

PV monitoring and fault detection

*Evaluation of machine learning for
prediction of PV soiling in Northern
Cape, South-Africa*

Gard Inge Rosvold



Thesis submitted for the degree of
Master in Informatics: Programming and Network
60 credits

Department of Informatics
Faculty of mathematics and natural sciences

UNIVERSITY OF OSLO

Spring 2017

PV monitoring and fault detection

*Evaluation of machine learning for
prediction of PV soiling in Northern
Cape, South-Africa*

Gard Inge Rosvold

© 2017 Gard Inge Rosvold

PV monitoring and fault detection

<http://www.duo.uio.no/>

Printed: Reprosentralen, University of Oslo

PV monitoring and fault detection

Gard Inge Rosvold

May 2, 2017

Abstract

Renewable energy sources, and thus PV are experiencing exponential growth due to most current energy production still relies on fossil fuels, and energy demands are steadily increasing. If the performance of PV could be increased, the result will be more production per installation.

One significant performance loss for PV is soiling on the modules. Research has been done to statistically indicate optimal cleaning intervals. Some attempts using conventional methods to predict soiling have been conducted as well, suggesting environmental features like wind and humidity are relevant factors for predicting soiling.

With the increase in popularity and availability of machine learning – is it possible to use machine learning to predict soiling? If it is possible, this could lead to quick and precise implementation of algorithms to predict instantaneous losses due to soiling. This would further lead to an exact optimal cleaning schedule, reducing both costs and losses.

With a test site in close proximity to a solar plant in Kalkbult, South-Africa, and the machine learning approach called artificial neural networks; this thesis tried to identify if this relationship exists, and if so, to what extent.

The results were encouraging, but not conclusive. There was indications the two features average humidity and maximum wind speed *could* relate to a daily change in performance with R2 scores around 0.1–0.28. However, more accurate data and designated experiments are needed to reduce uncertainties for a more conclusive remark.

Contents

1	Introduction	1
1.1	Motivation	3
1.2	Thesis overview	7
2	Background	9
2.1	State-of-the-art	9
2.1.1	Analytical monitoring	11
2.1.2	Failure Detection Routine	14
2.2	Soiling measuring	18
2.2.1	Wind and humidity	19
2.2.2	Precipitation (rainfall)	19
2.2.3	Temperature and humidity	19
2.2.4	Power reduction	20
3	Data Mining and PV	23
3.1	Data mining in PV	23
3.2	Approaches	24
3.3	Artificial Neural Network (ANN)	25
3.3.1	ANN Architecture	25
3.3.2	Learning	27
3.3.3	Model scoring and error estimation	33
4	Data collection	39
4.1	About the data	39
4.1.1	Module data	39
4.1.2	Weather data	42
4.2	Monitoring and Filtering of PV data	44
4.2.1	Requirements Specification	44
4.2.2	Functionality	44
4.2.3	Interfaces	45
4.2.4	Performance	45
4.2.5	Attributes	45
4.2.6	Design constraints	45
4.2.7	Prototyping	47
4.3	Data preparation	51
4.3.1	Variance in irradiance	52
4.3.2	The dataset	52

4.3.3	Preprocessing	53
4.4	Use of data	58
4.4.1	Irradiance variance	58
4.4.2	Choosing data from modules	58
4.4.3	Fuzzyfication (sorting) of the inputs features	59
5	ANN implementation	63
5.1	Models	63
5.1.1	Intervals	65
5.2	Neural net implementation	68
5.2.1	Programming tools and libraries	69
5.2.2	Designed classes and functions	70
6	Results & discussion	71
6.1	Model verification	71
6.2	Single hidden layer (SHL)	75
6.2.1	Model scores with unsorted inputs	75
6.2.2	Model scores with sorted inputs	80
6.3	Deep learning models	87
6.3.1	Model scores with unsorted input features	87
6.3.2	Model scores with sorted input features	89
7	Conclusion & recommendations	95
7.1	Feature relationship	95
7.2	Recommendations for future work	96
7.2.1	Secure more reliable data	96
7.2.2	Include data/measurement of airborne particles	97
7.2.3	Daily cleaning of a reference module	97
7.2.4	Expanding the neural network	97
7.2.5	Other machine learning techniques	97
7.2.6	Increase the dataset	98
	Appendices	99
A	Issues with weather data	101
B	UML diagrams	103
B.1	Data collection	103
B.2	Models	105
C	Aditonal result tables and plots	109
C.1	Unsorted inputs SHL second run and plots	109
C.2	Sorted inputs SHL plots	117
C.3	Unsorted inputs deep learning scores	123

List of Figures

1.1	World energy capacity additions by type and renewables share of total additions (IEA, nodate). (Note: Other renewables include biomass, CSP, geothermal and marine) . . .	2
1.2	Exponential global cumulative PV installation until 2015 (Fraunhofer-ISE, 2016) (Data: IHS. Graph: PSE AG 2016). . .	3
1.3	Illustration of PV-losses (<i>Energy Yield and Performance Ratio of Photovoltaic Systems</i> nodate) (more details on table 1.1) . . .	6
2.1	One week of basic monitoring data over time.	12
2.2	System yield (y_f) versus reference yield (y_r) for hourly averages over weeks in June/May 2012	14
2.3	System yield (y_f) versus reference yield (y_r) for 15-min averages in March/May 2012	14
2.4	Performance ratio (pr) versus module temperature (T_{mod}) for 15-min averages from (samples $G_I > 600W/m^2$); different subsequent weeks in March and May 2012	15
2.5	Fault detection procedure (Silvestre, Chouder, and Karatepe, 2013)	17
2.6	Fault diagnosis procedure (Silvestre, Chouder, and Karatepe, 2013)	18
3.1	An example of a neural network with 1 hidden layer.	26
3.2	Model of an artificial neuron.	26
3.3	The weight and bias adjusted functions, showing weight adjustment controls steepness and bias adjustments controls position of the function.	28
3.4	Illustration of the XOR problem (Kantardzic, 2011).	28
3.5	Gradient (in red) of a weight and its error-function (blue). . .	29
3.6	A known CNN called LeNet5 (LeCun et al., 1998), showing layer types (starting with convolution, then subsampling, convolution again etc.) and what the layer sees from the input throughout the network.	35
3.7	A quick example of the membership functions and their triangles.	35
4.1	Overview of all modules and other devices on site (Plessis, 2016)	40

4.2	Cleaning strategy of the stationary PV modules at the test site (Øgaard, 2016)	42
4.3	An overview of the main menu and their submenus	49
4.4	An entity relation diagram of entities in Faultication	50
4.5	Flowchart showing how software will enable and run monitoring, or disable it	51
4.6	The yield ratios and average irradiation on dates with calculatable points shows yield ratio often follows irradiation.	53
4.7	Shows the average yield ratios for modules 1, 3, 6, 8, 9, 10, 11, 12, 14 and 16 are somewhat restored after rain events. . .	59
4.8	An overview of S_{rate} for the applicable modules and dates. It is hard to see on paper, but Polycrystalline1 are least fluctuating module, and most aligned with the majority of other modules.	60
4.9	The yield ratio of Polycrystalline1, where it looks like the Y_R of Polycrystalline1 is somewhat restored after rain has occurred.	60
4.10	The S_{rate} values of Polycrystalline1, which will be the target values to predict.	61
4.11	Overview of how 24 sorted inputs looks like on two different days. Higher values now occur closer to eachother, reducing the dependency for a value to happen on the same hour for two different days.	61
4.12	Overview of how 24 unsorted inputs looks like on two different days. This shows ie. the highest wind speed (WS_{avg}) does not occur on the input node i for both days.	61
5.1	Showing the nodes and weight annotations for model F0 (combination of B0 and D0), where $i = \{2, \dots, n - 1\}$ and $n = 10$. Each hidden node also has a <i>bias</i> node, in order to shift all values of the activation function output.	68
6.1	These plots show the irradiation and temperature inputs of the three verification dates, along with production, which is output. Note: All input values are normalized in range -1 to 1 , and output is normalized in range 0 to 1 according to Table 4.9.	72
6.2	The production in black and predicted data in red on 16th July with $R^2 = 0.963$	74
6.3	The production in black and predicted data in red on 16th September with $R^2 = 0.845$	74
6.4	The target values with dates and distance from the horizontal 0.5 normalized soiling rate implying $S_{rate} = 0$. Positive values are increase in soiling, and negative values a decrease in soiling (improvement in performance since last day).	77
6.5	The predictions against their target values for test data using SHL on unsorted inputs. The black line on 0.5 implies $S_{rate} = 0$. Several values are spot on, with good R^2_{test} score and low RMSE and MAPE errors.	79

6.6 All predictions against their targets. Predictions on training data in orange, and test data in green. The test data consists of 39 points, approximately 33% of all targets. The black line on 0.5 implies $S_{rate} = 0$. As indicated, the test data performs vastly better than the training data for prediction. 80

6.7 The scores of Model D3, with the maximum points annotated. 82

6.8 The unzoomed scores of Model D3, with the maximum points annotated. 83

6.9 The test predictions values against their target values for Model D2 with SHL and sorted input features. There are a few predictions spot on their targets, but most is opposite of their measured value. 84

6.10 The best predictions for Model E5 for whole data and test data using sorted input features and SHL. 85

6.11 The predictions values against all targets on ModelI5 from the best R^2_{train} scores. However, it does not show much variance in output targets. 86

6.12 The predicted values against their targets for Model D5 using deep layer and unsorted input features. This was best R^2_{test} score of the unsorted deep learning models. 89

6.13 The predicted values against their targets for Model E5 using deep layer and unsorted input features. It had the best R^2_{all} score of the deep learning models. 90

6.14 The predicted values against their targets for K5, the most complex model in the top 10 R^2_{test} scores. 91

6.15 The predicted values against their targets for B1, the simplest model in the top 10 R^2_{test} show better potential than K5, even with only 8 input nodes. 91

6.16 Prediction values against their test targets for the best encountered model, D4, with deep learning and sorted input features. 92

6.17 The prediction values against their targets for Model F4, the next best encountered model with deep learning and sorted input features. The values are more spread than the best model. 93

6.18 The predicted values against their targets for Model D3, the third best encountered score with deep learning and sorted input features. Although some values are spot on, most seem to go too high, suggesting poor predictability. 93

C.1 The test predictions values against their target values for Model D3, #1 from last runs R^2_{test} with SHL and unsorted inputs. 111

C.2 The test predictions values against their target values for Model D2, #2 from last runs R^2_{test} with SHL and unsorted inputs. 112

C.3 The test predictions values against their target values for Model D5, #3 from last runs R^2_{test} with SHL and unsorted inputs. 112

C.4	The test predictions values against their target values for Model G5., #4 from last runs R_{test}^2 with SHL and unsorted inputs.	113
C.5	The test predictions values against their target values for Model G2, #5 from last runs R_{test}^2 with SHL and unsorted inputs.	113
C.6	The test predictions values against their target values for Model D4, #6 from last runs R_{test}^2 with SHL and unsorted inputs.	114
C.7	The test predictions values against their target values for Model F3, #7 from last runs R_{test}^2 with SHL and unsorted inputs.	114
C.8	The test predictions values against their target values for Model K1, #8 from last runs R_{test}^2 with SHL and unsorted inputs.	115
C.9	The test predictions values against their target values for Model E2, #9 from last runs R_{test}^2 with SHL and unsorted inputs.	115
C.10	The test predictions values against their target values for Model G3, #10 from last runs R_{test}^2 with SHL and unsorted inputs.	116
C.11	The predictions on test data against their targets for Model F2 ranked #3 on SHL with sorted input features.	119
C.12	The predictions on test data against their targets for Model G3 ranked #4 on SHL with sorted input features.	119
C.13	The predictions on test data against their targets for Model E4 ranked #5 on SHL with sorted input features.	120
C.14	The predictions on test data against their targets for Model D4 ranked #6 on SHL with sorted input features.	120
C.15	The predictions on test data against their targets for Model G4 ranked #7 on SHL with sorted input features.	121
C.16	The predictions on test data against their targets for Model I5 ranked #8 on SHL with sorted input features.	121
C.17	The predictions on test data against their targets for Model F5 ranked #9 on SHL with sorted input features.	122
C.18	The predictions on test data against their targets for Model I2 ranked #10 on SHL with sorted input features.	122

List of Tables

1.1	PV-losses (<i>Energy Yield and Performance Ratio of Photovoltaic Systems</i> nodate)	5
2.1	Parameters to be measured in real-time (Woyte et al., 2014) .	10
2.2	An example of mean and standard deviation for reference errors(Silvestre, Chouder, and Karatepe, 2013)	16
2.3	The proposed relationship between wind and humidity on soiling (Naeem and Tamizhmani, 2015).	19
3.1	A selection of neuron’s activation functions.	36
3.2	A selection of neuron’s activation functions derivatives.	37
4.1	Cleaning strategy for both polycrystalline and thinfilm modules (Weldemariam, 2016). Strategy C and D can be represented by A and B respectively.	41
4.2	Attributes for one row of data from the modules every ten minutes.	43
4.3	Attributes for one row of data from the weather station every minute.	43
4.4	An overview of the software users (entities) and their interfaces	46
4.5	The schedule of some time variables the software needs to design around	47
4.6	The key weather parameters on the reference day used to indicate yield ratio during analysis period. The back surface module temperature for both types of modules given as the midday (12:00 – 12:50) average on 11.05.2016. The given back surface module temperature are the average for all the modules of the same type.	54
4.7	Each module with their reference yield on the designated date, calculated from Eq. 2.13 with $(P^*/G_t)_{ref} = 1$	54
4.8	An overview of the valid keys, and their function when calculating interval value	55
4.9	The ranges are found by using the smallest and largest value found in the dataset within period of study for each measured variabel. Except Power, which has d_H given by specification as peak output for the Polycrystalline modules.	56

4.10	An overview of the dates with valid data. The circled date is reference day, transparent days are invalid, and striked out days are used for calculating soil rate for the following day. Striked out dates are thus invalid as target dates.	57
5.1	Output from selected modules and the rows dew are possible to occur. Note: this is all available data, not just the dates used in training the neural network.	65
5.2	An overview of the model inputs, the interval ranges and each model denotation.	66
6.1	A recap on the overview of features and its models with interval denotation.	76
6.2	The top 10 best R^2 values encountered on the test data with SHL on unsorted inputs.	81
6.3	The top 10 best R^2 values encountered using SHL on the test data with sorted inputs have a little lower values than the unsorted scores on test data.	81
6.4	The top 10 best R^2 values encountered on the train data of deep learning models with unsorted input features had better scores than the SHL setup.	88
6.5	The top 10 best R^2 values encountered on the test data of deep learning models with unsorted feature inputs had somewhat similar scores as the SHL.	88
6.6	The top 10 best R^2 values encountered on the test data of deep learning models with sorted input features encountered generally higher scores, including the highest before the last run available in appendix.	92
A.1	First the interpolated dates and their row counts, and then the invalid dates and their row counts. The date with 1441 values had a duplicate that was removed.	102
B.1	An UML diagram of the 'ProductionData' class and the relevant variables and functions.	104
B.2	An UML diagram of the 'Model' class and its variables and functions.	105
B.3	An UML diagram of the subclasses of 'Model'.	106
B.4	An UML diagram of the two subclasses for 'ANN'. They specify different layer setups with regards to their initialization. . . .	107
B.5	An UML diagram of all subclasses for both 'Basic' and 'Advanced'. These are the model classes that are initialized, and have predefined 'input_keys' as shown in notes for most classes.	108
C.1	The top 10 best R^2 values encountered with SHL on whole data for unsorted input features.	110
C.2	The top 10 best R^2 values encountered with SHL on the training data for unsorted input features.	110

C.3 The top 10 best R^2 values encountered on the test dataset after a second run with unsorted inputs on SHL. 111

C.4 The top 10 best R^2 values encountered on the whole dataset for sorted inputs and SHL. 118

C.5 The top 10 best R^2 values encountered on the train data with sorted inputs and SHL. 118

C.6 The top 10 best R^2 values encountered on all data of the deep learning models with unsorted input features. 124

Nomenclature

α	momentum constant
η	learning rate constant
ζ	Cleanness ratio
E_m	Error function
E_i	Current error
E_v	Voltage error
$G_{t.measured}$	Tilted measured global irradiation
$G_{t.ref}$	Tilted reference measured global irradiation
I_L	Irradiation
I_{dc_meas}	Measured current
I_{dc_sim}	Simulated current
I_{L0}	Irradiation at STC
P^*	Temperature corrected power output
P_{MPP}	The power output at maximum power point (MPP)
PR	Performance Ratio for a year (or other long interval)
pr	Instantaneous performance ratio
pr_0	Instantaneous performance ratio at 25\degree C
S_{rate}	Soiling rate
T_d	Dew temperature
T_{amb}	Ambient temperature
T_c	Cell temperature
T_{mod}	Module temperature
T_{STC}	Temperature at STC, 25°C

V_{dc_meas}	Measured voltage
V_{dc_sim}	Simulated voltage
Y_f	Final yield for a year (or other long interval)
y_f	Instantaneous final yield
Y_R	Yield ratio
Y_r	Reference yield for a year (or other long interval)
y_r	Instantaneous reference yield
RH_{avg}	Average humidity
$T_{env.avg}$	Average ambient temperature
$T_{mod.avg}$	Average module temperature
WS_{avg}	Average wind speed
WS_{max}	Maximum wind speed
net	A neural network
wind directions	The 12 wind directions $\{WS_N, WS_{NNE}, WS_{NE}, WS_{ENE}, WS_E,$ $WS_{ESE}, WS_{SE}, WS_{SSE}, WS_S, WS_{SSW}, WS_{SW}, WS_{WSW}, WS_W, WS_{WNW},$ $WS_{NW}, WS_{NNW}\}$

Preface

Thank you

my significant other,
for rekindling my passion for science and technology many years back
– this is the result

my two supervisors at Institute for Energy Technology (IFE):
Øystein Ulleberg & Josefine H. Selj,
for taking me on as a student of MSc., and all support given underway

prof. Yan Zhang,
for our meetings and discussions on machine learning and neural
networks

all who read through my thesis,
providing feedback and encouraging words, spending your time on me
results from the neural network(s),
for ending encouragingly, even though many of you were sacrificed
along the way

With these last letters, my thesis has come to an end.

Gard Inge Rosvold
May 2nd, 2017 Kjeller

Chapter 1

Introduction

World energy demands are currently dominated by fossil fuels. However, there exist a broad agreement the release of green house gases (GHG) from fossil fuels are related to the global increase in temperature and subsequently climatic problems. Simultaneously, the world energy demand increased steadily by around 1.7% per year the past decade (IEA., 2015a). This growth is largely due to the continuous increase in world population, which reached 7 billion in 2015. Also, most of the population increase occurs in areas experiencing economic growth and improvement in living standards – both boosting energy demand.

In order to mitigate this, the world realizes the need to save energy and replace fossil electricity production with renewable energy (ipcc-contributors, 2014). This is reflected by agreements like Kyoto Protocol and the recent Paris agreement, which influence the world energy outlook:

- *Renewables grow rapidly, almost quadrupling by 2035 and supplying a third of the growth in power generation.* (BP, 2016).
- *Electricity consumption grows by more than 70% to 2040, but 550 million people still live without any access to electricity at that time. Renewables overtake coal as the largest source of power generation by the early 2030s and account for more than half the growth in the Outlook. By 2040, renewables-based generation reaches 50% in the EU, around 30% in China and Japan, and above 25% in the United States and India. In contrast, coal is just 13% of electricity supply outside of Asia.* (IEA., 2015b).

Even though the outlooks have different scale (third vs. half) in power generation, both specify renewable energy sources to be a rapidly increasing market in both the near and far future.

IEA has stated that *The markets for wind power and solar photovoltaics (PV) are currently the most dynamic, with falling technology costs (in particular for solar PV), expanding policy support and potential for increased deployment around the world* (IEA., 2015b). The recent outlook provided numbers that renewables-based power capacity additions set a new record in 2015, and exceeded all other fuels for the first time, as shown in

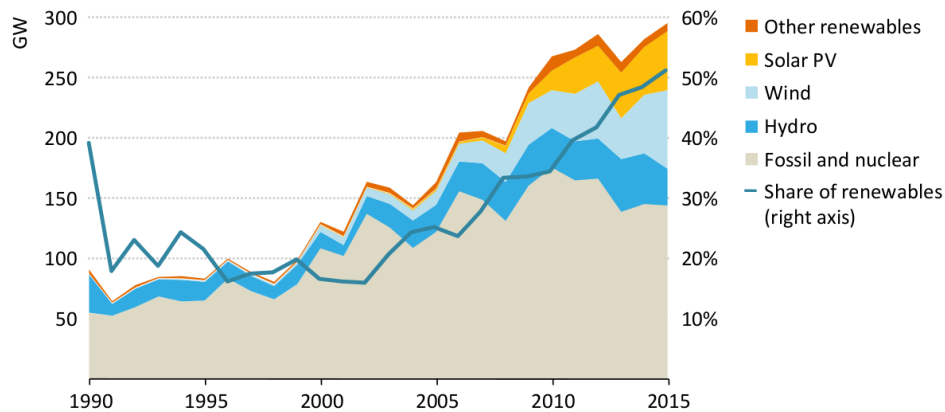


Figure 1.1: World energy capacity additions by type and renewables share of total additions (IEA, nodate). (Note: Other renewables include biomass, CSP, geothermal and marine)

Figure 1.1. One reason for this is the fact that PV has had an exponential growth as can be seen in Figure 1.2. This growth is due to PV has experienced a drastic cost reduction the last few years. This has made it commercially available to both private households and businesses, or even as power and utility plants in many countries. This is especially true for the poor regions along equator and countries south of Sahara where direct sunlight is abundant and the energy demand is rising rapidly. The increase in PV-market and especially installment of larger PV-plants have made research in improving system performance more relevant.

From the reports and figures it becomes clear that it will be more and more important to improve performance of PV in the future – even a small increase of 0.1-1% today would increase performance and power production by 5-50 GW.

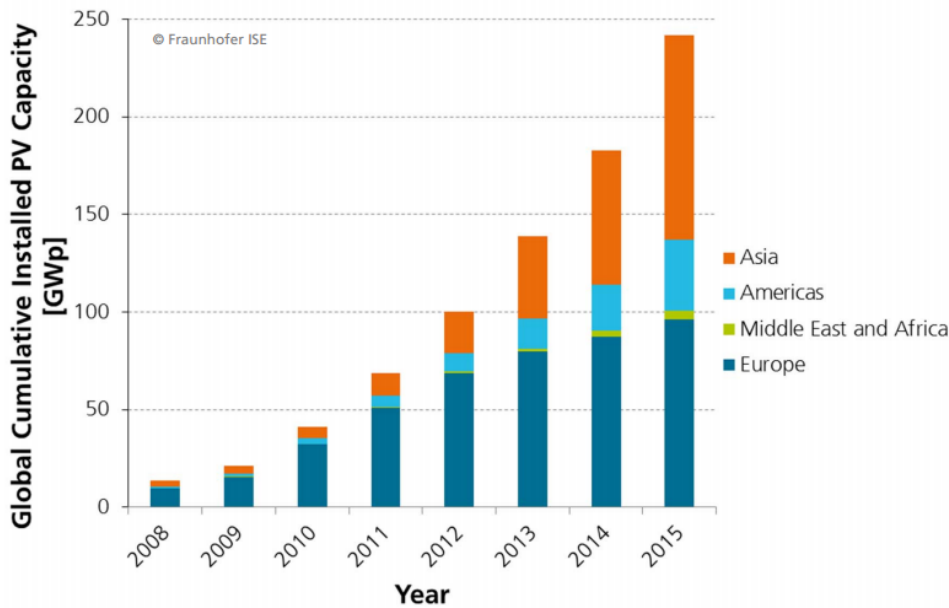


Figure 1.2: Exponential global cumulative PV installation until 2015 (Fraunhofer-ISE, 2016) (Data: IHS. Graph: PSE AG 2016).

1.1 Motivation

The efficiency of the most commonly used solar cells (Si-based PV) is in the range 15-17%. Most of the inefficiencies in the PV systems come from the energy losses within the modules themselves, but a small part are external losses during operation. Locating the *cause* of these losses and finding methods to prevent or reverse them will increase PV performance. Figure 1.3 shows an example of a simplified diagram of average losses in PV systems; where the largest losses occur in the PV module. Accordingly research has been on increasing module effectivity by improving the cell efficiency. It is still possible to increase this in the order of a few percent, but the majority of these losses are unavoidable in normal c-Si cells. The other losses from the figure; pre-photovoltaic (ie. shading, reflection, dirt or snow) and system losses account for around 20% together – but many are reversible. This means these losses can be reduced by *reversing* the incident that caused the loss. Brief descriptions of some losses shown in Figure 1.3 are presented in Table 1.1. The pre-PV losses are both the most challenging and significant losses to identify. They are significant because the losses account for around 8% *reversible* loss. They are challenging because they are hard to predict, even while they are happening, ie. dirt accumulating on the panels or an unknown shadow from a new building or other tall objects. Even though the system losses make up for about 14%, they are usually easy to predict or calculate, and are both reversed and improved by changing or upgrading to more efficient equipment, ie. better inverters, wires, modules etc. Thus in order to increase system performance, it is necessary to detect loss (failure) during operation. This is possible to achieve by monitoring

a set of parameters and analyzing the system behaviour. Monitoring is gathering a collection of real-time quantities and their historical values. A change in system behaviour gives a quantifiable difference between these values, and analyzing this difference is needed to determine how to reverse failure, if possible.

