Prediction of PV Power Production with Neural ODEs on the base of Weather data

Lukas Schwab[1], Louis Emier[1], Paul Machauer[1], Michael Quarti[2], and Rainer Gasper[2]

[1] University of Applied Science Offenburg
{lschwab,lemier,pmachaue}@stud.hs-offenburg.de
[2] INES – University of Applied Science Offenburg
{michael.quarti,rainer.gasper}@hs-offenburg.de

Abstract. Predicting energy production from photovoltaics (PV) is crucial for efficient energy management. In order to apply different operating strategies, it is necessary to predict the expected amounts of PV energy. The operating strategies are typically optimized with regard to economic or technical goals or a combination of both. Within this work, we show a possibility to predict PV power production using local weather data and Neural Ordinary Differential Equations (NODE). Based on the measured values from the PV system and an associated weather station, the NODE is trained and validated with regard to PV production. The measurement data are collected from the PV system of the former Campus North of Offenburg University of Applied Sciences.

Keywords: PV Power Production, time series modeling, NODE

1 Introduction

Predicting energy production from photovoltaics (PV) is crucial for efficient energy management. In order to be able to apply different operating strategies, it is necessary to forecast the expected generation of PV power. Models for forecasting PV generation can be categorized by the forecast horizon like (1) short-term forecasting for minutes or an hour [1, 2], (2) middle-term forecasting for about a day [3, 4] and (3) long-term forecasting for a couple of days up to months [5, 6]. It is common to use neuronal networks for all of the forecasting horizon. Within this work, we show a possibility to predict PV power production using local weather data using Neural Ordinary Differential Equations (NODE) [7]. The authors of Ref. [8] modelled a lithium-ion battery to predict the behavior of a lithium-ion battery using NODE and grey-box modelling. Based on the measured values from the PV system and an associated weather station, our model is trained and validated with regard to PV production. The measured data is collected from the PV system of the former Campus North of Offenburg University of Applied Sciences from the years 2017-2021. The PV system consisted of 3 strings with a nominal output of 2.16 kWp each. The strings have a collector inclination of 30° and 40° and an orientation of 18° south.

The article will be organized as follows. In the Introduction we will give an overview of the state of the art for PV power prediction and the use of Neural ODEs for modeling of physical systems. In the second chapter we give a short introduction to Neural ODEs for the prediction of time series. After that, we present the data and will describe the data preparation steps. The data preparation includes the resampling to a half hourly base. We also applied Principal Component Analysis and Fourier Transform to the data. After the preparation of the data we will discuss the hyper parameter tuning for the

NODE. In the next step we present the training and validation results of the NODE for the PV power prediction. We will conclude the article with a conclusion and discussion.

2 Overview Neural Ordinary Differential Equations

We will use Neural Ordinary Differential Equations (NODEs) [7] for the prediction of the PV power production. NODEs can be understood as a Residual Neural Networks, given by

$$x\left(t_{k+1}\right) = hf\left(x\left(t_k\right), (t), \Theta\right) + x\left(t_k\right) \text{ where } h = \Delta t, \tag{1}$$

where the discretization $h = \Delta t$ goes to zero and the function changes from discrete to continuous

$$\frac{\mathrm{d}x}{\mathrm{d}t} = f\left(x\left(t\right), t, \Theta\right) \tag{2}$$

where the Neural Network $f\left(x\left(t\right), t, \Theta\right)$ with the parameters Θ represents the continuous change of the hidden state $x\left(t\right)$. Or in other words Residual Neural Networks are the Euler discretization of NODEs [9–11]. In contrast to traditional neural network architectures with discrete layers, NODEs model the continuous evolution of hidden states with ordinary differential equations. More concrete, the input features don't pass through a fixed number of layers, the hidden states evolve over a time interval by using a solver. The differential equations themselves are represented by Neural Networks and describe the rate of change of the hidden states over time. The Neural Network is then integrated with a solver to obtain the trajectory of the hidden states over the time. In the context of solving ordinary differential equations, the input features can be interpreted as initial values for the hidden states. For the training of the NODEs we are using the so-called discretize-then-optimize approach where we backpropagate through the internal operation of the ode solver to obtain the gradients [12]. Because of the mentioned properties, NODEs are very well suited for the modeling of time series. For the same reasons NODEs are also well suited for the modeling of physical systems that can be expressed as differential equations. The NODEs defined above solve an initial value problem and the number of inputs/features has to be the same as the number of outputs. In the prediction of the PV power, the system depends not only on the initial values but also on additional inputs, most of all the weather. Therefore, the number of inputs is greater than the number of outputs. We consider a so called forced nonlinear system [13]

