsimmodule.py*

For the simulation of the PV-system *Python* and *Microsoft Excel* (incl. *Excel VBA*) were used. As already discussed, especially the toolbox of pvlib python, played a major role concerning simulation and accomplishing further relevant tasks linked to the simulation. Very rich and helpful insights for working with pvlib python are offered in *YouTube*-tutorial videos by Sasha Birk. [18] The next chapters should:

- describe the data-retrieving.process with usage of PVGIS,
- give a small insight on the web interface of PVGIS and
- explain the essential parts of the scripts `poadata.py` and `pvsimmodule.py` step by step

4.2.1 Retreiving data from PVGIS

To calculate the output power[3] for a given PV-system, information about solar radiation and the data about the photovoltaic system performance are required. For this means we used pvlib to access open source data from the *Photovoltaic Geographical Information Service (PVGIS).* PVGIS is a web application which is managed by the *EU Science Hub* of the European Commission.[4] [25]

[3] and respectively the output energy
[4] Description of PVGIS online: *"It allows the user to get data on solar radiation and photovoltaic (PV) system energy production, at any place in most parts of the world." [24].*

To get the data from PVGIS with a function call inside the script, so called "*iotools"* from pvlib python were used. With these tools it's possible to access and retrieve data from extrernal providers. More on pvlib's iotools in the publication of Jensen, Anderson et al. .[20]

4.2.2 PVGIS interface – example:

Figure 15 illustrates the PVGIS interface for getting relevant data for an grid-connected PV-system. The user is able to provide input informations about the PV-system specifications by filling out insert- and drop down options. For our case, the required parameters for the EVT Leoben site, were entered.(latitudinal and longitudinal coordinates, solar radiation database, PV technology, installed peak power, system losses, Mounting position, slope, azimuth).[5] As result the service provides data about the yearly PV-system performance (e.g.: the average energy output [kWh] for each month over one year).

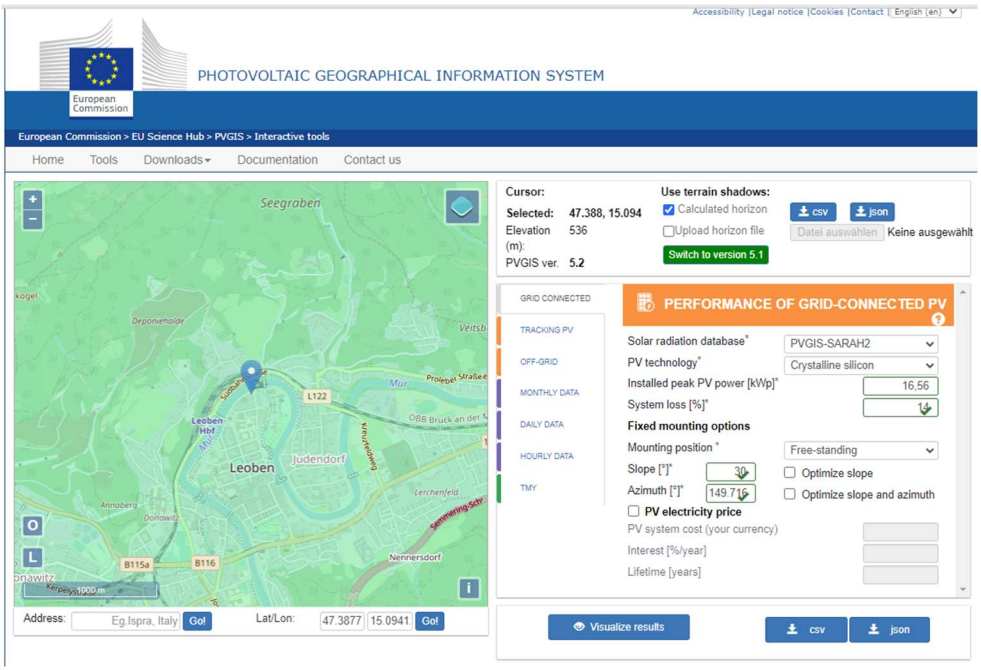


Figure 15: PVGIS interface with details from the PV-system at EVT Leoben

However, in `poadata.py` the data from PVGIS is requested and handled within the code and not manually as just described above. More details in 4.2.3.

[5] Especially crucial for the resulting output is the choice of the solar radiation database.

4.2.3 poadata.py: get_pvgis_data

`poadata.py` uses the just mentioned pvlibs iotools to retrieve historical weather data from within the code. This script is imported in `pvsimmodule.py` so that it's functions can be called directly from this script and one just has to execute one script when wanting to get the results of the simulation. The chosen solar radiation database in our case was `"PVGIS-SARAH2"`.

The variables latitude, longitude, tilt and azimuth have to be provided via user input and are needed to set the geographical position and the orientation of the PV-panels from the PV system at EVT Leoben. This is needed later on in *pvsimmodule.py* to create an instance of a `Location` object, which is defined in the *pvlib* library. Figure 2 illustrates the principle of measuring the zenith-, altitude- and azimuth angle.

The global radiation values are calculated in *poadata.py*:

```
poa_data['poa_global'] = poa_data['poa_diffuse'] + poa_data['poa_direct']
```

4.2.4 *pvsimmodule.py*: Parameters and helper functions

At the beginning of the script, after importing the needed packages, the most important parameters (start time, end time, specifications from data sheet etc.) are stored in the dictionary `PARAMS`.

The first helper function `get_date_attributes`, takes a `datetime` object as input and returns a tuple containing three attributes of the date: the weekday (1 for Monday to 7 for Sunday), the hour (0-23), and the month (1-12). It extracts these attributes using the `weekday()`, `hour`, and `month` methods from the `datetime` class.

The second helper function, `execute_vba_actions`, performs various data preparation actions on an Excel workbook. These actions ensure that the data is cleaned, formatted, and enriched, improving readability and enabling effective post-processing for deep learning applications.

The function utilizes the `openpyxl` library to manipulate the Excel workobook (`results.xlsx`). Specifically, it acts on two worksheets within this file: 'Model Chain Results' and 'POA Data'. The modifications include adjusting column widths for better readability, filling in missing data to ensure consistency, copying relevant data between worksheets to facilitate data integration, and adding derived features such as circular

encodings for time attributes. These circular encodings are particularly useful for representing temporal relationships effectively in downstream machine learning models.

The overall goal is to prepare the dataset for analysis by automating the cleaning, enrichment, and formatting steps, ensuring that the data is suitable for further processing, including deep learning workflows. By making these adjustments, the function streamlines the preparation process, ultimately improving the efficiency and quality of predictive modeling.

4.2.5 *pvsimmodule.py*: calculate_power_ouput

The function, `calculate_power_output`, calculates the power output of the photovoltaic (PV) system over a specified time period. Let's break down each part of the function step-by-step:[6]

Step 1

As mentioned in 4.2.3 a `Location` object is created including the informations about where the PV-system is located.

```
98. location = Location(latitude=latitude, longitude=longitude,
99.                         tz='Europe/Vienna', altitude=547.6, name='EVT')
```

Step 2

For retrieving the historical weather data, the in 4.2.3 discussed function `get_pvgis_data` from *poadata.py*, is used.

The call delivers "*plane of array*" (poa)- irradiance data. After that, this data is saved to a CSV-file, to then read it back to create a *pandas-DataFrame* (`poa_data_2020`) which stores the received data.[7] The index is set to a datetime range and with `poa_data = poa_data_2020[start:end]` the data is filtered to the specified start and end times.

```
101.     poa_data_2020 = poadata.get_pvgis_data(params['latitude'], params['longitude'],
2020, 2020, params['tilt'], params['azimuth'])
102.     poa_data_2020.to_csv("poa_data_2020_Leoben_EVT_io.csv")
103.     poa_data_2020 = pd.read_csv('poa_data_2020_Leoben_EVT_io.csv', index_col=0)
104.     poa_data_2020.index = pd.date_range(start='2020-01-01 00:00',
periods=len(poa_data_2020.index), freq="h")
105.     poa_data = poa_data_2020[params['start']:params['end']]
```

[6] Due to the fact that there is a lot of documentation (various docstrings and commentaries) inside the code, not every single detail should be explained in this section.
[7] *Pandas DataFrames* in Python are well suited for data handling and analysing in a convenient and easy interpretable way.

Step 3

In the next step solar position, effective irradiance, cell temperature and the angle of incidence (AOI) including the corresponding incidence angle modifier (IAM) are calculated. The angle of incidence, in degrees, is the angle between the solar vector and the surface normal of the PV-module. This is once again done by using some of pvlib functions. **Figure 16** illustrates the aoi.

```
solarpos = location.get_solarposition(times=pd.date_range(params['start'],
end=params['end'], freq="h"))
108.     aoi = pvlib.irradiance.aoi(params['tilt'], params['azimuth'],
solarpos.apparent_zenith, solarpos.azimuth)
109.     iam = pvlib.iam.ashrae(aoi)
110.     effective_irradiance = poa_data["poa_direct"] + iam + poa_data["poa_diffuse"]
111.     temp_cell = pvlib.temperature.faiman(poa_data["poa_global"],
poa_data["temp_air"], poa_data["wind_speed"])
```

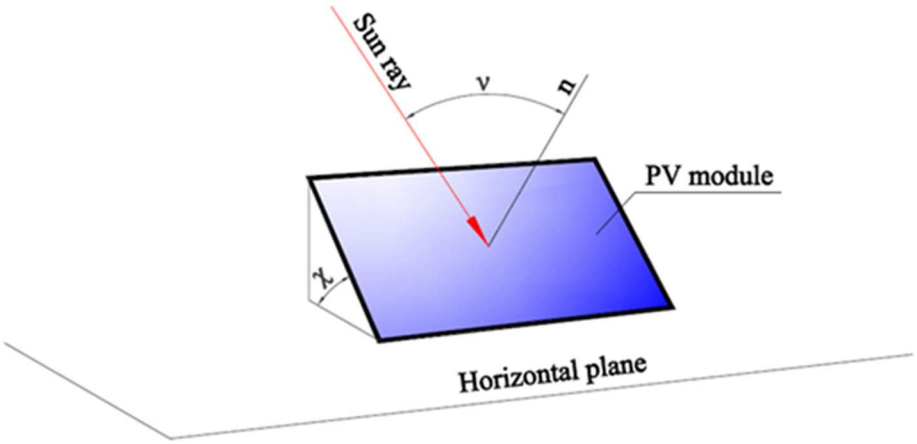


Figure 16: Illustration of the angle of incidence

Step 4

Now in *Step 4* the, in 2.1.4 discussed, characteristic single-diode model (SDM) parameters $(I_{L,ref}, I_{0,ref}, R_S, R_{sh,ref}, a_{ref})$ are calculated. As arguments the specific parameters from the existing PV-module specifcation-sheet, stored in `PARAMS`, are given to the function.

```
113.     I_L_ref, I_o_ref, R_s, R_sh_ref, a_ref, Adjust = pvlib.ivtools.sdm.fit_cec_sam(
114.         celltype=params['celltype'], v_mp=params['v_mp'], i_mp=params['i_mp'],
115.         v_oc=params['v_oc'], i_sc=params['i_sc'], alpha_sc=params['alpha_sc'],
116.         beta_voc=params['beta_voc'], gamma_pmp=params['gamma_pdc'],
cells_in_series=params['cells_in_series'])
```

With this calculated values the SDM equation, at a certain temperature and effective irradiance, can be computed using the CEC-model with the function call:

```
119.     cec_params = pvlib.pvsystem.calcparams_cec(effective_irradiance, temp_cell,
params['alpha_sc'], a_ref, I_L_ref, I_o_ref, R_sh_ref, R_s, Adjust)
```

The function returns the following parameters listed in **Table 3**.