Although one of the reversible losses, soiling, has been researched for over seven decades, it is still not fully understood, nor has it been given much effort until recently. This is likely due to earlier research was located in temperate areas with frequent precipitation, generating insignificant soiling loss. However, because of the aforementioned increase in use of PV in Middle-East, Asia, and North and South Africa, these dry and less humid areas have been observed to be prone to as much soiling in hours, as months would soil in temperate areas. (Sarver, Al-Qaraghuli, and Kazmerski, 2013), increasing the need to understand soiling.

This is the main motivation for the work of this thesis, where the goal is to continue the work of others on soiling of PV, to evaluate the possibility of using data mining to predict the degree of soiling at a specific geographic location. To the knowledge of this author, there exists little to none research in this specialized field. There are several machine learning studies on predicting PV output or other influencing factors like rain, but none on soiling prediction. The motivation to evaluate soiling prediction is to determine *when* the optimal time to clean modules will be. Other studies have concluded with general guidelines about when to manually clean for various regions, ie. once halfway through summer drought period in California, USA (Mejia and Kleissl, 2013), or a more comprehensive and technology specific *Optimal days to next cleaning* overview from Saudi Arabia available in the appendix (Jones et al., 2016). Of course these recommendations depend on the size of the plant, where bigger plants will more likely be better off with more frequent washes, while smaller systems may not need to wash at all because of the small gains. Another problem with guidelines are the risk of washing a clean system, but in our computer age it should be possible to measure some selected system features, and determine *how soiled* the system really is. If this is predicted in real-time, the system can easily calculate instantaneous losses in both *kW* and revenue lost from real-time electricity rates. Comparing the real-time revenue lost against system-cleaning costs gives a precise optimum-cleaning schedule – the moment losses are bigger than costs, with no indication of near-precipitation.

The data used in this study comes from a test site adjacent to Scatec Solar's solar park in Kalkbult, Northern Cape, South-Africa (latitude: -30.2, longitude: 24.1). This site includes regularly cleaned and uncleaned panels, and its own weatherstation. The thesis will use humidity and wind (speed/force and possibly direction) as input features against power output. These features are chosen based upon the conclusions from a study of the climatological relevances to soiling in Mesa, AZ-USA (Naeem and Tamizhmani, 2015). On regular days with neither rainfall nor duststorm it was shown that both wind speed and humidity have influential roles on both soiling increase and decrease. The relevant key conclusions at that location

Table 1.1: PV-losses (*Energy Yield and Performance Ratio of Photovoltaic Systems* nodate)

Pre-PV Losses	Tolerance of rated power	Consider that the module does not deliver the power as stated in the data sheet. Manufacturers provide a tolerance, often up to 5%.
	Shadows	Shadows may be caused by trees, chimneys etc. Depending on the stringing of the cells, partial shading may have a significant effect.
	Dirt	Losses due to dirt up to 4% in temperate regions with some frequent rain. Up to 25% in arid regions with only seasonal rain and dust.
	Snow	Dependant on location and maintenance effort.
	Reflection	Reflection losses increase with the angle of incidence. Also, this effect is less pronounced in locations with a large proportion of diffuse light, i.e. clouds.
Module Losses	Conversion	The nominal efficiency is given by the manufacturer for standard conditions.
	Thermal losses	With increasing temperatures, conversion losses increase. These losses depend on irradiance (i.e. location), mounting method (glass, thermal properties of materials), and wind speeds. A very rough estimate is 8%.
System Losses	Wiring	Any cables have some resistance and therefore more losses.
	MPP	Ability of the MPP tracker to consistently find the maximum power point.
	Inverter	Inverter efficiency.
	Mis-sized inverter	If the inverter is undersized, power is clipped for high intensity light. If it is oversized, the inverters's efficiency will be too low for low intensity light.
	Transformer	Transformer losses in case electricity has to be connected to a high-voltage grid.
Operations & Maintenance	Downtime	Downtime for maintenance is usually very low for photovoltaic systems.

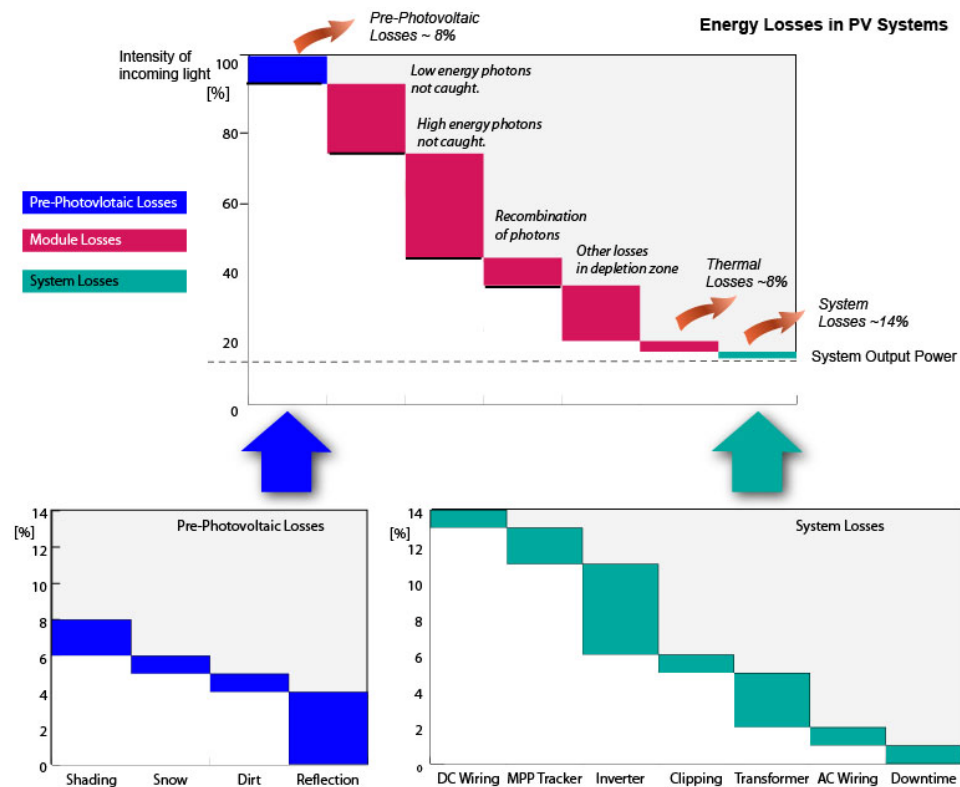


Figure 1.3: Illustration of PV-losses (Energy Yield and Performance Ratio of Photovoltaic Systems nodate) (more details on table 1.1)

were found to be:

- *Relative humidity and wind speed are the main climatological factors relevant to the soiling loss. As relative humidity increases, soiling loss increases. As wind speed increases, soiling loss decreases, provided that the wind is not high enough to lift up/carry dust with it.*
- *The cleaning effect of high winds gets even higher when they stay for multiple consecutive days.*
- *It was noticed that the highest daily wind speeds occur when the relative humidity is at its lowest. Thus, the cleaning potential due to high wind speeds is higher during such times.*

The extraction and selection of periods and parameters will be chosen based upon other studies within the same project, other articles, and possibly induced experimental values.

1.2 Thesis overview

The remainder of this thesis is divided into these chapters.

Background

More detailed background on how monitoring is done, and some of the common ways to detect and examine what failure that has occurred. It also includes information on studies done with regards to soiling.

Data Mining and PV

This section provides some introduction to Data mining techniques, and an overview of some contributions it has done to PV.

Data collection

Description of the data used in this thesis is done in this section. It also provides information about the PV monitoring system the start of thesis consisted of. Last it explains the preprocessing that is done before usage of the data.

ANN implementation

This chapter tries to show how the previous chapters is applied to define and create and model different setups of the artificial neural network.

Results

First a verification of the neural network and its ability to predict power over different selected dates is shown - to provide trust in the models. Then the results from the defined modules are presented and discussed.

Conclusions

This chapter provides some thoughts on what and why the results were. There are also some suggestions and recommendations on how to improve the results.

Appendices

- Appendix A: An overview of some problems with the weather data
- Appendix B: UML diagrams of relevant classes for the neural network
- Appendix C: Extra tables and plots of results

Chapter 2

Background

This chapter provides background for PV monitoring and fault detection. This is necessary for subsection 4.2, and the understanding of the implementation of data mining techniques. In the following section, state of the art analytical monitoring principles, and how they can be used to detect faults are reviewed from a thorough paper on analytical monitoring (Woyte et al., 2014). After that comes a section on failure detection routine (FDR), before the last section with more details on soiling of PV.

2.1 State-of-the-art

Although PV monitoring is a relative new research topic, there has been an International Electrotechnical Commission standard (IEC std.) on PV performance monitoring since 1998 (*TC 82 Solar photovoltaic energy systems (61724) 1998*).

The IEC std. (61724) has a range of requirements for what they call analytical or detailed monitoring. It defines an automatic dedicated data acquisition system with a minimum set of parameters to be monitored (IEC., 1998). The standard is currently under revision, and may include classification of monitoring system with sensor requirements, new parameters for measuring, new temperature-corrected performance ratios and other metrics, among others (Gostein, 2014). Table 2.1 provides an overview of some core parameters together with their symbol, units and priority. There are three parameters with priority 1: In-plane irradiance, ambient temperature and power to utility grid. These three are important to measure because they are used in order to evaluate how well a system is performing. The priority 2 parameters are used to detect and determine faults within the system. Last parameters in priority 3 are used alongside the other parameters to more precisely identify the actual fault.

The first introduction to guidelines on PV monitoring were *originally developed to establish the main operating characteristics of systems in demonstration projects without providing any guidance for reducing output losses over system lifetime*. Thus the new monitoring guidelines include the Failure Detection Routine (FDR). It consists of three different parts; failure detection system, failure profiling method and footprint method. Basically

Table 2.1: Parameters to be measured in real-time (Woyte et al., 2014)

Parameter	Symbol	Unit	Priority
In-plane Irradiance	G_I	W/m^2	1
Ambient temperature	T_{amb}	deg C	1
Module temperature	T_{mod}	deg C	2
Wind speed	S_w	m/s	3
PV array output voltage	V_{DC}	V	3
PV array output current	I_{DC}	A	3
PV array output power	P_{DC}	kW	2
Utility grid voltage	V_{AC}	V	3
Current to utility grid	I_{AC}	A	3
Power to utility grid	P_{AC}	kW	1
Durations of system outage	t_{outage}	s	2

FDR compares real-time monitored data against simulated data for the same period. If monitored data diverts from simulated, a failure may have been detected. A failure profile is then created using correlation between the monitored data and predefined profiles. This gives a statistical overview of what failure may have caused the diversion. The footprint method analyzes patterns in dependencies of three domains: normalized monitored power, time of day and sun elevation. This method has been developed and validated with data from the German 1000 roofs program. Another verified FDR model is the *Sophisticated Verification Method*, which utilizes fundamental system specifications and four simple measurable quantities to detect and classify failure (Kurokawa et al., 1998).

In the end, monitoring of the system and its environment is required to profile a system and implement a statistical approach on likely and unlikely cause of change in system behaviour. Considering that many of the non-module losses are reversible, and early detection prevents further deterioration, optimizing through monitoring can increase performance by reducing the average non-module losses.

2.1.1 Analytical monitoring

There are many parameteres to monitor in a system, though not every parameter on Table 2.1 is required. However, more measurements improve the likelihood to detect failures in operation earlier and more precisely.

Various methods and models exists in order to analyze a PV system. One way is simply visualization of the data recorded, this is referred to as *stamp plots*. An example of a stamp plot from a weekly output of some basic parameters is shown on Figure 2.1. These stamp plots show a monitored measurement and its relation to time over one week.

Some major faults is possible to detect in a system using stamp plots, but *linear regression* using scatter plots is a more effective method. The linear regression line for a weekly output expects a similar regression line the next week under normal circumstances. In other words, a gradual change in the line express a gradual change in the system, while a major change in the regression line proves some major fault (or action) has taken place. This linear regression are created with the relationship between two monitored measurements, instead of one against time. An overview of some common relationships will be explained in the following sections. Other relationships and equations to calculate non-monitored data exists (*PV Modeling Collaborative 2016*).

PV system performance

Performance of a system is measured as *Performance Ratio (PR)*. *PR* is the normalized relation with regards to irradiation between *system yield (Y_f)* and *reference energy yield (Y_r)*. Y_f is the *final* energy produced and measured by the system, while Y_r is the *reference* production the system is supposed to produce if under the same conditions with regards to standard test condition (STC). STC is defined as irradiance of $1kW/m^2$, $T_{cell} = 25 \text{ deg C}$ and air

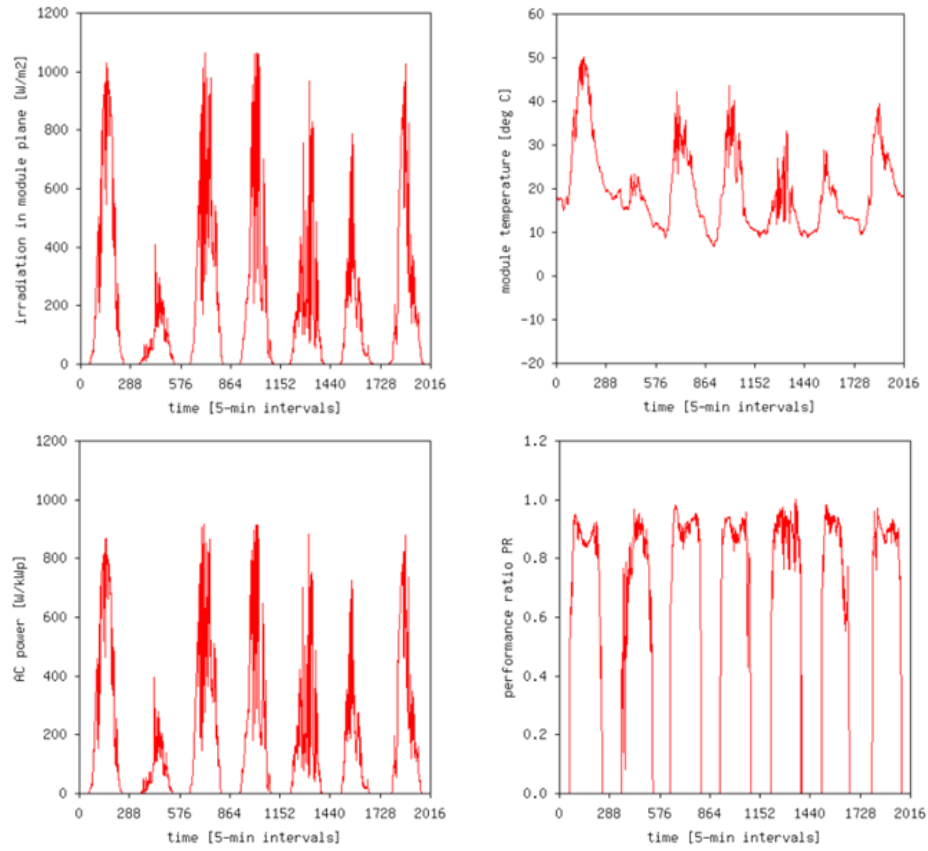


Figure 2.1: One week of basic monitoring data over time.

mass (AM) of 1.5. A PR value of as close to 1 is desired - but a PR above .9 (90 %) is seldom achieved because Y_r has low cell temperature under Standard Testing Conditions (STC). Equation 2.1 can be used to describe the relation:

$$PR = \frac{Y_f}{Y_r} \quad (2.1)$$

where:

PR : is the system performance (pr is the instantaneous performance ratio),

Y_f : Measured system yield over a period of time (y_f is instantaneous value),

Y_r : Simulated reference yield over same period using STC (y_r is instantaneous value)

It is possible to detect changes in system performance by using linear regression on PR scatter plots. In Figure 2.2 a normal (a) and gradually dropping (b) regression line is observable. During the normal operation, the regression lines are fairly atop on each other, however a gradually declining slope is visible on (b). This was due to increased shadow from vegetation. Another example is seen on Figure 2.3 where normal operation is on the left (a) and the inverter failure (b) is seen on the right. Notice here how a

regression line seems to float, not only between other regression lines but also between the scatter points. This signifies a sudden change in PR – inverter failure.

Using periodical linear regression (ie. on each week) indicates the same ongoing deterioration if the lines have regular changing, or an extensive failure if the line has shifted significantly from one period to another. Figure 2.4 shows another example of a floating regression line between both scatter points on 2.4b, and its before line on 2.4a.

Temperature

One of the most influential parameters on PV is temperature. The performance ratio (pr) vs. module temperature can be seen as a linear function that decreases as temperature rises. Notice it is lower case pr which indicates a shorter interval. The previous defined PR usually indicates performance ratio of a whole year. A simple model on temperature is system level pr given by Equation 2.2:

$$pr = pr_0(1 + \gamma\Delta T) \quad (2.2)$$

where:

ΔT : $T_{\text{mod}} - T_{\text{STC}}$ the difference to 25°C under STC,

γ : the temperature coefficient of power over the measured range of irradiance (usually negative),

pr : the instantaneous performance ratio,

pr_0 : the model performance ratio at 25°C

The coefficient γ is usually specified in the datasheet of a module, and pr_0 is 1 because it should be perfect pr in STC conditions. If they are not available, it is possible to determine them by linear regression if the module temperature is measured, γ should not change throughout a modules life. This only works well on high irradiance levels, any values with $G_I < 600\text{W/m}^2$ should be omitted. If module temperature are not measured, it can be calculated using Equation 2.3.

$$T_{\text{mod}} = T_{\text{amb}} + k_{\text{th}}y_r \quad (2.3)$$

where:

T_{mod} : the module temperature,

T_{amb} : the ambient temperature,

y_r : the instantaneous reference yield and,

k_{th} : the equivalent thermal resistance.

The thermal resistance k_{th} is not a strictly thermal resistance but comprises all mechanisms of heat transfer within the module. It can be calculated by measuring the other variables for several weeks, because the k_{th} should not change. When dealing with temperature, another factor to consider is wind. Larger wind speed effectively cools modules and will

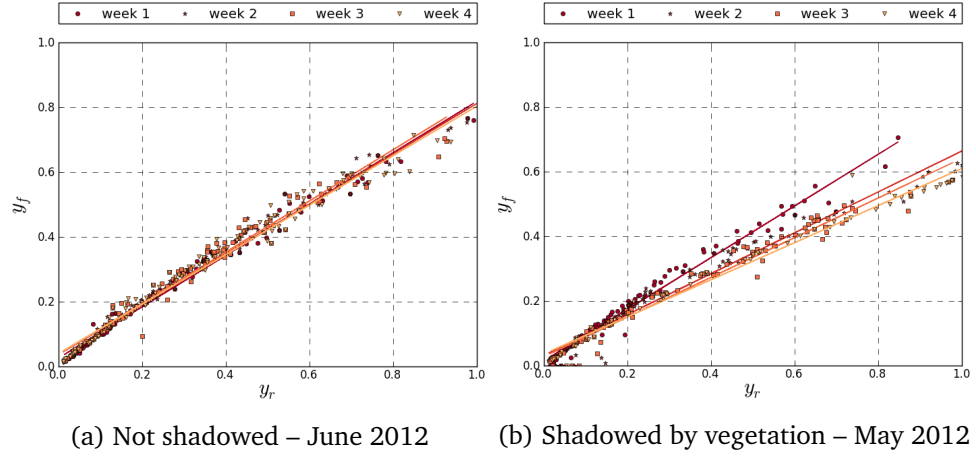


Figure 2.2: System yield (y_f) versus reference yield (y_r) for hourly averages over weeks in June/May 2012

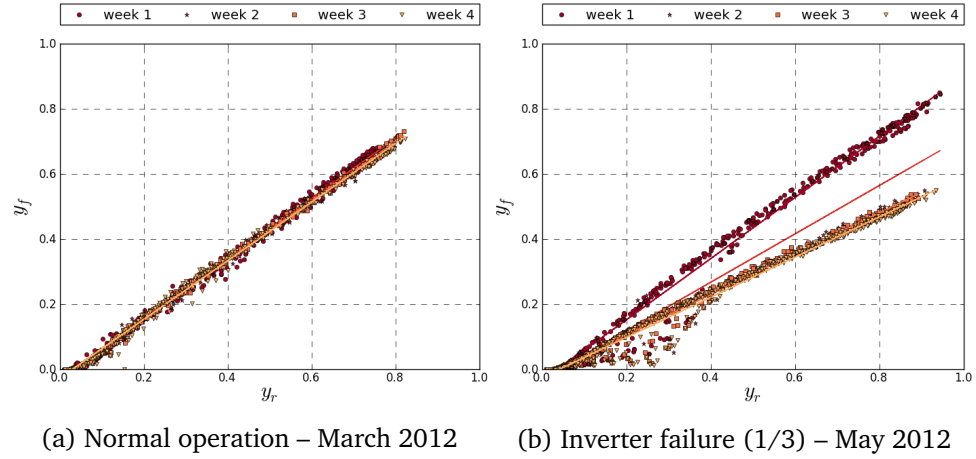


Figure 2.3: System yield (y_f) versus reference yield (y_r) for 15-min averages in March/May 2012

reduce temperature. The adjusted thermal resistance can thus be calculated with Equation 2.4:

$$k_{th} = k_{th0} e^{-C_{th} S_W} \quad (2.4)$$

where:

k_{th} : the equivalent thermal resistance,

k_{th0} : the equivalent thermal resistance without wind,

C_{th} : coefficient for thermal convection and S_W : wind speed.

2.1.2 Failure Detection Routine

As mentioned in section 2.1 the failure detection routine (FDR) consists of the three parts; failure detection, failure profiling and footprint method.

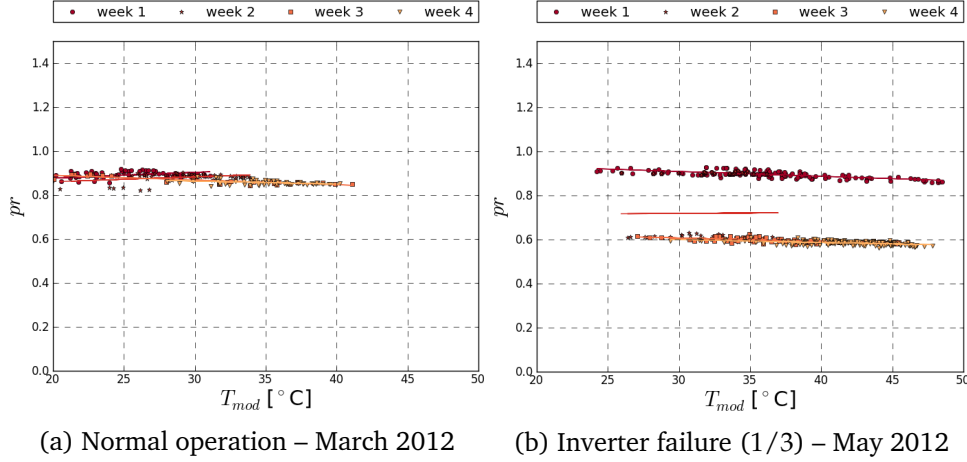


Figure 2.4: Performance ratio (pr) versus module temperature (T_{mod}) for 15-min averages from (samples $G_I > 600W/m^2$); different subsequent weeks in March and May 2012

Failure detection system

The failure detection system is the continued checking of measured actual values in the system against simulated values from forecast measurements and expected output. If the actual values are within a given threshold, the system is considered to be working as expected. If the measured values goes outside of the threshold however, there is a *possible* failure detected. The detection system then either alerts the system owner/user (supervised), or the failure profiler if applicable.

Below is an example of a fault detection (Silvestre, Chouder, and Karatepe, 2013). The overall power losses are defined by the normalized total capture losses L_c which can be calculated from Expression 2.5

$$L_c = Y_r(G, T_c) - Y_a(G, T_c) = \frac{H_i}{G_{ref}}(G, T_c) - \frac{E_{dc}}{P_{ref}} \quad (2.5)$$

where:

$Y_r(G, T_c)$: is the reference yield,

$Y_a(G, T_c)$: is the array yield,

G : real working irradiance,

T_c : real module temperature,

H_i : is the total irradiation in array plane,

G_{ref} : is the reference irradiance at standard testing conditions,

E_{dc} : is the energy produced by PV array,

P_{ref}^* : is the maximum power output of PV array

The fault detection calculates the instantaneous capture losses L_c using measured weather and electrical parameters. While the simulation model are evaluated with measured weather variables G and T_c . This makes it possible to find an error parameter, EL_c ,

$$EL_c = |L_{c_meas} - L_{c_sim}| \quad (2.6)$$

where:

L_{c_meas} : indicate measured values,

L_{c_sim} : indicate simulated values

To determine if the error EL_c is indeed an error, a deviation threshold should be established. The standard and mean deviation can be derived when the system is working fault free. An example found by trial and error showed a PV system to work fault free with values from Table 2.2 when a reference error, EL_{c_ref} , is in between the following thresholds (Silvestre, Chouder, and Karatepe, 2013):

$$EL_{c_ref} - 2\sigma(EL_{c_ref}) \leq EL_c \leq EL_{c_ref} + 2\sigma(EL_{c_ref})$$

The flow diagram on Figure 2.5 the systems follows the *Yes* arrow and recalculates EL_c infinitely as long as the calculated EL_c is within the established threshold. Once the calculated EL_c is outside the established threshold, the software follows *No* and starts a *fault diagnosis procedure* (failure profiling).

