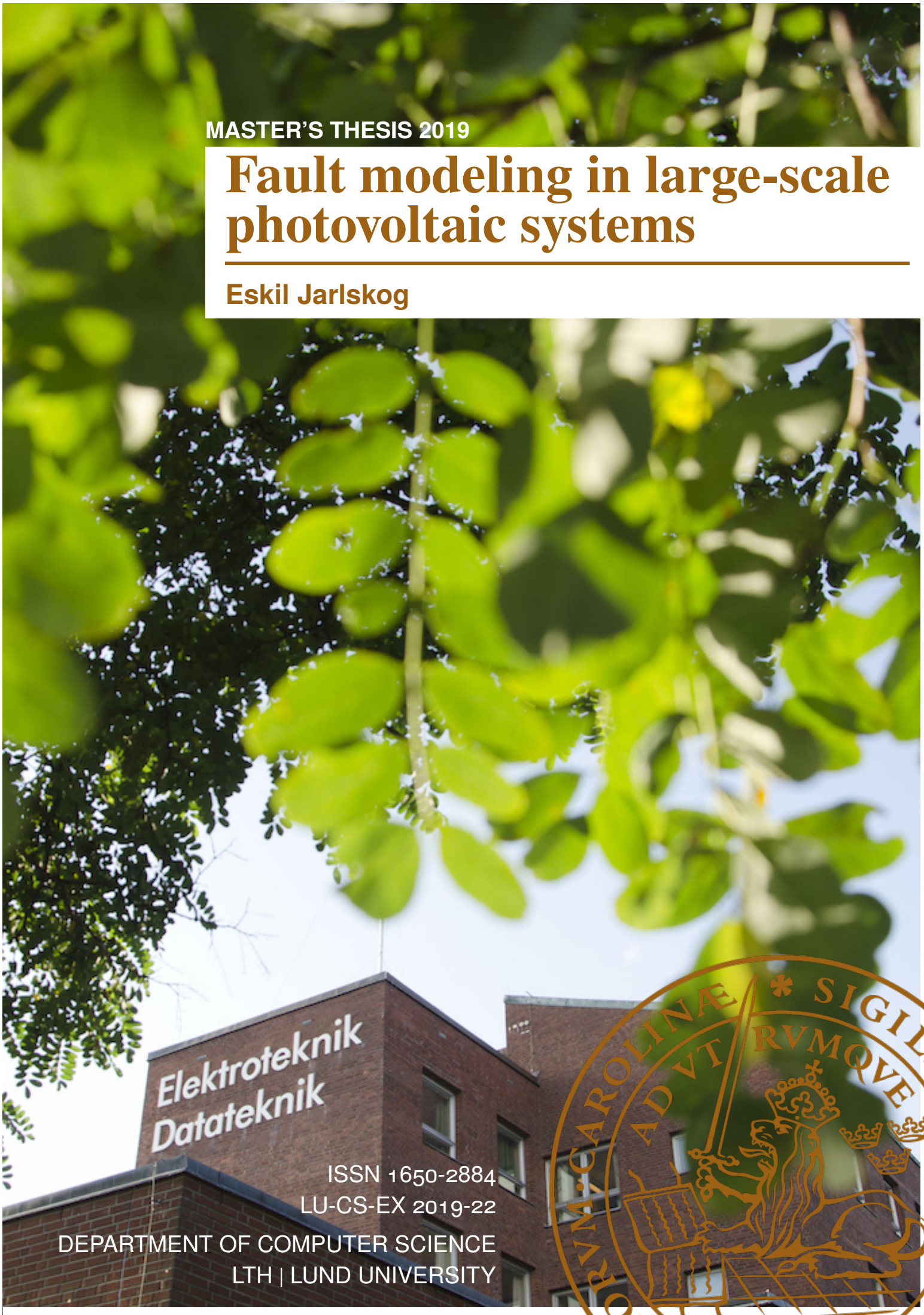

MASTER'S THESIS 2019

Fault modeling in large-scale photovoltaic systems

Eskil Jarlskog

ISSN 1650-2884
LU-CS-EX 2019-22

DEPARTMENT OF COMPUTER SCIENCE
LTH | LUND UNIVERSITY

EXAMENSARBETE
Datavetenskap

LU-CS-EX: 2019-22

Fault modeling in large-scale photovoltaic systems

Eskil Jarlskog

Fault modeling in large-scale photovoltaic systems

Eskil Jarlskog
tpi14eja@student.lu.se

August 23, 2019

Master's thesis work carried out at Nispera AG.

Supervisors: Pierre Nugues, pierre.nugues@cs.lth.se
Gianmarco Pizza, gianmarco.pizza@nispera.com

Examiner: Flavius Gruian, flavius.gruian@cs.lth.se

Abstract

Photovoltaic (PV) power systems are designed to supply power obtained through the photovoltaic effect. Large-scale PV systems consist of PV strings, which are serially connected PV modules, which produces direct current. After transformation to alternating current, the electricity is supplied to the grid. The objective of this work is to understand how different faults in PV systems affect the power output and to find, on a daily basis, where in the PV system the faults occur. The focus is on the faults that can be avoided and solved with proper maintenance.

More precisely, we examined three categories of faults, namely: availability, trackers, and soiling. We model these faults individually, on string level, to find how each string is affected by each fault. We implemented the models using neural networks that we trained on large-scale data. We optimize and evaluate the models and we describe the results we obtain. These results are promising and we discuss methods to improve them further.

Keywords: Photovoltaic systems, fault detection, solar trackers, neural networks, solar energy

Acknowledgements

We would like to thank Nispera AG for providing us with all the necessary equipment and data to perform this work.

We would also like to thank our supervisor, Pierre Nugues, for advising us during the work.

Lastly we would like to thank our examiner, Flavius Gruain, for providing us with useful comments and suggestions of how to improve the paper.

Contents

Chapter 1
Introduction

Renewable energy is nowadays a hot topic since it works against global warming and climate changes. When it comes to solar power, one does not simply build a solar park without maintaining it because that would be less economically beneficial. Usually in large-scale solar parks there are a lot of sensors installed sending data received by different components in the park. An important part of maintenance of solar parks, when having access to sensor data, is to analyse that data. The analysis can enable one to understand what one should do in order to perform optimal maintenance, both by looking at historical data and by using models to predict if faults will occur in the future.

This work will focus on analysing data from solar parks. We will in this chapter give an introduction to solar energy and maintenance, and explain terminology that is essential to know in order to understand the work. We would like this project to contribute to the knowledge about maintenance of solar parks, but also to more efficient usage of solar power through data analysis.

1.1 Background

To understand the content of this work, we describe the specific terminology that we use in this work. A summary of this terminology is provided in this section.

The photovoltaic (PV) effect is the creation of voltage and electric current upon the exposure of light. Solar cells, a component that can transform light into electricity, are the building blocks when it comes to exploiting the photovoltaic effect. The effect is a source of renewable energy and therefore it is seen as a energy source for the future and its usage is growing more and more popular.

The total power supplied by solar energy through photovoltaics in the world has been increasing exponentially since 1992 as can be seen to the left in Figure 1.1. The reason for this growth has to do with the decreasing prices of PV cells that decreased rapidly during since 1977 and especially during the last years, see the right graph in Figure 1.1.

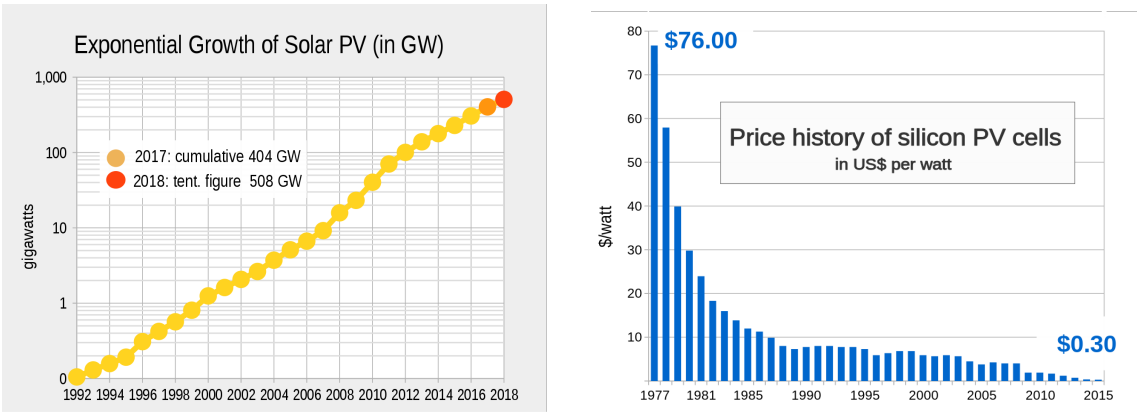

Figure 1.1: Left: Worldwide growth of photovoltaics on a semi-log plot since 1992. Right: Price per watt history for conventional (c-Si) solar cells since 1977. (Source of images: Wikipedia, Photovoltaics)

As the usage of photovoltaics has been continuing to grow rapidly during the last years it is projected that it will have significant role in the future. During 2018, 2.58% of the total energy consumption in the world was generated through PV (Masson et al., 2019). This paper states that China, the country with the highest installed PV capacity (176 GW), obtained 3.3% of its total power production from PV in 2018. Germany, which has a combined PV capacity of 45 GW, is the country with the highest installed capacity in Europe and 7.9% of their power is generated by PV.

1.1.1 Photovoltaic Power Station

A photovoltaic system uses the PV effect to supply usable power. An overview of a photovoltaic system is shown Figure 1.2. A solar panel consists of modules that are serially or parallely connected. In the case of serially connected modules, the solar panel can be referred to as a string. Moreover, a solar array consists of either one or more panels.

A photovoltaic power station, also referred to as a solar park, is a large-scale PV system. In the power stations, measurements are usually made on string level, inverter level and park level. The strings produce direct current (DC) which is transformed into alternating current (AC) by the inverter. This transformation is necessary in order to supply electricity to the grid.

To maximize the power output from the strings, the solar arrays can be tilted so that they become perpendicular to the rays of the sun. The system that handles this angling is called a *tracker system*. The main component in the tracker system responsible for the mechanical angling is called a *tracker*. Most trackers used in PV power stations are either moving the PV arrays along one axis, called one-axis trackers, or two axes, called two-axes trackers. In Figure 1.3, operation of a one-axis tracker is visible.

The power per area unit from sun light is called solar irradiance (often W/m^2). Solar irradiance is the biggest factor when it comes to how much energy a solar cell produces. The temperature also plays a part in the power output of a solar cell. Equation 1.1 specifies

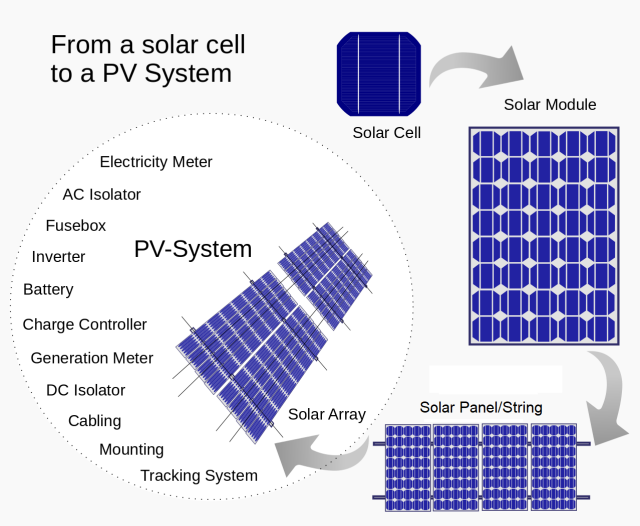

Figure 1.2: A visualization of the possible components and an overview of a photovoltaic system. (Source of image: Wikipedia, Photovoltaic system)

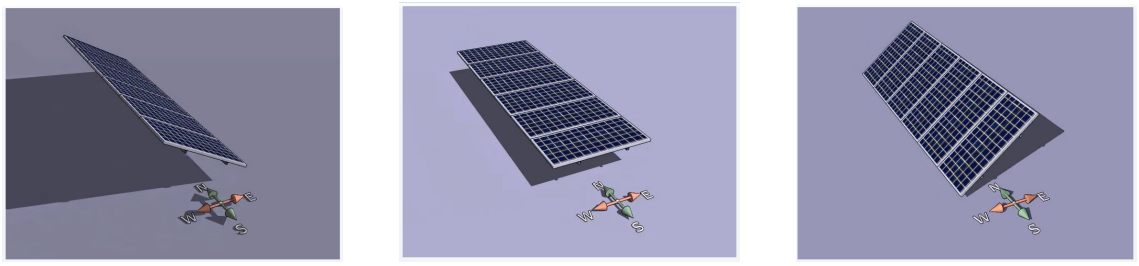

Figure 1.3: A visualization showing how the trackers operate during a day. The PV array has different angles throughout the day depending on where the sun is, from morning (left), noon (middle) and evening (right). (Source of images: Wikipedia, Inseguitore solare (Solar tracker), Language: Italian)

the power output function of a solar array (Chen et al., 2012):

$$P = \eta A I (1 - \frac{1}{200}(T - 25)), \tag{1.1}$$

where P denotes the power output (W), η denotes the efficiency of the cells in the PV array, A, denotes the area of the PV array (m^2), I denotes the absorbed irradiance (W/m^2), and T is the temperature of the cells in the PV array (°C). The tracking system increases the absorbed irradiance and therefore the power output.