Table 3: Resulted parameters solving the single diode equation [26]

Parameter	unit
Photocurrent – Light-generated current in amperes	A
saturation_current – Diode saturation curent in amperes	A
resistance_series – Series resistance in ohms	Ω
resistance_shunt – Shunt resistance in ohms	Ω
nNsVth – The product of the usual diode ideality factor (n, unitless), number of cells in series (Ns), and cell thermal voltage at specified effective irradiance and cell temperature.	V

Taking the received values from *cec_params*, the maximum power point (MPP) for each time step for a single module can finaly be calculated using *"Newton-Raphson"*-method.

```
122.     mpp = pvlib.pvsystem.max_power_point(*cec_params, method="newton")
```

Step 5

In step 5, the PV system configuration is realised. To recreate the existing system on top of EVT Leoben, 23 pv-modules per string and 3 strings per inverter were chosen. The function call *system.scale_voltage_current_power(mpp)* scales the MPP results to the system configuration for the DC site. After that, the desired inverter model can be chosen, in order to receive specific inverter data from the CEC database. Afterwards the calculation of the AC-power output, using the *"Sandia"-Model",* is possible.

```
123.     system = PVSystem(modules_per_string=23, strings_per_inverter=3)
124.     dc_scaled = system.scale_voltage_current_power(mpp)
126.     cec_inverters = pvlib.pvsystem.retrieve_sam('CECInverter')
127.     inverter = cec_inverters['Advanced_Energy_Industries__AE_3TL_23_10_08__480V_']
```

Step 6

At the end of the function the AC results (AC output power after the inverter) and the calculated DC parameters (current, voltage and power at MPP) are organized into a pandas `DataFrame` (`results_df`). At last this data and the original poa-data is written to an Excel file (*results.xlsx"*) with two sheets *("Model Chain Results"* and *"POA Data"*)

Finally the AC results and the poa-data are returned as a tuple of DataFrames.

```
128.     ac_results = pvlib.inverter.sandia(v_dc=dc_scaled.v_mp, p_dc=dc_scaled.p_mp,
inverter=inverter)
129.
130.     results_df = pd.concat([ac_results, dc_scaled.i_mp, dc_scaled.v_mp,
dc_scaled.p_mp, temp_cell], axis=1)
131.     results_df.columns = ['AC Power', 'DC scaled I_mp', 'DC scaled V_mp', 'DC scaled
P_mp', 'Cell Temperature']
132.
133.     with pd.ExcelWriter("results.xlsx") as writer:
134.         results_df.to_excel(writer, sheet_name='Model Chain Results')
135.         poa_data_2020.to_excel(writer, sheet_name='POA Data')
136.
137.     return ac_results, poa_data_2020
```

4.2.6 *pvsimmodule.py* - plot_results

The function *"plot_results"* should plot the results when executing the script. Therefore two plots are generated. The first one shows the obtained energy yield from the calculated time period. The second plot shows the calculated average value of the summed up energy yield for each month.

4.3 LSTM methodology

The dataset used in this study comprises photovoltaic (PV) system data, including hourly observations of AC power output (target variable), meteorological parameters such as air temperature, wind speed, and global irradiation, along with temporal information (e.g., hour of the day and month of the year). To enhance model performance, cyclical features were generated for time-related data, using sine and cosine transformations of the hour and month to capture seasonal and diurnal patterns. The processed dataset was split into training (80%) and testing (20%) sets.

The input features were normalized using MinMax scaling to ensure compatibility with the deep learning model. This scaling was applied independently to the training and

testing sets. The target variable, AC power, was also scaled using the same normalization technique.

4.3.1 Model Architecture

A deep learning model based on Long Short-Term Memory (LSTM) networks was employed for PV power prediction. The architecture was designed to capture temporal dependencies in the data, with the following configuration:

- **Input Layer:** Accepts sequences of input features over a sliding window of 24 time steps (i.e., a full day of hourly data).
- **LSTM Layers:** Three stacked LSTM layers were used, with the first two configured for sequence output. The first layer was bidirectional, enhancing the ability to capture both forward and backward temporal dependencies. Each LSTM layer contained 120 neurons and employed the ReLU activation function.
- **Dropout Regularization:** A dropout rate of 0.3 was applied to prevent overfitting.
- **Output Layer:** A dense layer with a single neuron was used to output the predicted AC power.

The model was compiled with the mean squared error (MSE) loss function and optimized using the Adam optimizer with a learning rate of 0.0005. Performance metrics included MSE and mean absolute error (MAE).

4.3.2 Sequence Construction

To prepare the data for the LSTM model, a sliding window approach was used. Each input sequence consisted of 24 consecutive hourly observations, and the corresponding target value was the AC power at the final time step of the sequence. This method ensured the model could learn temporal relationships in the data.

4.3.3 Training and Validation

The model was trained on the prepared dataset using early stopping and model checkpointing to enhance generalization and avoid overfitting. The training process was

stopped if the validation loss did not improve for 10 consecutive epochs, with the best model weights saved for further evaluation.

The model was trained for 10 epochs, with training and validation losses monitored during each epoch to assess convergence. The results showed stable learning behavior with minimal overfitting, as evidenced by the similarity of training and validation loss trends.

4.3.4 Evaluation Metrics

The performance of the model was evaluated on the test set using the following metrics:

- **Mean Squared Error (MSE):** Measures the average squared differences between predicted and observed values, penalizing larger errors.
- **Mean Absolute Error (MAE):** Reflects the average magnitude of errors in predictions, providing an intuitive measure of accuracy.
- **Mean Absolute Percentage Error (MAPE):** Captures the relative error as a percentage of actual values, providing insights into the prediction accuracy relative to the magnitude of the target variable.

4.3.5 Visualization

To validate the model predictions, the following visualizations were generated:

- **Time-Series Comparison:** A plot comparing the actual and predicted AC power over a sample period. This visualization highlighted the model's ability to capture the variability in power output, including peak generation periods.
- **Scatter Plot of Predictions vs. Actual Values:** Demonstrated strong agreement between predicted and observed values, with points closely aligned along the diagonal.
- **ROC-like Curve for Regression:** An experimental visualization based on true and false positive rates calculated for different thresholds. This plot provided insights into the model's ability to differentiate between high and low power outputs.

4.4 Forecasting – *forecast.py*

In the next chapter the handling of forecast-weather data will be discussed. This data handling is essential for generating the desired power-forecast of the PV-system, which is the main goal of this project. For this purpose, another python-script, `forecast.py`, was created. In the following section, the functions of `forecast.py` should be discussed.[8]

At the beginning the needed packages (e.g.: `pandas`, `numpy`, `joblib`, `json`, ...) are imported. The the saved models and scalers are definded.

```
18. # Load the saved model and scalers
19. model = load_model('final_model.h5', compile=False)
20. sc_x = joblib.load('scaler_x.pkl')
21. sc_y = joblib.load('scaler_y.pkl')
```

4.4.1 *forecast.py* - fetch_weather_data

In the first step string variable *`lat_lon`* is defined which holds the information about the longitude and latitude of the location of the PV-system. For the site of EVT Leoben the latitude value is given with *`47.38770748541585`* and the longitude value is given with *`15.094127778561258`*.

Then, the function creates variables which should hold the current time at the moment the script is executed and the desired start and end time of the forecast. (`current_time`, `fc_start_time` and `fc_end_time`)

Therefore several functions of the package `datetime` are used to get this values in "YYYY-MM-DDThh:mm"-format.

For retrieving weather forecast data from the “GeoSphereAustria”-API the function the API has to be addressed via it’s url.

To fetch data from the API, *`requests.get(url)`* is used. The url needs to be written in so-called *“Endpoint-structure”*. More information about how to correctly enter the url in

[8] Similarly to the code from the already discussed function `calculate_power_ouput` *in 4.2.5,* there is already a lot of documentation (various docstrings and commentaries) inside the code. Therof not every detail of the code should be explained in this section. The full code can be found in the

Appendix in sub-cahpter 8.3.

"Endpoint-structure"-syntax can be found in the user guide within the "GeoSphereAustria" dataset API documentation. [27]

The just mentioned variables *lat_lon*, *fc_start_time* and *fc_end_time* are integrated in the Endpoint-structure of the API-call.

In this way a user can choose the desired:

- historical, current, forecast- data
- Type of data (geospatial raster and station – grid or timeseries)
- Parameters of the dataset (e.g.: air temperature, global radiation, accumulated duration of sunshine, wind speed, etc.)

At the end of this function the dictionary `data` is returned.

```
if response.status_code == 200:
 39.         data = response.json()
 40.         print("Weather data fetched successfully.")
 41.         return data
 42.     else:
 43.         print(f"Failed to fetch data. Status code: {response.status_code}")
 44.         return None
```

4.4.2 *forecast.py* - process_weather_data

This process_weather_data function processes weather forecast data (`data`) and returns a structured pandas DataFrame.

Firstly several parts from data are extracted. Therefore the variables `time_list` (for the timestamps) and `parameters` (holds the weather forecast data, such as wind speed, global irradiation and sunshine duration) are created. Based on that the values for global irradiation are extracted and handled separately.