$$\frac{\mathrm{d}x}{\mathrm{d}t} = f\left(x\left(t\right), u\left(t\right), t, \Theta\right) \tag{3}$$

where additional information about the weather, e.g., is cumulated in the input vector $u\left(t\right)$. Note that the solution of the nonlinear system does not depend only on the initial condition of the states $x\left(0\right)$ but also on the input vector $u\left(t\right)$. It is important to note that the forced nonlinear system with the additional input vector $u\left(t\right)$ can be solved and trained with the same methods as the initial value problem. It has to be taken care that the input vector $u\left(t\right)$ is an external function of time and is not computed by the ode solver (as the state vector x) and if the inputs $u\left(t\right)$ are given as data points at discrete time steps t_k instead as an explicit function of time, an appropriate interpolation method has to be chosen.

3 Data Preparation and Analysis

Before we can train and test the NODE, we have to do the post processing of the data. That includes the resampling to a half hourly and hourly base. We also applied Principal Component Analysis and Fourier Transform to the data.

3.1 Resampling

To process the data, it must first be converted to the equidistance format. To this end, a resampling procedure was devised for the purpose of incorporating hourly and half-hourly values.

In the case of hourly sampled data at 11 a.m., the sampled values are calculated by aggregating all values between 10:30 a.m. and 11:30 a.m., thereby avoiding any consideration of future values. In order to guarantee the inclusion of data within the resampled data set at all times, any absent values are incorporated through the utilisation of a forward fill procedure. This is only feasible due to the fact that the recorded values of the sensors are only stored when a change is detected; otherwise, there are no values at specific points in time. It is important that the forward fill does not use the final value

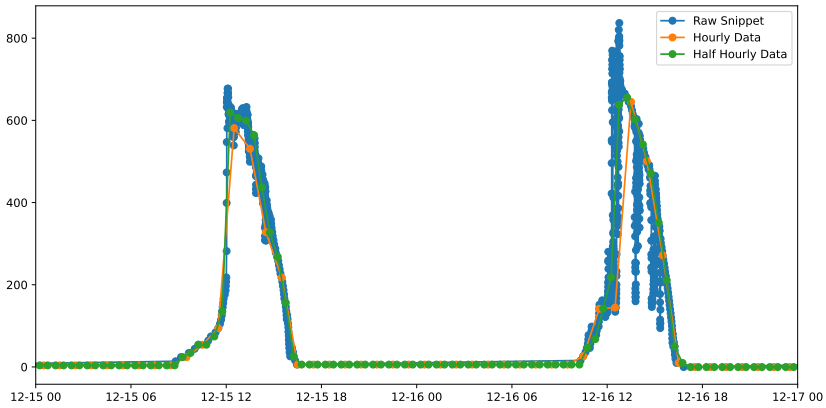

Fig. 1. Comparison between raw data, half-hourly re-sampled data and hourly re-sampled data for 48 hours, so that you can recognize the noon high twice.

from the resampling, but rather incorporates the last known value of the raw data into the missing values. This guarantees the precision of the resampled value over the course of the night. The resampled value is the average of the last hour with sunshine. As the measured values remain constant during the night, the period without sunshine is not considered. By using the last raw data value, the values over the night are accurate. Figure 1 shows an example of the resampled data for 32 hours.

3.2 Principal Component Analysis (PCA)

For further insight on the relevance of the features and comparison of the resampling rates (3.1) principal component analysis (PCA) is used. It is for data analysis purposes

only and is not used to transform the inputs of the NODE. PCA is suitable because the maximum variance of the data is retained in the principal components. The highest eigenvalues in table 1 thus show the relevance of the individual features in the data set from the point of view of variance. S represents the covariance matrix and u_1 the highest eigenvector [14].

$$u_1^T S u_1 = \lambda_1 \tag{4}$$

where λ_1 is the first eigenvalue. In the hourly data set, the three largest principal components explain 92.11 % of the variance and in the half-hourly 93.21 %. Since the three principal components are able to explain the data set with little loss in variance, the mean of u_1 - u_3 is used for each feature. Considering the variance to be obtained, the relevance and order of the features of the hourly and half hourly dataset is similar. So the hourly data set contains hardly any less information.