1.1.2 Maintenance of a PV power station

Operation and maintenance of PV systems has become a standalone segment within the solar industry (SolarPower Europe, 2018). Specialized companies different from the owners, often operate the PV power stations. Such companies developed competences in running

stations that ensure a better production and monitoring of components failures. Contracts with these companies usually include guarantees about production and response time to component failures. This ensures that the plant owners will have a guaranteed profit from the solar park while eliminating risks. A long description of operation and maintenance, and contractual best practices guidelines can be found in SolarPower Europe (2018).

In order to improve operation and maintenance results, the specialized companies should somehow analyze the data retrieved from the sensors of the solar park. The data analysis can either be done by the company itself or they could hire another company to do this analysis for them. The analysis of data should be automatic since the amount of data often is too big to analyse manually. The automatic data analysis enables detection and prediction of faults from data.

In this work, we will focus on detecting failures or issues of large-scale PV power stations that can be avoided or fixed with proper maintenance. We will develop a model that will give an indication on what to do to improve the power production. The goal is that the models developed in this work will make maintenance of PV systems easier. The idea is that this will be done by providing the maintenance team with which issues that are present on which strings in the PV power station. These issues we will consider will be introduced in the next section.

1.1.3 Maintenance related issues

The three major issues that we have classified as being fixable with good maintenance are availability issues, tracker issues and soiling. We chose these issues, together with Nispera AG, because they are thought to be the most important ones to consider. This since they are believed to occur frequently and contribute to significant losses in large-scale PV systems. There are also other issues such as aging of components, however in this project we decided to focus on these three. We explain these issues by means of how they affect the power output and what could have caused them.

- *Availability issue.* A string is defined as being unavailable if it has a zero production when it should produce. That is, the string is said to have an availability issue if it does not produce at all when the sun is shining. This can be due to many reasons such a disconnected or damaged cables. This unavailability of strings is the most severe issue since there will be no production until the issue has been resolved. Since sensors measuring production can be poorly calibrated we will later define a threshold for how much a string should produce in order to be defined as available.

- *Tracker issue.* When a tracker is not working, when its solar arrays are not angled properly, we say that there is a *tracker issue*. Since most of the absorbed irradiance comes from direct sunlight a tracker system can drastically increase the production. In fact, the production loss of direct sunlight due to having the wrong angle is $1 - cos\varphi$, where φ is the difference between the optimal angle and the actual angle (Wikipedia, solar tracker). The gain having a tracker-system as in previous work been estimated to be 13-25 % on average depending on the type of tracking system (Dhanabal et al., 2013).

- *Soiling* is the effect of dust accumulation of the surface of the PV panels, see Figure 1.4 for an example. How soiling affect the production accumulates over time but can

be avoided by cleaning the panels, however also heavy rains have this effect (Kimber et al., 2006). Soiling on the panels prevents some of the irradiance to be absorbed by the PV cells which makes the production lower. Observe that a theoretical worst case of soiling is also an availability issue, which is the case if there is so much dust on the surface of a solar panel so that no light reaches the solar panel. This case should only occur in theory though and therefore the availability issue and soiling should not overlap in practice. In most large-scale photovoltaic systems two pyranometers, measuring the irradiance, are available. The pyranometers are located next to each other and one of them is cleaned regularly (typically once a week) to capture the true irradiance, not affected by soiling. The other pyranometer is cleaned whenever a cleaning of the strings in the solar park is performed to capture the irradiance absorbed by the strings.

Figure 1.4: A visualization showing clean panels (top) and soiled panels (bottom). (Source of image: `http://pannelplus.divegroup.it/zh/press_scheda.aspx?press=9`)

1.2 Problem Statement

Formulating the problem statement in this work requires a definition of *producible*. The producible is defined as the power that a string or set of strings would produce with a theoretically optimal maintenance. Note this means that the producible is the power that a set of strings would produce if they all were completely clean and if all components of the strings were working properly during the designated time period.

The objective of this work is to find a model that is able to calculate the producible of a set of strings. If we find the producible we would also want to be able to identify the reasons why the strings (possibly) are not producing at this level. In other words, we have two objectives, but the first one needs to be reached before being able to satisfy the other one. If the objective, by any reason, cannot be fulfilled a detailed explanation of why it could not be fulfilled will be provided and suggestions of what can be done in order to satisfy the objective will be presented.

1.3 Related Work

We spent some time on studying previous works in the subject to get ideas and inspiration of how to design our work. We read articles to learn about the different issues and how they were modeled in the past. In this section we will summarize the related work to give an overview of what has previously been done in this subject.

In previous work Dhanabal et al. (2013) found that a PV array with a single-axis tracker system will produce on average around 13% more during a day compared to a PV array that has a fixed mount. In this same paper they stated that a dual-axis tracker system will increase the power output by approximately 25%. This paper will be helpful for us since it gives us information of how much loss a solar panel on stuck tracker at least has.

Al Dahoud et al. (2015) detected tracker issues by capturing pictures of PV arrays and then applying image recognition algorithms. However, to the best of our knowledge, no one has yet modeled tracker issues by using the power output of strings which our approach will do. For us, it is important to know about other possible approaches if our approach does not work.

A significant amount of work has been done in modeling soiling. The previous works were all modeling soiling as a function of several factors only depending of weather conditions. Our work will instead consider the power output of the strings of the solar park to estimate the loss due to soiling. Modeling of soiling was done with neural networks in Laarabi et al. (2019), but also more physical approaches have been tested. An approach to calculate the level of soiling by simulating the accumulation of dust particles was done in Barth et al. (2017).

The studied articles report work where only a few solar panels were used when evaluating the models describing the effect of soiling. In our work we will try to find a model that works well for large-scale PV systems however. Therefore it cannot be assumed that the previous methods will work work well for large-scale PV systems, but it is still important for us to know about alternatives if our initial approach does not work. The soiling effect affects PV systems differently depending on the environment where the system is located (Cordero et al., 2018). This suggests that unique parameters should be determined for every unique site. In our work we will only look at one specific solar park however, so the parameters presented later in this paper might not work as well for other solar parks.

It turns out that PV modules can be self cleaned in the case of heavy rain (Kimber et al., 2006). This does also suggests that the soiling issue should be considered differently depending on the surrounding environment. For instance, if there is going to be a heavy rainfall within a week it could be better to not spend money on cleaning and wait for the heavy rain to clean the panels instead. This does not help us in our current work but could

be worth taking into account for users of our soiling model when determining a date for cleaning the solar park.

The approach we will use is based on using the power output on string level which has not been done before, to the best of our knowledge. Also in the majority of the articles the models were made for small PV systems (less than 10 solar panels). Therefore, we will contribute to growing a new branch of the subject, by focusing on large-scale PV systems (thousands of solar panels) and modeling the issues by using the production.

1.4 Theory

In this section we will describe some of the theory behind the tools used in the implementation step of this work. The content in this section is not directly related to the work but is necessary to know when developing a model that uses neural networks. We believe that it is important to show the possession of this knowledge, it gives credibility to the work. However a reader can skip this section and still understand the content of the work which is recommended for readers that are not eager to learn the basics of neural networks. The knowledge presented in this section is taken from Chollet (2017).

1.4.1 Neural Networks and Keras

A neural network consists of nodes and directed edges. These are inspired by the neurons and synapses of the human brain. The nodes are arranged in what is called different layers, see Figure 1.5. Normally the nodes in a layer have incoming edges from the nodes of the previous layer and outgoing edges to the nodes of the next layer in the hierarchy, neural networks with this hierarchy are in literature called feed-forward networks.

The first layer in the hierarchy has no incoming edges, and is called input layer. It is into these nodes that the values of the input variables are inserted. The last layer in the hierarchy, containing the output node(s), is called output layer. All the layers between the input layer and the output layer are called hidden layers. A neural network where all nodes in every layer is connected to all nodes in the next layer is said to be fully connected.

In each node, the sum of the incoming values is always calculated and then this value is applied to a function, called activation function, and then sent as an outgoing value. This for a node is summarized in Equation 1.2.

$$y = \varphi(\sum_i x_i), \tag{1.2}$$

where y is the output of the node, x_i are the different inputs to the node and φ is the activation function. Along every edge there is a weight that the outgoing value is multiplied with before coming to the next node. If we take the weights into account, the output of the j-th node in a layer, z_j, as a function of the outputs of the previous layer, y_i, is stated in Equation 1.3.

$$z_j = \varphi(\sum_i w_{ij} y_i), \tag{1.3}$$

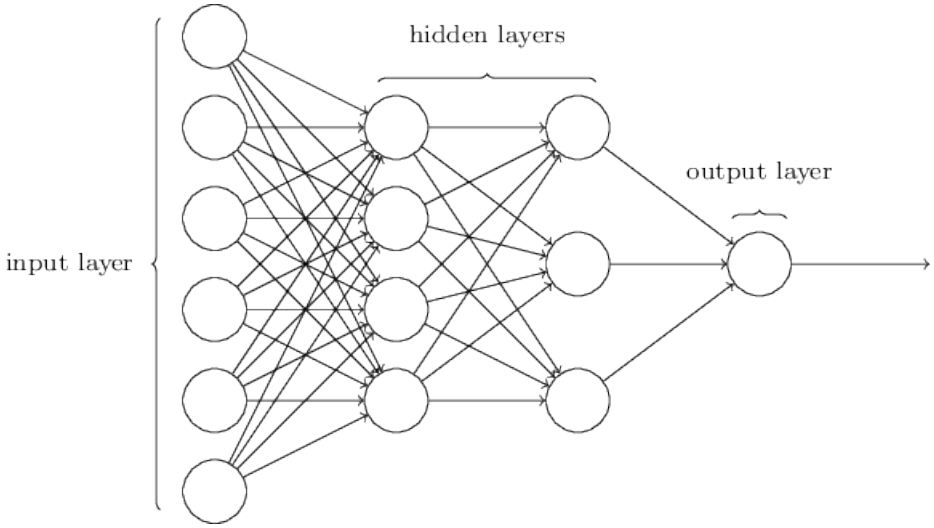

Figure 1.5: A visualization of the architecture of an example neural network that is fully connected. Observe that there is not necessarily only one output node in the output layer, even though it is the case in this example. (Source of image: `http://neuralnetworksanddeeplearning.com`)

where w_{ij} is the weight along the edge between node i in the previous layer and node j in the current layer. The sum goes over all the nodes in the previous layer and φ is the activation function.