This values getting added up in `parameters`, which is not wanted for our purposes. Hence the difference between every value and it's value before is calculated, using a for-loop. Doing that manipulation, it's necessary to add a *"placeholder value"* ,which of course has to be 0 to get the correct calculated values. Also a convertion factor for the desired dimension of [J/m²s] is applied.

```
# Calculate global irradiation difference and scale to J/(m²*s)
 60.     global_irradiation = [(parameters['grad']['data'][i + 1] - val) / 3600 for i, val
in enumerate(parameters['grad']['data'][:-1])]
 61.
 62.     # Add a placeholder value for the first global_irradiation to match the length of
other data
 63.     global_irradiation.insert(0, 0)  # Adding 0 as a placeholder
```

In the next step a new dataframe `new_data` is created which should contain:

- the values from `time_list`, converted to the datetime-format
- the values for the air temperature from `parameter`
- the (absolute) values for the wind speed and
- the just discussed processed global irradiation values

In the last step, the in 4.2.4 explained method of circular encoding is applied to the hourly and monthly values from `timestamp`, in order to make this values independent from their datetime-format.

```
73.     # Extract hour and month features from the timestamp and apply circular encoding
74.     new_data['hour'] = new_data['timestamp'].dt.hour
75.     new_data['sin_hour'] = np.sin(2 * np.pi * new_data['hour'] / 24.0)
76.     new_data['cos_hour'] = np.cos(2 * np.pi * new_data['hour'] / 24.0)
77.     new_data['month'] = new_data['timestamp'].dt.month
78.     new_data['sin_month'] = np.sin(2 * np.pi * new_data['month'] / 12.0)
79.     new_data['cos_month'] = np.cos(2 * np.pi * new_data['month'] / 12.0)
```

Finally the dataframe `new_data` is returned.

```
113.         # Print the results to the console
114.         print("Predicted AC Power for the next day (in W):")
115.         print("| {:<20} | {:<15} | {:<15} | {:<15} | {:<25} |".format('Time', 'Power
(W)', 'Temp (°C)', 'Wind Speed (m/s)', 'Global Irradiation (J/(m²*s))'))
116.         print("|" + "-" * 102 + "|")
117.         print("-" * 90)
```

4.4.3 *forecast.py* – main()

The `main-function` is used for it's conventional purpose to execute the written functions of the program. Hence, `fetch_weather_data` is executed to get the weather data from the GeoSphere Austria API and the call of `process_weather_data` processes the fetched data in the in 4.4.2 explained way.

Furthermore the main function predicts the PV power output for the next 24 hours. It uses the LSTM functionalities from `lstm.py` to make predictions based on weather features like temperature, wind speed, and global irradiation. To have a more detailed insight the next steps are explained in subsections:

Definition of prediction features:

The function defines the set of the, already in 4.3 mentioned, "*features*" to be used for the prediction. These include the hour and month features from the timestamp `sin_hour`, `cos_hour`, `sin_month`, `cos_month` and the weather parameters `temp_air`, `wind_speed` and `global_irradiation`.

```
91. # Define features to be used for prediction
92. features = ['sin_hour', 'cos_hour', 'sin_month', 'cos_month', 'temp_air',
'wind_speed', 'global_irradiation']
```

Hourly power prediction:

The function iterates over the next 24 hours (line 96) and makes predictions for each hour.

For each hour, it:

- extracts the relevant features from the processed weather data (line 98).

```
97. # Get features for each hour
98. X_new = new_data[features].iloc[i].values.reshape(1, -1)
```

- scales the features using a preloaded scaler (sc_x, line 101), ensuring the input is in the correct format, to then reshape this data in into a suitable format for the LSTM-model

```
100. # Scale the features using the loaded scaler
101. X_new_scaled = sc_x.transform(X_new)
102.
103. # Reshape the data to fit the model's expected input shape (1, n_steps, n_features)
104. X_new_scaled = X_new_scaled.reshape((1, 1, len(features)))
```

- uses a preloaded LSTM (`model`, line 107) to predict the AC power output for that hour.

```
106. # Make predictions using the loaded model
107. prediction = model.predict(X_new_scaled)
```

- inversely transforms the prediction to return the actual power value (line 110) and appends the predicted power to the predicted_powers list (line 111).

```
109. # Inverse transform the predicted value to get the actual power output
110. predicted_power = sc_y.inverse_transform(prediction)
111. predicted_powers.append(predicted_power[0][0])
```

At the end the results get printed in the console in tabular format showing the power output for each hour.

5 Results

In this chapter the obtained results, when executing the above described python scripts, should be discussed. For this purpose, the seperate outputs of the executed scripts should be examined. To simplify this observation, the flow chart of the project results, which was already presented in 3.1, is shown again in this context.

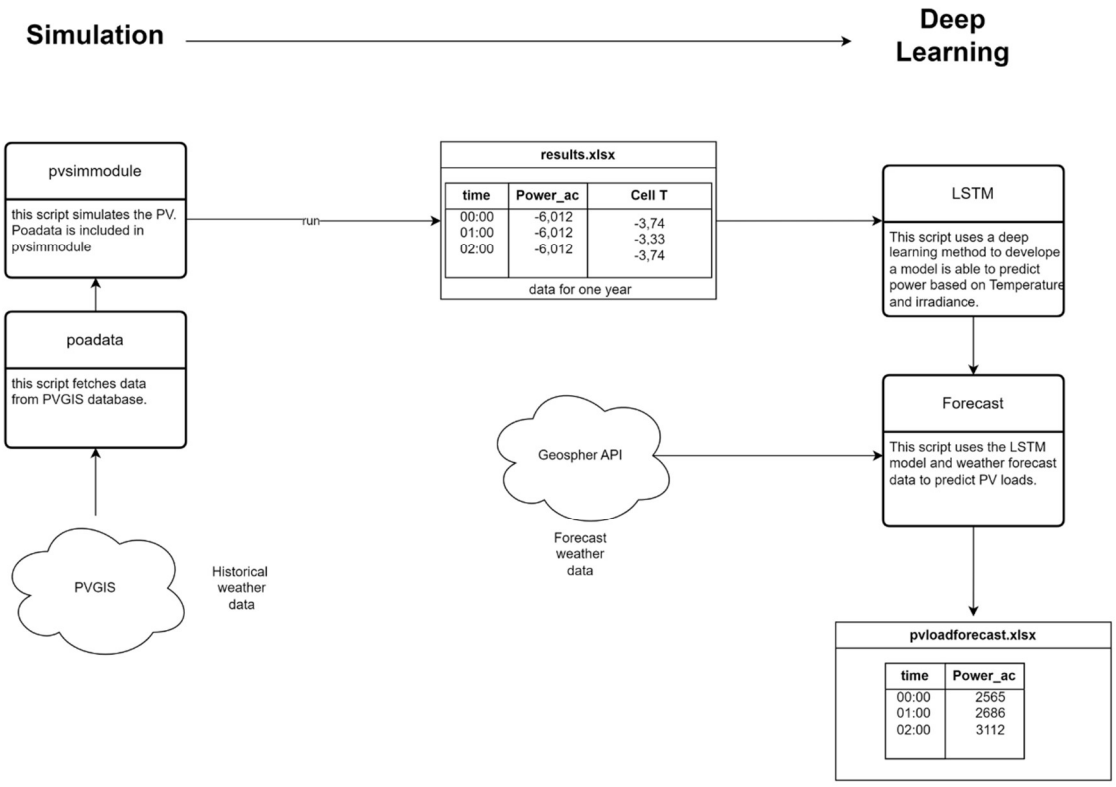


Figure 17: Illustration of the project results in form of a flowchart

5.1 Results of simulation (`poadata.py` and `pvsimmodule.py`)

The performed steps within the scripts, leading to the here described output, were mainly discussed in 4.2. As also already mentioned in 4.2, before running `pvsimmodule.py`, several user inputs have to be done in order to get the desired results.

Firstly the user needs to input several parameters (`latitude, longitude, tilt, azimuth angle, start year, end year`) inside `poadata.py`, for retrieving historical weather data from PVGIS.[9] Secondly the input of the PV-module specifications inside `pvsimmodule.py` has to be done. This data can be found on the corresponding data sheet of the PV-module. Then `pvsimmodule.py` can be executed by clicking "Run".[10] After performed calculations the script creates an Excel-file (`results.xlsx`) which stores the results of the simulation inside two data sheets.

The sheet "Model Chain results" stores performance related data (AC Power, I_mp, V_mp, P_mp, Cell_Temperature) and the sheet "POA Data" stores the retrieved data from PVGIS. Additionally to that in "Model Chain results" holds the circular encoded time data for "hour", "month" and "Weekday".[11] The relevant data for further processing with lstm.py, where the already described LSTM-methods are applied, are the following (see also in full code from `lstm.py` in 8.4)

- `'timestamp'`
- `'AC_Power'`
- `'temp_air'`
- `'wind_speed'`
- `'global_irradiation'`

Furthermore the plots of the Energy yield [Wh] and the corresponding summation curve [Wh] [12] for the inputted time period are generated. The plots for EVT Leoben can be observed in Figure 18 and Figure 19. For 2020 the months with the highest average energy-yield are April and July. The lower values for the autumn and winter months fit the expectations.

---

[9] The inputs for the project scenario for the PV-system at EVT Leoben can be examined in the Appendix 0.

[10] Since `poadata.py` was imported `pvsimmodule.py` the function `get_pvgis_data` is callable within `pvsimmodule.py`.

[11] The purpose of circular encoded data was mentioned in 4.2.4.

[12] The single calculated values can be summed with desired frequency in order to get a trend curve which shows the medium values for each time step. This can be done with `pandas.DataFrame.resample()`. In example for a monthly trend curve all values inside one month get added up and will be divided through the number of days of this month.