Failure profiling method

The profiling method is used to create an error profile after a possible failure has been detected. The profiling can easily exclude the most unlikely faults using the profile, and give a list of likely faults together with the error profile values for a footprint method.

Continuing the same example, an automated flow-chart for error profiling (fault diagnosis) can be seen on Figure 2.6. In the flow example the indicators are based upon voltage error, E_v and current error, E_i given by Equations 2.7 and 2.8 using measured and simulated values, respectively. The values are then evaluated if they exceed thresholds given by Equations 2.9 and 2.10 using the standard deviations from Table 2.2.

$$E_v = |V_{dc_meas} - V_{dc_sim}| \quad (2.7)$$

where:

Table 2.2: An example of mean and standard deviation for reference errors(Silvestre, Chouder, and Karatepe, 2013)

	Standard deviation σ	Mean value
$EL_{c_ref}(Wh/Wp/day)$	1.55×10^{-4}	1.8×10^{-4}
$E_{i_ref}(mA)$	108	136
$E_{v_ref}(V)$	4.30	4.65

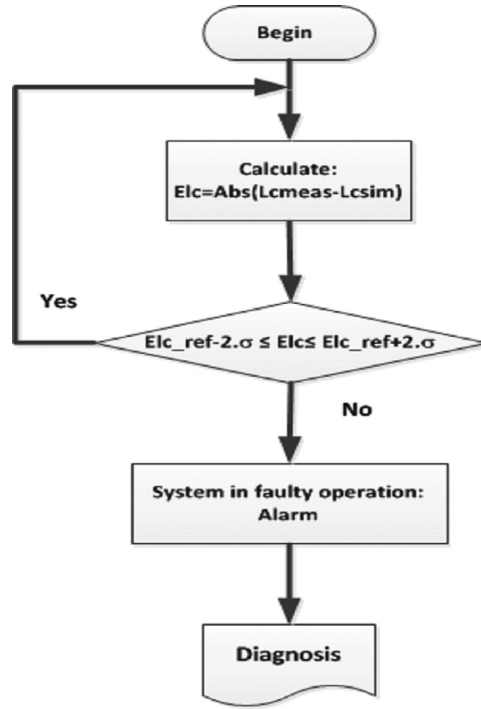


Figure 2.5: Fault detection procedure (Silvestre, Chouder, and Karatepe, 2013)

V_{dc_meas} : is the measured voltage,

V_{dc_sim} : is the simulated voltage

$$E_i = |I_{dc_meas} - I_{dc_sim}| \quad (2.8)$$

where:

I_{dc_meas} : is the measured current,

I_{dc_sim} : is the simulated current

$$E_{v_ref} - 2\sigma(E_v) \leq E_v \leq E_{v_ref} + 2\sigma(E_v) \quad (2.9)$$

$$E_{i_ref} - 2\sigma(E_i) \leq E_i \leq E_{i_ref} + 2\sigma(E_i) \quad (2.10)$$

The flow-chart in Figure 2.6 with the actual voltage and current errors show the most probable faults in the bottom of the diagram. An example flow when neither current or voltage is within their standard deviations, equivalent to *No* followed by *No* in the flowchart (*No*⇒*No*), could be; presence of shade, ground fault, line to line fault or others. This is likely because if neither values are as expected, a significant reduction in measured against simulated values are expected. On the other hand, a flow of *Yes*⇒*Yes* is a false alarm, because both current and voltage are actually within their expected values. The other two paths (*No*⇒*Yes* and *Yes*⇒*No*) lead to several different possible faults which needs *footprint methods* in order to identify the most likely faults.

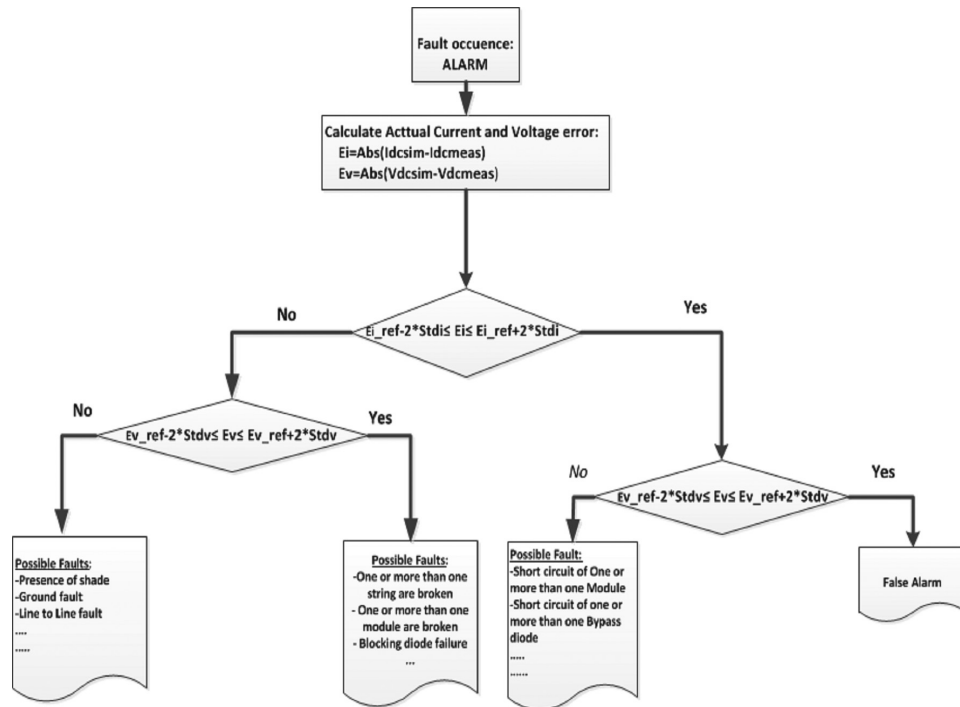


Figure 2.6: Fault diagnosis procedure (Silvestre, Chouder, and Karatepe, 2013)

Footprint method

The footprint method is used to identify the exact cause of failure. Example of difficult failures are shading and inverter failures. The method compares the current fault profile together with its footprints (data over one or more different time periods) against predefined faults and their footprints (Lorenz et al., 2004).

2.2 Soiling measuring

Soiling as a reducing factor in PV have long been studied as mentioned in the introduction, and in recent years, even more so. This is because the focus on field performance has increased, together with the fact that soiling *is recoverable*. To reduce the impact of soiling, panels needs to be cleaned either manually or by precipitation. Nature does provide periodically *chance* of rain according to the local climate, but is thus not a guarantee. In addition it is in the dry periods, with little to no precipitation, soiling increases the most. This is because accumulated soiling effects depend primarily on time since previous rainfall, and are being modeled as a linear degradation (Mejia and Kleissl, 2013).

2.2.1 Wind and humidity

During the normal circumstances of a dry period, the interaction between windspeed and humidity have different results depending on their values as explained in the introduction and on Table 2.3 . The two most important relationships here are; high humidity and low wind speed which increases soiling the most, and its opposite: low humidity and high wind speed which *decreases* soiling. The decrease is likely due to high wind easily moves particles when they are less dense without the moisture. The soiling increase is parallel to increased humidity as the particles get heavier, and thus more affected by gravity to fall down on the panels. Simultaneously the water in these particles form a bonding force to the surface of the PV module, effectively sticking the particles to the panel. Later on when humidity decreases, the cementation process increase particle adhesion so the fallen particles get strongly bonded to the surface (Naeem and Tamizhmani, 2015; Guo et al., 2015).

2.2.2 Precipitation (rainfall)

It has been shown that precipitation can both increase and decrease soiling. Depending on how clean the modules are before the rain, the amount of rain and the composition of the dust – especially its ability to stick to the panels (Naeem and Tamizhmani, 2015; Guo et al., 2015). One study has shown daily rainfall needed to clean panels completely requires 4-5mm (García et al., 2011), while another show only 1mm is needed (Caron and Littmann, 2013). It is hard to establish a definite limit of how much precipitation is needed in order to clean the module. What can be established is heavy rainfall clean solar panels if they are dirty.

2.2.3 Temperature and humidity

It is important to note it is possible to have a partial cleaning event without rain. This happens when temperature and humidity is able to create dew on the frontside of the panels. The dew could in these situations act as a small rain event by moving soil towards the ground. Unless the panels are horizontally inclined, then the dew will not move anything (Caron and

Table 2.3: The proposed relationship between wind and humidity on soiling (Naeem and Tamizhmani, 2015).

	Low WS	High WS
Low RH	Low increased soiling	Decreased soiling
High RH	High increased soiling	Medium increased soiling

Littmann, 2013). None of the modules at the test site are horizontally inclined, and thus dew can form and clean the panels. The Magnus-Tetens Formula given by Eq 2.11 and 2.12 calculates and identify these events by comparing when module temperature (T_{mod}) is lower than the dew temperature (T_d): $T_{\text{mod}} < T_d$.

$$T_d = \frac{b\gamma(T, RH)}{\alpha - \gamma(T, RH)} \quad (2.11)$$

where:

b : 237.7°,

T : Ambient temperature,

RH : Relative humidity,

α : 17.271

$$\gamma(T, RH) = \frac{\alpha T}{b + T} + \ln(RH/100) \quad (2.12)$$

2.2.4 Power reduction

It is widely accepted and proven that soiling decreases the power production of PV. The problem is that PV performance is affected by a range of different parameteres, making it hard to quantify loss due to soiling. In addition, it may not always be a significant daily reduction either. In order to significantly and notably reduce power output on PV, longer periods of soiling is needed. This includes the given test site (Øgaard, 2016). The two main reasons for reduced power production are *reduced insolation* and *change in incident angle*.

Reduced insolation is the most apparent and prominent of the two. The soiling particles covers a percentage area of the panel, reducing the total insolation the panel receives, and thus its production (Ramli et al., 2016).

Change in incident angle is due to the soiling particles ability to act as a intermediate layer between the air and surface of the panels, thus changing the incident angle of the light. PV panels are produced with best efficiency at perpendicular incident angle. With the change of this angle due to the soiling layer, it reduces the efficiency of the panels (Zorrilla-Casanova et al., 2011).

Assesing performance loss

Yield ratio will be used to evaluate a modules performance, instead of efficiency. A reduction in yield ratio (based upon a defined reference) *could* indicate an erroneous module. It has been defined by Eq. 2.13.

$$Y_R = \frac{P_{\text{measured}}^*/G_{t,\text{measured}}}{P_{\text{ref}}^*/G_{t,\text{ref}}} \quad (2.13)$$

where:

Y_R : Yield ratio,

P_{measured}^* : The measured temperature corrected power output,

$G_{t,\text{measured}}$: Measured global tilted irradiance,

P_{ref}^* : The temperature corrected power output at reference date,

$G_{t,\text{ref}}$: Global tilted irradiance at reference date

For calculating the Y_R , temperature corrected power (P^*) is required and defined by Eq. 2.14:

$$P^* = \frac{P_{\text{MPP}}}{1 + \gamma(T_c - T_{\text{STC}})} \quad (2.14)$$

where:

P_{MPP} : maximum power point from IV-curve,

γ : temperature coefficient from module specification,

T_c : estimated cell temperature,

T_{STC} : temperature at STC conditions (25°C)

And the temperature corrected power requires estimated cell temperature (T_c) given by Eq. 2.15:

$$T_c = T_{\text{mod}} + \frac{I_L}{I_{L0}} \Delta T \quad (2.15)$$

where:

T_{mod} : is the module temperature,

I_L : is the irradiation,

I_{L0} : is the irradiation at STC (1000kW/m²)

Cleanness ratio (CR) has been used in other studies (Øgaard, 2016; Plessis, 2016; Guo et al., 2015) to indicate the soiling level on an unclean moduled against a reference cleaned model that is regularly cleaned. The best reason to do this is for eliminating the variance in efficiency due to irradiation. In the Kalkbult data we see a correlation between increased irradiation towards the end of the year, and a drop in yield ratio. By comparing the modules directly, this relation is eliminated as shown on Eq. 2.16

$$\zeta = \frac{Y_{R_{\text{unclean}}}}{Y_{R_{\text{clean}}}} \quad (2.16)$$

where:

$Y_{R_{\text{unclean}}}$: Yield ratio of the unclean module,

$Y_{R_{\text{clean}}}$: Yield ratio of the clean module

Soiling rate are the variable this thesis wants to predict. Based upon some daily footprints, what is the daily soiling rate. Hence equation 2.17 have been defined to calculate soiling rate S_{rate} . The S_{rate} is the difference in yield ratio from previous day. A positive value means it was a better yield yesterday, which could be due to increased soiling level.

$$S_{\text{rate}} = Y_{R^{i-1}} - Y_{R^i} \quad (2.17)$$

where:

Y_{R^i} : The yield ratio on the current (i_{th}) day,

$Y_{R^{i-1}}$: The previous ($i_{\text{th}} - 1$) day yield ratio

Chapter 3

Data Mining and PV

Data mining is a category within computational science used to detect knowledge in patterns of datasets, hence it is also known as Knowledge Discovery in Dataset (KDD). The purpose of data mining is to explore and discover interesting information in data that are not yet known. Data mining is the general term used for the process from data preprocessing, through knowledge discovery in dataset, to post conclusion and consideration. In order to discover knowledge in datasets, machine learning is an increasingly popular approach; training models and algorithms of computers to statistically discover/recognize patterns in the data previously unknown. It is thus often conflated with data mining, which is the more broad term.

3.1 Data mining in PV

Data mining has received increased popularity within PV monitoring the recent years, with several approaches – all achieving positive and encouraging results:

- Using neuro-fuzzy logic on the IV-curve of a PV-system using the parameters *module temperature*, *global irradiation on the plane*, I_{mpp} , V_{mpp} , I_{sc} and V_{OC} to detect *diode short-circuit fault*, *lower earth fault*, *partial shading condition* and *upper earth fault* with good results (Bonsignore et al., 2014)
- Another study tried to omit environmental variables to detect faults using *total energy*, *hours in service*, *direct current*, *input voltage*, *nominal voltage* and *insulation resistance* to classify state of the system into one of six categories. This approach showed the importance of monitoring the irradiation, and the difficulty of detecting (correct) faults without environmental measurements (Serrano-Luján et al., 2016).
- A *fuzzy logic* approach was used on *temperature*, *humidity*, *dew point*, *wind speed*, and *pressure* to predict rainfall intensity with 68.926% accuracy (Agboola et al., 2013)

- Another attempt used fuzzy logic in order to detect partial shading, increased series resistance and potential induced degradation using light I-V measurements (Spataru et al., 2015).
- Artificial neural network proved more accurate than conventional methods for predicting solar radiation. Sunshine hours and air temperature were the most important inputs for ANN among other, with correlation coefficient of 97.65%. (Yadav and Chandel, 2014).
- And the most similar experiment used regression for analysing feature influence, and artificial neural network to accurately predicting power output (Pulipaka, Mani, and Kumar, 2016). This study showed particle composition is an important factor regarding soiling of PV along with the conclusion that artificial neural network is somewhat better for predicting than Multivariabel regression.

3.2 Approaches

The two primary approaches in data mining are *predictive* and *descriptive* (Kantardzic, 2011).

Descriptive data mining produce new, nontrivial information based on the available data set.

Predictive data mining produces the model of the system described by the given data set.

The descriptive approaches is typically a *classification* problem, like classifying if a image is of a cat or something else. The predictive is as the name say an attempt to predict what the next value is, given previous values. Regardless of approach, both require some core steps; preparation of data for machine learning model into a training set, validation set and testing set, training (and validating) the model, before finally testing (scoring) the model.

1. Data preparation:

- The datasets will be built from the observations and measurements.
- Data pre-processing – extraction from database and processing according to datasets.
- From the dataset, training, validation and testing datasets are built.

It is good practice to create training and testing with unique values. Not sharing values strengthens the models scoring on unseen data. Thus the validation set is often a cross between training and testing sets. The training data is usually the largest portion of the total data.

2. Training step:

- From the training set, the algorithm will modify its weights according to desired output. After each *epoch* (a full iteration over training set), the model is validated against the validation set if available. If a desired scoring is achieved in validation, the model is considered trained. If not, another training epoch is undertaken as long as the maximum number of epochs have not been reached.

3. Testing step:

- This step is necessary to assess the obtained model. The model is better if accuracy is higher. More information in Subsection 3.3.3

3.3 Artificial Neural Network (ANN)

The neural network machine learning approach thinks of the 'task' as a neural network, where different input nodes affect the output with various *weights* (w), using one or more *hidden layer(s)* between the input and output layers. An example of a neural network with one input, hidden and output layer are shown on Figure 3.1.

3.3.1 ANN Architecture

The way a neuron works in a neural network is by summing up all the inputs x multiplied by their weights w together with a possible bias b as shown by Eq. 3.1 and passing the result (net) through an *activation function* f as shown by Eq. 3.2, before the functions result is passed to the next layer.

$$\text{net}_k = \sum_{i=0}^n (x_i w_{ki}) + b_k \quad (3.1)$$

where:

net_k : Is the summation of k 'th neuron,

x_i : Is the i 'th input,

w_{ki} : Is the k 'th neurons i 'th weight,

b_k : Is the k 'th neurons bias value

$$y_k = f(\text{net}_k) \quad (3.2)$$

where:

y_k : Is the neurons output to the next layer,

f : Is the activation function

Figure 3.2 shows an example of a neuron with multiple inputs and a single output, which is typical for the output layer in regressional predicting neural networks when only one value is desired output.

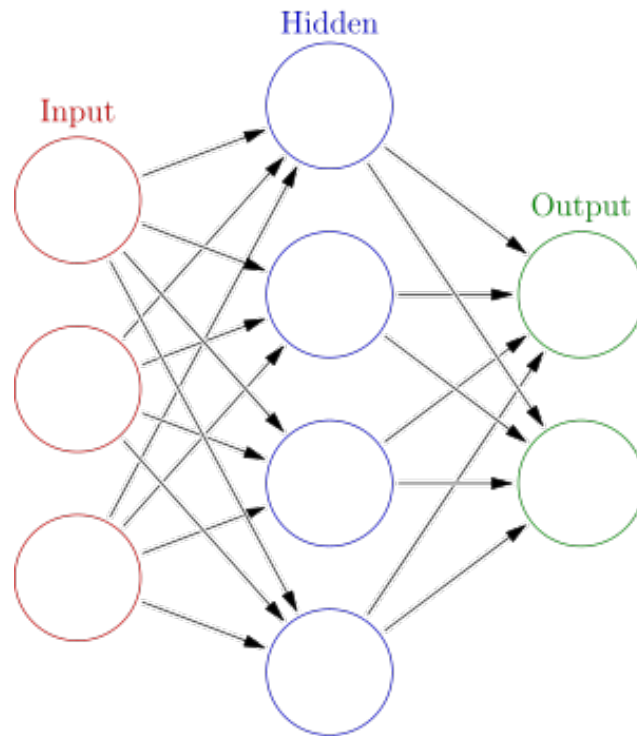


Figure 3.1: An example of a neural network with 1 hidden layer.

Activation functions

There are several different activation functions, as seen in Table 3.1. The most know is the *sigmoid* function, but *tanh* and *hard limit* are also popular depending on both the networks purpose and the layer it is in. Lately, when training deep networks, the *Rectified Linear Units* (RELU) has shown promise (Heaton, 2015). Deep learning is a category within artificial neural network when there are more than 2 hidden layers, hence *deep*.

The most important difference to note is the non-linear activation functions positive and negative limits, and their outputs thereby constrained to $\{0, \dots, 1\}$ or $\{-1, \dots, 1\}$ in contrast to linear functions. This means that regardless of input, a non-linear activation function would limit its output

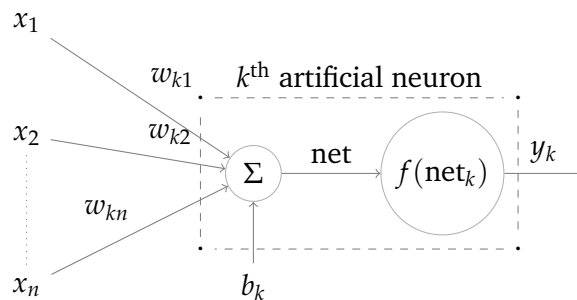


Figure 3.2: Model of an artificial neuron.

to the applicant range. While a linear function are more likely to pass on the actual value, depending on the function properties.

Bias nodes

The role of a bias node is only to shift the output, especially when the input value is 0. The best way to describe this is using illustrations of sigmoid weight function $f(x, w, b) = \frac{1}{1+e^{-(wx+b)}}$ (Heaton, 2015). First only the weights are adjusted and calculated, and it is clear only the slope is affected on 3.3a. Next the weight is constant but the bias is changed, and now gives different values for $x = 0$, as shown in 3.3.

Typical architectures

There are two typical architectures for ANNs; *feedforward* and *recurrent* interconnected neural network. The feedforward architecture consists of data being *forwarded* through the neurons, with no connections or loops backwards. Thus there are no connections between nodes in the same layer, or to previous layers. When there are a feedback link (usually with a delay) forming a loop, the network is recurrent. Multilayer feedforward with backpropagation-learning mechanism is probably used in over 90% of the industrial and commercial applications (Kantardzic, 2011). A simple and good illustration of why multilayers is common can be shown by the *exclusive-OR* (XOR) problem. A binary XOR function with two binary inputs $f(x_1, x_2)$ is classified to class 0 if both inputs are equal, else class 1. This can be depicted as classes shown on Figure 3.4. The dotted lines on the same figure proves there are no possible ways to create *one* straight line to separate the classes. This example show single layer ANNs are convenient for simple problems based on linear models. However, most real world problems are highly nonlinear, and thus require multilayered ANNs.

3.3.2 Learning

The purpose of machine learning is of course for an algorithm to *learn* the environment it is trying to represent. For this to happen, a fundamental difference between classical information processing and ANN must be explained. The classically approach is to gather information, and then build a model to represent the gathered information. However, in machine learning the model is built *by* the data *given* to it.

Teaching

The most agreed upon approach to teach an ANN is to adjust weights for the neurons by minimizing an error-cost function. This can be explained by using the earlier Figure 3.2 having n inputs with corresponding weights, and one output (Kantardzic, 2011). The bias is never changed. Output of this neuron given input $X(m)$ is denoted by $y(m)$:

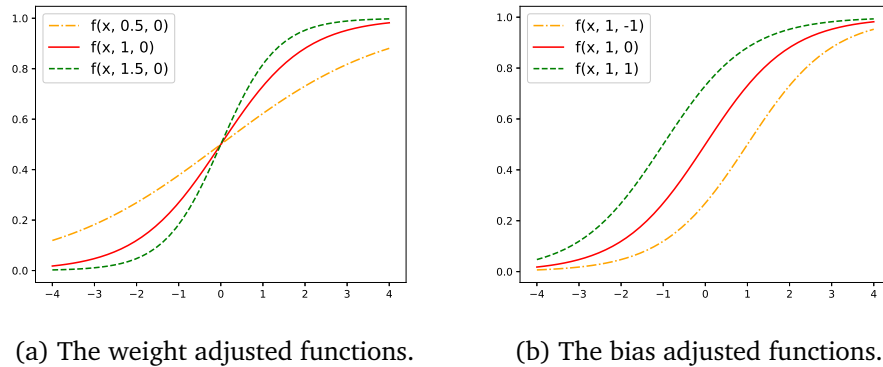


Figure 3.3: The weight and bias adjusted functions, showing weight adjustment controls steepness and bias adjustments controls position of the function.

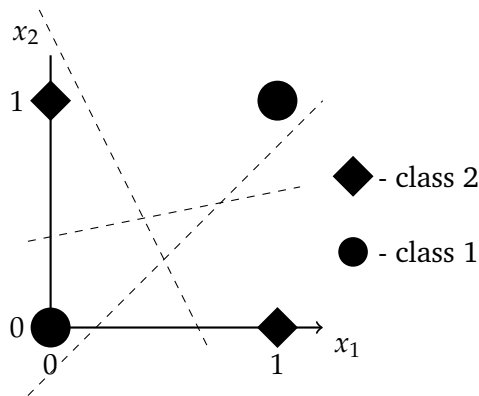


Figure 3.4: Illustration of the XOR problem (Kantardzic, 2011).

$$y = f \left(\sum_{i=1}^n x_i w_i \right)$$

When given the input, an error in the estimation is by definition

$$e(m) = d(m) - y(m)$$

With the error value, it is now possible to make corrective adjustments to the weights of the neuron. These adjustments are designed to make the estimated outcome $y(m)$ come closer to the desired outcome $d(m)$. This objective is achieved by minimizing error-cost function $e(m)$ by using gradient descent. Consider Figure 3.5 showing an example of an error-cost $e(w)$ for all w , and a single gradient at given point. What the neural network is trying to achieve is finding the global minimum of this function. This is done by calculating the gradient, and increasing the weight if the gradient is negative or decreasing the weight if it is positive. If the gradient is zero, it means a minima has been found – although this could be *local* and not the desired *global* minima.