Some of the most common activation functions are stated below and visualized in Figure 1.6.

- linear - $\phi(x) = x$
- relu - $\phi(x) = max(0, x)$
- sigmoid - $\phi(x) = 1/(1 + e^{-x})$
- tanh - $\phi(x) = \tanh(x) = 2/(1 + e^{-x}) - 1$

In cases of regression, where the output of the network should be able to range from large negative numbers to large positive numbers, the activation function of the output layer should be linear. However the activation functions of the other layers should not be linear. In fact if all activation functions of a neural network are linear then it can be replaced with a neural network with no hidden layers independently of how many hidden layers it has. This can proven by induction using Equation 1.3.

Activation functions, the number of layers and the number of hidden layers are examples of what is called hyperparameters. These must be chosen before doing the training of the neural network. What is optimized during training is only the value of the weights edges in the neural network.

An advantage in using neural networks is that by training them one can find a function that well approximates an unknown function, without any knowledge of the function that is being approximated. What is needed in order to efficiently use neural networks is to

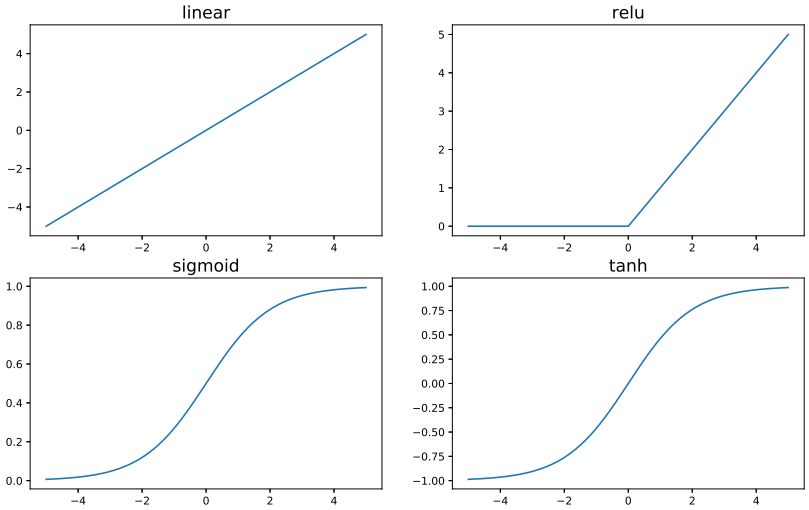

Figure 1.6: A visualization of some of the typical activation functions used in a neural network. Make sure to notice the different scaling in the four plots.

identify which variables that the output variable are dependent on. The observations of these variables can be used as input to the neural network and are then thereof called input variables. Training data must contain the observations of the input variables along with the corresponding observations of the output variable(s) so that a function between the two can be found by finding proper weights.

The library Keras in python which uses Tensorflow as a back-end is an efficient way of using neural networks. With Keras we can choose our neural network architecture freely and different training algorithms are implemented. With Keras we do not need to know the exact theory of how the algorithms work. In order to use the library efficiently, however, it is beneficial to know which architecture and hyperparameters that are suitable for which problems. We will summarize some of the most important algorithms related to neural networks in the following sections.

1.4.2 Backpropagation

We determine the neural network parameters through a technique called backpropagation. We will here describe how it works. During this section we will assume that we are analyzing a multilayer perceptron (MLP), which is a fully connected neural network with only one output node. We will also assume that the number of nodes in the input layer is m.

First we need to define a training set $D = \{\mathbf{x}_n, y_n\}_{n=1,\dots,N}$, where $\mathbf{x}_n \in \mathbb{R}^m$ is the input and $y_n \in \mathbb{R}$ is the target output of the n-th pattern in the training set. We also define an error function, $E(\omega)$, where the error is a function of all the weights, ω, in the neural network.

The idea now is to investigate how the error function changes as a function of the weights. We start by looking at the weights between the last hidden layer and output layer, denote by ω_j the weight of the edge connecting the j-th node in the last hidden layer and the node in the output layer. Denote by $\hat{y}_n$ the current output of network when using the n-th input pattern. Since we have N patterns in our training set we can write error function as

an average of the individual errors. We can then calculate the derivative the error function with respect to any weight ω_j, see Equation 1.4.

$$\frac{\partial E}{\partial \omega_j} = \frac{1}{2N}\sum_{n=1}^{N}\frac{\partial E_n}{\partial \omega_j} = \frac{1}{2N}\sum_{n=1}^{N}\frac{\partial E_n}{\partial \hat{y}_n}\frac{\partial \hat{y}_n}{\partial \omega_j}, \tag{1.4}$$

But the derivative $\partial E_n/\partial \hat{y}_n$ is known for a given error function, and also we have that

$$\frac{\partial \hat{y}_n}{\partial \omega_j} = \frac{\partial \varphi(\sum_{j'} \omega_{j'} h_{nj'})}{\partial \omega_j} = \varphi'(\sum_{j'} \omega_{j'} h_{nj'}) h_{nj}, \tag{1.5}$$

where the sums over j' goes over the last hidden layer, $h_{nj'}$ is the output of the j'-th node in the last hidden layer for the n-th pattern and φ is the activation function of the last layer. For a given network all the values in Equation 1.5 can be calculated by inserting the n-th input pattern, so the derivative in Equation 1.5 can be calculated. We denote by δ_n the expression in Equation 1.6, and when implementing the algorithm we would store these values,

$$\delta_n = -\frac{1}{2}\frac{\partial E_n}{\partial \hat{y}_n}\varphi'(\sum_{j'} \omega_{j'} h_{nj'}). \tag{1.6}$$

We can calculate the derivatives for the weights between the two last hidden layers, $\tilde{\omega}_{jk}$, in a similar way, see Equation 1.7.

$$\frac{\partial E}{\partial \tilde{\omega}_{jk}} = \frac{1}{2N}\sum_{n}\sum_{j'}\frac{\partial E_n}{\partial \hat{y}_n}\frac{\partial \hat{y}_n}{\partial h_{nj'}}\frac{\partial h_{nj'}}{\partial \tilde{\omega}_{jk}} = -\frac{1}{N}\sum_{n}\delta_n \omega_j \varphi_h'(\sum_{k'} \tilde{\omega}_{jk'} h_{nk'}) h_{nk}, \tag{1.7}$$

where h_{nk} is the output from the k-th node in the second last hidden layer, the sum over k' goes through all the nodes of the second last hidden layer and the function φ_h is the activation function of the last hidden layer. Since δ_n was calculated in the previous step we do not need to recalculate it.

In the algorithm we would continue by calculating the values specified in Equation 1.8 and use these when calculating how the weights between the next pair of layers affect the loss function.

$$\delta_{nj} = \varphi_h'(\sum_{k'} \tilde{\omega}_{jk'} h_{nk'}) \delta_n \omega_j \tag{1.8}$$

We have derived the derivatives of the loss function with respect to the weights on the edges between the last two pairs of layers. However, this derivation can be done inductively for an arbitrary amount of layers.

The algorithm is called backpropagation since we are calculating the deltas, δ, for each layer starting at the last one. We then "back-propagate" the deltas through the layers to calculate the derivatives. This algorithm is the foundation in many training algorithms. The weight updates are based on optimization algorithm, where gradient descent is one of the most common ones. The gradient descent optimization algorithm will be explained in the next section.

1.4.3 Gradient Descent

The training algorithm gradient descent is alone one of the most basic training algorithms. However, it is the foundation of many currently used training algorithms. It is an optimization algorithm that is following the direction of the negative gradient in order to find a minimum.

As for all optimization algorithms, we would like to find a minimum of a function. In this case we want to find the minimum of the loss function, E, defined in the previous section. To do this we will find the gradient and move in its opposite direction in the domain of E, which is the weight space. The step length we take in the negative gradient direction is in the application of neural networks called learning rate and is often denoted by η. The weight update of an iteration of gradient descent with a learning rate η is given in Equation 1.9.

$$\omega_{n+1} = \omega_n - \eta\nabla(\omega_n), \tag{1.9}$$

where ω_n is the vector containing the current weights, after the n-th iteration of the gradient descent algorithm. Note that the backpropagation is not a training algorithm itself but it enables an efficient implementation of training algorithms, like gradient descent, since it efficiently finds the gradient.

An alternative version of gradient descent is stochastic gradient descent which, instead using of all patterns in every iteration, uses only one random pattern in each iteration. This means that the algorithm is less computationally heavy and therefore usually converges faster than the normal gradient descent. It will require more iterations than the gradient descent though, but for huge datasets the stochastic version is preferable over the non-stochastic one. There is also a version of stochastic gradient descent were we, instead of using one random pattern, use a random subset of them in each iteration. This version is called mini-batch gradient descent.

1.4.4 Universal Approximation Theorem

Neural networks are widely used because it has a capability of extending a set of patterns to a general function. An important result in the theory of neural network is that there is always a neural network that can approximate any continuous function on a compact subset of $\mathbb{R}^n$ arbitrarily well.

Universal approximation theorem. Assume that we have a continuous function $f : C \to \mathbb{R}$ on a compact subset $C \subset \mathbb{R}^n$. Then there exists a neural network with one hidden layer with output function $y : \mathbb{R}^n \to \mathbb{R}$ such that

$$\sup_{\mathbf{x}\in C} |f(\mathbf{x}) - y(\mathbf{x})| = |f(\mathbf{x}) - \sum_{i=1}^{N} \alpha_i \varphi(\mathbf{w_i^T x} + b_i)| < \varepsilon \tag{1.10}$$

for all $\varepsilon > 0$. In other words there exist constants $N \in \mathbb{Z}_+$, $\mathbf{w_i} \in \mathbb{R}^n$ and $\alpha_i, b_i \in \mathbb{R}$ such that Equation 1.10 holds with these conditions.

1.5 Contributions

We have contributed to the area of monitoring of photovoltaic power stations which is an important part of making them more efficient in the future. Even though our algorithm can be significantly improved by further development this could be a basis for future automatic monitoring.

We hope that the approach we have presented will continue to be developed and successfully used to find issues in large-scale PV systems. In the big picture, we hope to contribute to the growth of solar energy and that it will make an impact to counteract global warming.

Chapter 2
Approach

The idea of the approach is to identify the different possible issues that could have a negative impact on the production for a set of strings. When these issues have been identified, the idea is to try to model them individually to find which strings are affected by the different issues. When modelling the issues the weather should, to the greatest extend possible, not be used since these are usually only measured at park level. In contrary to previous work, we will be developing models for large-scale solar parks, that contains thousands of solar modules. This implies that these parks will be large and therefore clouds will affect the production locally. Hence, we will use the production data obtained at string level in order to adjust for local phenomenon in the parks such as passing clouds. We want to use the production at string level since the weather data is usually only measured at park level.

In this chapter, we will specify the data we have access to and describe, more in detail, how we used it to reach our goal.

2.1 Method

In Section 1.1.3 we introduced the major issues that can be fixed and avoided by proper maintenance. We decided to focus on the issues, which are availability issues, trackers issues and soiling. In this section, we details of how the modelling was done for each of identified issues.

We decided that our model should run to detect issues on a daily basis. That is, we would like our model to run once a day, at the end of each day, and output the issues that were present during the last 24 hours.

2.1.1 Availability model

The availability issue is classified as when zero production occurs. To detect if a string is available we will look at the production during the day, in the time interval when the string

should produce, and see if it is available.

In order to detect an availability issues we will start by identifying the time interval when the strings should produce. This will be done by looking at the irradiance and observing if it is high enough for production to occur. Usually, we only have irradiance at park level which means that we typically have access to one or a few sensors per park. In our case we will use data from one irradiance sensor. Therefore, we will have to shrink this interval to ensure that all strings are exposed to a sufficiently high level of irradiance, so that we can guarantee that they should produce if working properly. Observe that since the measuring equipment can be poorly calibrated we will say that the string is producing if its current output power exceeds some small positive threshold. This threshold was determined together with Nispera AG and is specified in the implementation section.