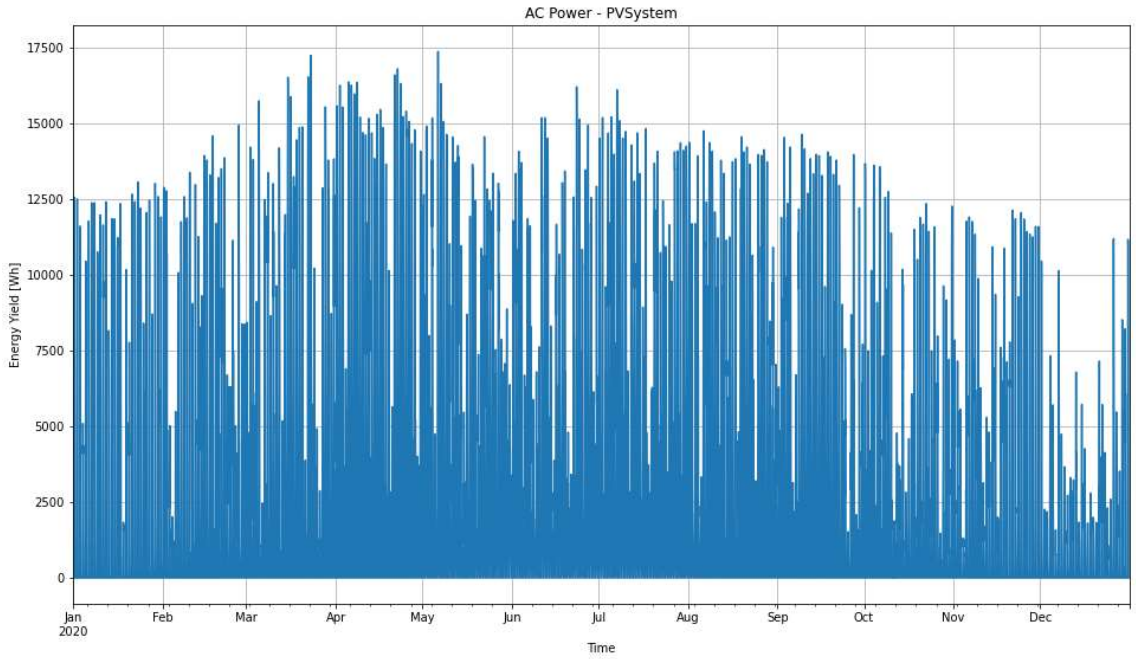


Figure 18: PV simulation results – AC Power PVSystem

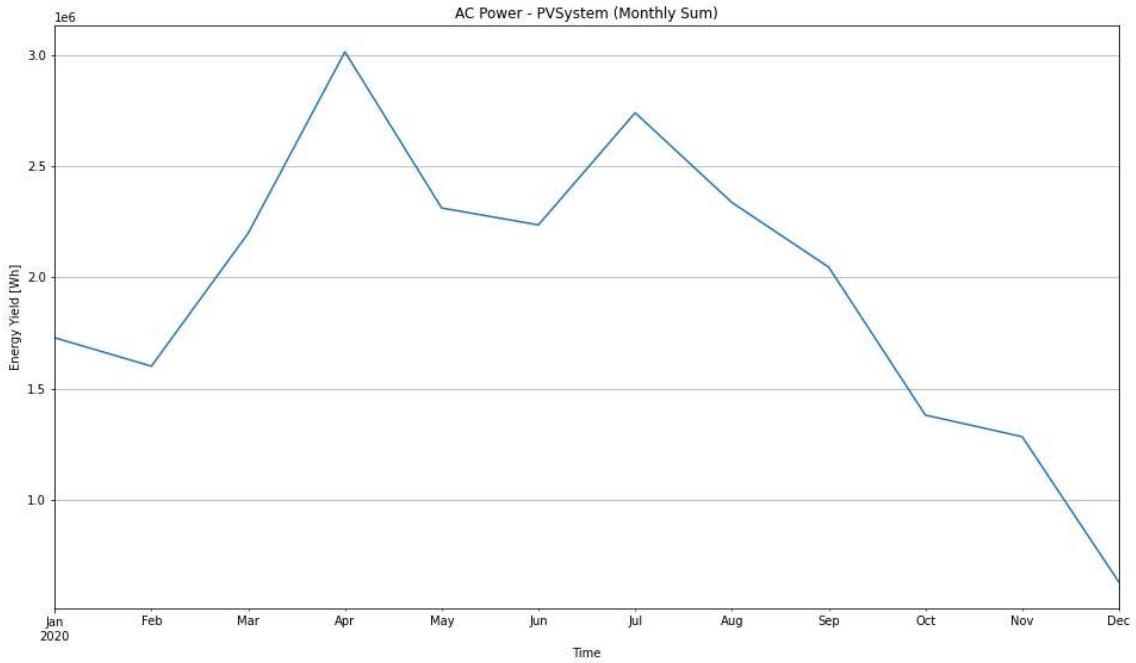


Figure 19: PV simulation results – AC Power PVSystem: Monthly sum

5.2 Results and accuracy of deep learning

To evaluate the performance of the developed deep learning model, we employed three key metrics: Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and the Coefficient of Determination (R^2). These metrics provide a comprehensive assessment of the model's accuracy, precision, and ability to capture the variability of the target variable.

1. **Root Mean Squared Error (RMSE):**

The RMSE quantifies the average magnitude of prediction errors by penalizing larger deviations more heavily. It is calculated as:

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

Where y_i and $\hat{y}_i$ are the observed and predicted values, respectively, and n is the number of data points. RMSE provides insights into the overall error magnitude and is particularly sensitive to outliers.

2. **Mean Absolute Error (MAE):**

The MAE measures the average absolute difference between the predicted and observed values, offering an intuitive metric for understanding model accuracy. It is defined as:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

Unlike RMSE, MAE treats all deviations equally and is less sensitive to large outliers, making it suitable for assessing the consistency of predictions.

3. **Coefficient of Determination (R^2):**

The R^2 score, or the coefficient of determination, evaluates the proportion of variance in the observed data that the model explains. It is expressed as:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

Where $\bar{y}$ is the mean of the observed values. An R^2 score close to 1 indicates that the model captures a significant portion of the variability in the target variable.

These metrics collectively provide a robust framework for assessing the predictive performance of the deep learning model, addressing both error magnitude and the model's explanatory power.

5.2.1 Dataset Summary

The dataset used for training and testing the deep learning model contains 8,784 hourly observations of AC power output and associated meteorological variables. Summary statistics for the target variable, **AC Power**, are presented below:

- **Mean:** 2,675.23 W
- **Standard Deviation:** 4,224.59 W
- **Minimum:** -6.01 W (system idle or measurement errors)
- **Median:** -6.01 W
- **Maximum:** 17,380.93 W

The data exhibits high variability, with a significant portion of observations corresponding to idle periods (AC Power ~ -6.01 W), reflecting the natural variability in PV system output due to weather and diurnal effects.

5.2.2 Model Performance

The test set performance of the deep learning model is evaluated using Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and the Coefficient of Determination (R^2):

Metric	Value	Unit
Root Mean Squared Error (RMSE)	1,311.74	W
Mean Absolute Error (MAE)	682.72	W

R^2 Score	0.9072	(Unitless)

Table 4: Test set performance metrics

- **Root Mean Squared Error (RMSE):** The RMSE of 1,311.74 W is modest compared to the mean AC power output (2,675.23 W) and standard deviation (4,224.59 W). This indicates the model captures the general trends in the data while being sensitive to larger deviations.
- **Mean Absolute Error (MAE):** The MAE of 682.72 W reflects the average prediction error magnitude, which is small relative to the mean power output. This suggests that the model performs consistently across the dataset.
- **Coefficient of Determination (R2*R*2):** An R^2 score of 0.9072 indicates that the model explains 90.72% of the variance in the target variable, demonstrating strong predictive capability.

5.2.3 Visual Analysis

The visual performance of the model is illustrated in Figures 20, and 21 which collectively demonstrate the model's effectiveness in learning and predicting AC power output.

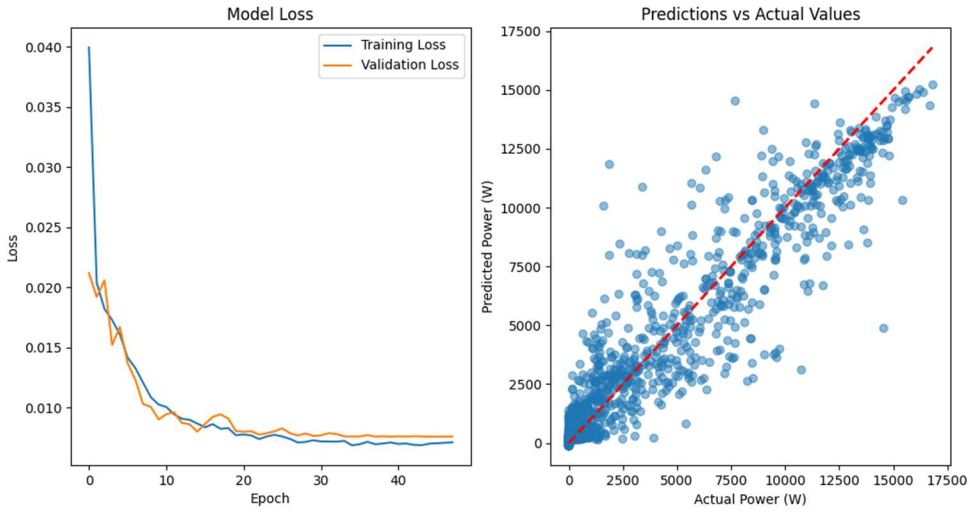


Figure 20: Model Loss and comparison of actual and predicted values

Figure 20 presents the convergence of the training and validation loss over epochs, highlighting a stable reduction in both metrics. This indicates effective model learning while avoiding significant overfitting, as evidenced by the similar trends between training and validation losses throughout the training process. Additionally, Figure 20 includes a scatter plot comparing the predicted and actual AC power values. The points in the scatter plot are tightly clustered along the diagonal line, reflecting strong agreement between the model's predictions and the ground truth values across the test set. This alignment suggests that the model accurately captures the variability of the target variable, even under conditions of high power output.

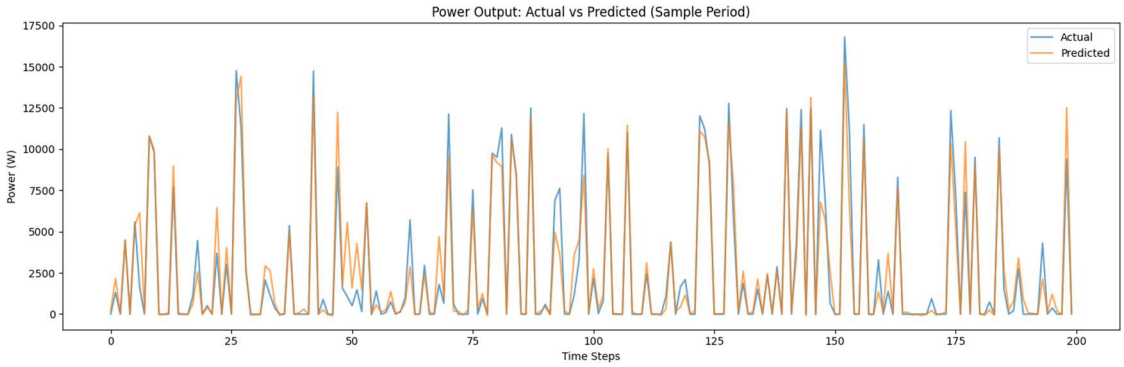


Figure 21: PV power actual vs predicted in time series

Further insight is provided by Figure 21, which shows a time-series comparison of predicted and actual AC power over a sample period. The predicted values closely follow the actual values, with minimal deviations observed during peak power output. This consistency underscores the model's ability to effectively generalize across various scenarios, capturing both baseline and extreme power outputs with high fidelity. These visualizations confirm the model's robustness and validate its suitability for practical photovoltaic power forecasting applications.