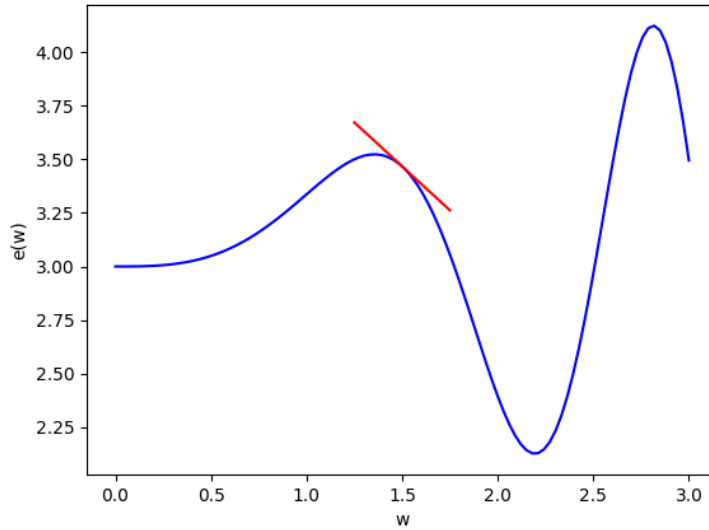


Figure 3.5: Gradient (in red) of a weight and its error-function (blue).

The steps involved of calculating the gradient for each weight can be summarized to (Heaton, 2015):

- Calculate the error, based on the ideal of the training set.
- Calculate the node (neuron) delta for the output neurons.
- Calculate the node delta for the interior neurons.
- Calculate individual gradients.

Error estimation

The error of a regression neural network is calculated using the common mean square error (MSE) function represented by Eq. 3.3.

$$E_m = \frac{1}{m} \sum_{k=1}^m (t_k - y_k)^2 \quad (3.3)$$

where:

t_k : target value,

y_k : neural network output

Each weight is updated by an amount proportional to the partial derivative (gradient) of the error function. The partial derivative with respect to w_{ik} for MSE is:

$$\frac{\delta E_m}{\delta w_{ik}} = \sigma(y_k - t_k) y_k (1 - y_k) z_i$$

When the output node delta has been calculated, the remaining node deltas for the interior nodes can be calculated by Eq. 3.4:

$$\delta_i = \phi'_i \sum_k w_{ki} \delta_k \quad (3.4)$$

where:

ϕ'_i : the derivative of the activation function

Earlier the activation functions was depicted in Table 3.1, for the weight update their derivatives are needed and the most common are shown on Table 3.2.

Backpropagation

Backpropagation in order to train the weights are done in one out of two ways; batch or online. Usually there are a large amount of training elements, and thus the gradients must be calculated an equal amount of times for each training element (and for each weight). This process is called *online training*, because the weights are updated for each training element, and is not done until all elements have been iterated over – in which an *epoch* has been completed. The other method, *batch training* does not update the weights for each element, but after a batch of elements have been iterated over. Instead it sums up the gradients before updating. The most used and original propagation is the online training, as it is logical to update the weights for each error calculated, instead of waiting. However, if training speed is desired - not updating the weights for each element can be advantageous.

An extended functionality is random selection of batches for training the network, before updating (either online or after batch). This continues until the validation of the model reaches a desired level. Random training sets will usually converge faster than looping through the entire training set for each iteration (Heaton, 2015).

The explained way of training by gradient descent is known as *Stochastic Gradient Descent* (SGD), which can be used in either online or batch fashion. It originated in 1960 and is today known as *ADALINE* (**A**daptive **L**inear **N**euron).

Backpropagation weight update is done by Eq. 3.5. Two new important variables here are *learning rate* and *momentum*. The learning rate is how hard the learning for each update should be, and are usually a value around 0.02. Momentum is used to help the algorithm skip through a local minima, by pushing with the previous weight update. Both of these values are important with regard to performance of the neural network.

$$\Delta w_{(t)} = -\eta \frac{\delta E}{\delta w_{(t)} + \alpha \Delta w_{(t-1)}} \quad (3.5)$$

where:

η : the learning rate constant,

α : the momentum constant

Nesterov momentum was invented by Yu Nesterov in 1983 and updated in 2003. It is defined by Eq. 3.6. In the equation, the current iteration is given by t with the previous iteration being $t - 1$. This partial weight update is called n and it starts out at 0. As before, the partial derivative is the gradient of the error function at the current weight. In addition, Eq. 3.7 shows Nesterov update which replaces the standard backpropagation weight update in Eq. 3.5. This update equation is calculated as an amplification of the partial weight change from the momentum. Using Nesterov momentum with SGD is one of the most effective training algorithms for deep learning (Heaton, 2015).

$$n_0 = 0, n_t = \alpha n_{t-1} + \eta \frac{\delta E}{\delta w_t} \quad (3.6)$$

where:

η : is the learning rate constant,

α : is the momentum constant

$$\Delta w_t = \alpha n_{t-1} - (1 + \alpha)n_t \quad (3.7)$$

where:

α : is the momentum constant

Other update techniques

The above explained update and training is the core idea behind machine learning in ANN. In the years since machine learning's birth, several other techniques have been proposed.

RMSProp (for **Root Mean Square Propagation**) is an extended variant of SGD with Nesterov momentum and the idea to divide learning by a running average of the previous recent gradients. The running average is given by:

$$v(w, t) = \gamma v(w, t - 1) + (1 - \gamma)(\nabla Q_i(w))^2$$

and the weight is updated by:

$$w = w - \frac{\eta}{\sqrt{v(w, t)}} \nabla Q_i(w)$$

AdaGrad is termed from adaptive gradient algorithm, and seeks to update by taking notice of the data it is updating from (Duchi, Hazan, and Singer, 2011). This is done by modifying the general learning rate to do larger updates for frequent inputs, and smaller updates for infrequent inputs. The biggest benefit AdaGrad thus gives is elimination of the need to tune learning rate.

Adam (short for **Adaptive Moment Estimation**) is a combination of *RMSProp* and *AdaGrad* functions. The extension is the use of running averages from both the gradients and second moments of the gradients. It was concluded to work well with high-dimensional parameter spaces (Kingma and Ba, 2014). The parameter update is given by Eq. 3.8:

$$w^{(t+1)} \leftarrow w^{(t)} - \eta \frac{\hat{m}_w}{\sqrt{\hat{v}_w + \epsilon}} \quad (3.8)$$

where:

ϵ : is a small number used to prevent division by 0,

β_1 : are forgetting factor for gradients,

β_2 : are forgetting factor for second moment of gradients

$$\begin{aligned} \hat{m}_w &= \frac{m_w^{(t+1)}}{1 - \beta_1^t} \\ \hat{v}_w &= \frac{v_w^{(t+1)}}{1 - \beta_2^t} \\ m^{(t+1)} &= \beta_1 m_w^{(t)} + (1 - \beta_1) \nabla_w L^{(t)} \\ v^{(t+1)} &= \beta_2 v_w^{(t)} + (1 - \beta_2) \left(\nabla_w L^{(t)} \right)^2 \end{aligned}$$

Node count in one fully connected ANN layer is proposed calculated by Eq. 3.9 (Yadav and Chandel, 2017), before assessing ± 5 nodes what is most likely the best value.

$$H_n = \frac{i_n + o_n}{2} + \sqrt{s_n} \quad (3.9)$$

where:

H_n : the number of hidden neurons,

i_n : the number of input nodes,

o_n : the number of output nodes,

s_n : numer of data-samples

Normalizing

It is common practice to normalize data within machine learning. The two standard procedures are *range normalizing* and *z-score normalization*. The first transforms the data values into a predefined range, usually -1 to 1 or 0 to 1 . Z-score normalization transforms the data into a balanced distribution defined by the data it is transforming using mean and standard deviation. This gives a score from mean in units of standard deviation. The range normalization is thus better for scaling all the input features into the same range, and is given by Eq. 3.10 for downscaling and Eq. 3.11 for upscaling.

$$x_N = \frac{(x - d_L)(n_H - n_L)}{d_H - d_L} \quad (3.10)$$

where:

x_N : Is the normalized value,

x : Is the value to be normalized,

d_L : Is the lowest possible value for that key,

d_H : Is the highest possible value for that key,

n_L : Is lowest normalized range (default -1),

n_H : Is highest normalized range (default 1)

$$x = \frac{x_N(d_L - d_H) - (n_H d_L) + (d_H n_L)}{n_L - n_H} \quad (3.11)$$

where:

x_N : Is the normalized value,

x : Is the original value,

d_L : Is the lowest possible value for that key,

d_H : Is the highest possible value for that key,

n_L : Is lowest normalized range (default -1),

n_H : Is highest normalized range (default 1)

3.3.3 Model scoring and error estimation

For some statistics on how well a model function maps inputs to outputs, several approaches is used.

Mean squared error

Mean squared error is a common way of assessing the quality a mapping of inputs to outputs. It is defined by Eq. 3.12:

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^n (\hat{Y}_i - Y_i)^2 \quad (3.12)$$

where:

\hat{Y} : Vector of n predictions,

Y : Vector of n observed values

Root mean squared error (RMSE), also known as Root Mean Square Deviation (RMSD) is the root of MSE. It is a good indication for accuracy when comparing models of same variable. It is given by Eq. 3.13

$$\text{RMSE} = \sqrt{\text{MSE}} = \sqrt{\frac{\sum_{i=1}^n (\hat{Y}_i - Y_i)^2}{n}} \quad (3.13)$$

Mean absolute percentage error

The **mean absolute percentage error** is given by Eq. 3.14 and indicates the accuracy of estimated output.

$$\text{MAPE} = \left(\frac{1}{n} \sum_{i=1}^n \left| \frac{\hat{Y}_i - Y_i}{Y_i} \right| \right) \times 100 \quad (3.14)$$

where:

\hat{Y} : Vector of n predictions,

Y : Vector of n observed values

R² scoring

The R^2 score, *coefficient of determination*, is a measure of how well future samples are likely to be predicted by a predicting model. Values indicating prediction are in range $(0, 1]$, where 1.0 is the best possible score and 0.0 is a overfit model always giving the expected value of Y_i regardless of input. The score is given by Eq. 3.15

$$R^2(\hat{Y}, Y) = 1 - \frac{\sum_{i=1}^n (\hat{Y}_i - Y_i)^2}{\sum_{i=1}^n (\hat{Y}_i - \bar{Y})^2} \quad (3.15)$$

where:

\bar{Y} : is $\frac{1}{n} \sum_{i=1}^n Y_i$,

the mean of the observed data

Convolutional Neural Network (CNN)

CNN is a common technique used to identify features in images. The layers most associated with neural networks are dense layers, where each node from one layer is fully connected to all nodes in the next layer. CNN utilizes the spatial of input to detect some frequent behaviour of a less connected neural network. The two layers used for this is the *convolutional* layer and a *pooling* layer. The convolution layer moves a rectangular window over the n -dimensional data and dot-multiplies the values for a new value, which represents a new dimensional feature-map. The pooling layer is simply a downsizing layer, usually maxpool, selecting the highest value within its window. A representation of the most known CNN, LeNet-5, is shown on Figure 3.6, where the convolution layers are exemplified with a window calculating a new value for the next layer, and the pooling table downsizes its received input.

Fuzzy logic

Fuzzy logic is a machine learning approach where the data are given linguistic annotations to represent degrees, ie. *low* wind speed or *high* humidity. These are defined by *membership-functions* μ to determine in what range the value is within. An example can be seen in Figure 3.7.

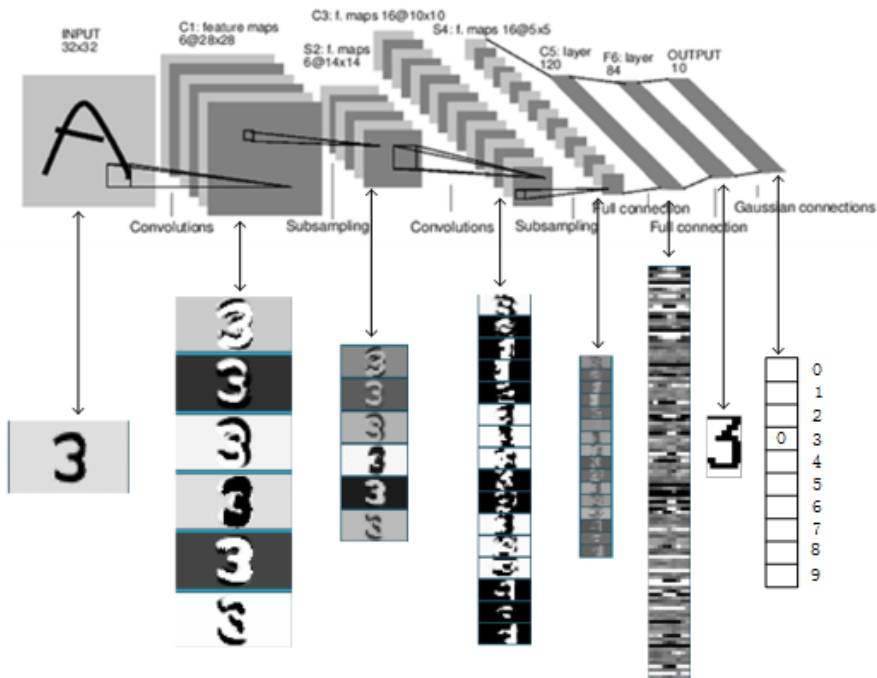


Figure 3.6: A known CNN called LeNet5 (LeCun et al., 1998), showing layer types (starting with convolution, then subsampling, convolution again etc.) and what the layer sees from the input throughout the network.

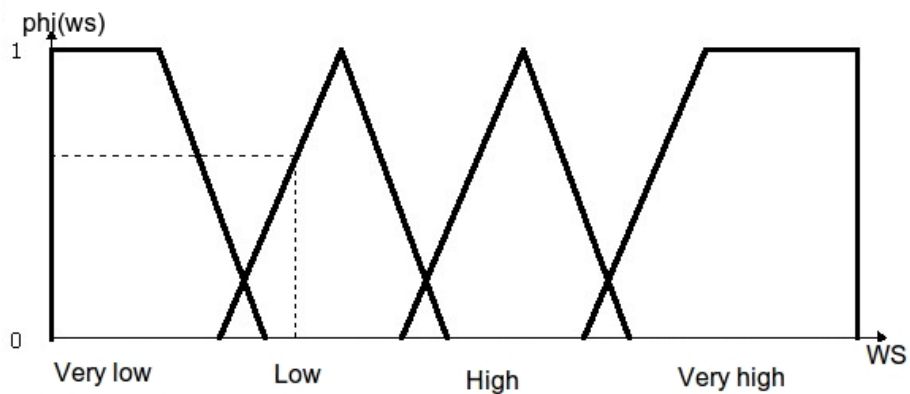
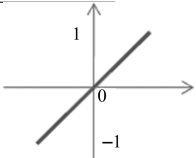
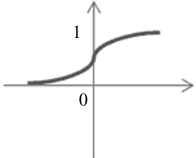
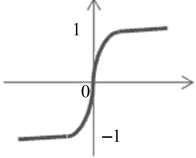
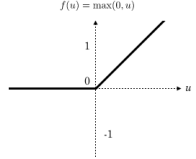


Figure 3.7: A quick example of the membership functions and their triangles.

Table 3.1: A selection of neuron's activation functions.

Activation Function	Input/Output relation	Graph
Hard limit	$y = \begin{cases} 1 & \text{if net} \geq 0 \\ 0 & \text{if net} < 0 \end{cases}$	
Symmtrical hard limit	$y = \begin{cases} 1 & \text{if net} \geq 0 \\ -1 & \text{if net} < 0 \end{cases}$	
Linear	$y = \text{net}$	
Saturating linear	$y = \begin{cases} 1 & \text{if net} > 1 \\ \text{net} & \text{if } 0 \leq \text{net} \leq 1 \\ 0 & \text{if net} < 0 \end{cases}$	
Symmetric saturating linear	$y = \begin{cases} 1 & \text{if net} > 1 \\ \text{net} & \text{if } -1 \leq \text{net} \leq 1 \\ 0 & \text{if net} < -1 \end{cases}$	
Log-sigmoid (sigmoid)	$y = \frac{1}{1+e^{-\text{net}}}$	
Hyperbolic tangent sigmoid (tanh)	$y = \frac{e^{\text{net}} - e^{-\text{net}}}{e^{\text{net}} + e^{-\text{net}}}$	
Reactified Linear Unit (RELU)	$y = \begin{cases} \text{net} & \text{if net} > 0 \\ 0 & \text{if net} \leq 0 \end{cases}$	

Table 3.2: A selection of neuron's activation functions derivatives.

Activation Function	Input/Output relation	Graph
Linear	$\phi'(x) = 1$	
Log-sigmoid (sigmoid)	$\phi'(x) = \phi(x)(1 - \phi(x))$	
Hyperbolic tangent sigmoid (tanh)	$\phi'(x) = 1.0 - \phi^2(x)$	
Rectified Linear Unit (RELU)	$\frac{\delta y}{\delta x} \phi(x) = \begin{cases} 1 & x > 0 \\ 0 & x \leq 0 \end{cases}$	

Chapter 4

Data collection

The three sections of this chapter covers information about the data, a software created to monitor and filter PV data, which is partly used before the last section about data preparation and what ends up being used.

4.1 About the data

The testing site is located at Kalkbult, Northern Cape, South-Africa (latitude: -30.2, longitude: 24.1). It consists of 16 255W Virtus II Modules from Rene Sola (polycrystalline silicon) and 8 100W First Solar Cadmium-Telluride (CdTe) modules. Half of the CdTe are FS-4100A with anti-reflective coating, the other half are FS-4100 without coating, both from First Solar. All modules have tilt angle of 30° towards north, and are grouped in rows of eight, as seen in Figure 4.1 with the exception of single-axis tracking (SAT) modules which are 4 modules pairwise tracking west to east.

Table 4.1 provides an overview of eight different cleaning strategies applied on the modules shown on Figure 4.2. The figure also shows four polycrystalline modules installed on single-axis sun-tracker, with anti-soiling coating applied diagonally on module.. From both of these the important note is that modules on strategy A, B, C and D are *not* cleaned, while modules on strategy E, F, G and H are cleaned every second week. For the period in question, C and D are equivalent to A and B respectively.

This is chosen to determine the best cleaning strategy for lager scale PV plant, and have been studied in other thesies but will not be in the scope of this thesis. They are included for the reader to know. The site also has its own weather station for local enviromental measurements.

Previous experiments within the project have indicated that the anti-soiling coating is in fact working against its purpose (Plessis, 2016). Thus modules with coating may provide the best data to work with.

4.1.1 Module data

All modules store one row of data into their respective SQL-table every tenth minute, where the row consists of the data shown in Table 4.2.

Table 4.1: Cleaning strategy for both polycrystalline and thinfilm modules (Weldemariam, 2016). Strategy C and D can be represented by A and B respectively.

Strategy	Treatment	Duration
A	Cleaned with distilled water and treated with anti-soiling(AS) product.	Left for long term. After 12-18 months, apply the AS product again.
B	Cleaned with distilled water only.	Left for long term (indefinitely)
C (A)	Cleaned with distilled water and treated with AS product. Then afterward only dry-cleaned for the remained of the testing phase.	Dry clean after a long-term exposure (6 months). After 12-18 months apply the AS product again.
D (B)	Cleaned with distilled water. Then afterward only dry cleaned for the remained of the testing phase.	Dry clean after a long-term exposure (6 months).
E	Cleaned with distilled water and treated with the AS product. Then cleaned with distilled water only.	Cleaned again with distilled water after short-term exposure (once every two weeks). After 12-18 months, apply the AS product again.
F	Cleaned with distilled water only.	Cleaned again with distilled water on a regular basis (once every two weeks).
G	Cleaned with distilled water and treated with the AS product. Then afterward only dry cleaned for the remained of the testing phase.	Dry-cleaned again on a regular basis (once every two weeks). After 12-18 months apply the AS product again.
H	Cleaned with distilled water. Then afterward only dry-cleaned for the remainder of the testing phase.	Dry.-cleaned again on a regular basis (once every two weeks).

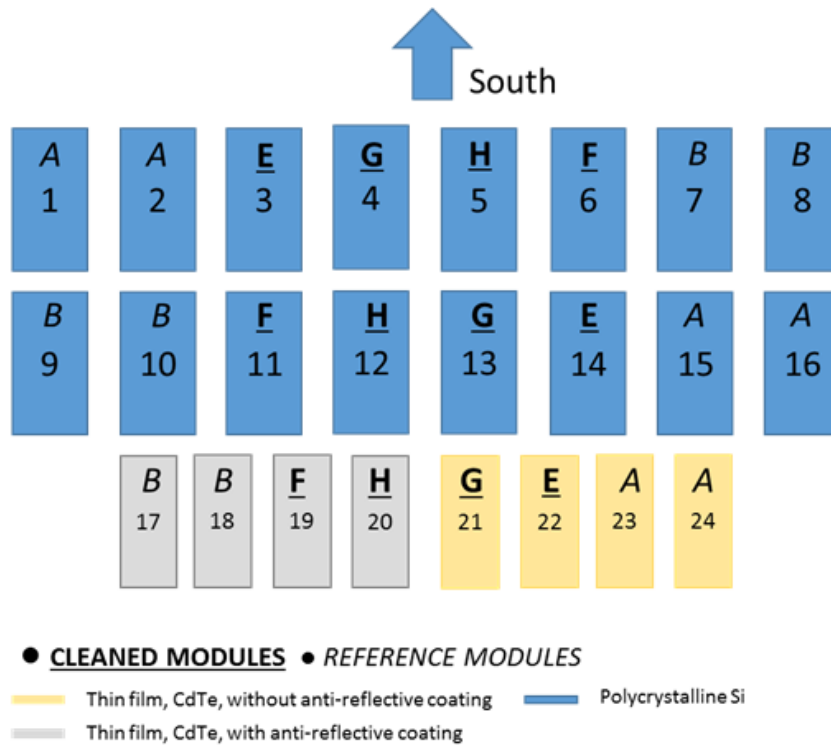


Figure 4.2: Cleaning strategy of the stationary PV modules at the test site (Øgaard, 2016)

It also shows the thinfilm rows are slightly different from the polycrystalline, because thinfilm modules are connected pairwise to the same active load, and thus needs separate identifiers for temperature and current. The 20 IV-pairs are measured within two seconds for every module with the help of device called ActiveLoad (Ndapuka, 2015).

4.1.2 Weather data

The on-site weather station is a Met StationOne provided by Met One Instruments, and measures wind speed including direction, ambient temperature, pressure, relative humidity, and amount of precipitation. How these values are stored can be seen in Table 4.3. The table is compressed with `{prefix(es)}``{suffix(es)}` in order shorten the list of all 80 values. Other studies indicate the most important variables with regards to soiling in the table are wind speed and humidity, and in a small way temperature because of dew formation.

Table 4.2: Attributes for one row of data from the modules every ten minutes.

(a) Shared attributes	(b) Polycrystalline	(c) Thinfilm
ID	TEMPERATURE	TEMPERATURE1
FIFOSPACE	CURRENT0	TEMPERATURE2
YEAR	CURRENT1	CURRENT10
MONTH	CURRENT2	CURRENT20
DATE	CURRENT(3,...,18)	CURRENT11
HOUR	CURRENT19	CURRENT21
MINUTE		CURRENT12
SECOND		CURRENT22
VOLTAGE0		CURRENT1(3,...,18)
VOLTAGE1		CURRENT2(3,...,18)
VOLTAGE2		CURRENT119
VOLTAGE(3,...,18)		CURRENT219
VOLTAGE19		

Table 4.3: Attributes for one row of data from the weather station every minute.

ID, FIFOSPACE, YEAR, MONTH, DATE, HOUR, MINUTE, SECOND, {Avg_, Max_, Min_}{Wind_Spd, Temp, Humidity, Pres, GHI1, GHI2, GHI3, Temp_Pyranol, Temp_Pyranol2, Temp_Pyranol3, DNI, Temp_Pyrhelio}, Interval_Rain, Avg_V_WS, Total_Rain_Today, {WinDir_}{N, NNE, NE, ENE, E, ESE, SE, SSE, S, SSW, SW, WSW, W, WNW, NW, NNW}, Saxis_Angle1, Saxis1_Mode, Saxis_Angle2, Saxis2_Mode, Saxis_Angle3, Saxis3_Mode, Daxis_Azimuth, Daxis_Beta, Failed_WS, Failed_Pyranol1, Failed_Pyranol2, Failed_Pyranol3, Failed_Pyrhelio, Failed_Saxis1, Failed_Saxis2, Failed_Saxis3, Failed_Daxis

4.2 Monitoring and Filtering of PV data

In the beginning of the thesis, a system to gather and monitor data from the Kalkbult plant, or any SQL server, was implemented. For this system some requirements specification were set up – and the 'Database'-module created were used and extended for data collection to the neural network Python-modules. The only related part of this section to the ANN part of the thesis is subsection 4.2.7, where some attributes of the 'Database'-module are described.