Note that filtering out the unavailable strings is necessary in order to detect other issues. When modeling the other faults we do not want to compare the strings we are analyzing with string that are unavailable. Since this issue is the most severe one, the production is zero, we will not investigate if unavailable strings have other faults.

2.1.2 Tracker model

The purpose of a tracker model is to detect when the trackers are not moving properly. Issues are usually, a tracker not moving for a couple of days, a day or a shorter period of time. The focus will be on detecting issues that occur during an entire day to ensure that we detect the long term issues during the first whole day when they are present.

In Figure 2.1 we can see how the production of a PV array would look like during a day with a working tracker and with trackers being stuck in the same position. The angles corresponding to the different power outputs are specified in the figure. Observe that the power output is the same during the mid hours of the day. This happens because the solar panels become warmer during these hours and therefore they are not producing even more.

The first step in the model is to split the time interval, when the strings are producing, into three subintervals. This was decided since there are three naturally different intervals in the power output function; an increase during the start of the day, an almost constant power during mid day and a decrease of power during the end of the day. We will here denote these subintervals by I_j, $j \in \{1, 2, 3\}$, see Figure 2.2. Then we choose a reference string in these subintervals. The production of the reference string should match the production of the string we are investigating if it did not have a tracker issue. Observe that Figure 2.1 and Figure 2.2 are similar, but the first one shows how the power output with a stuck tracker while the second one explains the idea behind the algorithm we have discussed.

To continue to explain the main idea of the tracker algorithm we will let $P_i(t)$ denote the power output of string i at time t and $P_r(t)$ denote the power output of the reference string at time t. In the algorithm we will, for each string i, calculate three ratios r_1, r_2 and r_3 defined as

$$r_k = \frac{\int_{t \in I_k} P_r(t)}{\int_{t \in I_k} P_i(t)} dt. \tag{2.1}$$

After we have calculated the three ratios we would like to calculate a value corresponding to how bad the tracker performed during the day. From Figure 2.2 we see the three

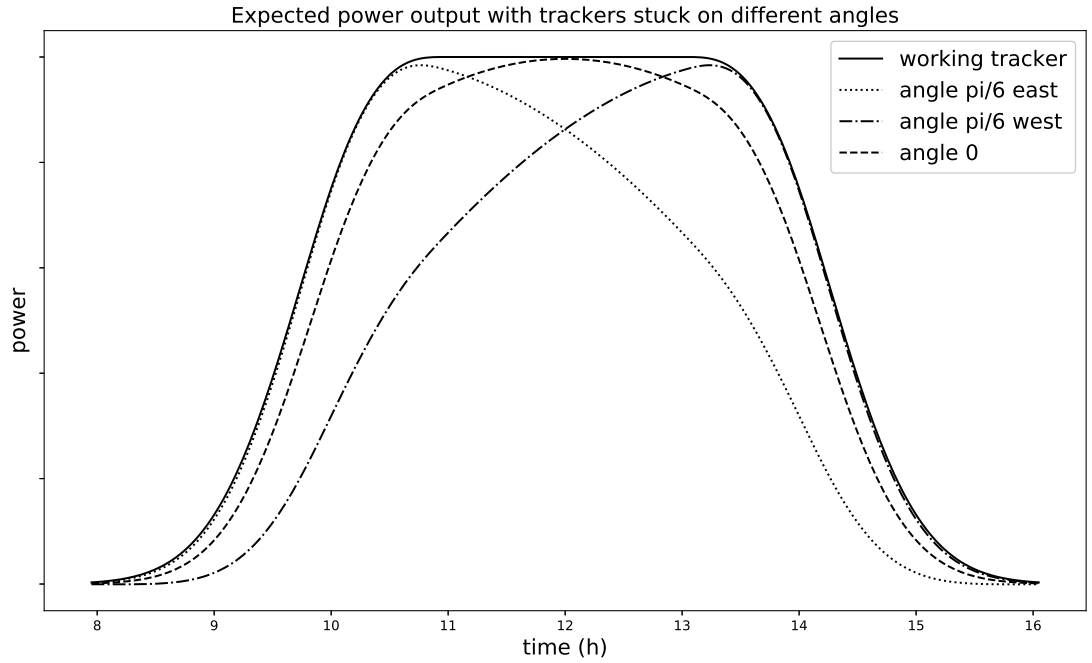

Figure 2.1: A visualization of how the power output of a single-axis (the case for our dataset) tracker that is stuck at different positions should look like. The plot was obtained by multiplying the function corresponding to the working tracker by $\cos\varphi$, where φ is the difference of the optimal angle and the real angle.

major cases of tracker issues that we would like to detect. In all these cases the ratios will not be the same in all intervals. Together with Nispera we came up with the idea to therefore introduce a penalty function, $M : \mathbb{R}_+^3 \to \mathbb{R}_+$, that will give a large value for varying ratios and a small value for similar ratios. If all the three ratios are the same we would like the penalty function to return 0 since this should only occur when a string is permanently producing a fraction of what its reference does. This can occur if the panels are at a constant offset during the entire day, that is not perpendicular to the sun. We have however decided that, in this work, we will be focusing on trackers that are not moving.

We chose the penalty function M defined in Equation 2.2. The term $1/\sqrt{3}$ inside the sum comes from that it is the value of $r_k/\|\mathbf{r}\|_2$ if $r_j = r_k$ for all $j, k \in \{1, 2, 3\}$. Also in the equation we have denoted the vector (r_1, r_2, r_3) by $\mathbf{r}$.

$$M(\mathbf{r}) = M((r_1, r_2, r_3)) = \sum_{k=1}^{3} \left(\frac{r_k}{\|\mathbf{r}\|_2} - \frac{1}{\sqrt{3}}\right)^2, \tag{2.2}$$

where the 2-norm is defined as $\|x\| = (\sum_i x_i^2)^{1/2}$. The reference string was, initially, chosen as the string that was the 90th percentile producing string. This means that the reference is the string such that 90% of the strings produce less than the reference and 10% of the strings produce more than the reference. This turned out to be a bad choice, therefore we will chose another way of finding a reference string. This new approach uses the correlation between the power production time series. To get an as accurate reference as possible we used regression with neural networks. How the correlation and neural

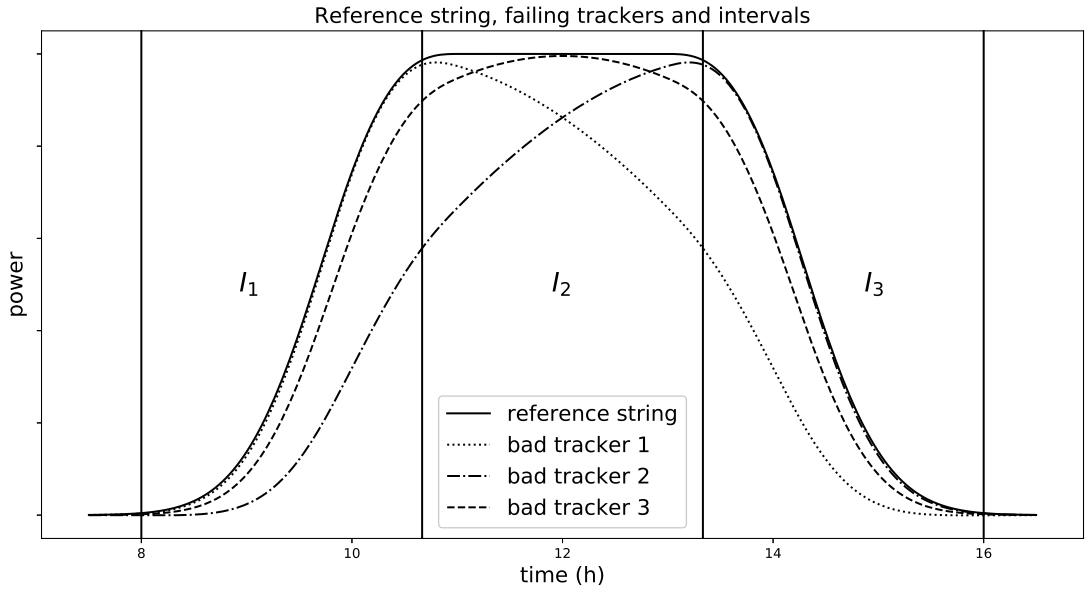

Figure 2.2: A visualization showing the idea of the algorithm designed to detect tracker issues. In the figure, four sample power outputs are visualized. The correspondence of each line is specified in the legend of the figure. Figure 2.1 specifies the angles of malfunctioning trackers.

networks were used is specified in Section 2.3. The theory behind neural networks is explained in Section 1.4.

We would like to use this method as a binary classifier to see if a tracker is working properly or not. If a tracker has been classified as not working, then we can estimate the loss due to the tracker as the reference production subtracted by the actual one.

2.1.3 Soiling model

As we explained in Section 1.1.3, there are typically to pyranometers available in large-scale photovoltaic plants. One of them is cleaned with the same frequency as the panels, and one is cleaned more regularly, typically once a week. These pyranometers enables us to get an estimate of the irradiance loss due to soiling.

In Equation 1.1 we showed how the power output function of a string looks like. To estimate the loss due to soiling we can use the measurements of irradiance and temperatures in order to estimate the parameters of the function.

We start by denoting the irradiance from the cleaned pyranometer by I_c and the irradiance measurement by soiled pyranometer by I_s. Then the loss due to soiling, ΔP, at a time when the temperature is T would be

$$\Delta P = P(I_c, T) - P(I_s, T) = \eta A(1 - \frac{1}{200}(T - 25))(I_c - I_s). \tag{2.3}$$

Observe that we do not need to find the parameters η and A individually, but only their product ηA. This can be done by performing a linear regression with the least squares approach. Since we have measurements of the temperature, irradiance and the production

for each string we can estimate the product. Observe that the irradiance measurements that we will use are the ones from the soiled pyranometer since the strings are also affected by soiling.

To derive how the estimation will be performed we start by assuming that we have m strings and n observations of irradiance I_i, temperature T_i and power output P_{ij} with $i \in \{1, 2, \ldots, n\}$ and $j \in \{1, 2, \ldots, m\}$. Let $b_j = \eta_j A_j$, for the j-th string, and $f(T) = 1 - (T - 25)/200$. Then, if we denote $b = [b_1\ b_2\ \ldots\ b_m]^T$ and $y = [I_1 f(T_1)\ \ldots\ I_n f(T_n)]^T$ we have that $P = yb^T$, where $P = (P_{ij}) \in \mathbb{R}^{n \times m}$. From this we get that the following derivation to obtain the least squares estimate

$$\begin{aligned} Pb^{-T} &= y \\ P^T Pb^{-T} &= P^T y \\ b^{-T} &= (P^T P)^{-1} P^T y \\ b &= ((P^T P)^{-1} P^T y)^{-T}. \end{aligned}$$

Observe that if x is a vector then $x^{-T} = (x^T)^{-1}$ is a trivial operation to perform. The equation that is derived in from step one to step two in the sequence of equations above is called a normal equation and is one of the standard ways to perform a least squares estimate. When $b = [\eta_1 A_1\ \eta_2 A_2\ \ldots\ \eta_m A_m]$ has been estimated we plug in the individual $\eta_j A_j$ into Equation 2.3 to get the estimated loss due to soiling for the j-th string.

2.2 Dataset

The data that was used in order to develop the models comes from a solar park whose information can be seen in Table 2.1.

Table 2.1: Park information.

Capacity	24 MW
Surrounding environment	Desert
Size	66.13 ha = 661300 m^2
Ground cover ratio	0.33
Number of strings	4074 with 19 modules each (77,406 modules)
Number of trackers	378 in total 246 with 12 strings 62 with 10 strings 41 with 8 strings 29 with 6 strings

We can see in Table 2.1 the scale of the PV power plant we have been working with. The size of the solar park, the surrounding environment and more is provided so that people that are trying to use this work will know the conditions of the solar park that was the source of data.