5.2.4 Discussion

The results highlight the model's effectiveness in predicting AC power output with high accuracy. The low RMSE and MAE values indicate robust performance, while the high R^2 score suggests that the model captures the inherent variability in the dataset. However, challenges remain in predicting extreme values, especially during periods of low or zero power generation. These observations suggest the need for further optimization, particularly under adverse or unusual conditions

6 Summary and outlook.

This study explored the integration of machine learning techniques with simulated photovoltaic (PV) system data to enhance the accuracy of short-term load forecasting (STLF). By employing a Long Short-Term Memory (LSTM) neural network, the research demonstrated the feasibility of predicting PV power output using simulated weather data obtained through PVGIS and processed with the pvlib Python library. The simulation accuracy achieved using pvlib Python validated the ability to replicate real-world PV system behavior under varying conditions. Furthermore, the predictive performance of the LSTM model was robust, as reflected by low error metrics such as Mean Squared Error (MSE) and Mean Absolute Error (MAE). These results underscore the model's ability to effectively capture temporal dependencies in the data, thereby enhancing the reliability of PV power forecasts.

The developed pipeline for handling historical and forecasted weather data ensured seamless integration with the predictive model. This comprehensive framework facilitated the efficient processing of data and streamlined model training and forecasting. Visualization techniques further supported the validation of the model's predictions, showcasing strong alignment between the actual and predicted values. The findings highlight the potential of hybrid approaches, combining data-driven machine learning methods with physical PV simulations, to significantly improve the quality and reliability of STLF in smart grids.

Looking forward, the growing penetration of renewable energy sources, especially PV systems, presents an urgent need for even more accurate and scalable forecasting models. Enhancing the LSTM model with advanced architectures, such as Transformer networks or hybrid models that integrate physical and data-driven methodologies, could provide more precise and adaptive solutions. Moreover, incorporating real-time weather forecasts and satellite imagery may improve data quality and enrich model inputs, further boosting prediction accuracy.

The scalability of these predictive models will also be critical as they are extended to forecast larger PV systems or entire regions. Such scalability will ensure that these models can be deployed in real-world scenarios to support grid management and

optimization. Additionally, integrating these forecasts into energy management systems for real-time grid balancing and operational efficiency represents an essential next step in leveraging predictive analytics for renewable energy systems.

Addressing the challenges of robustness and generalization will be vital for adapting the model to different climatic conditions and exploring its application to other renewable energy sources, such as wind or hydropower. These advancements will contribute to more resilient and versatile forecasting solutions, enabling better alignment with the dynamic needs of modern energy systems.

In conclusion, this research demonstrates the potential of machine learning models, combined with simulated data, to transform PV power forecasting. Continued innovation in model development, data integration, and system implementation will play a pivotal role in achieving global energy transition goals and fostering a more sustainable and efficient energy future.

7 Bibliography

[1] J. Kleissl, red., *Solar energy forecasting and resource assessment*. Amsterdam, Boston: Elsevier AP Academic Press in an imprint of Elsevier, 2013.

[2] S. A. KALOGIROU, *SOLAR ENERGY ENGINEERING: Processes and systems*. S.l.: ELSEVIER ACADEMIC PRESS, 2023.

[3] T. Stoffel *et al.,* "Concentrating Solar Power: Best Practices Handbook for the Collection and Use of Solar Resource Data (CSP),," National Renewable Energy Laboratory (NREL), September/2010.

[4] T. Stoffel *et al.,* "Concentrating Solar Power: Best Practices Handbook for the Collection and Use of Solar Resource Data (CSP), NREL (National Renewable Energy Laboratory),"

[5] GW Green Energie GmbH, *Wie Der Photoelektrische Effekt Die Sonnenenergie In Elektrischen Strom Verwandelt - Entdecken Sie Das Geheimnis Hinter PV-Anlagen! | GW Green Energie GmbH.* [Online] Hentet fra: https://gw-greenenergie.de/photoelektrische-effekt/. Lastet ned: aug. 18 2024.

[6] N. Gupta, G. F. Alapatt, R. Podila, R. Singh og K. F. Poole, "Prospects of Nanostructure-Based Solar Cells for Manufacturing Future Generations of Photovoltaic Modules," *International Journal of Photoenergy*, vol. 2009, nr. 1, 2009, Art. nr. 154059. [Online] Hentet fra: doi:10.1155/2009/154059.

[7] *Silizium-Solarzellen | LEIFIphysik.* [Online] Hentet fra: https://www.leifiphysik.de/elektronik/halbleiterdiode/grundwissen/silizium-solarzellen. Lastet ned: aug. 18 2024.

[8] S. Bader, X. Ma og B. Oelmann, "One-diode photovoltaic model parameters at indoor illumination levels – A comparison," *Solar Energy*, vol. 180, s. 707–716, 2019. [Online] Hentet fra: doi:10.1016/j.solener.2019.01.048.

[9] C. Gradwohl *et al.,* "A Combined Approach for Model-Based PV Power Plant Failure Detection and Diagnostic," *Energies*, vol. 14, nr. 5, s. 1261, 2021. [Online] Hentet fra: doi:10.3390/en14051261.

[10] G. C. Eder, K. A. Berger og G. Oreski, "IEA Photovoltaik (PVPS) Task 13: Zuverlässigkeit und Ertragssicherheit von Photovoltaik-Anlagen: Arbeitsperiode 2018 – 2021," Österreichisches Bundesministerium für Klimaschutz, Umwelt, Energie, Mobilität, Innovation und Technologie, Wien, Mai/2023.

[11] Österreichisches Bundesministerium für Klimaschutz, Umwelt, Energie, Mobilität, Innovation und Technologie, “Österreichische Photovoltaik-Strategie: Klimaneutral bis 2040 dank der Sonnenkraft – so geht’s!,”

[12] *Bundesgesetz über den Ausbau von Energie aus erneuerbaren Quellen (Erneuerbaren-Ausbau-Gesetz – EAG): EAG*, 2024.

[13] K. Leonhartsberger, H. Fechner og S. Savic, “Innovative Energietechnologien in Österreich Marktentwicklung 2023: Technologiereport Photovoltaik,” Berichte aus Energie- und Umweltforschung, Österreichisches Bundesministerium für Klimaschutz, Umwelt, Energie, Mobilität, Innovation und Technologie, Wien, 2024.

[14] Bundesverband Photovoltaic Austria, “DIE ÖSTERREICHISCHE PHOTOVOLTAIK- UND SPEICHER-BRANCHE 2023 IN ZAHLEN,” Bundesverband Photovoltaic Austria, Juni/2024.

[15] P. Biermayer *et al.,* “Innovative Energietechnologien in Österreich Marktentwicklung 2023: Biomasse, Photovoltaik, Photovoltaik-Batteriespeicher, Solarthermie, Großwärmespeicher, Wärmepumpen, Gebäudeaktivierung, Windkraft und innovative Energiespeicher,” Berichte aus Energie- und Umweltforschung, Wien, 2024.

[16] INTERNATIONAL ENERGY AGENCY PHOTOVOLTAIC POWER SYSTEMS PROGRAMME, “Snapshot of Global PV Markets 2024,” INTERNATIONAL ENERGY AGENCY PHOTOVOLTAIC POWER SYSTEMS PROGRAMME, 2024.

[17] *pvlib python — pvlib python 0.10.5 documentation.* [Online] Hentet fra: https://pvlib-python.readthedocs.io/en/stable/index.html. Lastet ned: jun. 05 2024.

[18] Sasha Birk, *pvlib python - YouTube*.

[19] K. S. Anderson, C. W. Hansen, W. F. Holmgren, A. R. Jensen, M. A. Mikofski og A. Driesse, “pvlib python: 2023 project update,” *JOSS*, vol. 8, nr. 92, s. 5994, 2023. [Online] Hentet fra: https://joss.theoj.org/papers/10.21105/joss.05994#. [Online] Hentet fra: doi:10.21105/joss.05994.

[20] A. R. Jensen *et al.,* “pvlib iotools—Open-source Python functions for seamless access to solar irradiance data,” *Solar Energy*, vol. 266, s. 112092, 2023. [Online] Hentet fra: https://www.sciencedirect.com/science/article/pii/S0038092X23007260. [Online] Hentet fra: doi:10.1016/j.solener.2023.112092.

[21] W. De Soto, “Improvement and validation of a model for photovoltaic array performance,” *Solar Energy*, nr. vol 80, s. 78–88, 2006.

[22] *Calculating a module’s IV curves — pvlib python 0.10.5 documentation.* [Online] Hentet fra: https://pvlib-python.readthedocs.io/en/stable/gallery/iv-

modeling/plot_singlediode.html#sphx-glr-gallery-iv-modeling-plot-singlediode-py. Lastet ned: jun. 05 2024.

[23] *API reference — pvlib python 0.11.0 documentation.* [Online] Hentet fra: https://pvlib-python.readthedocs.io/en/stable/reference/index.html. Lastet ned: jul. 26 2024.

[24] EU Science Hub, *PVGIS user manual.* [Online] Hentet fra: https://joint-research-centre.ec.europa.eu/photovoltaic-geographical-information-system-pvgis/getting-started-pvgis/pvgis-user-manual_en. Lastet ned: jun. 23 2024.

[25] EU Science Hub, *Photovoltaic Geographical Information System (PVGIS).* [Online] Hentet fra: https://joint-research-centre.ec.europa.eu/photovoltaic-geographical-information-system-pvgis_en. Lastet ned: jun. 23 2024.

[26] *pvlib.pvsystem.calcparams_cec — pvlib python 0.11.0 documentation.* [Online] Hentet fra: https://pvlib-python.readthedocs.io/en/stable/reference/generated/pvlib.pvsystem.calcparams_cec.html?highlight=pvlib%20pvsystem%20calcparams_ce. Lastet ned: jul. 26 2024.

[27] *Endpoints — Dataset API Documentation v1 documentation.* [Online] Hentet fra: https://dataset.api.hub.geosphere.at/v1/docs/user-guide/endpoints.html. Lastet ned: jun. 22 2024.

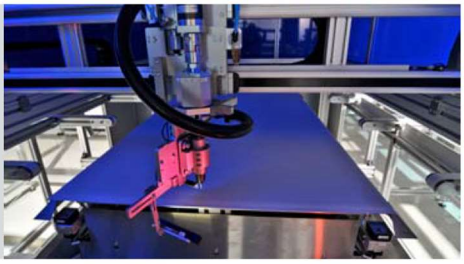
8 Appendix

The following chapter contains:

- the data sheet of the PV-module type which is installed at the PV-system at EVT Leoben (8.1)
- the python code of the "*pvlib python"* use case example (8.2)
- the codes of the created python-scripts within this project (8.2-8.4)
- and an example for the so called *"Endpoint-structure"* to retreive forecast-data from the *"GeoSphere Austra*"-API. (8.5)

8.1 Data sheet "*KPV PE NEC 235 / 240 Wp*"

KPV Photovoltaik-Module übertreffen alle Qualitätsanforderungen der europäischen Märkte. Die hocheffizienten, polykristallinen Solarmodule auf der Basis von 6" Wafern werden mit modernstem Equipment ausschließlich in Österreich produziert. Ein innovatives Messverfahren garantiert engste Toleranzen. Die Solarmodule mit einer Leistung von 235 bis 240 Wp werden mit Aluminiumrahmen und leitenden Eckverbindern ausgeliefert. Serienmäßig ist eine Tyco Anschlussdose angebracht. Es verlassen nur erstklassige Solarmodule die Fertigungsstraße der KIOTO Photovoltaics GmbH. Diese leistungs- und ertragsoptimierten Solarmodule sind vorwiegend zum Einsatz in netzgekoppelten Anlagen bestimmt.