4.2.1 Requirements Specification

A software requirements specification (SRS) is created to ensure the software works as desired by the end-users and with a quality required by the developer. The following subsections are inspired by (Japenga, 2016)

4.2.2 Functionality

The software must be able to

- be used with command based user interface through terminal.
- connect to external database.
- be updated with new filters/analyses.
- have existing filters/analyses edited.
- delete existing filters/analyses.
- run filter/analysis on a database.
- have filter/analysis that can warn user if a (un)desired condition is met.
- show textual outcome on terminal or to screen.
- store textual outcome to files.
- keep records of data in its own database.
- copy an existing external database into its database.
- update its database from the same database it copied.
- activate a monitoring of external database, which includes:
 - Continuous update of local database.
 - Enable or disable predefined filters/analyses on the continuous data in combination with local database.
 - Warn user if filter/analysis with warning is enabled and the conditions are met.
- disable an activated monitoring.

The software should be able to

- be used through a graphical user interface.
- show graphical outcome of filters/analyses on screen as graphs.
- store graphical outcome as graphs to files.
- backup the database, in order to manipulate database and revert unwanted changes.
- connect to several databases.
- do a cross-filtering and updating on several databases.

4.2.3 Interfaces

Interfaces are how the software will interact with users and/or applications. This includes external interfaces like the external database. The users and its interface is described on Table 4.4, where the most important note are filters and analyses that can be written in either Python, MatLab or SQL. The rest is more descriptive *how* the functional requirements will be implemented.

4.2.4 Performance

In order to optimize speed, the software should attempt to utilize low-level programming if a filter/analysis is taking a long time. To encourage this, a timing implementation should be implemented to warn user of low quality filters. On Table 4.5 there are an overview of the most important time schedules or limits the software must be aware of.

4.2.5 Attributes

Concerns that needs addressing:

- The user must be allowed by the external database to both read and also store read content on local machine.
- Should the data be encrypted on local end-point to secure it from external attacks.
- All database queries are to be implemented in such a fashion that SQL-injection cannot occur.

4.2.6 Design constraints

- When error occurs, *all* columns on Kalkbult database are 0.0 values, including time and date specific columns, with row-ID as the sole exception.

Table 4.4: An overview of the software users (entities) and their interfaces

Entity	Interaction
Users	Through a terminal with string input commands, or if possible through a graphical user interface (GUI), with buttons and possibility of enabling advanced options.
Administrators	Initially all users will be administrators (access to all features). This might change later during development, and thus needs some level of generalization to easier implement <i>restrictions</i> at a later date.
MatLab	The Python language enables MatLab scripts to be executed through MatLabPython (<i>MATLAB Engine API for Python 2016</i>). Many users would desire such a possibility when writing filters and algorithms on data.
Filters/Analyses	The filters can be written in either MatLab, Python or SQL, whichever fits the users experience in the language and preferably its task.
Database	The softwares database will be implemented using SQLite and SQLAlchemy. SQLite does not need a local server, and can be executed on a database file and in memory. SQLAlchemy is chosen for its broad features to integrate several SQL database types, continuous update and well documented API (Bayer, 2016).

- The local database might not have consistent data (ie. the 0.0 values on all columns except ID), it needs to be edited. Since the invalid value rows are stored on the external database, only without values, the local database can update *those* rows with interpolated values locally.
- In order to reduce risk of loosing database integrity, the external database will always be read-only. The local database will be read-only unless software is told otherwise. Some filters and custom SQL statements may override this.
- Software will be used on several machines, thus it requires portability and the possibility to be run on different operating systems (OS), specifically Windows, Linux and MacOS.
- The software should be modulable, to increase usability on later stages and different systems, not just on the project it is designed for initially. However, testing on other systems are of no priority.

Table 4.5: The schedule of some time variables the software needs to design around

Description of action	Time variable/description
Attempts to connect to external database before timeout	5
Time to wait before updating local database during monitoring	30 minutes
Designated weekday start and end	Monday-Sunday
Kalkbult test plant specifics	
Update of weather data on table 'MasterController25'	every minute
Update of modules system data on tables 'Polycrystalline{1,...,16}', 'ThinFilm{17,...,20}' and 'STracker{21,...,24}'	every 10 minutes

4.2.7 Prototyping

All entries in the software *must* list will be a part of the prototype. The prototype will be used for demonstration on the Kalkbult database in order to prove purpose of software.

Diagrams and charts of design

On Figure 4.3 is a chart of the main menu, and its current submenus that are available. The temporary name of software is *Faultication* and has the three menus; Monitor, Filters and analyses and Manage Database. The *Monitor* menu is used to start and stop continuous monitor of external database to update local database, and enable or disable filters/analyses that are to be run during monitoring. The *Filters and analyses* menu is used to manage filters/analyses by creating new, update/edit existing, deleting, or applying them to the local database. Filters are either used to extract simple information from the database, or edit it in order to increase consistency. Analyses are used to analyze the data from filters (or extracted themselves) using scientific algorithms. The last menu, *Manage database*, is used to work with the local database more directly. In this menu the external database is set up for monitoring/updating local database. It will also include the possibility to write custom SQL statements for those with experience and knowledge in the field. For others there will be a view option with predefined SQL statements to show the current state of the databases

(simple extraction filters). This is also the menu that has the option to do a single update of the local database from the external databases.

To achieve the desired menu, a entity relation diagram is shown on Figure 4.4. This shows that Faultication needs one local database, one analysis manager and one filter manager to function, and an optional monitor for monitoring.

On Figure 4.5 is a flow diagram on how the software will behave while monitoring. The software must have the possibility to run several filters and/or analyses in a specified order on the updated values. This is reflected by fetching a filter/analysis while there are some assigned that has not yet been applied to the updated values. Also, if a filter fails, it must have a procedure on what to do when the filter does not return OK. This could be a range of different things, including sending an e-mail with warning, update data with new values, or nothing at all. The filter must also have a continue/stop attribute in order for the monitoring to know *how* to proceed after a filter have failed. A stop in this sense is *stop filter/analysis fetching*, and *not* stop monitoring. Later updates might include the data necessary to fix the previous failure, and thus monitoring is only stopped when user disables it. Analyses on the other hand do not require a continue/stop attribute, as they *expect* the data to be correct when applied, and thus cannot fail as they only acquire/compute results.

Database

The Database-module contains a 'Database' class which connects to the Kalkbult SQL server by default through the 'sqlalchemy-module'. It also contains its own 'command-loop' in order to do some predefined executions or a 'SELECT' statement if the user desire. Most of the system requirements are ensured in this class, like updating database (or one table), applying filters, extract predefined data collections and more. Some of the predifined functions for collecting data from a 'Database'-instance are:

- 'select(query, external=False)'
This function executes the 'query' on the local database, and returns the result. If 'external=True', the query is executed on the external connection, if any.
- 'select_from_until(table, start, end)'
This function collects all data from table, within the range [start,end). Notice how it is until not including the end. The 'start' and 'end' variables may either be 'datetime' or integers. If they are 'datetime', the database tries to find the row containing the desired datetime. If they are integers, they are handled as 'ID"s, and the row with same value are chosen as 'start' or 'end'. The returned value is a 'list' where each entry is a returned 'sqlalchemy.RowProxy'. Those classes operates similar to a read-only 'OrderedDict' Python-class.
- 'select_peak(date)'
This function returns an ordered list (biggest to smallest) with

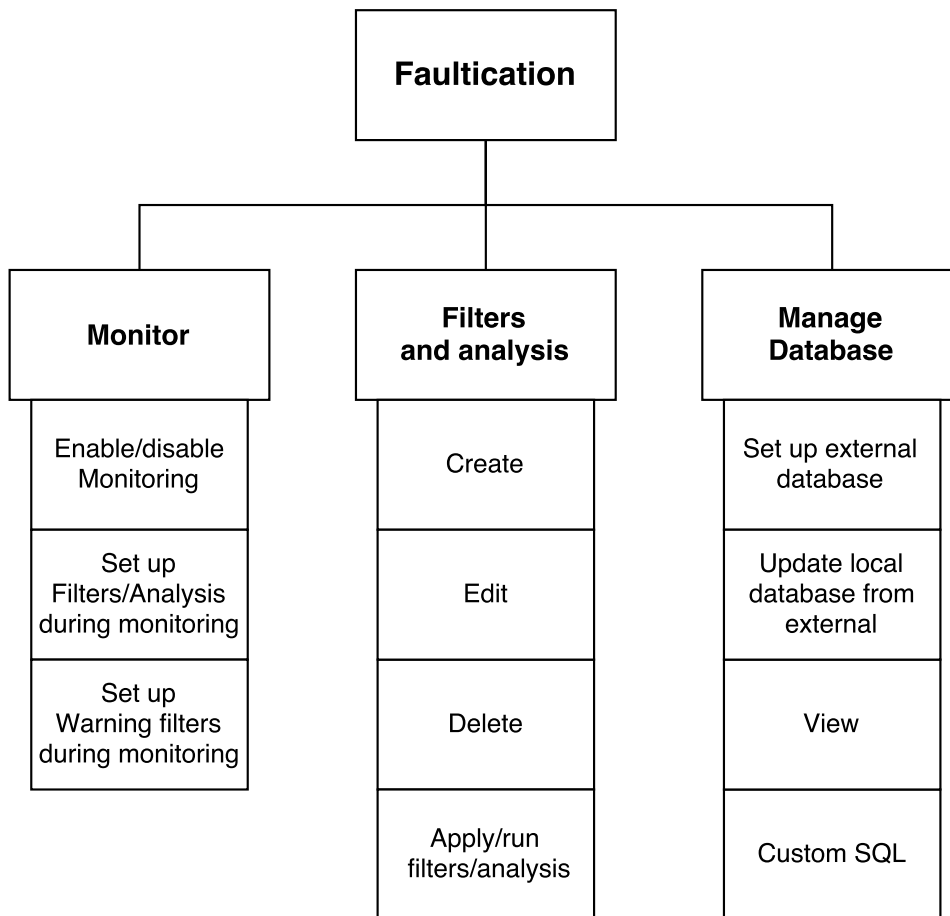


Figure 4.3: An overview of the main menu and their submenus

irradiation values between 800 – 1200 from key 'Avg_GHI1' in the 'MasterController25' table for the date in 'date' variable. The 'date' value should be either a 'date' or 'datetime' instance.

- `'select_production_and_weather(modules, dates)'`
This function selects production together with irradiation and ambient temperature for the given 'modules' from the given 'dates' (which must be tuples of '(start, end)' within that day). The returned value is a nested 'OrderedDict' with weather and module data for each timerange in 'dates'. The first level is a 'startdate: OrderedDict', where 'startdate' is the key, and the 'OrderedDict' contains a list for each key $\{I_{avg}, T_{avg}\}$ and each of the 'modules'.

Analysis and Filter manager

For the prototype, three filters were subclassed from the 'Superfilter'-module to show examples of implementation. For each filter, they are applied to either one row or an entire rowset. The superfilter ensures all filters have some default behaviour when applying filters like saving last

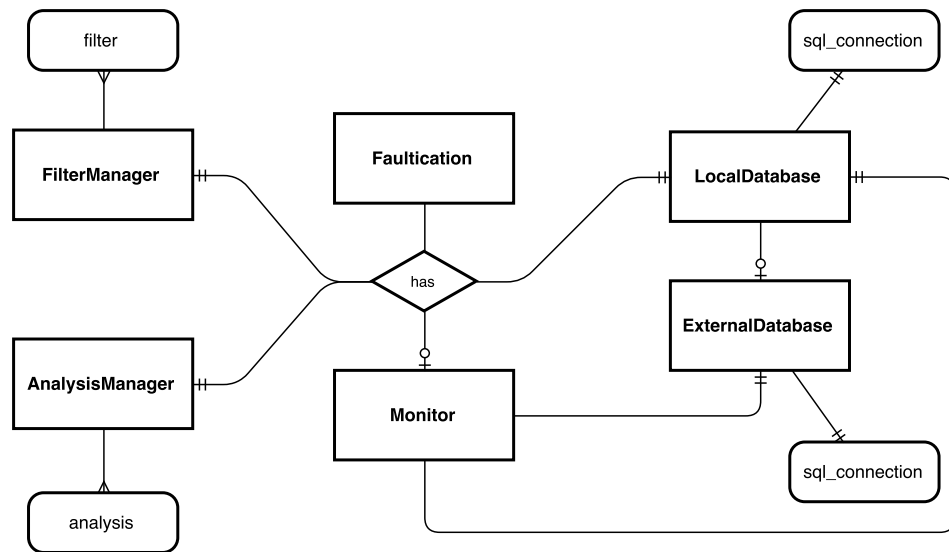


Figure 4.4: An entity relation diagram of entities in Faultication

applied row, building error messages or resetting filters. The subclassed filters created are:

Nullvalue Filter which checks if all values are 0's in a row, except ID and FIFOSPACE columns. This is because those columns requires unique keys, and a row is always inserted into the used SQL server. That is why these two columns must have valid values, and if all other values are 0, it means there was an error during save of that row on-site, and an error message warning about this is built.

IV Curve Filter is a simple IV-curve checking filter. If a row contains the applicable columns, all 'Voltage' columns are checked for continous increase, and all 'Current' columns are checked for continous decrease. To ensure non-erronous messages because of small change in a value to the next, an $\epsilon = 0.2$ are default parametervalue when creating the filter.

Continual date Filter is used to ensure that the previous row is within a specified timeframe to the current row. For the Kalkbult plant a value of 'minutes=15' is default, because the PV-modules saves measurements every 10 minutes. With a timeframe of 15 minutes, it has some leeway before announcing filter failed.

Monitor

The Monitor module needs a 'Database'-instance to monitor, or update. Any filters enabled in monitoring are only applied to the rows received during monitoring – not the existing rows in the local 'Database'. The monitor module also allows for e-mail messages if a filter does not succeed. For each SQL-table an error occurs on, an e-mail will be sent (once). However, the

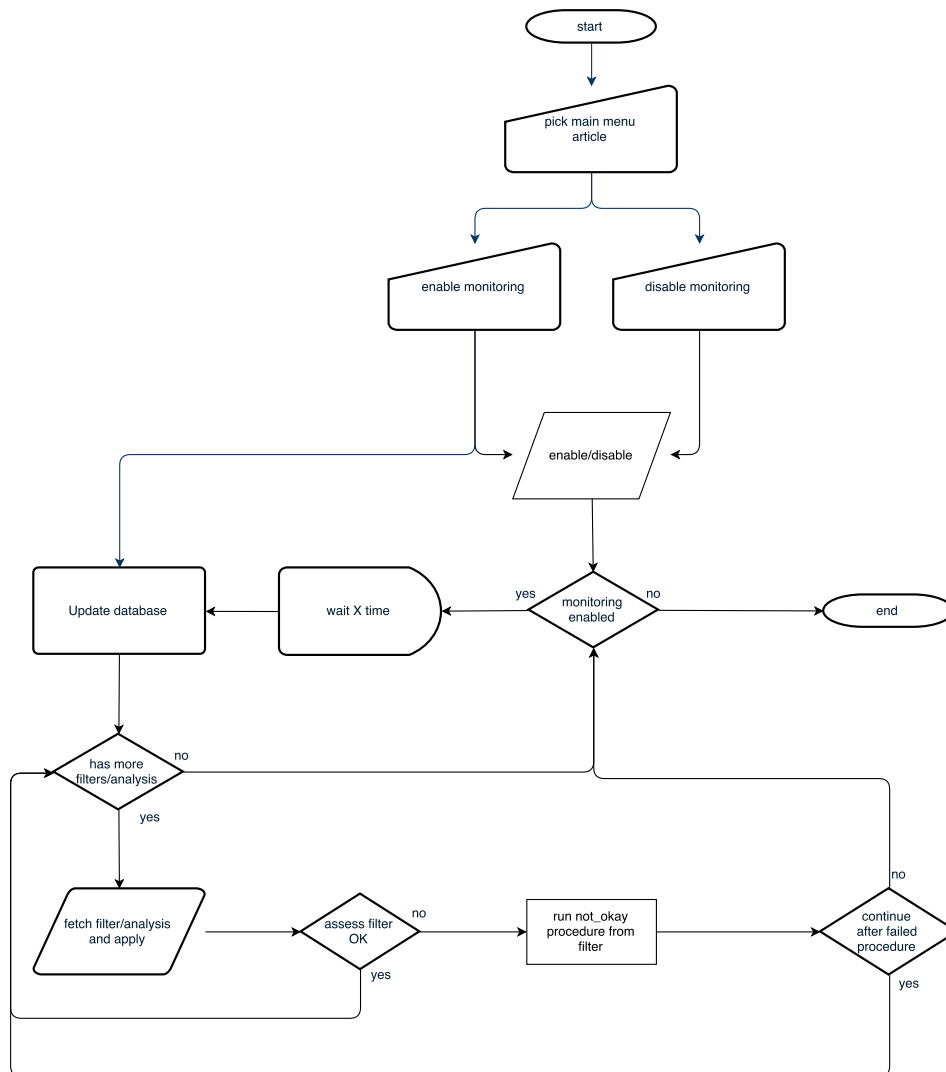


Figure 4.5: Flowchart showing how software will enable and run monitoring, or disable it

monitor will still receive data and insert it into the local database for the erroneous tables. This is because the user has been warned on email when data failed – and can manually run desired filters on the database in the timespan from when the filter failed.

4.3 Data preparation

In contrast with many other studies, data during the night (when there is no irradiation) could prove valuable, in order to catch the whole environmental daily footprints. However, it is impossible to know how each minute of values affect the soiling because it is hard to quantify soiling at that frequency. Also even if it was quantifiable, it would require continuous-manual observation or Υ_R calculation – either nearly impossible. Previous

research and their findings will thus be used in order to choose what approaches to use in preparation of the dataset.

4.3.1 Variance in irradiance

There are two periods without precipitation at the test site, leading to the most likely indications to presence of soiling (Øgaard, 2016): 14.05.2016 - 26.07.2016 and 18.09.2016 - 04.11.2016.

The biggest issue is the impossibility of measuring *instantaneous* soiling at every possible data-row moment. Primarily because the soiling per minute is not easily measurable, or even that much at the testsite (Øgaard, 2016). In order to mitigate this and be aligned to the other theses from the same testsite, each input/output will be daily values.

Using the yield ratio against a reference yield ratio raises another issue. The yield ratio of any PV module is dependent on a range of parameters, with irradiance being the most significant. However, other studies at the testsite indicates higher irradiance seem to lower efficiency, and lower irradiance above a limit seem to increase efficiency. This is noticeable on Figure 4.6. Two ways are proposed to counter this;

Using other module(s) for reference, because all modules should be almost equally affected by the varying light intensity, ie. the proposed *Cleanness Ratio CR* (Plessis, 2016). Or,

Account for the error by using either datasheet for the modules in order to correct the error at varying light intensities or by using statistical measured values from the dataset.

4.3.2 The dataset

The raw dataset to extract inputs from will include the following measured values; date in isoformat(ISO8601, 2000), the average, minimum and maximum; wind speed, relative humidity, irradiation throughout the day (including nighttime), ambient temperature and wind directions, along with module temperature, irradiation and power output at a specific timeframe of the day for yield ratio calculation. The dataset will also include days since clean (which includes rain).

Defining yield ratio reference

The date for defining a *reference* value close to STC with low wind speed, and after rainfall cleaning all modules was found on 11.05.2016 (Øgaard, 2016; Plessis, 2016). Key weather parameters between 12.00-12.50 are displayed on Table 4.6 and each applicable modules' Y_R is shown on Table 4.7.

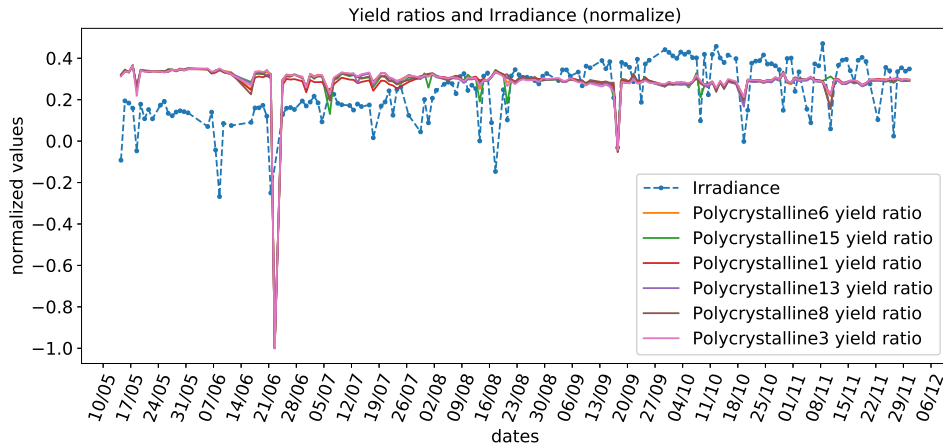


Figure 4.6: The yield ratios and average irradiation on dates with calculatable points shows yield ratio often follows irradiation.

Handling errors in the dataset

The dataset is not complete, and measurements are sometimes not stored. These events ranges from a minute to hours. When this occurs, all values in a row will be 0's (null fields), including the dates. However, the rows are still inserted indicating how many minutes were lost. These rows are dropped from the dataset, and the applicable day is set as invalid.

The range of values will be from May 12th (reference day + 1) until and including November 30th. One day is defined as all measurements from 12.31 on day 1 until and including 12.30 on day 2. The total number of days in the period is 203. For polycrystalline modules, a valid day in this range is a day where 4/7GHI values are within 800-1200 and no less than 150GHI from the middle value, together with valid production data generated from single diode equation defined by another study within the same project (Plessis, 2016). It was found to be a valid day on 168 of the 203 days for all modules except #1 and #5 which had 2 and 1 respectively less valid days. This reduces the valid module days to 165.

A valid weather day is a day with the *complete* 1440 rows of minute-values from day 1 until day 2. Out of 165 valid module days, 29 days did not have the complete number of rows. 7 of these days had at least 1397 rows. An interpolation will thus increase the number of valid dates for better training. It is after all minutely intervals, so an interpolation between a few minutes should not create too much deviation in the relevant measured features. The invalid and fixable dates and their row count are found on Table A.1 in the Appendix.

4.3.3 Preprocessing

Two Python-modules have been created in order to deal with the *weather data* and *module data*. Earlier a Database module has been created for the general system, and has been used and expanded in order to easily get the

Table 4.6: The key weather parameters on the reference day used to indicate yield ratio during analysis period. The back surface module temperature for both types of modules given as the midday (12:00 – 12:50) average on 11.05.2016. The given back surface module temperature are the average for all the modules of the same type.

Date	$G_t [W/m^2]$	$WS [m/s]$	$RH [\%]$	$T_{amb} [^\circ C]$	$T_{poly} [^\circ C]$	$T_{tf} [^\circ C]$
11.05	983.8	1.7	56.2	17.0	48	41

Table 4.7: Each module with their reference yield on the designated date, calculated from Eq. 2.13 with $(P^*/G_t)_{ref} = 1$

1	2	3	4	5	6	7	8
0.239	–	0.243	0.236	0.243	0.241	0.206	0.243
9	10	11	12	13	14	15	16
0.241	0.245	0.243	0.242	0.241	0.241	0.240	0.241

desired data based upon dates or other features. And a last module *dataprep* deals with preparation of output data.

Weather data

Because the environmental data is from Kalkbult, the Python-module is named `'kalkbult_weather.py'`. This Python-module contains constants regarding the database and two primary functions among others; `'create_input_intervals'` and `'get_all_weather_data'`. The first function takes weather data one day (1440 rows of values), and creates a new array consisting of n -intervals with min, max, average or incremented values – based upon the value keys. The different keys and their desired function for intervals are shown on Table 4.8, and each prepared interval row only contain these keys with values in the returned array. The rest of the data are discarded.

The other function requires a Database-instance and an array of end-datetime. These dates are used to gather all data from the 24 hours *before* that point in time, ie. If `'[datetime(2016, 5, 14, 12, 31)]'` (a single datetime element) is provided, all weather-data from 2016-5-13 12:31 until 2016-5-14 12:30 is returned. By default it also prepares all the date-sets using the first mentioned function `'create_input_intervals'` function with 12 intervals. However, this interval can set explicitly in the call, or ignored all together by using a `'pure=True'` parameter – returning the raw sql-rows gathered for each date mapped to its date in a key:value using the Python-class `OrderedDictionary`. This is recommended when the need to

Function	Key(s)
increment	Rain _{interval}
average	$T_{avg}, RH_{avg}, WS_{avg}, I_{avg}$
max	$RH_{max}, WS_{max}, I_{max}$
min	$RH_{min}, WS_{min}, I_{min}$
wind directions	$WS_N, WS_{NNE}, WS_{NE}, WS_{ENE}, WS_E, WS_{ESE}, WS_{SE}, WS_{SSE}, WS_S, WS_{SSW}, WS_{SW}, WS_{WSW}, WS_W, WS_{WNW}, WS_{NW}, WS_{NNW}$

Table 4.8: An overview of the valid keys, and their function when calculating interval value

process the data manually or in several intervals. For example when creating the different models.