In Table 2.2 one can see what was provided in the given dataset. Since we would like to run the model at the end of every day, we can use the data obtained during the day.

Table 2.2: Dataset information.

Variable (unit)	Component	Resolution
Power (kW)	String	10 minute (each string)
Irradiance (W/m^2)	Pyranometer	10 minute (2 values)
Temperature (C°)	Thermometer	10 minute (1 value)
Tracker angles (°)	Tracker	10 minute (each tracker)

How it is obtained and stored will not be mentioned in this paper. Observe that the tracker angles were given only to validate the model, so that we could easily find days that had failing tracker components. The failing tracker components was detected by calculating the mean squared error between the reference angles (predefined optimal angles) and the actual angles throughout a day.

At first we were given a dataset containing the information seen in Table 2.2 from the beginning of November 2018 till the end of January 2019. It was on this dataset which the models were developed and validated for. When the models had been developed, we were given an additional dataset containing the same information, but for the months February, March and April of 2019. This new dataset was only used for evaluation of the models we developed from the other dataset. Observe that the two datasets cover different seasons of the year and this will make it harder for us to develop an accurate model. We will try to create a model that is not dependent of season so therefore it should still be possible to develop a sufficiently accurate model.

Both datasets were given in .csv files, with separate files for each of the variables specified in Table 2.2. In Table 2.3, we show how the files containing the power output of the strings look like. As one can see, we have the power output with a 10-minute frequency. These values are the calculated averages of the power output over the 10-minute interval ending with the timestamp on the same row.

Table 2.3: A visualization how the 10 first rows of a .csv file that specifies the power output of the first dataset.

	datetime	power	string
0	2018-11-01 06:10:00	0.000000	STR-1.1.1.1
1	2018-11-01 06:20:00	0.000000	STR-1.1.1.1
2	2018-11-01 06:30:00	0.250686	STR-1.1.1.1
3	2018-11-01 06:40:00	0.501583	STR-1.1.1.1
4	2018-11-01 06:50:00	0.754164	STR-1.1.1.1
5	2018-11-01 07:00:00	1.108180	STR-1.1.1.1
6	2018-11-01 07:10:00	1.492260	STR-1.1.1.1
7	2018-11-01 07:20:00	1.952440	STR-1.1.1.1
8	2018-11-01 07:30:00	2.204640	STR-1.1.1.1
9	2018-11-01 07:50:00	3.786510	STR-1.1.1.1

Figure 2.3 visualizes a similar PV power station to the one we are analyzing. Observe the scale that can be seen in the lower right corner in the figure. What can also be seen in the figure, when looking closely, is that the solar park is partially covered by clouds.

This affects the power output of the strings covered by the shadow of the cloud and will be taken into account by our approach.

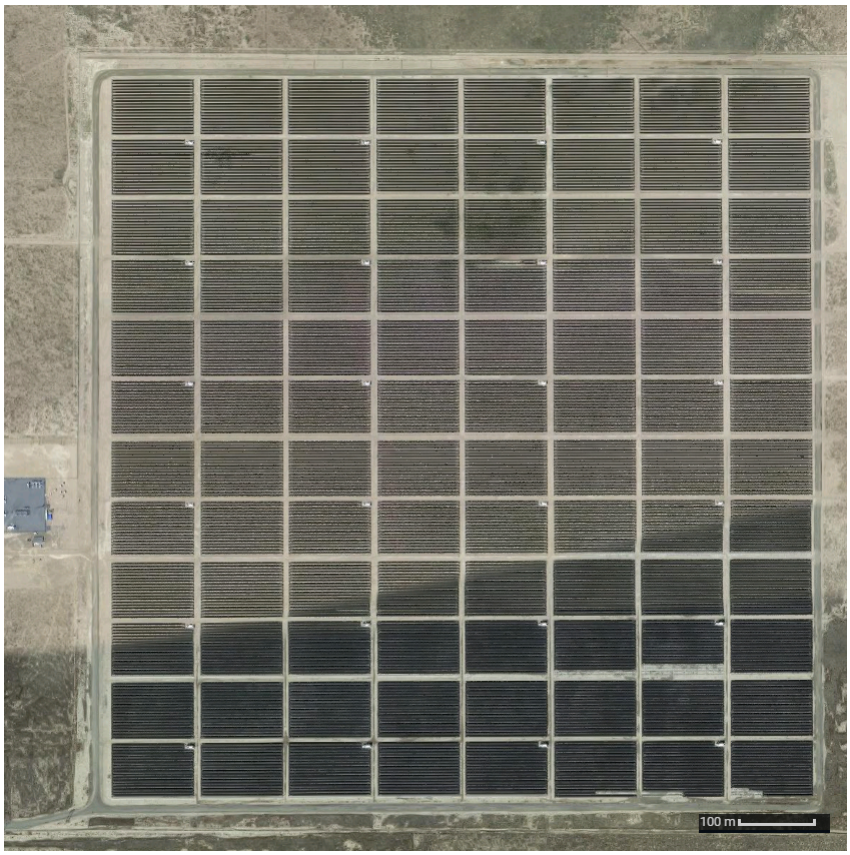

Figure 2.3: A screen shot from Google Maps showing a 30 MW solar park in a desert environment. This solar park similar to the one that has been analyzed which has a capacity of 24 MW. One can see from the scale in the lower right corner. The size of the solar park is about 1 km^2, which is around 1.5 times the size of the solar park we have been analyzing.

In addition to our dataset we were given two maps in PDF-format, one describing were each string is located in the plant and the other showing where the trackers are located. These maps made it possible for us to validate which string that corresponded to which tracker. This look up had to be done manually by looking at both of the maps however. Therefore, and due to the number of strings, we decided to not manually create a mapping between strings and trackers since it would be too time consuming. Thus no mapping between strings and trackers could be used in the models, but only for validation. In practice, this means that we can look up which strings that are on which tracker, but it takes a minute to do so for a tracker.

2.3 Implementation

The project involved parsing data and implementing models which required a significant amount of coding. We developed the models in Python 3.6 (and not a more recent version of Python so that Tensorflow and Keras would be compatible). Additionally we used Jupyter Notebook in order to be able to explore and visualize data in a simple way.

When implementing the models we discovered some problems with the data. One of those problems were missing values. The missing values could be seen in the data as entire missing rows. During some timestamps we did not have access to any measurement. In Table 2.3 we see an example of a missing value. Observe that row 8 contains the timestamp *2018-11-01 07:30:00* while row 9 contains the timestamp *2018-11-01 07:50:00* which is a 20-minute difference.

We came up with two different solutions of handling missing values. The first solution was to disregard all other measurements with the same timestamp as the missing one. This solution was not satisfactory due to the high frequency of missing values, which resulted in a removal of too many values.

Instead, we decided to implement an interpolation to approximate the missing measurement. We choose to approximate a value with the average of its surrounding two values. In the analyzed data we did not see two consecutive occurrences of missing values.

Due to the time schedule we decided to focus on implementing the availability model and tracker model. Eventually, we realized that we would not have enough time to implement the soiling model that we presented. Therefore we do not present any details about the implementation here and we will not present any results for the soiling model later in the document. We will leave the implementation of the soiling model for future work.

2.3.1 Availability Model

We implemented the algorithm for detecting availability issues described in Section 2.1.1. This algorithm is shown in Algorithm 1.

Data: Irradiance I, power output P
Result: Available: true or false
$S \leftarrow \{t \in \mathcal{D}(I) : I(t) > I_{th}\}$;
$t_{min} \leftarrow \min_{t \in S} t$;
$t_{max} \leftarrow \max_{t \in S} t$;
foreach $t \in \mathcal{D}(P)$ **do**
  if $t_{min} + d < t$ **and** $t < t_{max} - d$ **then**
    if $P(t) < P_{th}$ **then**
      return *false*;
    end
  end
end
return *true*;

Algorithm 1: Algorithm for detecting availability issues of a string.

In Algorithm 1, $\mathcal{D}(.)$ denotes the domain of a function, I_{th} is the irradiance threshold and P_{th} is the availability production threshold. The parameters we chose when running the model were $(I_{th}, P_{th}, d) = (30, 0.05, 30)$, where the unit of d is minutes.

Observe that we do not use the irradiance threshold to get the interval. We also crop it by 30 minutes on each side. This is done by choosing $d = 30$ minutes. This is because the park only has a single value for the irradiance. Since the site is very big we want to make sure that a sufficient magnitude of irradiance is present at all the strings in the solar park before filtering them about as not available.

2.3.2 Tracker Model

We implemented the algorithm described in Section 2.1.2 during the first three weeks of the project. The algorithm did not initially perform as well as we intended it to do. We realized that this occurred since the reference was not a good enough approximation to all the strings. This encouraged us to use another method for calculating the reference.

The information on where the strings are located within the solar park must play a big part on how the strings produce compared to each other. Thus, we somehow wanted to exploit the positions of the strings. We did not know these positions, however, so we came up with an idea to use the positions implicitly.

Therefore, we tried to model the positional dependencies by correlating the power outputs of the strings. We calculated the correlation matrix describing the correlation of the power outputs of the strings. We thought that if the correlation was calculated during a cloudy day then we would capture positional dependencies.

Using the correlation method we could calculate the reference output for a string in another way. We initially choose to consider the strings with the highest correlation and model the reference output as the mean of the output of these strings. This gave a huge improvement in estimating the reference for a string. This approach of determining the reference strings will be referred to as the *correlation approach* later in this document.

It is obvious that the mean estimate is not perfect since the best performing string will always have a reference output that is less than its real one. Therefore we developed a new approach that can estimate correct reference outputs even for these outlier strings.

The new approach uses neural networks to estimate the reference strings. For each string, we will use a neural network to estimate the reference output for that string. The code used to calculated the reference outputs is shown below. This approach of determining the reference strings will be referred to as the *neural network approach* later in this document.

```
1  def get_references(all_data, timestamps, dates, strings, date, top_correlated, sample_size):
2      references = {}
3      try:
4          nn_model = load_model()
5      except:
6          nn_model = build_model(sample_size+1)
7      for string in strings:
8          regression_strings = top_correlated[string][:sample_size]
9          x_test, y_test = get_test_data(all_data, timestamps, date, string, regression_strings)
10         file_path = MODEL_PATH + string
11         if os.path.isfile(file_path + '.h5'):
12             nn_model.load_weights(file_path + '.h5')
```

```
13          else:
14              x_train, y_train = get_training_data(all_data, timestamps, dates, string, regression_strings)
15              nn_model.load_weights(MODEL_PATH + 'initial_weights.h5')
16              nn_model.fit(x_train.T, y_train, epochs=15, verbose=0)
17              save_weights(nn_model, string)
18          reference[string] = nn_model.predict(x_test.T)
19      return references
```

In the code above we see the function for calculating the reference outputs for all the strings. The output of the function is a dictionary, a hash-table, containing the references which are represented by an array of floating point values. The keys of the dictionary are the string ids of the strings which corresponds to a reference.

Outside the function one needs have have the variable *MODEL_PATH* defined. Whenever this function runs for a new *MODEL_PATH* all the weights for the networks in this path will be saved to .h5 files so that when testing the model later on the weights can be loaded right into the model. Creating these .h5 files saves us a lot of time since we only need to create one model object and we can reuse it for all the different networks.

On line 5, the function load_model() returns an exception if there is a file describing a model in the MODEL_PATH.

On line 7, the function build_model specifies how the model should be built like number of layers, size of the layers, optimization method, activation functions etc. The input variable to this function is sample_size+1 since we will also have the time of the day as an input to the network. This function will also save the model, so that if the get_references function is called again the load_model will not give an exception.

When implementing the model we need to consider what we should do when an unavailable string is present when we are training or testing of the model. We decided replace the power outputs of the unavailable strings, and we implemented two different methods for doing this. The first one replaces the string with the mean production of all strings that were available during the entire day. The second method calculates the average of the n highest correlated strings that are available.