KPV PE NEC

Leistungsklassen:

Type	$Pmpp_{[Wp]}$	$Umpp_{[V]}$	$Impp_{[A]}$	$Uoc_{[V]}$	$Isc_{[A]}$	Wirkungsgrad	Flächenbedarf pro kWp
KPV 235 PE	235 Wp	29,82 V	7,97 A	37,24 V	8,61 A	14,22%	7,03 m²
KPV 240 PE	240 Wp	29,87 V	8,04 A	37,33 V	8,78 A	14,52%	6,89 m²

Elektrische Daten:

60 polykristalline Zellen:	156mm x 156mm
Anschlussystem:	Tyco-Solarlok®, Steckverbinder 4 mm²
Max. Systemspannung:	1000 V DC
Leistungstoleranz:	(+ 3% / - 0%)*
Temperaturkoeffizienten:	Pmpp= -0,405%/K / Uoc= -114mV/K / Isc= +4,1mA/K
Standard Test Bedingungen:	AM1,5 / 1000 W pro m² / 25°C
Umgebungstemperatur:	+ 85°C bis - 40°C
Kabellänge:	2000mm
Bypassdioden:	3 Stk. Tyco SL1515
Leistungsgarantie:	min. 97% im ersten Jahr, danach max. Reduktion um 0,70% p.a. bis zu 25 Jahren
Produktgarantie:	12 Jahre

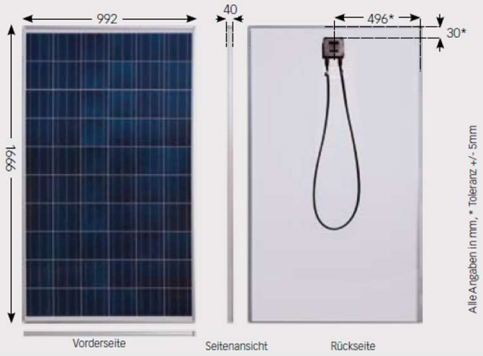
* Measurement: STC (standard test conditions)

Technische Daten:

inkl. Alurahmen:	1666 mm x 992 mm x 40 mm (+/-2 mm)
Laminate:	1659 mm x 985 mm x 4,5 mm (Dosenhöhe 22,5 mm)
Gewicht mit/ohne Rahmen:	19,50 kg / 17,50 kg
Glasspezifikationen:	Solarglas ESG 3,2 mm
Verkapselungsmaterial:	Etimex
Rückseitenmaterial:	Isovoltaic
Prüfzertifikat:	IEC 61215, Ed. 2, IEC 61730; IP 65, MCS - Zertifikat
Erweiterte Hageltests:	Hagelkorngröße 25mm, maximale Geschwindigkeit von 46m/s (165,6km/h) und Hagelkorngröße 55mm, maximale Geschwindigkeit von 33,5m/s (120,6km/h)
Salznebeltest:	Min. 96 Stunden in einem hochkonzentrierten Salznebel

www.tuv.com TÜVRheinland ID: 0000025707
- Qualified, IEC 61215
- Safety tested, IEC 61730
- Periodic Inspection

www.tuv.com TÜVRheinland ID: 0000025707
- Salt corrosion resistance tested
- Periodic Inspection



APPROVED PRODUCT
Certificate No.: MCS PV0025
Photovoltaic systems

qualityaustria SYSTEMZERTIFIZIERT ISO 9001:2008 NR.08987/0

IQNet

AUSTRIA

März 2012

Photovoltaic Power.

KPV Solar GmbH
Solarstrasse 1
9300 Sankt Veit/Glan
AUSTRIA

T +43 (0) 4212 33 300 0
F +43 (0) 4212 33 300 799
E office@kpv-solar.com
W www.kpv-solar.com

fig. 1: spec sheet *KPV PE NEC 235 / 240 Wp*

8.2 *Pvlib python*: use case example - IV curve modeling

The following python code was used to obtain the presented results from the use case example from chapter 3.2.1.

```
from pvlib import pvsystem
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# Example module parameters for the Canadian Solar CS5P-220M:
parameters = {
    'Name': 'Canadian Solar CS5P-220M',
    'BIPV': 'N',
    'Date': '10/5/2009',
    'T_NOCT': 42.4,
    'A_c': 1.7,
    'N_s': 96,
    'I_sc_ref': 5.1,
    'V_oc_ref': 59.4,
    'I_mp_ref': 4.69,
    'V_mp_ref': 46.9,
    'alpha_sc': 0.004539,
    'beta_oc': -0.22216,
    'a_ref': 2.6373,
    'I_L_ref': 5.114,
    'I_o_ref': 8.196e-10,
    'R_s': 1.065,
    'R_sh_ref': 381.68,
    'Adjust': 8.7,
    'gamma_r': -0.476,
    'Version': 'MM106',
    'PTC': 200.1,
    'Technology': 'Mono-c-Si',
}

cases = [
    (1000, 55),
    (800, 55),
    (600, 55),
    (400, 25),
    (400, 40),
    (400, 55)
]

conditions = pd.DataFrame(cases, columns=['Geff', 'Tcell'])

# adjust the reference parameters according to the operating
# conditions using the De Soto model:
IL, I0, Rs, Rsh, nNsVth = pvsystem.calcparams_desoto(
    conditions['Geff'],
    conditions['Tcell'],
    alpha_sc=parameters['alpha_sc'],
    a_ref=parameters['a_ref'],
    I_L_ref=parameters['I_L_ref'],
    I_o_ref=parameters['I_o_ref'],
    R_sh_ref=parameters['R_sh_ref'],
    R_s=parameters['R_s'],
    EgRef=1.121,
    dEgdT=-0.0002677
)

# plug the parameters into the SDE and solve for IV curves:
SDE_params = {
    'photocurrent': IL,
```

```
    'saturation_current': I0,
    'resistance_series': Rs,
    'resistance_shunt': Rsh,
    'nNsVth': nNsVth
}
curve_info = pvsystem.singlediode(method='lambertw', **SDE_params)
v = pd.DataFrame(np.linspace(0., curve_info['v_oc'], 100))
i = pd.DataFrame(pvsystem.i_from_v(voltage=v, method='lambertw', **SDE_params))

# plot the calculated curves:
plt.figure()
for idx, case in conditions.iterrows():
    label = (
        "$G_{eff}$ " + f"{case['Geff']} $W/m^2$\n"
        "$T_{cell}$ " + f"{case['Tcell']} $\\degree C$"
    )
    plt.plot(v[idx], i[idx], label=label)
    v_mp = curve_info['v_mp'][idx]
    i_mp = curve_info['i_mp'][idx]
    # mark the MPP
    plt.plot([v_mp], [i_mp], ls='', marker='o', c='k')

plt.legend(loc=(1.0, 0))
plt.xlabel('Module voltage [V]')
plt.ylabel('Module current [A]')
plt.title(parameters['Name'])
plt.gcf().set_tight_layout(True)


# draw trend arrows
def draw_arrow(ax, label, x0, y0, rotation, size, direction):
    style = direction + 'arrow'
    bbox_props = dict(boxstyle=style, fc=(0.8, 0.9, 0.9), ec="b", lw=1)
    t = ax.text(x0, y0, label, ha="left", va="bottom", rotation=rotation,
                size=size, bbox=bbox_props, zorder=-1)

    bb = t.get_bbox_patch()
    bb.set_boxstyle(style, pad=0.6)


ax = plt.gca()
draw_arrow(ax, 'Irradiance', 20, 2.5, 90, 15, 'r')
draw_arrow(ax, 'Temperature', 35, 1, 0, 15, 'l')
plt.show()

print(pd.DataFrame({
    'i_sc': curve_info['i_sc'],
    'v_oc': curve_info['v_oc'],
    'i_mp': curve_info['i_mp'],
    'v_mp': curve_info['v_mp'],
    'p_mp': curve_info['p_mp'],
}))

```

8.1 Poadata.py

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

"""
Author: Michael Gruen
Email: michaelgruen@hotmail.com
Created: 01.05.2024 08:17
Project: pvforecast
"""

```

```
import pandas as pd
import pvlib



def get_pvgis_data(latitude, longitude, start_year, end_year, tilt, azimuth):
    """
    Retrieve hourly plane-of-array (POA) irradiance data from the PVGIS API.

    Args:
        latitude (float): Latitude of the location.
        longitude (float): Longitude of the location.
        start_year (int): Start year for the data retrieval.
        end_year (int): End year for the data retrieval.
        tilt (float): Tilt angle of the PV surface in degrees.
        azimuth (float): Azimuth angle of the PV surface in degrees.

    Returns:
        pd.DataFrame: DataFrame containing the POA irradiance data.
    """
    poa_data, meta, inputs = pvlib.iotools.get_pvgis_hourly(
        latitude=latitude, longitude=longitude,
        start=start_year, end=end_year, raddatabase="PVGIS-SARAH2",
        components=True, surface_tilt=tilt, surface_azimuth=azimuth,
        outputformat='json', usehorizon=True, userhorizon=None,
        pvcalculation=False, peakpower=None, pvtechchoice='crystSi',
        mountingplace='free', loss=0, trackingtype=0, optimal_surface_tilt=False,
        optimalangles=False, url='https://re.jrc.ec.europa.eu/api/v5_2/',
        map_variables=True, timeout=30
    )

    # Calculate diffuse and global POA irradiance
    poa_data['poa_diffuse'] = poa_data['poa_sky_diffuse'] +
poa_data['poa_ground_diffuse']
    poa_data['poa_global'] = poa_data['poa_diffuse'] + poa_data['poa_direct']

    return poa_data


if __name__ == "__main__":
    latitude = 47.38770748541585
    longitude = 15.094127778561258
    tilt = 30
    azimuth = 149.716  # azimuth for SOUTH (pvlib = 180°, PVGIS = 0°)

    poa_data_2020 = get_pvgis_data(latitude, longitude, 2020, 2020, tilt, azimuth)

    # Save the data as a CSV file
    poa_data_2020.to_csv("poa_data_2020_Leoben_EVT_io.csv")

    print(poa_data_2020)

```

8.2 Pvsimmodule.py

```python
"""
This script calculates the power output of a photovoltaic system.