Table 1. Each column shows the values for the separate features. The values are calculated from the three largest eigenvectors like described in the section above. The table also serves the purpose of comparing the hourly and half hourly dataset.

Parameter	Hourly sampled data	Half hourly sampled data
Wind direction	0.508	0.515
Moist	0.493	0.486
Power	0.322	0.324
String 40° irradiation	0.247	0.270
Global solar irradiation	0.224	0.230
Pressure	0.198	0.105
Temperatur	0.139	0.075
Wind speed	0.093	0.033

Due to the similarity of the two datasets it is more reasonable to only focus on one dataset.

When observing the results of table 1 it is noticeable that wind direction, moist and power are considered more important than irradiation or temperature. This does not meet the intuitive expectations. One has to keep in mind, that the PCA components explain the variance in the data, and not a causal effect or dependency.

As soon as there are two similar features, usually one feature is left out. This was not done in the case of String 40° irradiation and global solar irradiation. Both features describe something similar.

In conclusion, PCA is used to recognise which features may be most important in order to test which data is necessary to collect in the future. However, reduced data sets would first have to be tested and compared with the actual application on the NODE. Additionally, it is beneficial to note that both resampling rates are good to use.

3.3 Fourier Transform

To further validate that the derived equidistant timeseries accurately represent the non-equidistant raw data, we generate spectrograms using Fourier transformations. The spectrograms are then utilized to gain insights into the prevalent frequencies within the

equidistant timeseries. We anticipate observing the daily solar cycle reflected in our derived timeseries of power production.

Different sampling frequencies are used for hourly and half-hourly timeseries to ensure good interpretability of frequencies across all spectograms. For the hourly timeseries we use a sampling frequency of 1/3600 Hz which equals 1 h and for half-hour timeseries we use 1/1800 Hz which equals half a hour.

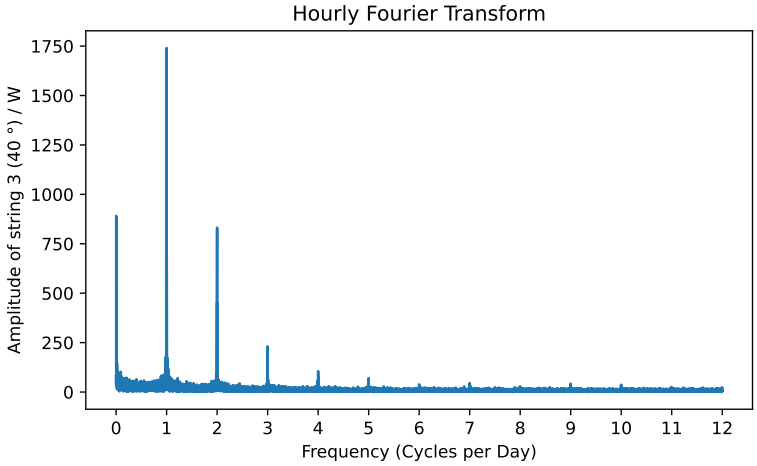

Fig. 2. Spectogram of the fourier transformation using the hourly sampled data and a sampling frequency of 1/3600 Hz.

Figure 2 shows the resulting spectrogram for the hourly timeseries. Because of our previously chosen sampling frequency we are able to interpret the frequency-axis in cycles per day. The highest amplitude appears at a frequency of 1, a strong indication for the presence of daily sun cycle within our dataset. Similar results are obtained when using the half-hourly timeseries.

3.4 Hyperparameter tuning

The time-consuming step of finding hyperparameters is automated with Optuna's Bayesian optimization [15] in combination with MLflow [16] for tracking. This allows for the collection of hyperparameters that result in a low loss within a reasonable time frame.

The optimization takes place in a separate validation loop. An average loss for a whole epoch is calculated from all weekly accumulated losses. Furthermore, the search space is defined by the following parameters: *tanh* and Rectified Linear Unit (ReLU) as activation functions for the neural network. ADAM, RMSprop and SGD for gradient descent. The number of neurons from 30 to 300. Learning rate in the of magnitude from 10^{-5} to 10^{-3}.

We used the optimization algorithms from PyTorch's torch.optim package [17].