We designed the neural network to have one hidden layer. We could not have just an input layer and a single output layer since we had the time of the day as an input to the network. If no hidden layer would be present, the time of the day would just be multiplied by a weight and added to the output. Therefore if we would like the time of the day to affect the output in any other way we need a hidden layer. We also chose to not have more than 1 hidden layer since we then got too many parameters in our network. Since we would like the training and evaluations of the neural network to run quite fast, we decided that using two layers would be too slow for our purpose.

Therefore we new that our neural network should be of the form $a - b - 1$, where a is the number of nodes in the input layer and b is the number of nodes in the hidden layer. To find a good architecture we varied the hyperparameters a and b in order to find optimal values. We also tried different activation functions.

When designing the neural network we needed to make sure that it could be trained in a reasonable time since we needed to try different combinations. When using five days as input data the training process took between 30 minutes and two hours for the architectures we tried.

Eventually network architecture we decided use was 41-64-1 since we found that it performed sufficiently well and was still fast enough. We used relu, $\varphi(x) = \max(0, x)$,

activation functions, since they are non-linear and computationally fast. The optimizer we chose to use was ADAM, which is a computationally efficient gradient descent algorithm.

During the implementation phase we decided to train the model on five days in November. These days were the 5th, the 6th, the 12th, the 18th and the 21st. These days were chosen since the irradiance showed different characteristics, so that no matter how the irradiance varies we would have a case that is close to that variation in the test data.

We also chose three different days in November to calculate the correlation matrix, namely, the 7th, the 17th and the 24th. These days were mainly cloudy, the idea of choosing cloudy days is in that way capture the positional dependencies with the help of clouds. The correlation matrix was created to determine which strings that should be the input to which neural network.

Chapter 3
Evaluation

In order to evaluate the models we used a dataset which was not intersecting to the dataset used for coming up with the models. The models evaluated have not been changed after the second part of the dataset, containing the data from February 2019 to April 2019, was given. As mentioned in Section 2.3 our timeline did not allow us to implement the soiling model. Therefore results of the soiling model will not be present in this chapter.

For the tracker model we provide results for two different ways of obtaining the reference strings. The approach were we find the reference strings by neural networks will be referred to as the *neural network approach*. The approach that finds the top correlated strings and assigns the average of these to the reference string will be referred to as the *correlation approach*.

3.1 Experimental Setup

At first, we were given three months of data (Nov 2018 to Jan 2019). This data contained the data that is specified in Table 2.2 from the park which information is specified in Table 2.1. By using this data the models were developed, but these should be tested on different data to get unbiased results. Therefore, after we had developed the models, we received more data so that we were able to make an objective evaluation of the model, see Section 2.2.

We will, before presenting the results in Section 3.2, explain how we set up the evaluation of the models. Here, we will explain how we extracted data to that we later used when evaluating the model.

3.1.1 Availability Model

In order test the availability model we will look at to what extent availability issues are removed. If correctly implemented, our model will filter out the strings which have a power

output that is lower than the availability threshold when the irradiance is sufficiently large.

The model is very straight forward, and we know that it will remove the availability issues that we implemented it to remove if implemented correctly. Therefore, we need to test if we have implemented the algorithm correctly or not. In order to do this, we looked through data manually and tested the algorithm for different examples of the two available cases; available or not available.

The evaluation of the model was performed on the dataset that contains data from February 1st to April 30th. These graphs will be presented later in this chapter.

3.1.2 Tracker Model

In the dataset containing the data from February to April 2019 there were four days containing tracker issues. The angles of the trackers, and their variation throughout the day, are during these days shown in Figure 3.1. The gray lines represent the angles of the working trackers while the red ones represent the angle of the malfunctioning tracker. In the figure, 180 degrees correspond to a tracker angled upwards, parallel to the ground. The expected behavior between the mid hours of the day is a linear increase of the angle from around 125 degrees to 235 degrees. We can see from the figure that this behavior is observed for all the trackers represented by the gray lines while it is not the case for the trackers represented by the red lines.

In order to find these cases we looked through the corresponding graphs for all days in the dataset and extracted the ones having major tracker angling issues.With the tracker map and string map we could figure out which strings are on which tracker. In Table 3.1 we show what strings were located on the stuck trackers.

Table 3.1

Date	Stuck tracker	Strings on stuck tracker
2019-02-22	TC-3.1.2	STR-2.2.10.x ($5 \leq x \leq 16$)
2019-02-23	TC-3.1.2	STR-2.2.10.x ($5 \leq x \leq 16$)
2019-04-05	TC-8.4.4	STR-8.2.2.x ($5 \leq x \leq 14$)
2019-04-06	TC-8.4.4	STR-8.2.2.x ($5 \leq x \leq 14$)

3.2 Results

In this section we will present results we got with our models. These will consist of figures or tables with text explaining unclarities and the circumstances around these. The results will however not be interpreted in this section, this will instead be done in Section 3.3.

3.2.1 Availability model

By looking at Algorithm 1 we can see that it should either work, if correctly implemented, or else it will not work. Therefore we looked at hundreds of graphs and controlled manually

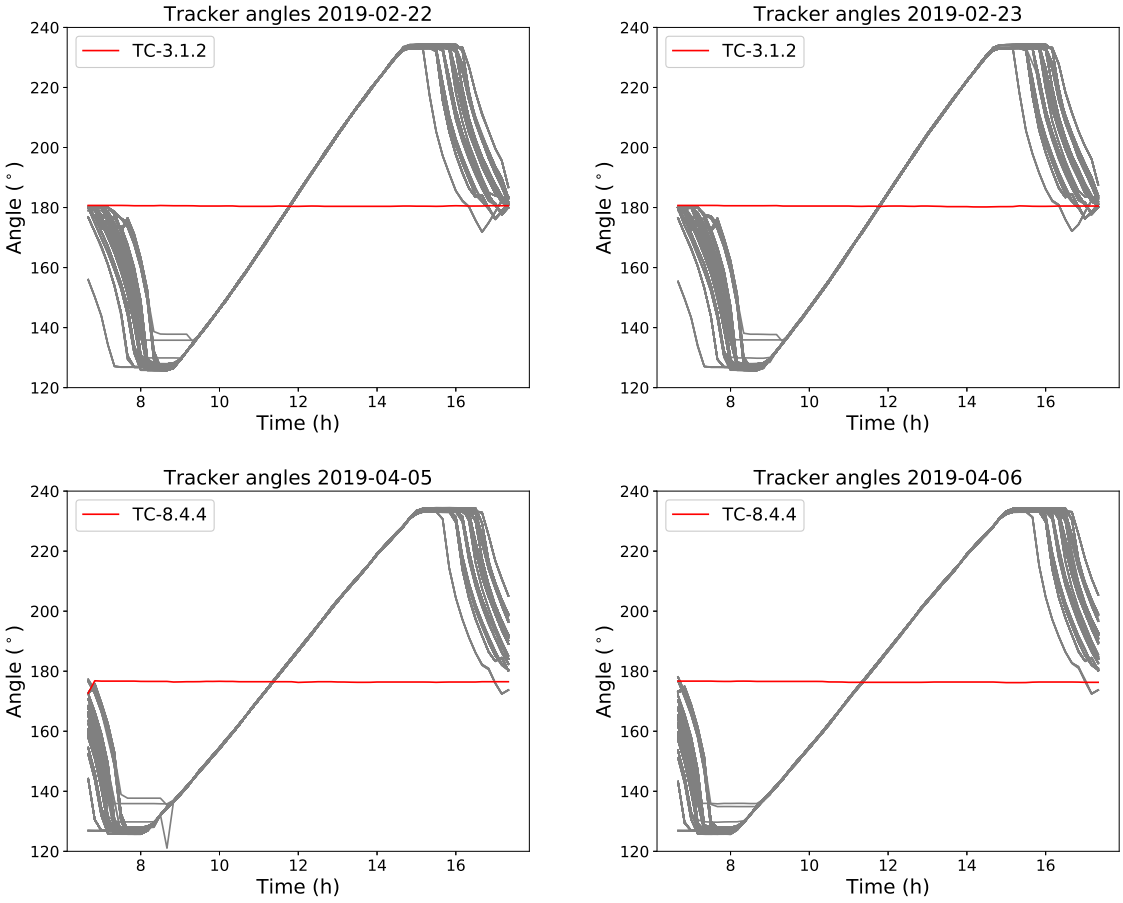

Figure 3.1: A visualization of the tracker angles during the four days were tracker whole day tracker issues occurred in the evaluation data. The angle on the y-axes is 180° when the PV arrays on the tracker are parallel to the ground for reference.

that the algorithm gave the correct result in each case. Some of these cases are visible in Figure 3.2 and Figure 3.3.

In figure 3.4 we see an example of an availability issue that was not detected by the availability model. The right graph shows that there were no clouds present during the day, while the left one shows two major drops in the power output. These two drops are availability issues that last less than 20 minutes.

3.2.2 Tracker model

The algorithm was then tested on the days in the new dataset which had a tracker issue present. We present results for two different ways of estimating the reference string. In Table 3.2, Table 3.4 and Table 3.6 the results obtained by using neural networks to estimate the reference strings are shown. In Table 3.3, Table 3.5 and Table 3.7 we present the results obtained by only using the correlation to estimate the reference stings. Observe that none of these tables include results from 22nd of February, since we found that all the strings corresponding to the faulty tracker were unavailable during that day.

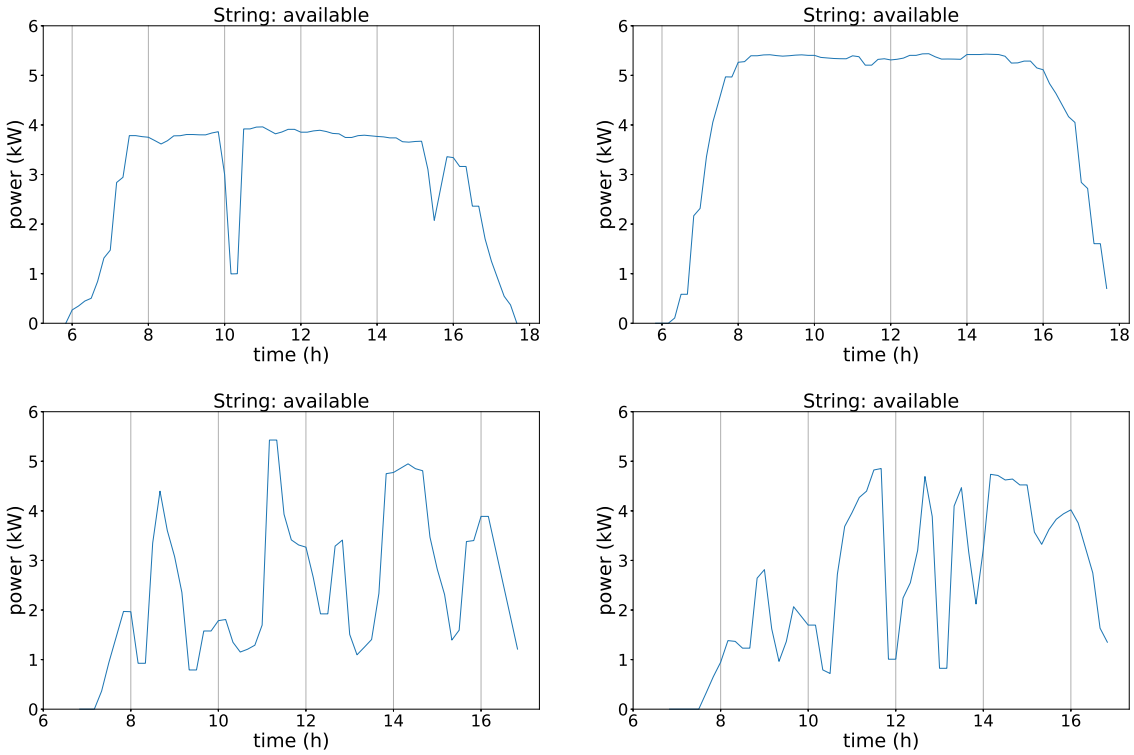

Figure 3.2: A visualization of some cases for which are algorithm flagged a string as being available.