The script uses the PVLib library to calculate the power output of a photovoltaic system based on provided parameters.
It also processes and saves the results in an Excel file and generates plots for visualization.

Author: Michael Grün
Email: michaelgruen@hotmail.com
Version: 1.0
Date: 2024-10-13
"""

import os
from typing import Tuple

import matplotlib.pyplot as plt
import openpyxl
import pandas as pd
import pvlib
from openpyxl.utils import get_column_letter
from pvlib.location import Location
from pvlib.pvsystem import PVSystem
import poadata
from datetime import datetime, timedelta
import sys

# Define parameters
PARAMS = {
    'start': '2020-01-01 00:00',
    'end': '2020-12-31 23:00',
    'latitude': 47.3877,
    'longitude': 15.0941,
    'tilt': 30,
    'azimuth': 149.716,
    'celltype': 'polycrystalline',
    'pdc0': 240,
    'v_mp': 29.87,
    'i_mp': 8.04,
    'v_oc': 37.33,
    'i_sc': 8.78,
    'alpha_sc': 0.0041,
    'beta_voc': -0.114,
    'gamma_pdc': -0.405,
    'cells_in_series': 69,
    'temp_ref': 25
}

# Define helper functions
def get_date_attributes(date: datetime) -> Tuple[int, int, int]:
    """
    Return the weekday, hour, and month for a given date.

    Args:
        date (datetime): The date for which attributes are to be extracted.

    Returns:
        Tuple[int, int, int]: A tuple containing the weekday (1-7), hour (0-23), and month (1-12).
    """
    return date.weekday() + 1, date.hour, date.month


def execute_vba_actions(file_name: str) -> None:
    """
```

```
    Execute various VBA-like actions on an Excel file to format and adjust column widths.

    Args:
        file_name (str): The name of the Excel file to be processed.

    Returns:
        None
    """
    workbook = openpyxl.load_workbook(file_name)
    model_chain_results = workbook['Model Chain Results']
    poa_data = workbook['POA Data']

    # Autofit all columns in 'Model Chain Results' and 'POA Data'
    for worksheet in [model_chain_results, poa_data]:
        for column in worksheet.columns:
            max_length = max(len(str(cell.value)) for cell in column)
            adjusted_width = (max_length + 2) * 1.2
            column_letter = get_column_letter(column[0].column)
            worksheet.column_dimensions[column_letter].width = adjusted_width

    # Save the modified workbook
    workbook.save(file_name)


def calculate_power_output(params: dict) -> Tuple[pd.DataFrame, pd.DataFrame]:
    """
    Calculate the power output of the photovoltaic system based on provided parameters.

    Args:
        params (dict): A dictionary containing various parameters for the PV system and simulation.

    Returns:
        Tuple[pd.DataFrame, pd.DataFrame]: A tuple containing the AC power results and POA data for 2020.
    """
    location = Location(latitude=params['latitude'], longitude=params['longitude'],
                        tz='Europe/Vienna', altitude=547.6, name='EVT')

    poa_data_2020 = poadata.get_pvgis_data(params['latitude'], params['longitude'], 2020, 2020, params['tilt'], params['azimuth'])
    poa_data_2020.to_csv("poa_data_2020_Leoben_EVT_io.csv")
    poa_data_2020 = pd.read_csv('poa_data_2020_Leoben_EVT_io.csv', index_col=0)
    poa_data_2020.index = pd.date_range(start='2020-01-01 00:00', periods=len(poa_data_2020.index), freq="h")
    poa_data = poa_data_2020[params['start']:params['end']]

    solarpos = location.get_solarposition(times=pd.date_range(params['start'], end=params['end'], freq="h"))
    aoi = pvlib.irradiance.aoi(params['tilt'], params['azimuth'], solarpos.apparent_zenith, solarpos.azimuth)
    iam = pvlib.iam.ashrae(aoi)
    effective_irradiance = poa_data["poa_direct"] + iam + poa_data["poa_diffuse"]
    temp_cell = pvlib.temperature.faiman(poa_data["poa_global"], poa_data["temp_air"], poa_data["wind_speed"])

    I_L_ref, I_o_ref, R_s, R_sh_ref, a_ref, Adjust = pvlib.ivtools.sdm.fit_cec_sam(
        celltype=params['celltype'], v_mp=params['v_mp'], i_mp=params['i_mp'],
        v_oc=params['v_oc'], i_sc=params['i_sc'], alpha_sc=params['alpha_sc'],
        beta_voc=params['beta_voc'], gamma_pmp=params['gamma_pdc'], cells_in_series=params['cells_in_series']
    )

    cec_params = pvlib.pvsystem.calcparams_cec(effective_irradiance, temp_cell, params['alpha_sc'],
                                               a_ref, I_L_ref, I_o_ref, R_sh_ref, R_s, Adjust)

```

```
122.     mpp = pvlib.pvsystem.max_power_point(*cec_params, method="newton")

123.     system = PVSystem(modules_per_string=23, strings_per_inverter=3)
124.     dc_scaled = system.scale_voltage_current_power(mpp)
126.     cec_inverters = pvlib.pvsystem.retrieve_sam('CECInverter')
127.     inverter = cec_inverters['Advanced_Energy_Industries__AE_3TL_23_10_08__480V_']
128.     ac_results = pvlib.inverter.sandia(v_dc=dc_scaled.v_mp, p_dc=dc_scaled.p_mp,
inverter=inverter)
129.
130.     results_df = pd.concat([ac_results, dc_scaled.i_mp, dc_scaled.v_mp,
dc_scaled.p_mp, temp_cell], axis=1)
131.     results_df.columns = ['AC Power', 'DC scaled I_mp', 'DC scaled V_mp', 'DC scaled
P_mp', 'Cell Temperature']
132.
133.     with pd.ExcelWriter("results.xlsx") as writer:
134.         results_df.to_excel(writer, sheet_name='Model Chain Results')
135.         poa_data_2020.to_excel(writer, sheet_name='POA Data')
136.
137.     return ac_results, poa_data_2020
138.
139.
140. def plot_results(ac_results: pd.DataFrame) -> None:
141.     """
142.     Plot the AC power results over time and as a monthly sum.
143.
144.     Args:
145.         ac_results (pd.DataFrame): The DataFrame containing the AC power results.
146.
147.     Returns:
148.         None
149.     """
150.     ac_results.plot(figsize=(16, 9))
151.     plt.title("AC Power - PVSystem")
152.     plt.xlabel("Time")
153.     plt.ylabel("Energy Yield")
154.     plt.grid(True)
155.     plt.savefig("energy_yield_start_to_end.png")
156.     plt.show()
157.
158.     monthly_sum = ac_results.resample('M').sum()
159.     monthly_sum.plot(figsize=(16, 9))
160.     plt.title("AC Power - PVSystem (Monthly Sum)")
161.     plt.xlabel("Time")
162.     plt.ylabel("Energy Yield")
163.     plt.grid(True)
164.     plt.savefig("energy_yield_monthly_sum.png")
165.     plt.show()
166.
167.
168. def main() -> int:
169.     """
170.     Main function to run the power output calculation, plotting, and Excel file
processing.
171.
172.     Returns:
173.         int: The exit status code.
174.     """
175.     ac_results, poa_data_2020 = calculate_power_output(PARAMS)
176.     plot_results(ac_results)
177.     execute_vba_actions('results.xlsx')
178.     return 0
179.
180.
181. if __name__ == '__main__':
182.     sys.exit(main())
183.
```

8.3 Forecast.py

```
# -*- coding: utf-8 -*-
"""
Author: Michael Grün
Email: michaelgruen@hotmail.com
Description: This script loads a trained LSTM model and scalers to predict the power output of a photovoltaic system for the next day using features downloaded from an API.
Version: 1.0
Date: 2024-05-03
"""

import numpy as np
import pandas as pd
import joblib
from tensorflow.keras.models import load_model
import requests
import json
from datetime import datetime

# Load the saved model and scalers
model = load_model('final_model.h5', compile=False)
sc_x = joblib.load('scaler_x.pkl')
sc_y = joblib.load('scaler_y.pkl')

def fetch_weather_data():
    """
    Fetch weather forecast data from the API.

    Returns:
        dict: The fetched weather forecast data as a dictionary, or None if the request failed.
    """
    lat_lon = "47.38770748541585,15.094127778561258"
    current_time = datetime.now().replace(minute=0, second=0, microsecond=0)
    fc_start_time = current_time.strftime("%Y-%m-%dT%H:%M")
    fc_end_time = (current_time + pd.Timedelta(hours=24)).strftime("%Y-%m-%dT%H:%M")

    url = f"https://dataset.api.hub.geosphere.at/v1/timeseries/forecast/nwp-v1-1h-2500m?lat_lon={lat_lon}¶meters=grad¶meters=t2m¶meters=sundur_acc¶meters=v10m&start={fc_start_time}&end={fc_end_time}"
    response = requests.get(url)

    if response.status_code == 200:
        data = response.json()
        print("Weather data fetched successfully.")
        return data
    else:
        print(f"Failed to fetch data. Status code: {response.status_code}")
        return None

def process_weather_data(data):
    """
    Process the fetched weather forecast data and create a DataFrame.

    Args:
        data (dict): The fetched weather forecast data.

    Returns:
        pandas.DataFrame: A DataFrame containing the processed weather data with extracted features.
    """
    time_list = data['timestamps']
    parameters = data['features'][0]['properties']['parameters']

    # Calculate global irradiation difference and scale to J/(m²*s)
    global_irradiation = [(parameters['grad']['data'][i + 1] - val) / 3600 for i, val in enumerate(parameters['grad']['data'][:-1])]
```