A third function gets the peak values at given dates. This is used by the PV Python-module in order to get the most valid data to work on for each day when creating daily yield ratio, and thus also soiling rate using equations (2.13) and (2.17).

The last function is 'extract_inputs_from_keys' that is used by the machine learning model instances to prepare weather data into correct shape of numpy-arrays. This function requires keys and the weather data as parameters, in order to only gather value from the desired keys (features) by the models. For example Model A has WS_{avg} as its parameter, and thus the only key will be '[AVG_WS]'. The function also normalizes the values according to Eq. 3.10 into values between $-1-1$ using the values from Table 4.9 as d_L and d_H depending on the key. This is because the most common activation function *sigmoid* outputs values from $0 - 1$, and is more dynamic when influenced by smaller values within that range.

Finally, the valid dates to use as S_{rate} as target values for the ANN are the dates with a calculated previous day. This is because S_{rate} needs to have yield ratio for a valid previous day. If the last valid day are longer timespan than a day, it will not be one day difference, but a change from a longer timeframe. An overview of dates with valid yield ratios are shown on Table 4.10, where transparent days are invalid, striked out dates are used to calculate S_{rate} target days, and the solid days are valid target days. This is because striked out days have more than one day since last valid Y_R .

PV data

In the 'ProductionData' module, the function 'fetch_raw_output(dates)' finds all valid dates, and creates an averaged production and yield ratio value for each day. This is done in a step-like manner:

- For each date in period, find the largest irradiation ($800 < G_t < 1200$) value from database using 'get_peak_data'. This will return 7 values,

Table 4.9: The ranges are found by using the smallest and largest value found in the dataset within period of study for each measured variabel. Except Power, which has d_H given by specification as peak output for the Polycrystalline modules.

Keys	d_L	d_H
$T_{env.avg}$	-6.1	40.0
$T_{mod.avg}$	-16	77.875
$RH_{min}, RH_{avg}, RH_{max}$	0.0	100
$WS_{min}, WS_{avg}, WS_{max}$	0.0	20.5
$I_{min}, I_{avg}, I_{max}$	0	1600
$Rain_{interval}$	0	3
Wind directions	0	12
S_{rate}	-0.3	0.3
Power	0kW	255kW

3 values before and after the peak and the peak itself.

- For each module:
 - For each date in period
 - * If date was invalid from peak values, skip, else:
 - * Get production values from database given timeunits from the 7 peak units.
 - * Ensure there are production data to work on (ensuring there was production data for the given time).
 - * Average the valid production data. It is valid if at least 4/7 have both valid irradiation ($G_t > 800$) and production larger than 0. Only the valid points are used.

To create the production values, the gathered IV-curve values are sent through a nested function hierarchy:

- 'poly_yield_ratio' in Python-module 'pv_module' calculates the yield_ratio against the reference value using Eq. 2.13 on the row data and irradiance at the specified time
- The production is calculated by finding the MPP using 'iv_fit' in Python-module 'pv_module'. This function returns the 'Perform_SingleDiode_Adjustment' (Plessis, 2016). If the calculation of MPP fails, a value of -1 is returned to indicate erroneous values and IV-curve calculation. If the iv-fit returned value larger than

0, a temperature correction is done using Eq. 2.14, else the power production is returned as 0 for the given data.

After all the modules have prepared production data, an instance of the 'Dataprep' class is created to store yield ratios, and create soil rates and soil ratios from the valid production data. These values are stored in a dictionary for each module instance. These arrays are ready to use as target values for the neural network.

4.4 Use of data

Some notes on selection of data, and what modules to use.

4.4.1 Irradiance variance

As mentioned in previous sections on the other studies performed on the same data from Kalkbult, it is hard to quantify how much soiling occurs. The same problem has thus affected this thesis' calculations and how the data performed. On Figure 4.6 is the display of soiling ratio for each (valid) day from May 12th until September 12th, along with average irradiation used in calculating the average yield ratios. If looking closely each change in irradiance most often resulted in equivalent change in ratio the next day, ie. increase in irradiance increased soiling the next day, and opposite.

The other studies within the project used a cleanness index to compare a clean and unclean module against each other for a clean-ratio. This thesis is trying to identify soiling rate from day to day, making the same approach not applicable. This is because the clean modules are only cleaned every 14 days. This means it should not be a considerable difference between clean and unclean until at least 14 days has passed. The other approach was to account for the error. However, this approach was studied and found too uncertain because of large deviance from another study within the same project (Øgaard, 2016).

That is why the proposed Y_R leading to S_{rate} from Eq. 2.13 and 2.17 respectively will be used, even though it is most likely influenced by irradiation and/or other factors. This is still done because there *are* small noticeable tendencies for decreasing yield ratio between cleaning events, as seen on Figure 4.7. This figure shows the Y_R and S_{rate} of modules Polycrystalline1 and Polycrystalline3, where there are areas where both modules seem to drop a little on the day-to-day yield ratio – indicating the defined Y_R and S_{rate} can indicate soiling of modules despite being influenced by irradiation and possibly other values.

4.4.2 Choosing data from modules

There are an overview of S_{rate} for the modules without problems and that had valid creation of Y_R on Figure 4.8. From this graph, it is possible to see Polycrystalline1 and to some extent Polycrystalline3 are without the largest spikes in S_{rate} . In other studies within the same project, Polycrystalline1

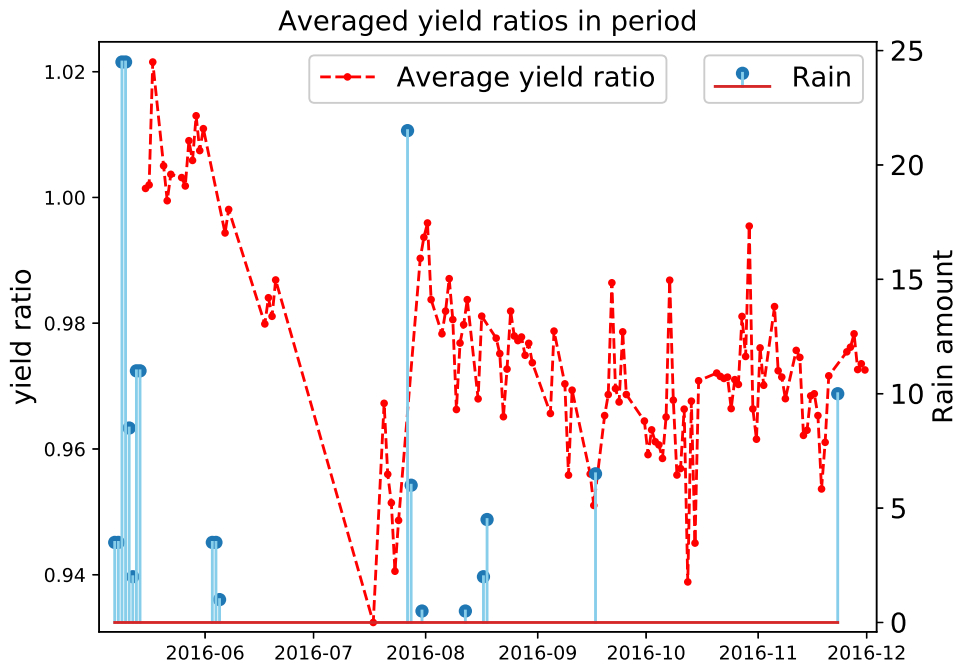


Figure 4.7: Shows the average yield ratios for modules 1, 3, 6, 8, 9, 10, 11, 12, 14 and 16 are somewhat restored after rain events.

was indicated to be the modules with most soiling. From the figure it is also the module which is seldom away from the majority of modules. This indicates Polycrystalline1 is least affected by other factors, unlike the other modules. Together with the fact that Polycrystalline1 is never cleaned *and* coated with anti-soiling (which was proposed to work against its purpose), Polycrystalline1 seems like the best data to use for target values.

Looking closer at the yield ratio (Y_R) of Polycrystalline1 in Figure 4.9, it looks like recovery after the rainfalls. Although, there are recoveries at other times as well.

The target values for the neural network will thus be the S_{rate} of Polycrystalline1 as shown in Figure 4.10.

4.4.3 Fuzzyfication (sorting) of the inputs features

Fuzzification is the process of preparing the data for a fuzzy inference. Although the direct use of this will not be utilized – the principles of it may be. If each of the feature values were given inputnode based upon the *time* it occurred – it gives some meaning to that *time*. However, the interesting part is the *values* and their *interactions*. This may be achieved by sorting each feature inputs; this causes the lowest value of a feature to always come to the same input node, next lowest to the same node etc. Capturing the footprint of a day having similar values at different hours of the day is possible by using this sorted approach. An example of two different days with and without sorted features are shown on Figure 4.11 & 4.12 respectively.

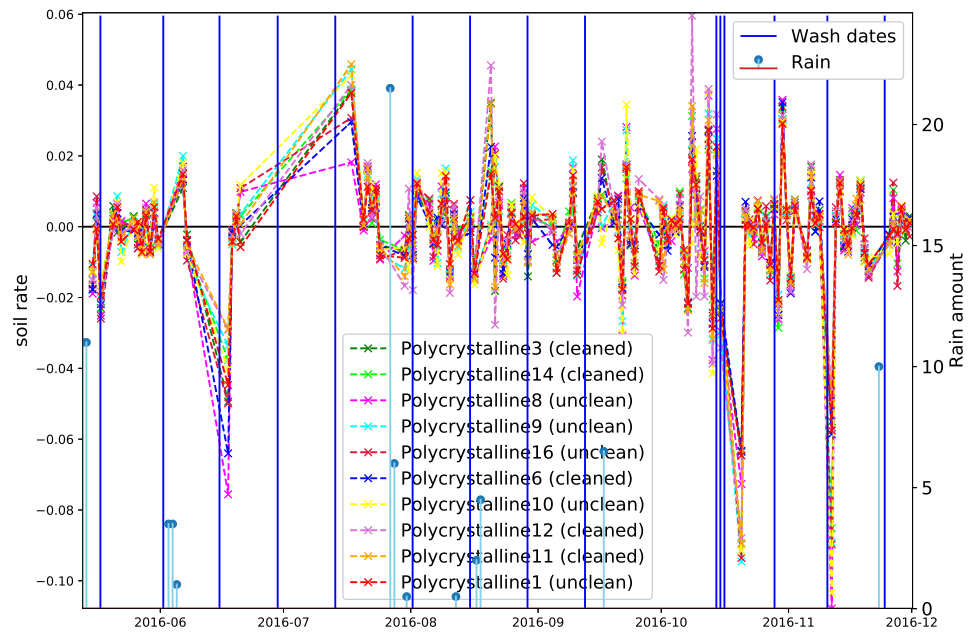


Figure 4.8: An overview of S_{rate} for the applicable modules and dates. It is hard to see on paper, but Polycrystalline1 are least fluctuating module, and most aligned with the majority of other modules.

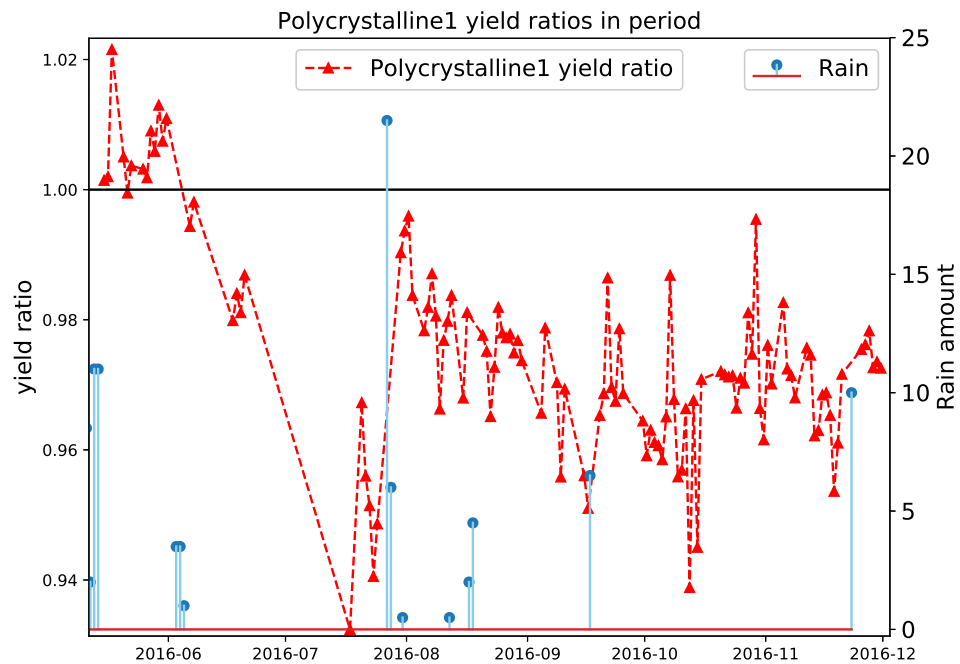


Figure 4.9: The yield ratio of Polycrystalline1, where it looks like the Y_R of Polycrystalline1 is somewhat restored after rain has occurred.

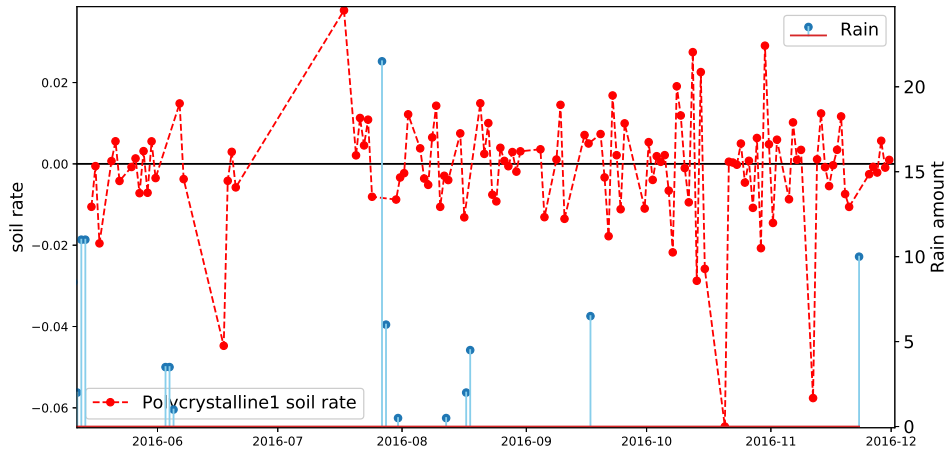
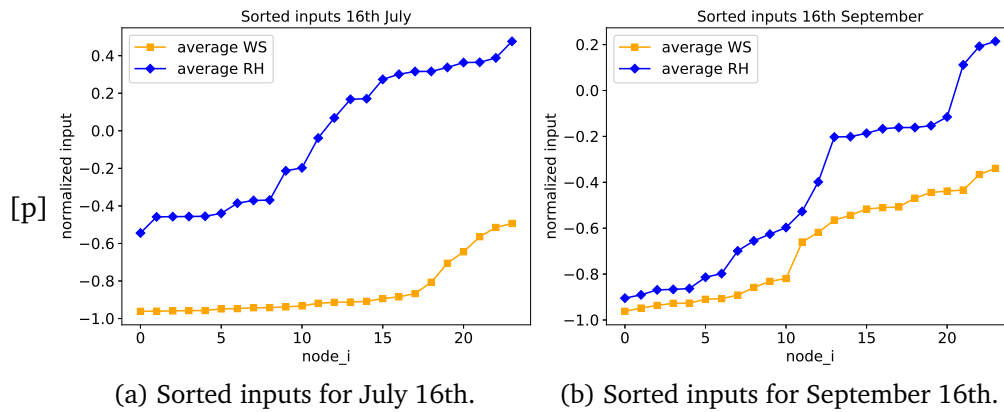


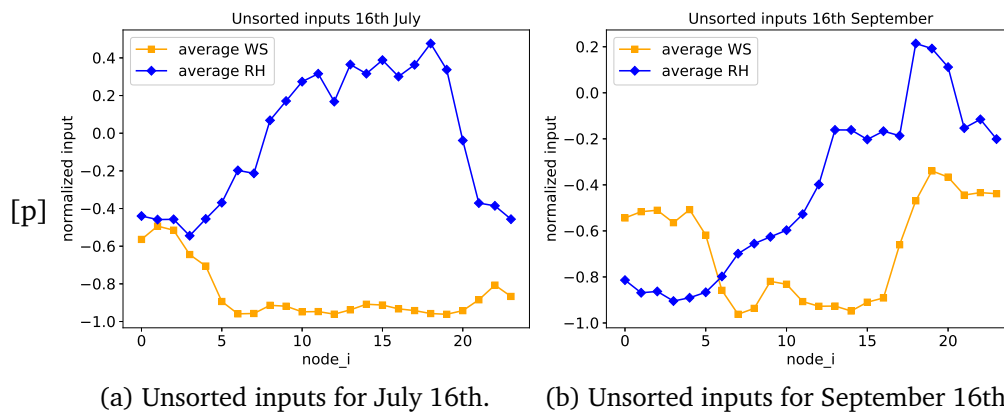
Figure 4.10: The S_{rate} values of Polycrystalline1, which will be the target values to predict.



(a) Sorted inputs for July 16th.

(b) Sorted inputs for September 16th.

Figure 4.11: Overview of how 24 sorted inputs looks like on two different days. Higher values now occur closer to eachother, reducing the dependency for a value to happen on the same hour for two different days.



(a) Unsorted inputs for July 16th.

(b) Unsorted inputs for September 16th.

Figure 4.12: Overview of how 24 unsorted inputs looks like on two different days. This shows ie. the highest wind speed (WS_{avg}) does not occur on the input node i for both days.

Chapter 5

ANN implementation

Every model will be used to predict only one output value; soiling rate (S_{rate}). However, to ensure that the library and data preparation has been done correctly, a model verification will be done on prediction of power output for every ten minutes over several different days.

5.1 Models

It is necessary to start with a simple model in order to evaluate *how much* information is needed in order to get some valuable output. Then expand the model with more varied and increased inputs. In the models with more than one input, ordering of the input values may be done as explained in the fuzzification paragraph in subsection 4.4.3. Again, the reason for this is to dismiss a significance to the hour a value occurred on. As it is the values and their interaction which is interesting, and not the hour it happened in. To elaborate, the scope of this thesis is to predict *daily* soiling rate, so the daily values which occurred *at any point* during the day is the important information – not *when* it happened.

Eg. if a model uses hourly average wind speeds throughout the day, totaling to 24 values as input, not sorting them would imply the *hour* a value happened on is important. It is the position of an input which is significant. By sorting these averages from lowest to highest, the values are more likely to occur on a similar position, which increases the likelihood to *catch* interaction between certain input values following each other instead of certain hourly values following each other.

The basic setup of each model will have hidden nodes given by Eq. 3.9 as the size of the first (fully connected) layer. An overview of all starting models, their interval and inputs, and the corresponding name can be seen on Table 5.2.

Model A will use only the average wind speed, as this has been proposed to have a minor influencing feature for soiling rate.

Model B is maximum wind speed and average wind speed. This is chosen because gusts of wind provide more force to move heavier particles upon

the panels, or even push them off. An average might not catch these gusts, but the maximum may do that.

Model C is the average wind speed along with the 12 wind directions inputs. Because the solar panels are inclined, the direction of the wind is more likely to play a more influencing role in the soiling rate. Simultaneously, at the given test site there is a road due west. Traffic on this road together with east oriented winds are expected to increase soiling.

Model D is only the average relative humidity. Partitioning up all features may help distinguishing the influencing parameters more easily, especially when combining features on the other models.

Model E is average humidity together with ambient and module temperature. The reason for adding both ambient and module temperature is because of possibility of dew formation. Table 5.1 shows dew formation could occur on around 11-13% of the time using Eq. 2.11 all data in the selected period. Other studies have shown dew formation may lead to a minor cleaning event. Minor cleaning can lead to a decrease in soiling, or reduce the increase in soiling.

These conclude the basic models. The following models are combined versions of the previous core models.

Model F is the model having both average wind speed and humidity (combined A and D). Other studies have proposed these are the most influential parameters with respect to soiling.

Model G is models B and D, maximum wind speed with the average wind speed and humidity.

Model H is models C and D, average wind speed and humidity together with the wind directions.

Model I is models A and E, temperature variables together with average wind speed and humidity.

Model J is models C and E, wind directions with average wind speed, humidity and temperatures.

Model K is models B and E, average and maximum wind speed, average humidity and temperatures.

Table 5.1: Output from selected modules and the rows dew are possible to occur. **Note:** this is all available data, not just the dates used in training the neural network.

Module name	dew-formation rows	all-rows	percentage
Polycrystalline1	3038	26062	11.66
Polycrystalline3	3327	26398	12.60
Polycrystalline6	3381	26392	12.81
Polycrystalline8	3355	26228	12.79
Polycrystalline9	3239	26402	12.27
Polycrystalline10	3358	26402	12.72
Polycrystalline11	3381	26402	12.81
Polycrystalline12	3307	26401	12.53
Polycrystalline14	3247	26364	12.32
Polycrystalline16	3389	26402	12.84

Model L is all features mentioned, average and maximum wind speed with directions, and average humidity, and temperatures.

Irradiation is not included as a feature because it is not expected to affect the soiling rate significantly. It was discussed whether irradiation could impact through heating by drying of particles, effectively sticking them to the module/panel. However, for this to occur, the particles must arrive there (by wind) initially. This makes it somewhat possible to detect the suggested effect from selected features. Also the heat and dry effect are not expected to have significant impact on the low soiling rates.

5.1.1 Intervals

The chosen intervals are a combination of both a measure of how long an interval can be, and what is most likely measurement intervals. At the test site, IV-curves are calculated every 10 minutes - and are thus the shortest interval. Another study proposed 15 minutes to be a good choice, and will also be included. The longest interval in these models is 1 day, because the soiling rate is calculated from day to day. The three other intervals, 1, 6 and 12 hours are chosen because of their hourly relevance of a partitioned day; $\frac{1}{24}$, $\frac{1}{4}$, $\frac{1}{2}$ respectively.

Table 5.2: An overview of the model inputs, the interval ranges and each model denotation.

Model:Inputs	Interval count (length per interval)					
	1 (1 day)	2 (12h)	4 (6h)	24 (1h)	96 (15m)	144 (10m)
A: WS_{avg}	A0	A1	A2	A3	A4	A5
B: WS_{avg}, WS_{max}	B0	B1	B2	B3	B4	B5
C: WS_{avg} with wind directions	C0	C1	C2	C3	C4	C5
D: RH_{avg}	D0	D1	D2	D3	D4	D5
E: $RH_{avg}, T_{env.avg}, T_{mod.avg}$	E0	E1	E2	E3	E4	E5
F: WS_{avg}, RH_{avg}	F0	F1	F2	F3	F4	F5
G: $WS_{avg}, WS_{max}, RH_{avg}$	G0	G1	G2	G3	G4	G5
H: WS_{avg}, RH_{avg} with wind directions	H0	H1	H2	H3	H4	H5
I: $WS_{avg}, RH_{avg}, T_{env.avg}, T_{mod.avg}$	I0	I1	I2	I3	I4	I5
J: $WS_{avg}, RH_{avg}, T_{env.avg}, T_{mod.avg}$ with wind directions	J0	J1	J2	J3	J4	J5
K: $WS_{avg}, WS_{max}, RH_{avg}, T_{env.avg}, T_{mod.avg}$	K0	K1	K2	K3	K4	K5
L: $WS_{avg}, WS_{max}, RH_{avg}, T_{env.avg}, T_{mod.avg}$ with wind directions	L0	L1	L2	L3	L4	L5

Example of the neural network using *one* day as interval length gives 10 number of hidden nodes from Eq. 3.9 with 71 samples. With one hidden layer the ANN can be illustrated by Figure 5.1.

Another example, the maximum number of inputs will be L5. This is *10-minute* averages and give a total of 144 values for each feature. These features are *maximum wind speed, average wind speed and humidity, ambient temperature and module temperature*, and the 12 *wind directions*. The model requires $144 * (5 + 12) = 2448$ inputs.