In the majority of runs, Bayesian optimisation utilizes the rectified linear unit (ReLU) activation function and the adaptive moment estimation (ADAM) optimizer. For this reason NODE seems to work best with ReLU and ADAM in the current use case. The number of neurons and the learning rate don't differ significantly for the best runs.

Table 2. The columns 1 to 3 show the three best runs of the hyperparameter tuning under consideration of the weekly accumulated loss.

	1.	2.	3.
Activation function	ReLU	ReLU	ReLU
Optimizer	ADAM	ADAM	ADAM
Neurons	171	192	140
Learning rate	$8.77 \cdot 10^{-4}$	$6.68 \cdot 10^{-4}$	$5.84 \cdot 10^{-4}$
Weekly accumulated loss	94803	95228	95334

Notably, the row learning rate could be replaced by a clever choice of learning rate scheduler.

4 Results for the prediction of PV power production

The following results were achieved using our best hyperparameter configuration (see table 2). For our experiments we split both, the hourly and half-hourly, dataset into training (70 %) and validation (30 %) sets. A single training example contains one week of data. The neural network receives 6 weather features (wind direction, wind speed, temperature, global irradiation, moisture & pressure) as well as two positional encodings (time of day & current month) and the last preceding prediction. We observe promising convergence results with just a few weeks of training data, reaching diminishing convergence results within the first epoch of training (167 given example weeks).

Training is stopped after the first epoch to prevent overfitting on the dataset. The trained model achieves a mean error of 265.40 W and a median error of 30.24 W. These errors amount to 12.29 % and 1.40 % respectively when measured in relation to the 2.16 kWp of installed PV capacity over the validation set of 41 weeks. The large differences between these metrics can be explained by outliers which are caused by sudden cloud shading. It's important to mention that the focus of the project is not to predict the impact of individual clouds on the power production but rather larger timespans (e. g. > 3 h).

The results for predicting one week of PV power are shown in figure 3. As mentioned above, the model is not able to predict the peaks due to cloud shading. However, the overall prediction is in good agreement with the measurement.

A more informative (conclusive) way to measure the error might be to exclusively evaluate the day time phases where the model is not able to receive a "free lunch" by predicting a power output of 0 W. In this case the mean error is 401.37 W (18.58 %) and the median error 214.12 W (9.91 %).

4.1 Conclusion

The study has demonstrated that it is feasible to forecast PV power data using a neural ordinary differential equation (ODE). The model is capable of accurately forecasting the photovoltaic (PV) output of individual days.

There are minor discrepancies between the original and predicted values, particularly at the peak output observed at midday. Furthermore, the model does not anticipate abrupt declines in output, which may be attributed to cloud cover obstructing the photovoltaic panels. Furthermore, in exceptional circumstances, the model may occasionally

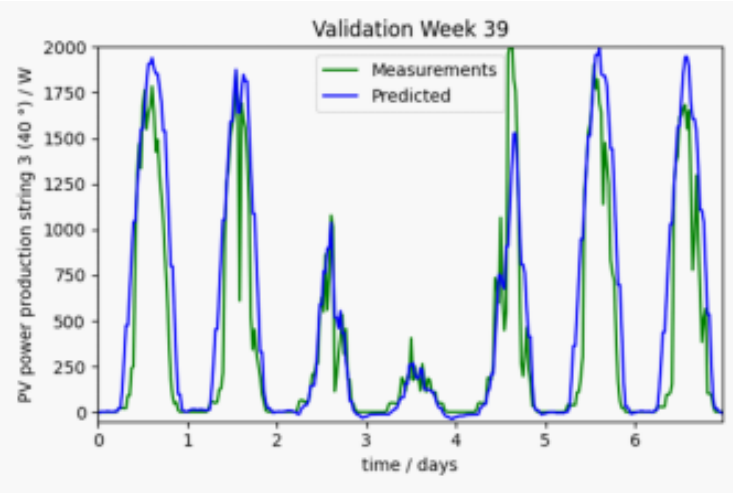

Fig. 3. Comparison of predicted and measured PV power production of string 3 with a collector inclination of 40°

predict negative PV power values for the night. It is not possible for negative power values to exist in the real world. Despite the aforementioned limitations, a generalised prediction of PV output for the day can be made.

A comparison of the prediction results with those of other methods and models was not conducted. This represents a potential avenue for further investigation. At present, the model is configured to receive only the most recent power value in order to facilitate the next prediction. One potential avenue for further investigation would be to modify the model in a way that incorporates a greater quantity of historical data into the next prediction step. The current model is only capable of making predictions one step into the future at a time. The incorporation of a multi-step output could facilitate the prediction of PV output at an extended temporal horizon.