```
61.
62.     # Add a placeholder value for the first global_irradiation to match the length of
other data
63.     global_irradiation.insert(0, 0)  # Adding 0 as a placeholder
64.
65.     # Create DataFrame ensuring all arrays have the same length
66.     new_data = pd.DataFrame({
67.         'timestamp': pd.to_datetime(time_list),
68.         'temp_air': parameters['t2m']['data'],
69.         'wind_speed': np.abs(parameters['v10m']['data']),
70.         'global_irradiation': global_irradiation
71.     })
72.
73.     # Extract hour and month features from the timestamp and apply circular encoding
74.     new_data['hour'] = new_data['timestamp'].dt.hour
75.     new_data['sin_hour'] = np.sin(2 * np.pi * new_data['hour'] / 24.0)
76.     new_data['cos_hour'] = np.cos(2 * np.pi * new_data['hour'] / 24.0)
77.     new_data['month'] = new_data['timestamp'].dt.month
78.     new_data['sin_month'] = np.sin(2 * np.pi * new_data['month'] / 12.0)
79.     new_data['cos_month'] = np.cos(2 * np.pi * new_data['month'] / 12.0)
80.
81.     return new_data
82.
83. def main():
84.     """
85.     Main function to fetch weather data, process it, and predict power output for the
next day.
86.     """
87.     weather_data = fetch_weather_data()
88.     if weather_data is not None:
89.         new_data = process_weather_data(weather_data)
90.
91.         # Define features to be used for prediction
92.         features = ['sin_hour', 'cos_hour', 'sin_month', 'cos_month', 'temp_air',
'wind_speed', 'global_irradiation']
93.
94.         # Iterate over 24 hours and predict power for each hour separately
95.         predicted_powers = []
96.         for i in range(24):
97.             # Get features for each hour
98.             X_new = new_data[features].iloc[i].values.reshape(1, -1)
99.
100.            # Scale the features using the loaded scaler
101.            X_new_scaled = sc_x.transform(X_new)
102.
103.            # Reshape the data to fit the model's expected input shape (1, n_steps,
n_features)
104.            X_new_scaled = X_new_scaled.reshape((1, 1, len(features)))
105.
106.            # Make predictions using the loaded model
107.            prediction = model.predict(X_new_scaled)
108.
109.            # Inverse transform the predicted value to get the actual power output
110.            predicted_power = sc_y.inverse_transform(prediction)
111.            predicted_powers.append(predicted_power[0][0])
112.
113.        # Print the results to the console
114.        print("Predicted AC Power for the next day (in W):")
115.        print("| {:<20} | {:<15} | {:<15} | {:<15} | {:<25} |".format('Time', 'Power
(W)', 'Temp (°C)', 'Wind Speed (m/s)', 'Global Irradiation (J/(m²*s))'))
116.        print("|" + "-" * 102 + "|")
117.        print("-" * 90)
118.        for i, power in enumerate(predicted_powers):
119.            future_time = (datetime.now() + pd.Timedelta(hours=i)).strftime("%Y-%m-%d
%H:%M")
120.            temp_air = new_data['temp_air'].iloc[i]
121.            wind_speed = new_data['wind_speed'].iloc[i]
122.            global_irradiation = new_data['global_irradiation'].iloc[i]
123.            print("|  {:<18} |  {:<15.2f} |  {:<15.2f} |  {:<15.2f} |  {:<25.2f}
|".format(future_time, power * 1000, temp_air, wind_speed, global_irradiation))
```

```
124.
125. if __name__ == "__main__":
126.     main()
127.
```

8.4 Lstm.py

```
  1. # -*- coding: utf-8 -*-
  2. """
  3. Author: Michael Grün
  4. Email: michaelgruen@hotmail.com
  5. Description: This script calculates the power output of a photovoltaic system
  6.              using the PVLib library and processes the results in an Excel file.
  7. Version: 1.0
  8. Date: 2024-05-03
  9. """
 10.
 11. import sys
 12. import pandas as pd
 13. import numpy as np
 14. from datetime import datetime
 15. import matplotlib.pyplot as plt
 16. from sklearn.preprocessing import MinMaxScaler
 17. from sklearn.metrics import mean_squared_error, mean_absolute_error
 18. import tensorflow as tf
 19. from tensorflow.keras.models import Sequential
 20. from tensorflow.keras.layers import Dense, LSTM, Bidirectional
 21. from tensorflow.keras.optimizers import Adam
 22. from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint
 23. from sklearn.metrics import roc_curve
 24. from tensorflow.keras.layers import Input
 25.
 26. def split_sequences(X, y, n_steps):
 27.     X_, y_ = list(), list()
 28.     for i in range(len(X) - n_steps + 1):
 29.         seq_x, seq_y = X[i:i + n_steps], y[i + n_steps - 1]
 30.         X_.append(seq_x)
 31.         y_.append(seq_y)
 32.     return np.array(X_), np.array(y_)
 33.
 34.
 35. def build_lstm_model(n_steps, n_features):
 36.     model = Sequential([
 37.         Input(shape=(n_steps, n_features)),
 38.         Bidirectional(LSTM(120, activation='relu', return_sequences=True)),
 39.         LSTM(120, activation='relu', dropout=0.3, return_sequences=True),
 40.         LSTM(120, activation='relu'),
 41.         Dense(1)
 42.     ])
 43.     model.compile(loss='mse', optimizer=Adam(learning_rate=0.0005), metrics=['mse',
'mae'])
 44.     return model
 45.
 46.
 47. def plot_predictions(data, test_indices, predictions, y_test_unsc):
 48.     pred_df = data.iloc[test_indices[:len(predictions)]][["timestamp",
"AC_Power"]].copy()
 49.     pred_df.loc[:, "y_hat"] = predictions
 50.     pred_df["LocalDt"] = pd.to_datetime(pred_df["timestamp"], format="%Y-%m-%d
%H:%M:%S")
 51.     pred_df["RealTime"] = pred_df["LocalDt"] - pd.Timedelta(hours=1)
 52.     pred_df = pred_df.sort_values(["RealTime"], ascending=True)
 53.     pred_df = pred_df.set_index(pd.to_datetime(pred_df.RealTime), drop=True)
 54.
 55.     plt.figure(figsize=(12, 5))
```

```
    plt.plot(pred_df.index, pred_df["y_hat"], color='blue', label="1-Day-Ahead PV Power Forecast (KW)")
    plt.plot(pred_df.index, pred_df["AC_Power"], color='red', label="Actual PV Power (KW)")
    plt.ylabel('PV Power (KW)', fontsize=12)
    plt.title('Actual PV Power vs. 1-Day-Ahead Forecast')
    plt.xlabel('Date')
    plt.xticks(rotation=45)
    plt.legend()
    plt.grid(True)
    plt.show()


def plot_roc_curve(y_test_unsc, predictions):
    threshold_values = np.linspace(min(y_test_unsc), max(y_test_unsc), 10)
    tpr_values, fpr_values = [], []

    for threshold in threshold_values:
        binary_predictions = (predictions > threshold).astype(int)
        fpr, tpr, _ = roc_curve(y_test_unsc > threshold, binary_predictions)
        tpr_values.append(tpr[1] if len(tpr) > 1 else np.nan)
        fpr_values.append(fpr[1] if len(fpr) > 1 else np.nan)

    plt.figure(figsize=(8, 6))
    plt.plot(fpr_values, tpr_values, color='blue', lw=2, label='ROC-like Curve')
    plt.plot([0, 1], [0, 1], color='gray', linestyle='--', lw=2, label='Random Guess')
    plt.xlabel('False Positive Rate (FPR)')
    plt.ylabel('True Positive Rate (TPR)')
    plt.title('ROC-like Curve for Regression')
    plt.legend()
    plt.grid(True)
    plt.show()


def main():
    data = pd.read_excel("results.xlsx", sheet_name='Model Chain Results', usecols='A,B,L,M,P')
    data.columns = ['timestamp', 'AC_Power', 'temp_air', 'wind_speed', 'global_irradiation']
    data['timestamp'] = pd.to_datetime(data['timestamp'])
    data['hour'] = data['timestamp'].dt.hour
    data['sin_hour'] = np.sin(2 * np.pi * data['hour'] / 24.0)
    data['cos_hour'] = np.cos(2 * np.pi * data['hour'] / 24.0)
    data['month'] = data['timestamp'].dt.month
    data['sin_month'] = np.sin(2 * np.pi * data['month'] / 12.0)
    data['cos_month'] = np.cos(2 * np.pi * data['month'] / 12.0)

    features = ['sin_hour', 'cos_hour', 'sin_month', 'cos_month', 'temp_air', 'wind_speed', 'global_irradiation']
    target = 'AC_Power'

    train_size = int(len(data) * 0.8)
    train_df = data.iloc[:train_size].copy()
    test_df = data.iloc[train_size:].copy()

    sc_x = MinMaxScaler().fit(train_df[features])
    sc_y = MinMaxScaler().fit(train_df[target].values.reshape(-1, 1))
    train_df[features] = sc_x.transform(train_df[features])
    train_df[target] = sc_y.transform(train_df[target].values.reshape(-1, 1))
    test_df[features] = sc_x.transform(test_df[features])
    test_df[target] = sc_y.transform(test_df[target].values.reshape(-1, 1))

    X = train_df[features].values
    y = train_df[target].values
    n_steps = 24
    X, y = split_sequences(X, y, n_steps)

    x_train, y_train = X, y
    X_test = test_df[features].values
```

```
120.     y_test = test_df[target].values
121.     X_test, y_test = split_sequences(X_test, y_test, n_steps)
122.     x_test, y_test = X_test, y_test
123.
124.     model = build_lstm_model(x_train.shape[1], x_train.shape[2])
125.     checkpoint = ModelCheckpoint('best_model.keras', monitor='val_loss',
save_best_only=True, mode='min')
126.     history = model.fit(x_train, y_train, validation_data=(x_test, y_test),
epochs=10, callbacks=[checkpoint, EarlyStopping(monitor='val_loss', patience=10,
restore_best_weights=True)], verbose=1)
127.
128.     model.load_weights('best_model.keras')
129.
130.     predictions = sc_y.inverse_transform(model.predict(x_test).reshape(-1, 1))
131.     y_test_unsc = sc_y.inverse_transform(y_test.reshape(-1, 1))
132.     print(f"MSE: {mean_squared_error(y_test_unsc, predictions):.4f}")
133.     print(f"MAE: {mean_absolute_error(y_test_unsc, predictions):.4f}")
134.     mape = np.mean(np.abs((y_test_unsc - predictions) / y_test_unsc)) * 100
135.     print(f"MAPE: {mape:.4f}%")
136.     print(f"Average value of y_test_unsc: {np.mean(y_test_unsc):.4f}")
137.     print(f"Number of epochs made: {len(history.epoch)}")
138.
139.     plot_predictions(data, test_df.index, predictions, y_test_unsc)
140.     plot_roc_curve(y_test_unsc, predictions)
141.
142.
143. if __name__ == "__main__":
144.     sys.exit(main())
145.
```

8.5 Excample: Retreiving forecast-data from the *“GeoSphere Austra”*-API with *“Endpoint-structure”*

Example of an *“Endpoint-structure*” for forecast-data with the following specifications:

- **Type** = timeseries – forecast
- **Resource** (chosen weather model = *Numeriacal Weather Prediction*) = nwp-v1-1h-2500m

o **Parameters** =

- grad (surface global radiation [J/m²])
- t2m (Temperture 2m above ground [°C])
- sundur_acc (sunshine duration accumulated [s])
- v10m (wind speed northern direction [m/s])

o **Time interval** =

- start = 2024-04-05T10:00
- end = 2024-04-06T09:00

➔ https://dataset.api.hub.geosphere.at/v1/timeseries/forecast/nwp-v1-1h-2500m?lat_lon=47.38770748541585,15.094127778561258&parameters=grad&parameters=t2m&parameters=sundur_acc&parameters=v10m&start=2024-04-05T10:00&end=2024-04-06T09:00