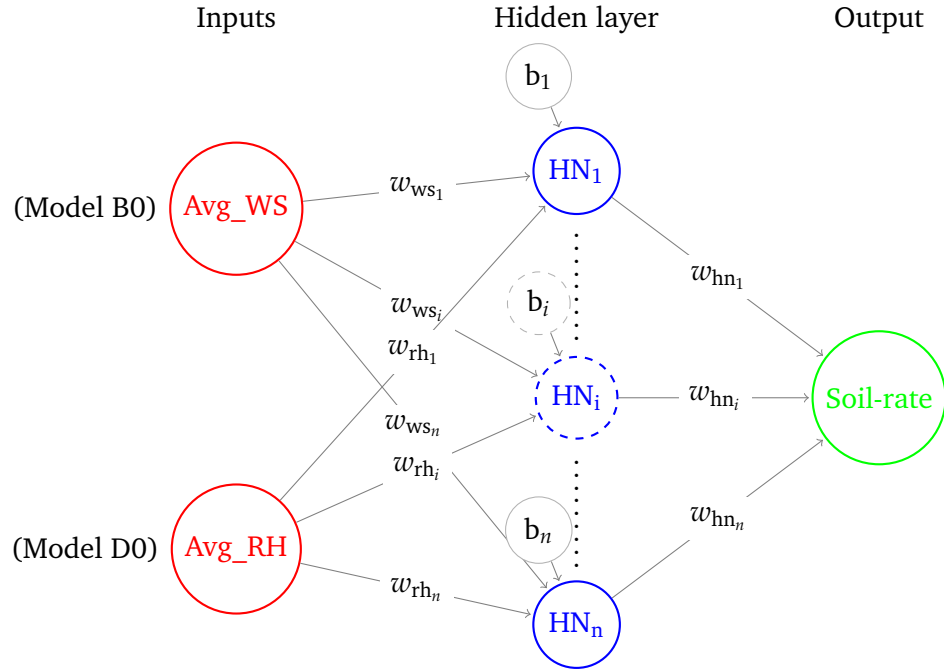


Figure 5.1: Showing the nodes and weight annotations for model F0 (combination of B0 and D0), where $i = \{2, \dots, n - 1\}$ and $n = 10$. Each hidden node also has a *bias* node, in order to shift all values of the activation function output.

5.2 Neural net implementation

In ANN, important parameters are the *update* function and *update learning rate*, as these two create the foundation for how a neural network will learn from the training. During experiments on the neural network, update function *ADAM* (3.8) proved to provide quick and accurate training, requiring less than 100 epochs for the same training result *SGD* (3.5) alone needed more than 10k for. The standard learning rate of 0.02 has also been chosen as update learning rate, as it had the same performance overall.

Another important feature of an ANN are the layers. The input layer will always have one node per input multiplied by interval count, ie. Models A0 (1 daily WS_{avg}) and B0 (1 daily WS_{avg} and WS_{max}) will have 1 and 2 nodes respectively in the input layer. While A5 and B5 will have 144 and 288 input nodes respectively, because there are 144 $10m$ values for each input feature. The output layer is one single output node, the soil rate (S_{rate} from Eq. 2.17), for all models. The hidden layer have nodes calculated from Eq. 3.9 initially, and will be tested with ± 5 nodes. It has been stated more hidden layers are required for non-linear discoveries. Each model are thus available to be created with deep learning capabilities. It will be two hidden layers for deep learning on models with a single input feature, and three hidden layers for all other models.

5.2.1 Programming tools and libraries

To implement the neural network models, some external Python libraries are used:

- **numpy** is a well known library for numerical computation (Walt, Colbert, and Varoquaux, 2011).
- **Theano** is a library for efficiently use of known mathematical expressions, integratable with numpy (Al-Rfou et al., 2016).
- **Lasagne** is a lightweight library for building and training neural networks using Theano (Dieleman et al., 2015). Example of usage from lasagne are update functions and layers classes:

```
- from lasagne.updates import adam
- from lasagne.layers import InputLayer, DenseLayer
```

- **NoLearn** is a library built upon lasagne, which enables quick implementation of neural networks upon the other libraries (Nouri, 2014). From this library, the 'NeuralNet' class is imported for quick creation of a neural net using Python-modules from Lasagne: 'from nolearn.lasagne import NeuralNet' and its use in this thesis 'Model' class.

```
def create_net(self,
                module,
                epochs=1,
                update=adam,
                update_learning_rate=0.02,
                verbose=False):
    """
    Creates a neural net with given parameters, and
    stores it in the local OrderedDict 'nets' on its
    'module' key.
    :param module: The module name, ie. 'Polycrystalline1'
    :param epochs: The number of epochs for each training/fitting session
    :param update: The update function, ie. 'sgd', 'rmsprop', 'adam' oo.
    :param update_learning_rate: The learning rate, usually 0.01 - 0.02
    :param verbose: If the NeuralNet is supposed to be verbose or not
    """
    self.nets[module] = NeuralNet(
        layers=self.layers,
        max_epochs=epochs,
        update=update,
        objective_loss_function=squared_error,
        update_learning_rate=update_learning_rate,
        regression=True,
        verbose=verbose
    )
```

5.2.2 Designed classes and functions

Python-module 'learning' has been created to define the neural net model class hierarchy available in the appendix. The most important classes are the 'Model' and 'ANN'-classes. The 'Model' is the superclass of all models, having most of the core functionality like creation of the neural net, fitting training data and predicting test data. The 'ANN'-classes define the layers with single hidden layer ('ANN_Basic') or multiple layers ('ANN_Advanced'). When initializing either subclassed model, it prepares the input values based upon what 'input_keys' (feature types) are provided using the 'extract_inputs_from_keys' defined.

Each earlier described model has been implemented to automatically pass the relevant feature keys upward in the class-hierarchy. Thus, the parameters needed for creation of a module are the valid dates, weather data (mapping of dates with mapping to all features for each date), the 'ProductionData'-module containing production values, and the number of intervals.

Chapter 6

Results & discussion

This chapter consists of three sections. First a model verification in order to prove the implementation are able to predict known relations. The second section is model training with a single hidden layer approach. The last section is an attempt at deep learning with 2 and 3 hidden layers.

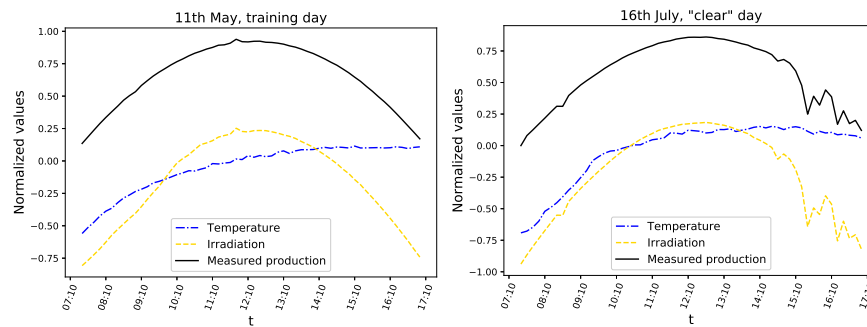
6.1 Model verification

In order to establish confidence that the suggested models can discover a pattern in the data if it exists, a verification using the same principles as another study (Yadav and Chandel, 2017) were performed. The reference study showed the method with best prediction (least MAPE and RMS) on power output through the day of study occurred when only using irradiation and temperature as inputs for every minute.

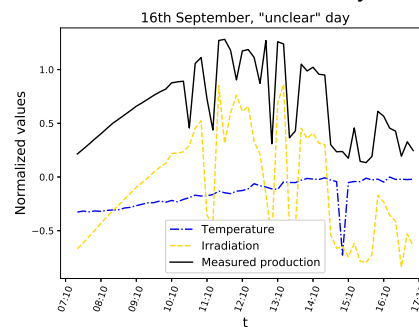
Hence an ANN model was built using the same principles as the reference study, using this thesis' described models to accommodate the two inputs I_{avg} and $T_{env.avg}$. There will be one hidden layer with 13 nodes, equal to the model with best results in the reference study. The most notable difference to the reference study and this dataset is frequency in production values. The Kalkbult data stores production value every 10 minutes, instead of every minute as was done in the reference study. Due to low irradiance in the morning and evening, values chosen were all 10-minute values from 7.30 until and including 17.00, for a total of 58 values for one day.

Three days were chosen for verification; the reference day 11th of May, 16th of July, and 16th of September. The first date was used for training, and the other two for testing. The dates were chosen because it is approximately the same length (2 months) between each date. This increases confidence the model can discover pattern throughout the period of study. In addition, the first two dates are close to clear sky dates, which is shown by a near perfect circular curve on Figure 6.1a & 6.1b. The last day is a cloudy day, which is indicated by the many dips in the curve on Figure 6.1c. Because cloudy days are significantly different from clear sky dates, it is a good verification of the neural network to predict input very different from the training dataset.

After training the verification model (hereby known as **model**) with



(a) The production in black, irradiation in gold and temperature in blue on 11th May. (b) The production in black, irradiation in gold and temperature in blue on 16th July.



(c) The production in black, irradiation in gold and temperature in blue on 16th September.

Figure 6.1: These plots show the irradiation and temperature inputs of the three verification dates, along with production, which is output. **Note:** All input values are normalized in range -1 to 1 , and output is normalized in range 0 to 1 according to Table 4.9.

6 epochs as the reference study suggested, the model did not converge or score equally well. This may be due to the datasets were different. However this is not likely, as the data in the reference study were similar to the data chosen in this study. A more likely cause was the different approaches in training the neural networks. The reference study used Levenberg–Marquardt (LM) for training, while this thesis used the stochastic gradient descent (SGD) technique ADAM (Eq.3.8). LM is an approximation to the Newton method, and has the benefit of being faster to find a solution than SGD. However, LM needs to be closer to the global minima because it does not include momentum which helps ADAM escape a local minima. Because of this, additional epochs of training were done on the verification model to a total of 40 epochs. At that point, the verification model had a very good prediction on both the clear and unclear sky days with R^2 scores of 0.963 and 0.845 as shown on Figure 6.2 & 6.3 respectively.

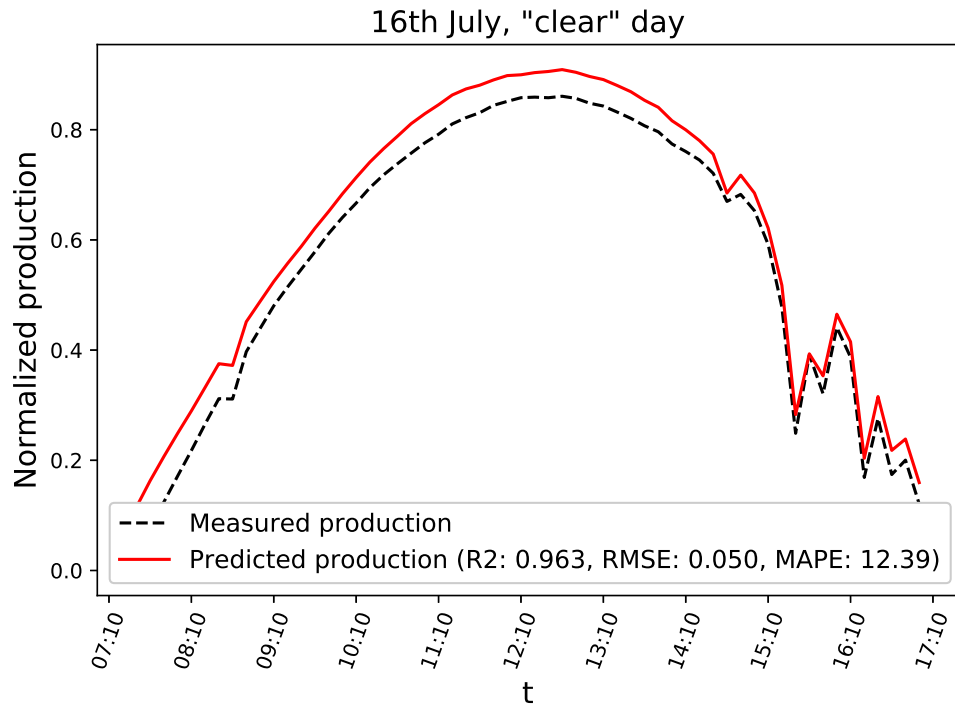


Figure 6.2: The production in black and predicted data in red on 16th July with $R^2 = 0.963$.

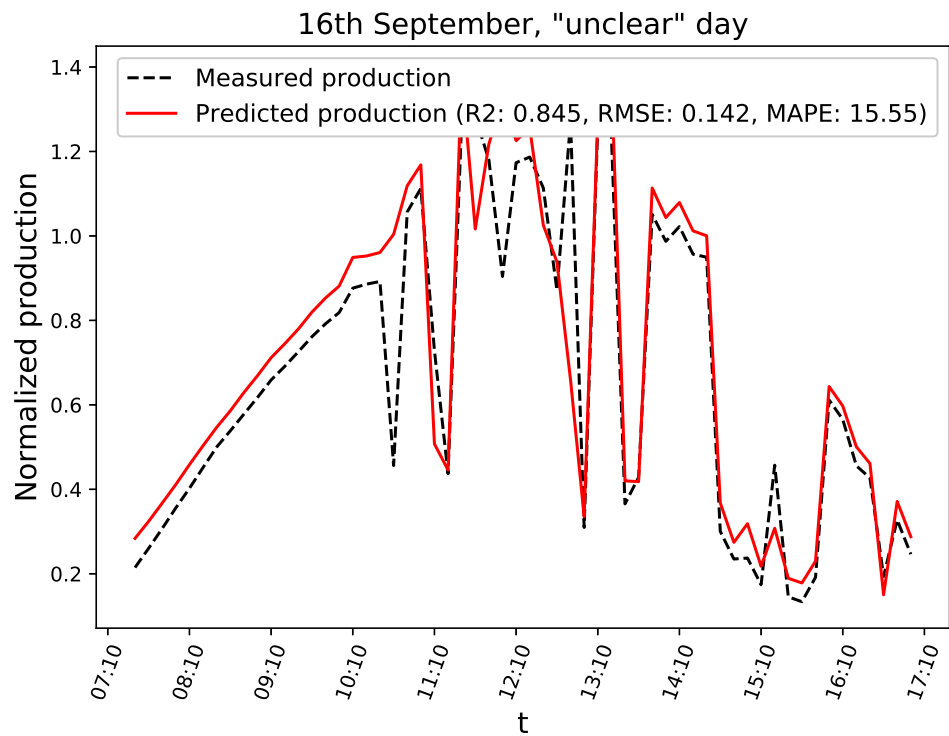


Figure 6.3: The production in black and predicted data in red on 16th September with $R^2 = 0.845$.

6.2 Single hidden layer (SHL)

Because the verification model showed 40 epochs of training were necessary on a dataset with *known* relation between input and output, each model from Table 5.2 were trained and scored with R^2 150 epochs (an entire run through the training data with weight updates for each run accordingly). The increase in training epochs were because it is not known how related the features are with soiling, if at all. Also, exploring if this relation exist is the goal of this thesis.

It is also important to remember the defined soil rate (S_{rate}) is a result of the change in yield ratio as defined by Eq. 2.17. *However*, the yield ratio is probably influenced by features besides soiling. This means it is not guaranteed the **target values** are only because of soiling. This is also indicated on the targets having negative S_{rate} in Figure 6.4 (all values below the horizontal 0.5 line). A negative S_{rate} implies a reduction in soiling (an *improvement* in performance) since the day before. Significant improvement is usually related to cleaning or rain events. Although some studies suggest a minor cleaning event from dew formation can improve performance, and also powerful wind speeds. That is why the targets are used despite the uncertainty. Additionally, if there is a relationship between a feature and soiling, they were expected to reduce the *magnitude* of negative S_{rate} .

6.2.1 Model scores with unsorted inputs

All models were trained and scored with 150 epochs for the 11 different node setups in a single hidden layer. The initial hidden nodes were calculated from Eq. 3.9, and varied with ± 5 . For each model, the highest values of R^2 and lowest values for MAPE and RMSE encountered during training of all node setups were stored along with that models scoring history. After training and scoring of each node setup, the best R^2 values encountered were from scoring R^2 on the test data (R^2_{test}), while the R^2 on whole dataset (R^2_{all}) set and training data (R^2_{train}) scores did not show similar promise. Additionally the best (lowest) MAPE and RMSE values had very poor R^2 values.

R2 on the test data

Both the overall and singular best R^2 values found were from scoring R^2 on the test data. The best value was ModelD3 with $R^2_{test} = 0.225$, as shown on Table 6.2.

Model D3 is among the few models with all R^2_{train} , R^2_{test} and R^2_{all} values being positive. The other models with all positive R^2 scores are the other D models; D4 and D5. This may be due to their positive, albeit low, score on R^2_{train} . It was thus not surprising these models held 3/4 top scores. What was surprising is Model D *only* has RH_{avg} as its sole input feature. Additionally, 8/10 top models had RH_{avg} as one of the input features. Two other notable inputs were WS_{avg} which is in all models other than D, and WS_{max} which were in models B, G and K. Comparing the models G2, G3 and

Table 6.1: A recap on the overview of features and its models with interval denotation.

Model:Inputs	Length per interval (number of inputs per feature)					
	1 day (1)	12h (2)	6h (4)	1h (24)	15m (96)	10m (144)
A: WS_{avg}	A0	A1	A2	A3	A4	A5
B: WS_{avg}, WS_{max}	B0	B1	B2	B3	B4	B5
C: WS_{avg} with wind directions	C0	C1	C2	C3	C4	C5
D: RH_{avg}	D0	D1	D2	D3	D4	D5
E: $RH_{avg}, T_{env.avg}, T_{mod.avg}$	E0	E1	E2	E3	E4	E5
F: WS_{avg}, RH_{avg}	F0	F1	F2	F3	F4	F5
G: $WS_{avg}, WS_{max}, RH_{avg}$	G0	G1	G2	G3	G4	G5
H: WS_{avg}, RH_{avg} with wind directions	H0	H1	H2	H3	H4	H5
I: $WS_{avg}, RH_{avg}, T_{env.avg}, T_{mod.avg}$	I0	I1	I2	I3	I4	I5
J: $WS_{avg}, RH_{avg}, T_{env.avg}, T_{mod.avg}$ with wind directions	J0	J1	J2	J3	J4	J5
K: $WS_{avg}, WS_{max}, RH_{avg}, T_{env.avg}, T_{mod.avg}$	K0	K1	K2	K3	K4	K5
L: $WS_{avg}, WS_{max}, RH_{avg}, T_{env.avg}, T_{mod.avg}$ with wind directions	L0	L1	L2	L3	L4	L5

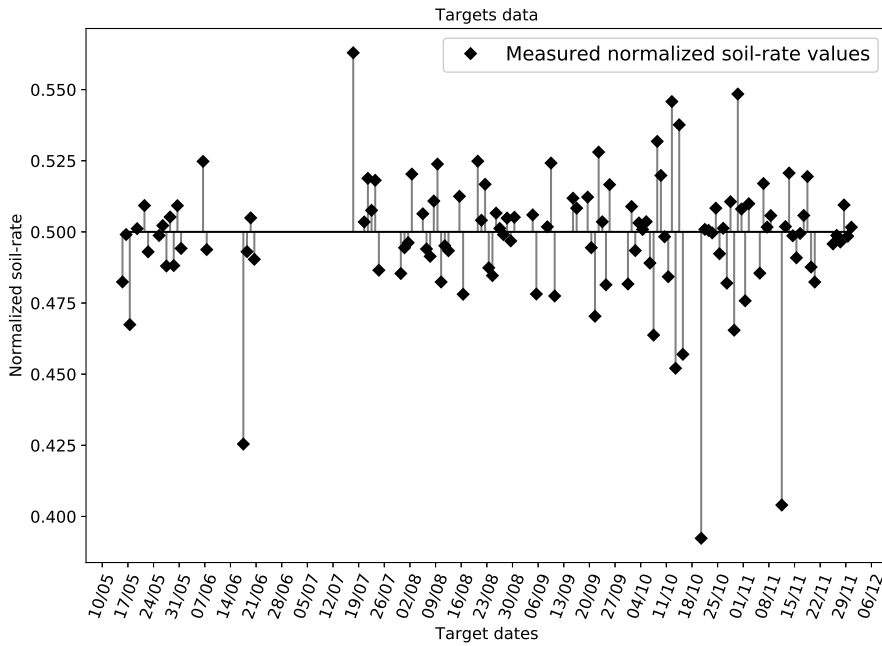


Figure 6.4: The target values with dates and distance from the horizontal 0.5 normalized soiling rate implying $S_{\text{rate}} = 0$. Positive values are increase in soiling, and negative values a decrease in soiling (improvement in performance since last day).

F2 may explain the importance of WS_{max} and interval lengths. Model G2 on rank #8 with all three features are somewhat lower than the similar F2 with only WS_{avg} and RH_{avg} on rank #5. The reason for this may be the interval length and its affect on the feature footprints. Interval-2 is 4 hour intervals, which may not be enough to capture the effect of WS_{max} in the interval. Removing the feature could give F2 an advantage over G2 because of this. By shortening interval length, model G3 on rank #3 with hourly values (from interval-3) were more likely to catch the effects of WS_{max} , and is thus the best of the three. Although model B on #6 and #7 consisting of only WS_{avg} and WS_{max} does not support this interval theory, with B1 having higher rank than B2. It is worth noting model B is the only model in top 10 without RH_{avg} . This could indicate it is better with short intervals when having both RH_{avg} and WS_{max} as features, but longer intervals are good enough when only WS_{avg} and WS_{max} are inputs.

Summarized, these observations indicate the *most* important feature is humidity (RH_{avg}), along with maximum wind speed (WS_{max}), and somewhat average wind speed (WS_{avg}). This is due to humidity being in almost all models, while average wind speed alone is not among the top scores. The maximum wind speed seems to be better with decreasing interval length, but even this value may create noise on the humidity models. Without humidity, maximum wind speed had better scores on the longer intervals (B and F in the table). None of the top ten models have wind directions among their inputs at all, indicating these features

create more noise than they contribute valuable data. Only model K2 on rank #10 had temperatures as input features. Indicating also these features provide more noise than valuable data, although not as bad as the wind directions. The best models usually had hourly intervals (interval-3) on each input, with the exception of B and F having best results on 12 (interval-1) and 4 (interval-2) hour intervals respectively. A possible reason for good scores despite having longer intervals may be the *low* number of total inputs. D3 has 1 feature with 24 values to the neural network. By contrast D4 and D5 has 4 and 6 times as many values, 96 and 144 values total respectively. These numbers are even higher for the more complex models with more than one feature. Because the rate of soiling is likely a non-linear relationship between humidity, wind, and other features, the relationship is best captured with lesser values rather than more. Many inputs to the neural network are more likely to contribute noise instead of valuable input to output relation.

Model D3 prediction

The prediction values on the test data from model D3 at epoch 18, when R^2 had best score, is visible on Figure 6.5.

The predicted test values (green) against their targets reveal several of the values are *spot on* and others really close to their target value. Considering all values are part of the test dataset, and not the training dataset, the model has *never* encountered these inputs and targets. Also over half of the points (21/39) are on the correct side of the no-soiling line as their targets. Although there is no clear pattern if the magnitude increase or decrease. Looking at the whole dataset in Figure 6.6, the (orange) training predictions does not align good with the (black) measured target values. However, this is not important because it is the unseen values the neural network is interested in predicting – the test data in this case.

Considering the low RMSE of 0.042 and MAPE of 6.150 together with some points *spot on* their targets and the overall good scores of humidity on Table 6.2 indicates there may be some correlation between hourly or shorter RH_{avg} values and the defined S_{rate} .

An observable difference compared to the verification model is the low number of epochs, with the best score encountered at 18 epochs. This was the best, because it did *not* encounter a better R^2_{test} in the remaining 132 epochs. An explanation for this is the model most overfitted on the training data. This is indeed indicated by looking how R^2_{test} are better than R^2_{train} only early epochs in Figure 6.7, and also by R^2_{train} having its best score on epoch 44, much later than 18. The figure stops around 50 epochs, as both scores gradually worsens and never recover (within the 150 training epochs) as shown in Figure 6.8.

Also, it is important not to let the spike for best R^2_{test} value go by unnoticed. The best score is reached after training from a $R^2_{test} \approx 0.01$ with an upward trend to $R^2_{test} = 0.225$. This could indicate among other things a random hit or too hard learning rate and/or momentum. First, if it were

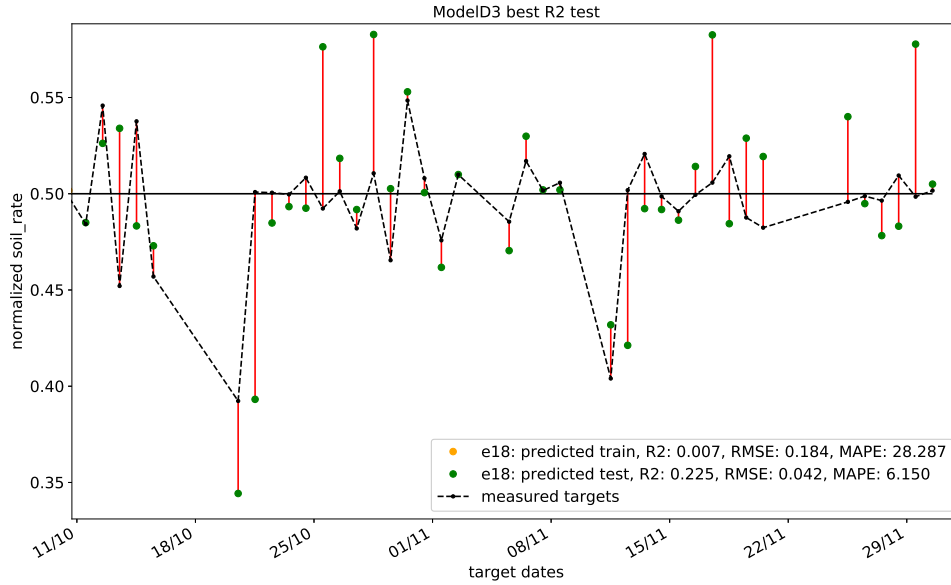


Figure 6.5: The predictions against their target values for test data using SHL on unsorted inputs. The black line on 0.5 implies $S_{rate} = 0$. Several values are spot on, with good R^2_{test} score and low RMSE and MAPE errors.

random – this model and other D models should not hit good on different node setups, which also happens on later results. So it is unlikely the inputs and weights happened to hit somewhat good, but it is possible. The more likely cause is learningrate $\eta = 0.02$ and/or momentum $\alpha = 0.9$ were too powerful, effectively pushing the network away from from the good weights unable to recover.