In the tables in this subsection, the function $M(\mathbf{r})$ in the tables refers the penalty function from Section 2.1.2, where $\mathbf{r}$ denotes the ratio vector. These values in the tables corresponds to the string left of the value. The third column in the tables, that is *Strings on TC-x.x.x*, refers to the strings that are located on the tracker, *TC-x.x.x*, which was the failing tracker during the day. The last column, *Rank*, specifies the ranking of the string based on the penalty values. If a value in the last two columns has a "-" it means that the corresponding string was not available during the day, in other words it was filtered out by the availability model.

In Table 3.2 the results are shown for the neural network approach, while Table 3.3 shows the result of the correlation approach. Both show results for the 23rd of February 2019. In Figure 3.5 we have visualized the power output of a string on a malfunctioning tracker and its reference found by the correlation approach. This is similar to the behavior we expected.

In Table 3.4 the results are shown for the neural network approach and in Table 3.5 we show the result of the correlation approach. These tables show results of the two approaches for the 5th of April 2019. Correspondingly, Table 3.6 shows the results of the neural network approach while Table 3.7 shows the results of the correlation approach for the 6th of April 2019.

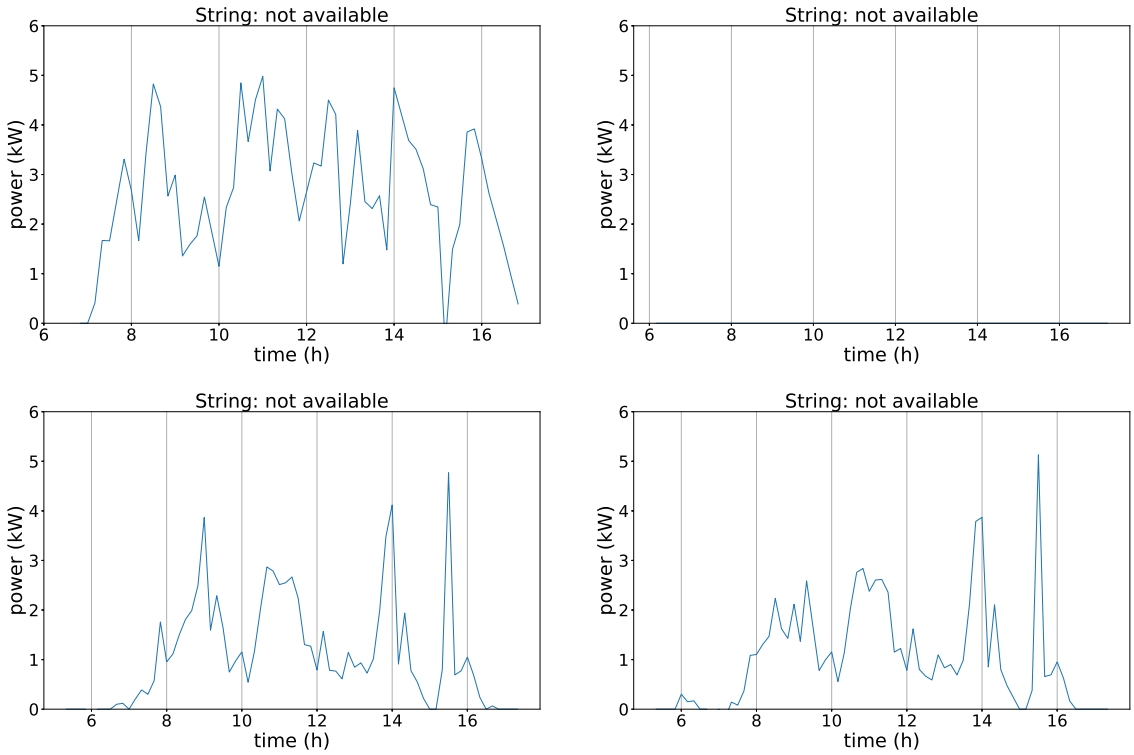

Figure 3.3: A visualization of some cases for which are algorithm flagged a string as being not available. Observe that the graph in the top right corner has a zero, or minimal production the entire day.

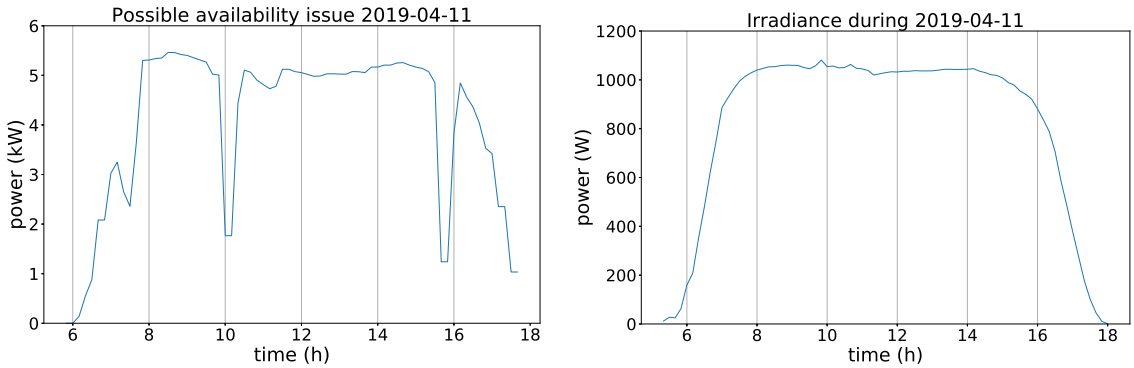

Figure 3.4: Left: A graph showing the power output of a string that was not classified as unavailable. Right: A graph showing the irradiance in the tracker plane on park level during the same day.

3.3 Discussion

In this section we will discuss the results and discuss why we believe them to be promising. We will also discuss what could be done differently in order to have achieved better results.

Table 3.2: Results of 23rd of February 2019 obtained by the neural network approach of the model.

Strings on TC-3.1.2	$M(\mathbf{r})$	Rank
STR-2.2.10.9	0.53887	1
STR-2.2.10.5	0.52570	2
STR-2.2.10.10	0.51468	3
STR-2.2.10.6	0.48083	4
STR-2.2.10.7	0.47168	5
STR-2.2.10.8	0.43701	6
STR-2.2.10.11	-	-
STR-2.2.10.12	-	-
STR-2.2.10.13	-	-
STR-2.2.10.14	-	-
STR-2.2.10.15	-	-
STR-2.2.10.16	-	-

Table 3.3: Results of 23rd of February 2019 obtained by the correlation approach of the model.

Strings on TC-3.1.2	$M(\mathbf{r})$	Rank
STR-2.2.10.10	0.40730	1
STR-2.2.10.9	0.40711	2
STR-2.2.10.6	0.40462	3
STR-2.2.10.5	0.40415	4
STR-2.2.10.7	0.40308	5
STR-2.2.10.8	0.39163	6
STR-2.2.10.11	-	-
STR-2.2.10.12	-	-
STR-2.2.10.13	-	-
STR-2.2.10.14	-	-
STR-2.2.10.15	-	-
STR-2.2.10.16	-	-

3.3.1 Availability model

We mentioned in Section 3.2.1 that the availability algorithm is simple. By looking at some examples of when the strings were available and not available we could conclude that it did what it was implemented to do.

We can see in Figure 3.4 that some availability issues are too short to be detected by the availability model with the 10-minute resolution. This is seen by the sudden loss of power which can be seen twice in the figure, this drop is of a magnitude which, due to the sunny day, must be an availability issue. In cases when availability faults occur during a time period than is shorter than twice the resolution there is no guarantee that a zero production will show in the data. These short term availability issues need to be addressed in a different way if all of these issues should be detected.

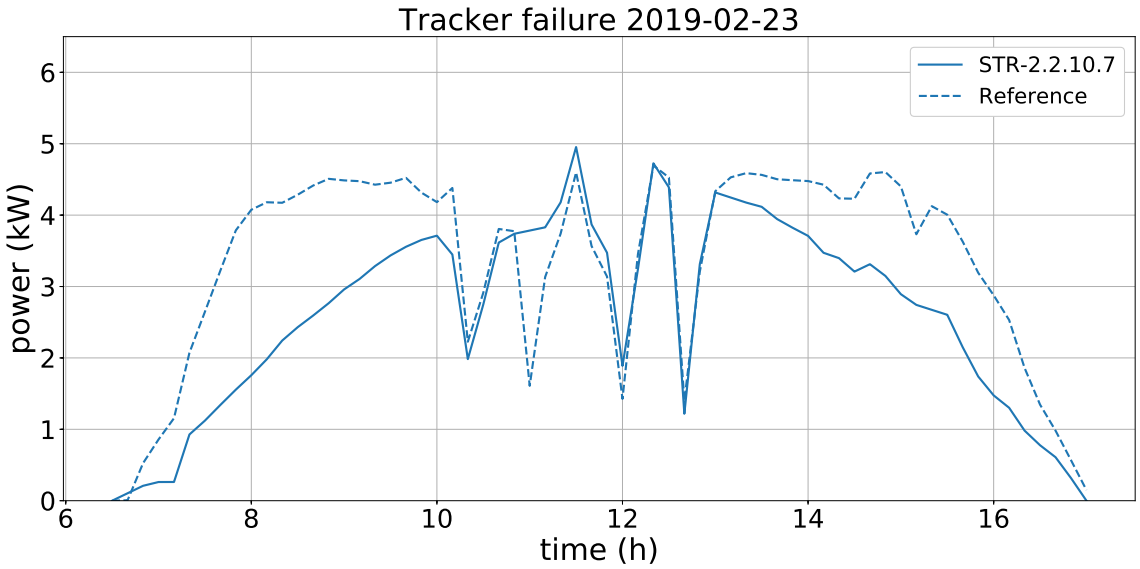

Figure 3.5: A visualization the power output of a string on a malfunctioning tracker and its reference string, found with the correlation approach.

Table 3.4: Results of 5th of April 2019 obtained by the neural network approach of the model.

Strings on TC-8.4.4	$M(\mathbf{r})$	Rank
STR-8.2.2.10	0.47124	3
STR-8.2.2.12	0.42336	6
STR-8.2.2.7	0.39360	8
STR-8.2.2.9	0.34306	17
STR-8.2.2.11	0.31718	22
STR-8.2.2.6	0.28646	31
STR-8.2.2.14	0.26334	41
STR-8.2.2.8	0.23757	54
STR-8.2.2.5	0.21737	71
STR-8.2.2.13	0.16606	146

3.3.2 Tracker model

While implementing the neural network approach it seemed very promising, but when evaluating the model with the newly obtained data we found that the results were not as good. For the 23rd of February we had good results, which can be seen in Table 3.2. We can see that the top trackers found were actually the ones having the tracker issue, and that there is a gap of the values between the penalty function of the strings with tracker issues and the strings without.

The results for the 5th of April, seen in Table 3.4, obtained with the neural network approach are not as good as we had hoped for. Even then we could argue that they are acceptable. This because all of the strings on the faulty tracker have high penalty value compared to the other strings, but some of the other strings also have penalty values of

Table 3.5: Results of 5th of April 2019 obtained by the correlation approach of the model.

Strings on TC-8.4.4	$M(\mathbf{r})$	Rank
STR-8.2.2.9	0.40544	2
STR-8.2.2.7	0.39553	5
STR-8.2.2.5	0.38706	6
STR-8.2.2.10	0.38274	7
STR-8.2.2.11	0.35780	11
STR-8.2.2.13	0.35580	12
STR-8.2.2.8	0.35575	13
STR-8.2.2.12	0.33826	17
STR-8.2.2.5	0.33039	18
STR-8.2.2.6	0.25243	32

Table 3.6: Results of 6th of April 2019 obtained by the neural network approach of the model.