4.2 Acknowedgement

We gratefully acknowledge financial support by the Carl-Zeiss Foundation through foundation professorship Mechatronic Systems Engineering.

References

1. Wolff, B., Kühnert, J., Lorenz, E., Kramer, O., Heinemann, D.: Comparing support vector regression for pv power forecasting to a physical modeling approach using measurement, numerical weather prediction, and cloud motion data. Solar Energy **135** (October 2016) 197–208
2. Mellit, A., Pavan, A.M., Benghanem, M.: Least squares support vector machine for short-term prediction of meteorological time series. Theoretical and Applied Climatology **111**(1–2) (May 2012) 297–307
3. Ekici, B.B.: A least squares support vector machine model for prediction of the next day solar insolation for effective use of pv systems. Measurement **50** (April 2014) 255–262
4. Lima, F.J., Martins, F.R., Pereira, E.B., Lorenz, E., Heinemann, D.: Forecast for surface solar irradiance at the brazilian northeastern region using nwp model and artificial neural networks. Renewable Energy **87** (March 2016) 807–818

5. Olatomiwa, L., Mekhilef, S., Shamshirband, S., Mohammadi, K., Petković, D., Sudheer, C.: A support vector machine–firefly algorithm-based model for global solar radiation prediction. Solar Energy **115** (May 2015) 632–644
6. Sobri, S., Koohi-Kamali, S., Rahim, N.A.: Solar photovoltaic generation forecasting methods: A review. Energy Conversion and Management **156** (January 2018) 459–497
7. Chen, R.T.Q., Rubanova, Y., Bettencourt, J., Duvenaud, D.: Neural ordinary differential equations (2019)
8. Brucker, J., Behmann, R., Bessler, W.G., Gasper, R.: Neural ordinary differential equations for grey-box modelling of lithium-ion batteries on the basis of an equivalent circuit model. Energies **15**(7) (April 2022) 2661
9. Lu, Y., Zhong, A., Li, Q., Dong, B.: Beyond finite layer neural networks: Bridging deep architectures and numerical differential equations (2020)
10. Haber, E., Ruthotto, L.: Stable architectures for deep neural networks. Inverse Problems **34**(1) (December 2017) 014004
11. Ruthotto, L., Haber, E.: Deep neural networks motivated by partial differential equations (2018)
12. Kidger, P.: On neural differential equations (2022)
13. Vidyasagar, M.: Nonlinear Systems Analysis. Society for Industrial and Applied Mathematics (January 2002)
14. M., C.: Pattern Recognition and Machine Learning. 1 edn. Information Science and Statistics. Springer, New York, NY (August 2006)
15. Akiba, T., Sano, S., Yanase, T., Ohta, T., Koyama, M.: Optuna: A next-generation hyperparameter optimization framework (2019)
16. Zaharia, M.A., Chen, A., Davidson, A., Ghodsi, A., Hong, S.A., Konwinski, A., Murching, S., Nykodym, T., Ogilvie, P., Parkhe, M., Xie, F., Zumar, C.: Accelerating the machine learning lifecycle with mlflow. IEEE Data Eng. Bull. **41** (2018) 39–45
17. Ansel, J., Yang, E., He, H., Gimelshein, N., Jain, A., Voznesensky, M., Bao, B., Bell, P., Berard, D., Burovski, E., Chauhan, G., Chourdia, A., Constable, W., Desmaison, A., DeVito, Z., Ellison, E., Feng, W., Gong, J., Gschwind, M., Hirsh, B., Huang, S., Kalambarkar, K., Kirsch, L., Lazos, M., Lezcano, M., Liang, Y., Liang, J., Lu, Y., Luk, C., Maher, B., Pan, Y., Puhrsch, C., Reso, M., Saroufim, M., Siraichi, M.Y., Suk, H., Suo, M., Tillet, P., Wang, E., Wang, X., Wen, W., Zhang, S., Zhao, X., Zhou, K., Zou, R., Mathews, A., Chanan, G., Wu, P., Chintala, S.: PyTorch 2: Faster Machine Learning Through Dynamic Python Bytecode Transformation and Graph Compilation. In: 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2 (ASPLOS '24), ACM (April 2024)