Strings on TC-8.4.4	$M(\mathbf{r})$	Rank
STR-8.2.2.8	0.18147	71
STR-8.2.2.7	0.15066	100
STR-8.2.2.10	0.13549	131
STR-8.2.2.6	0.13293	136
STR-8.2.2.5	0.12872	148
STR-8.2.2.12	0.11923	171
STR-8.2.2.14	0.09118	292
STR-8.2.2.9	0.08039	369
STR-8.2.2.11	0.08023	370
STR-8.2.2.13	0.05048	762

Table 3.7: Results of 6th of April 2019 obtained by the correlation approach of the model.

Strings on TC-8.4.4	$M(\mathbf{r})$	Rank
STR-8.2.2.14	0.17379	10
STR-8.2.2.13	0.16136	13
STR-8.2.2.5	0.15647	15
STR-8.2.2.6	0.14954	16
STR-8.2.2.10	0.14620	17
STR-8.2.2.12	0.12522	31
STR-8.2.2.11	0.11721	37
STR-8.2.2.8	0.11457	39
STR-8.2.2.9	0.10892	47
STR-8.2.2.7	0.10095	56

the same magnitude. However, by looking carefully at the table we can see that the other strings with large penalty values correspond to different trackers. This information could

be used to see if all the strings on a tracker have high penalty values or if this is just the case for some of the strings. The behavior of a tracker that is malfunctioning would be that the penalty values of all the strings are high. This can be used to extract the malfunctioning trackers from the other ones.

The results achieved with the neural network approach for the 6th of April are not good at all, see Table 3.6. So by looking at the results of the neural network approach we can say that the results are not sufficiently good.

We can explain the worse results in April by that we are having the time of the day, in hours, as an input to the neural networks in this approach. Since we were training the model for only days in November, we have trained the neural network for the sun trajectory of November. Since this one is quite different from the one in April the output of the neural network will not be accurate. The solar movement is however likely more similar to November in February than it is in April which could explain the good results obtained for February 23.

This problem could be resolved by having the solar incidence angle as an input to the neural networks instead of the time of the day. One option would also be to train separate networks for each month of the year.

Another way is to implicitly use the incidence angle, to have the time of the year also as an input to the network. This is because the time of the year and time of the day determines the incidence angle for a specific location. We can input this to the neural network and let the neural network determine this dependency. This will however require a lot of training data since we will need to use days from different times of the year to get a good approximation.

In Table 3.3 we can see the results of the correlation approach for the 23rd of February. This result is like the neural network approach very good for this date. The correlation approach was more successful than the neural network approach for the days in April however, as we can see in Table 3.5 and Table 3.7.

Since the correlation approach does not require any data to train on we expected to get similar results as we got when developing the model. The result of the correlation approach for the 6th of April, presented in Table 3.7, is much better than the result obtained by the neural network approach the same day, see Table 3.6. Even though this is the case, we cannot say with certainty, from Table 3.7, that the correlation approach finds the bad tracker during this day. This is because we need to make sure that the penalty values of the strings of the faulty tracker are high compared to the others, as mentioned above.

Even though we personally think that the correlation approach is sufficiently good we cannot say this with certainty. In order to be able to validate the model we would need the mapping between strings and trackers. We could then define a penalty function on tracker level instead of string level. Such a penalty function would give us more concrete results of how the models are performing. Unfortunately we do not have access to this mapping which makes it hard to evaluate more in detail how well the model performs. If we had the mapping, which should be the case in most applications, we could define the penalty function on tracker level as the average of the penalty values obtained by its strings.

Another flaw, which emerges due to not having the tracker to string mapping, is that when determining which strings to model the reference from we cannot be sure to avoid using strings on the same tracker. This is a problem we have for both approaches of determining the references.

Chapter 4
Conclusions

In this final chapter we will say a few words about ideas how the models could improved in future work. We will first give a short conclusive summary of the work.

As photovoltaics grows very rapidly, large-scale PV systems will become more and more common. It is important to develop strategies to monitor and analyse the data of these systems in order to maximize their efficiency. In this work, we have presented an approach that is based on finding the issues that a PV system might have, and then model these issues individually. The modeling was done on string level so that a user of this model could tell which string is affected by which issue. We are the first, to the best of our knowledge, to implement failure detection models with this approach. We successfully implemented an availability model and a tracker model, and we presented an idea for a soiling model. We showed by discussion that the results are promising and that future work could improve the models.

4.1 Future Work

If we would have had more time we would have implemented the soiling model. A model calculating the how the soiling affects the output would give us a result of the producible, which was our main goal. Future work includes implementing the suggested soiling model and evaluate if it works well. And if it does we could calculate the producible of the PV system. We think that the models we have implemented are sufficiently good, a soiling model is what is missing for us to achieve our objective.

For future work we encourage developing a tracker model that uses the solar incidence angle as a parameter, as discussed in Section 3.3.2. This is because the solar angle is very robust, it is for a day of the year and time of the day predetermined. We suggest using the solar angle since the relation between the power output of strings should logically mostly depend on the solar angle, except from the actual power output of course.

We would also like to see how the results would be with a complete, usable, string

to tracker mapping. This would give us more control of how to of to distribute the input strings to the neural networks used in the neural network approach. An obvious improvement for the tracker model would be to make sure that a reference of a string is not modeled by any strings on the same tracker as the string is located on.

We can see some possible future improvements that a string to tracker mapping could give us. One of them, if a lot of trackers fail, is to model the reference of a string using at most one string per tracker. This will give us a diversity and could give potential good results if multiple trackers are failing.

Differences between large-scale PV systems might have to be taken into account when extending this to other PV systems. For instance, the two pyranometers are necessary for the soiling model to work. Another limitation is that the idea might not be as efficient for solar parks with very different geometric properties. Other than that, we do not see any obvious limitations that would occur when applying the models to another PV system. However, when doing this one should spend some time to find an optimal architecture of the neural network in that approach. This is because the number of strings varies from solar park to solar park. This process would be time consuming but the models should still be compatible.

Finally, we hope that our work will help making monitoring of photovoltaic power stations more efficient in the future. We hope that the approach we have presented will continue to be developed and successfully used to find issues in large-scale PV systems. In the big picture, we hope to contribute to the growth of solar energy and that it will make an impact to counteract global warming.

Bibliography

Al Dahoud, A., Fezari, M., Alrawashdeh, T., and Jannoud, I. (2015). Improving monitoring and fault detection of solar panels using arduino mega in wsn.

Barth, N., Figgis, B., Abdallah, A. A., Aly, S. P., Ahzi, S., and Figgis, B. (2017). Modeling of the influence of dust soiling on photovoltaic panels for desert applications the example of the solar test facility at doha, qatar. In *2017 International Renewable and Sustainable Energy Conference (IRSEC)*, pages 1–6.

Chen, S. X., Gooi, H. B., and Wang, M. Q. (2012). Sizing of energy storage for microgrids. *IEEE Transactions on Smart Grid*, 3(1):142–151.

Chollet, F. (2017). *Deep Learning with Python*. Manning Publications Co., Greenwich, CT, USA, 1st edition.

Cordero, R. R., Damiani, A., Laroze, D., MacDonell, S., Jorquera, J., Sepúlveda, E., Feron, S., Llanillo, P., Labbe, F., Carrasco, J., Ferrer, J., and Torres, G. (2018). Effects of soiling on photovoltaic (pv) modules in the atacama desert. *Scientific Reports*, 8(1):13943.

Dhanabal, R., Bharathi, V., Ranjitha, R., Ponni, A., Deepthi, S., and Mageshkannan, P. (2013). Comparison of efficiencies of solar tracker systems with static panel single-axis tracking system and dual-axis tracking system with fixed mount. *Intern. J. Eng. Technol.*, 5:1925–1933.

Kimber, A., Mitchell, L., Nogradi, S., and Wenger, H. (2006). The effect of soiling on large grid-connected photovoltaic systems in california and the southwest region of the united states. In *2006 IEEE 4th World Conference on Photovoltaic Energy Conference*, volume 2, pages 2391–2395.

Laarabi, B., Tzuc, O. M., Dahlioui, D., Bassam, A., Flota-Bañuelos, M., and Barhdadi, A. (2019). Artificial neural network modeling and sensitivity analysis for soiling effects on photovoltaic panels in morocco. *Superlattices and Microstructures*, 127:139 – 150. Materials Science for Green Energy Ifran City.

Masson, G., Kaizuka, I., Detollenaere, A., and Lindahl, J. (2019). 2019 snapshot of global pv markets.

SolarPower Europe (2018). Operation & maintenance, best practices guidelines.

INSTITUTIONEN FÖR DATAVETENSKAP | LUNDS TEKNISKA HÖGSKOLA | PRESENTERAD 2019-06-11

EXAMENSARBETE Fault modeling in large-scale photovoltaic systems
STUDENT Eskil Jarlskog
HANDLEDARE Pierre Nugues (LTH), Gianmarco Pizza (Nispera AG)
EXAMINATOR Flavius Gruian (LTH)

Modellering för förbättrad övervakning av storskalig solkraft

POPULÄRVETENSKAPLIG SAMMANFATTNING **Eskil Jarlskog**

Maximering av effektiviteten hos storskaliga fotovoltaisk solkraft kräver analysering av data. I detta arbete har vi utvecklat och implementerat lovande modeller för att optimera övervakningen av storskalig fotovoltaik.

Den fotovoltaiska effekten är ett fysikaliskt fenomen som bygger på att ljus kan omvandlas till elektricitet. I världen har denna typ av solkraft ökat exponentiellt under de senaste årtiondena. Under 2018 genererades elektricitet motsvarande 2,6% av världens energiförbrukning på detta sätt. Med en fortsatt exponetiellt ökning av fotovoltaik i framtiden kommer solkraft att bli en stor del av energimarknaden. Vi ville därför analysera de fel som bidrar till att inte solkraft fungerar optimalt.

Vi skapade modeller för de tre största felen relaterade till övervakning av fotovoltaiska anläggningar. Dessa är otillgänglighet (utebliven produktion), defekta solspårare och nedsmutsning. Arbetets mål var att bestämma hur mycket varje solpanel hade producerat om de inte var påverkade av dessa fel. Vi ville också kunna avgöra hur varje fel påverkar varje solpanel. Därför valde vi att skapa individuella modeller för varje fel.

Under arbetets gång hade vi tillgång till data från ett 66 hektar stor anläggning med en maxkapacitet på 24MW. Vi gavs data angående effekten av varje solpanel, ljusstyrkan i solkraftverket och även utomhustemperaturen på anläggningen. Solfångarnas vinklar var även givna, men dessa skulle endast användas för validering av solfångarmodellen. Given data hade en upplösning på 10 minuter och var försedd av Nispera AG.

Modellen för otillgänglighet implementerades först och användes även som ett försteg för de andra modellerna så att otillgängliga solpaneler filtreras ut. Därefter implementerades två versioner av en solspårarmodell. En version var att implicit använda solpanelernas positioner och den andra var att med neurala nätverk använda sig båda av ett rumsligt och ett tidsligt beroende mellan solpaneler. Implementationen var den mest tidskrävande delen av arbetet och tiden räckte inte till för att implementera modellen för nedsmutsning.

Resultaten vi uppnådde var lovande och visade i var på rätt spår med modellerna. Modellen för otillgänglighet fungerade bra men kan förbättras på att upptäcka kortvariga fel. Resultaten visade att den rumsliga versionen av solspårarmodellen fungerade bra medan den andra versionen hade ett problem. Problemet bestod i att, på grund tillgänglig data, tränades modellen i november vilket gav sämre resultat då den evaluerades i april.

Ett användande av solvinkeln istället för temporal data tror vi hade förbättrat vår solfångarmodell avsevärt. På grund av bristande kartläggning av anläggningens komponenter var det svårt att göra en utförlig utvärdering av modellen, men metodiken lovar gott för framtiden. Vi hoppas att idéer från detta arbete ävänds i framtiden och kan bidra till expansionen av solkraft.