An explanation of the fluctuance in the R^2 scores may be the update function or the uncertainties in the target values. The update function used is ADAM (Eq. 3.8), which includes momentum. This momentum is in place to *push* the weight-error function past a local minima. Although on this data it looks like it keeps pushing it *away* from the best minima. This might be the result of too strong learning rate or momentum, or the minimas are not significantly lower than the error function. The uncertainties in target values could lead to inputs with similar footprints having opposite values. If such values exist, the training of weights may be shifted back and forth for each epoch (or weight update).

R2 from all data and R2 from training data

The scores encountered on R^2_{all} and R^2_{train} are available in the Appendix on Table C.1 & C.2 respectively were not as good as on R^2_{test} , but there are some elements to look at here as well. However, the primary reason these scores are poor is the epoch they mostly occur on is 0. Since the *best* scores achieved are encountered after *one* epoch of training, chance are the caught relation is more of a coincidence than an actual model of the

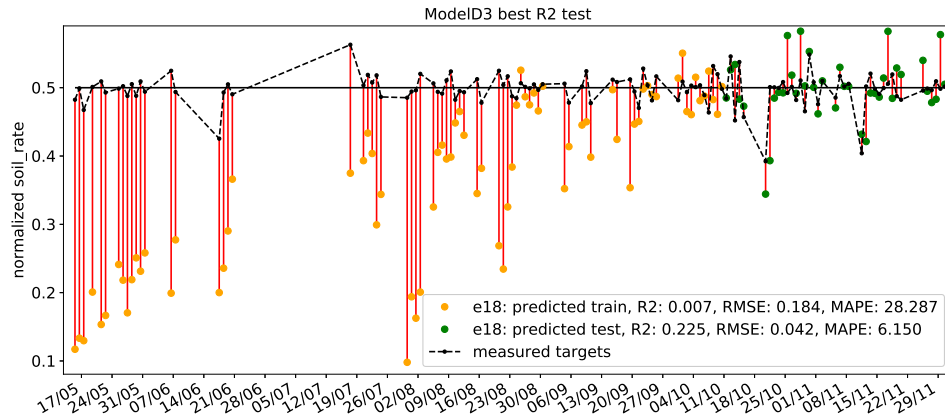


Figure 6.6: All predictions against their targets. Predictions on training data in orange, and test data in green. The test data consists of 39 points, approximately 33% of all targets. The black line on 0.5 implies $S_{\text{rate}} = 0$. As indicated, the test data performs vastly better than the training data for prediction.

input/output relation. However, because the models hit *something* with the initial training, there could be just this amount of relation available with the given uncertainties. Another reason to evaluate some of the scores are the *models* which occurred on all the top 10 scores (including R_{test}^2); models D3, D4 and D5. These models are always among the top 6 scores, and few from the same setup (same number of hidden nodes). This indicates *different* neural networks setups and weights achieved similar conclusions during training of the D models. Although two of the scores occur at epoch 0, the rest is a result of extended training. This strengthens previously indication of a relationship between RH_{avg} and the defined S_{rate} .

6.2.2 Model scores with sorted inputs

The following scores were created from training and testing the models 150 times on SHL with the 11 node setups and using *sorted inputs* defined in subsection 4.4.3.

R2 scores on test data

As with unsorted inputs, the best R^2 scores were achieved on the R_{test}^2 as seen on Table 6.3. Unlike the unsorted best R_{test}^2 , the best score and overall scores were not as good, and with other models among the top10.

The best was model D2 with $R_{\text{test}}^2 = 0.183$ and the test-predictions in Figure 6.9 look somewhat good. Around 16 of the 39 points are on the same side of the middle line as their targets.

More surprisingly were Model E5 on #2 with $R_{\text{test}}^2 = 0.168$. Earlier, E5 only appeared on the R_{all}^2 , and not on the R_{test}^2 . However, looking at the predicted values on Figure 6.10a reveals the predictions were much the same regardless of input. This proves the R^2 score are indicators, and

Table 6.2: The top 10 best R^2 values encountered on the test data with SHL on unsorted inputs.

#	Model	Hidden nodes	epoch	R^2_{train}	R^2_{test}	R^2	RMSE	MAPE
1	ModelD3	14(+2)	18	0.016	0.225	0.013	0.042	6.150
2	ModelD5	7(-5)	20	0.002	0.198	0.026	0.066	10.578
3	ModelG3	18(+5)	4	-0.150	0.195	-0.035	0.051	8.994
4	ModelD4	17(+5)	12	0.010	0.147	0.008	0.072	11.939
5	ModelF2	10(-3)	6	-0.343	0.145	-0.182	0.033	5.108
6	ModelB1	8(-5)	5	-0.147	0.137	-0.054	0.048	7.387
7	ModelB3	8(-5)	16	-0.136	0.126	-0.043	0.055	9.374
8	ModelG2	14(+1)	4	-0.053	0.118	0.009	0.075	12.779
9	ModelG1	12(-1)	0	-0.064	0.097	-0.005	0.080	12.404
10	ModelK2	17(+3)	29	-0.027	0.093	0.020	0.065	10.878

Table 6.3: The top 10 best R^2 values encountered using SHL on the test data with sorted inputs have a little lower values than the unsorted scores on test data.

#	Model	Hidden nodes	epoch	R^2_{train}	R^2_{test}	R^2	RMSE	MAPE
1	ModelD2	7(-5)	14	0.004	0.183	-0.004	0.042	6.545
2	ModelE5	10(-3)	16	-0.010	0.168	-0.023	0.038	5.610
3	ModelF2	9(-4)	10	0.001	0.160	-0.010	0.042	7.064
4	ModelG3	13	8	-0.024	0.132	0.032	0.070	11.207
5	ModelE4	11(-2)	18	-0.010	0.130	-0.018	0.066	11.753
6	ModelD4	15(+3)	11	0.004	0.129	0.000	0.069	11.464
7	ModelG4	10(-3)	18	-0.007	0.122	-0.020	0.044	7.465
8	ModelI5	19(+5)	8	-0.008	0.118	-0.021	0.045	7.534
9	ModelF5	13	16	0.003	0.115	-0.001	0.065	10.987
10	ModelI2	13(-1)	12	-0.070	0.094	-0.031	0.033	5.436

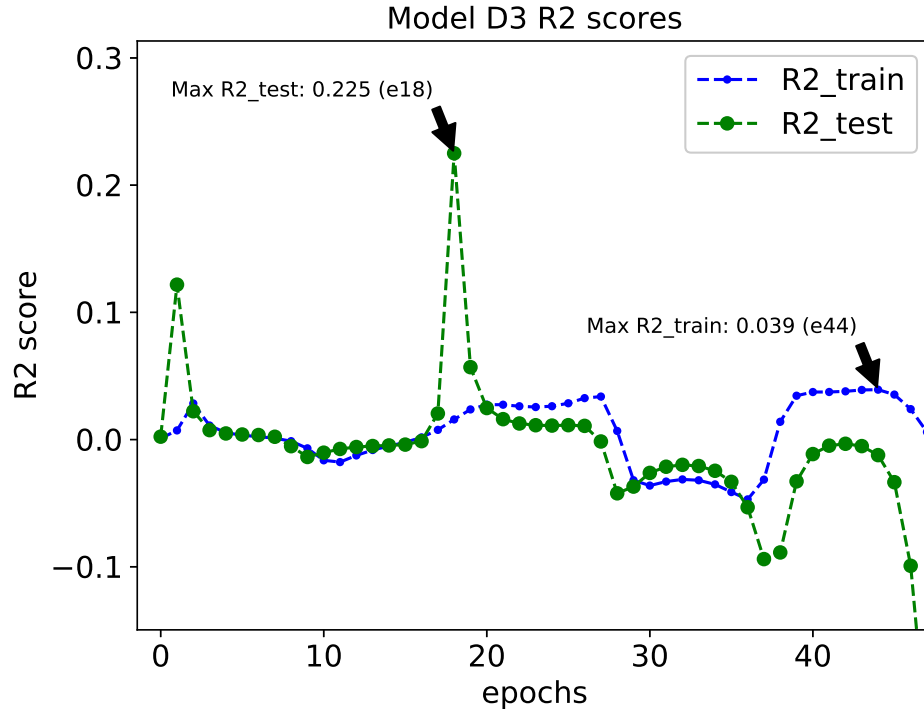


Figure 6.7: The scores of Model D3, with the maximum points annotated.

prediction plots are needed for confirmation. Although the two lowest target values received yet again indication of prediction. From other prediction plots, both values, with the last especially, are often well predicted. Because the R^2 only can be used as indication, the rest of top 10 R^2_{test} prediction plots are included in the Appendix in subsection C.1.

From all the prediction plots for the sorted input scores, both E4 and E5, together with I5, show poor predictions contrary to their R^2_{test} scores. These models similarities are their temperature features ($T_{\text{env.avg}}$, $T_{\text{mod.avg}}$) and the short intervals of 10 and 15 minutes. Only model I2 on #10 includes temperatures as feature, but this model has 6 hour intervals. It is thus suspected the sorted temperatures contribute more noise than valuable data for shorter intervals. A last interesting observation from the R^2_{test} values are RH_{avg} is a feature in *every* model.

R2 scores on train and all data

The R^2_{all} scores on Table C.4 looks more poor than the R^2_{all} scores from the unsorted input, both top score and in general. Still, some interesting trends continues with Model D3, D4 and D5 again among the top 10, together with F3 and G3. The most notable difference is model I2 and I3 are present on the sorted input scores. Model I were not among top 10 on any of the unsorted runs. It may seem that sorting the temperature features on the longer intervals provide new models some valuable data, but in general the sorting of input features lowers scoring on the previously well performing models.

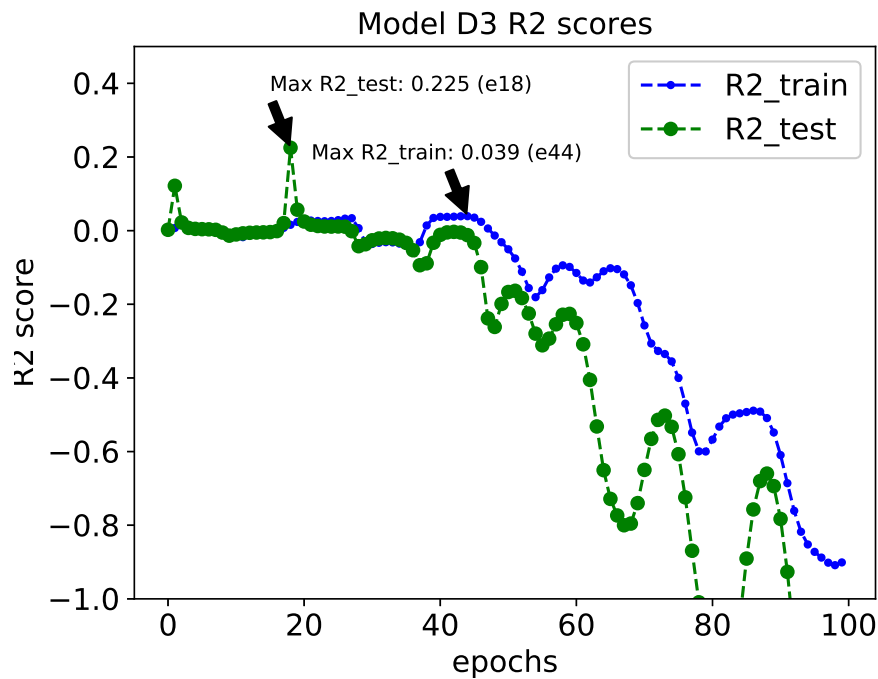


Figure 6.8: The unzipped scores of Model D3, with the maximum points annotated.

Finally, the R^2_{train} scores on Table C.5 in the appendix continues R^2_{all} observation with most models the same as in the unsorted score. One surprise is the addition of I5 on the top spot. However, the score of R^2_{train} is again not representative for prediction when looking at the poor predictions in Figure 6.11.

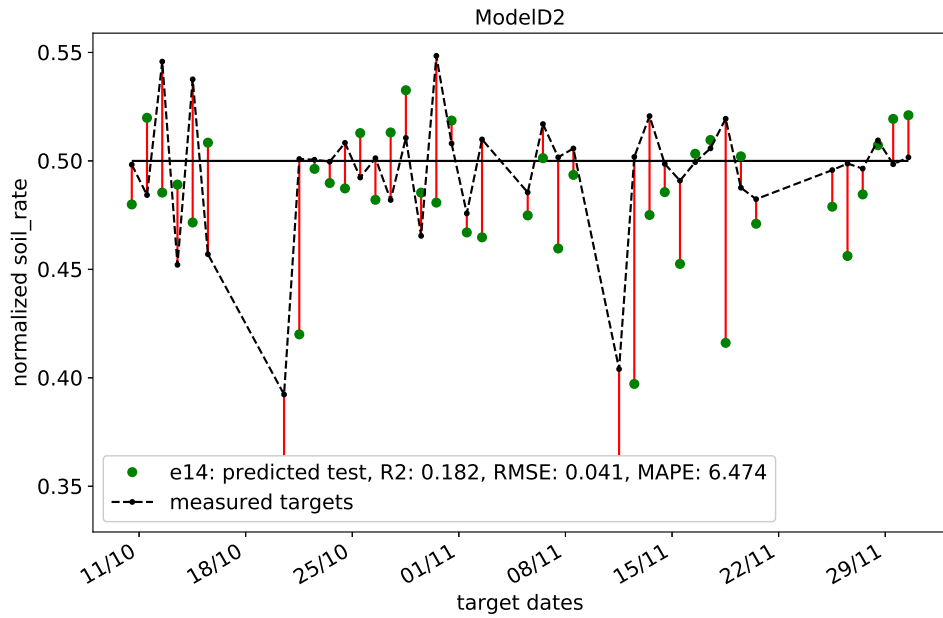
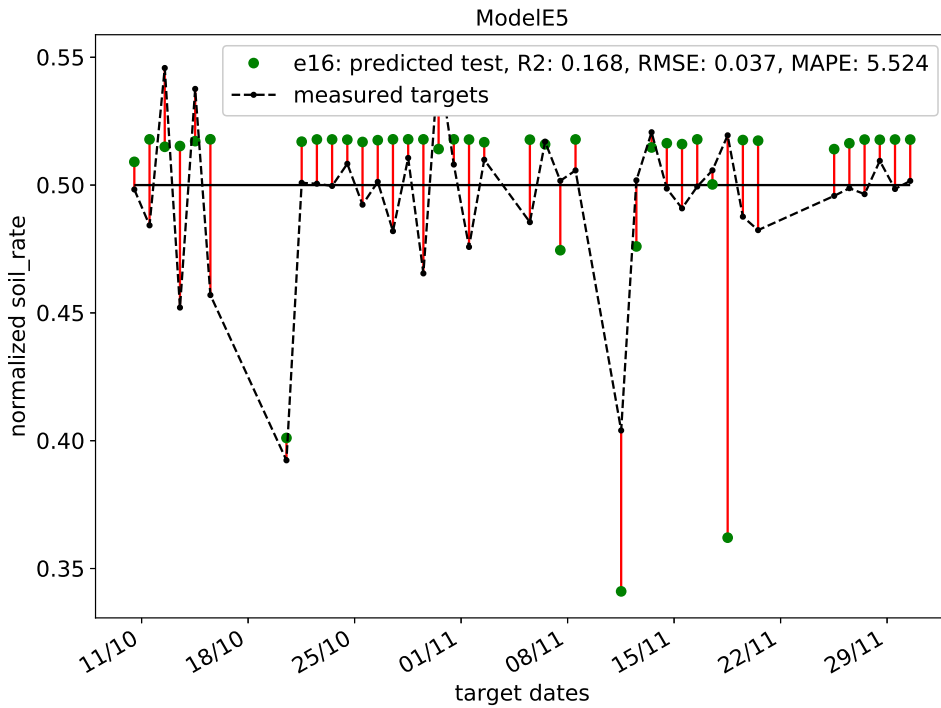
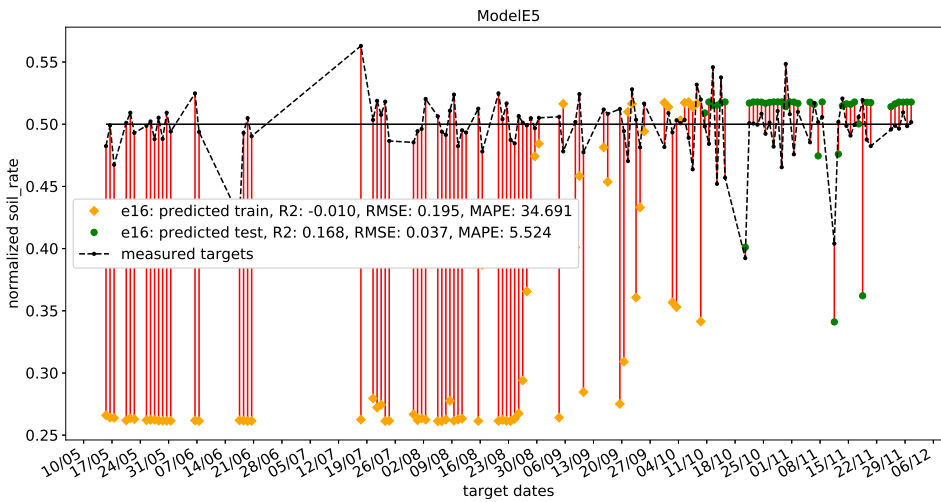


Figure 6.9: The test predictions values against their target values for Model D2 with SHL and sorted input features. There are a few predictions spot on their targets, but most is opposite of their measured value.



(a) The test predictions values against their target values for Model D2 with SHL and sorted input features.



(b) All predictions against their targets for E5 with sorted input features and SHL. Again, predictions on training data are poor compared to the predictions on test data.

Figure 6.10: The best predictions for Model E5 for whole data and test data using sorted input features and SHL.

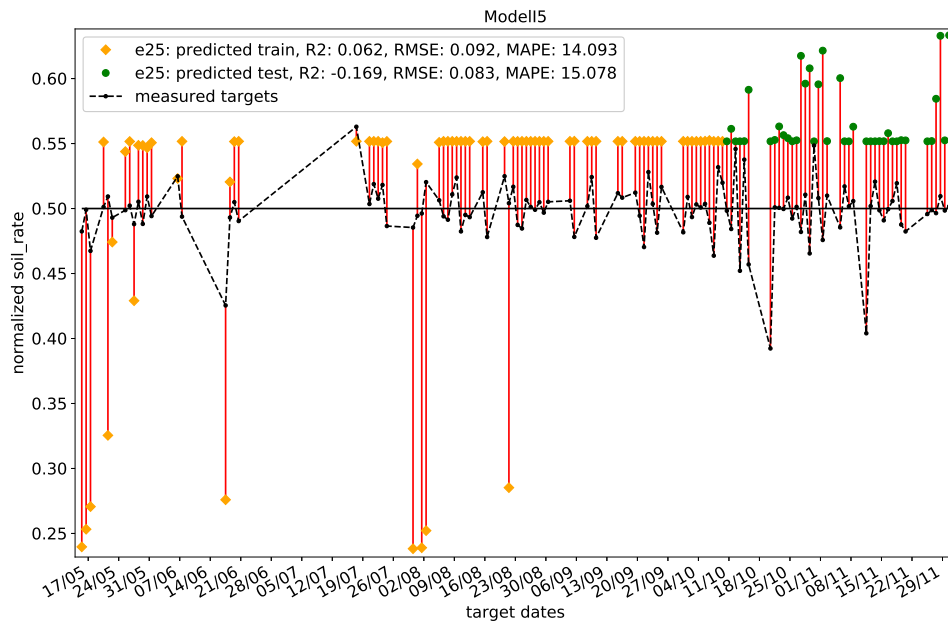


Figure 6.11: The predictions values against all targets on Modell5 from the best R^2_{train} scores. However, it does not show much variance in output targets.

6.3 Deep learning models

The deep learning (DL) models were trained and scored similar to the single layer models, 150 times for 11 different node setups. The only difference is the hidden layers consists of either 2 or 3 layers, depending on the number of inputs to the models. For each change in node count, all layers are changed. This means all the hidden layers have the same number of nodes at all times.

6.3.1 Model scores with unsorted input features

The R^2 scores encountered during training and scoring the deep ANNs are much alike the single layered setups on R_{test}^2 values. Additionally, the deep learning setups encountered better R_{train}^2 scores.

The most exciting result from deep learning was model D again achieving the best score on both R_{train}^2 and R_{test}^2 . Again indicating the influence humidity has on the S_{rate} . One difference from the single hidden layer scores were the setups hitting top spots were models with more (and thus shorter) intervals; D4 and D5 with 10 and 15 minute intervals respectively. Except on R_{all}^2 , which is held by E5 (1st), E3 (2nd) and E4 (3rd). Model E consists of humidity (RH_{avg}) and the average temperatures ($T_{\text{env.avg}}$ and $T_{\text{mod.avg}}$). This confirms deep learning is better to identify the non-linear relationships between features. As such, better results from models with more input features are expected by using deep learning. This includes shorter intervals, as the increase in inputs from the intervals can resemble non-linear interfeature relationships. The most surprising score is B1 on #5 with $R_{\text{test}}^2 = 0.106$. B1 has 4 inputs, WS_{avg} and WS_{max} averaged and maxed over 12 hours. Model B had decent scores on both single hidden layer and deep learning, and both times with *few (long)* interval counts. This indicates the WS_{max} is the most important feature – as this value is not averaged, but the maximum value within the period. If WS_{avg} were the significant feature, more shorter intervals would be expected to perform better with deep learning. However, the longer and few intervals were best on both approaches – indicating WS_{max} are the important feature of the two.

Predictions

Looking at the predicted values for D5 (best on R_{test}^2) and E5 (best on R_{all}^2) in Figures 6.12 & 6.13, it seems like E5 is better suited to become a more general model than D5, despite D5's better R^2 score.

Other interesting predictions from the deep learning score tables are the better results from Model K5. K5 is the combination of nearly all features (RH_{avg} , WS_{avg} , WS_{max} , $T_{\text{env.avg}}$, $T_{\text{mod.avg}}$), and has rank #4 on R_{test}^2 and #9 on R_{all}^2 . The predictions on (unseen) test data for K5 in Figure 6.14 from R_{test}^2 shows similar promise as the other models visually prediction plots. It is also worth noting K5 has both lowest RMSE and MAPE of all the top 10

Table 6.4: The top 10 best R^2 values encountered on the train data of deep learning models with unsorted input features had better scores than the SHL setup.

#	Model	Hidden nodes	epoch	R^2_{train}	R^2_{test}	R^2	RMSE	MAPE
1	ModelD4	16(+4)	54	0.179	-1.516	-0.357	0.047	6.113
2	ModelD5	16(+4)	149	0.115	-0.584	0.023	0.050	8.057
3	ModelE5	16(+3)	45	0.077	-0.014	0.042	0.090	13.467
4	ModelE4	8(-5)	46	0.065	-0.062	0.030	0.093	16.097
5	ModelE3	16(+3)	2	0.042	0.009	0.042	0.131	22.301
6	ModelG4	12(-1)	13	0.042	-0.170	-0.031	0.074	12.582
7	ModelG1	9(-4)	5	0.033	-0.031	-0.016	0.166	32.753
8	ModelI5	14	10	0.029	-0.007	0.009	0.524	94.163
9	ModelD3	10(-2)	13	0.029	-0.004	0.004	0.120	22.740
10	ModelF4	12(-1)	5	0.024	-0.015	-0.012	0.246	46.221

Table 6.5: The top 10 best R^2 values encountered on the test data of deep learning models with unsorted feature inputs had somewhat similar scores as the SHL.

#	Model	Hidden nodes	epoch	R^2_{train}	R^2_{test}	R^2	RMSE	MAPE
1	ModelD5	15(+3)	7	-0.025	0.243	-0.013	0.069	11.494
2	ModelD4	11(-1)	7	-0.044	0.217	-0.006	0.057	10.164
3	ModelF4	11(-2)	4	-0.058	0.112	-0.018	0.090	15.313
4	ModelK5	13(-1)	9	-0.002	0.107	0.009	0.049	7.877
5	ModelB1	10(-3)	6	-0.190	0.106	-0.054	0.064	10.176
6	ModelG4	13	24	-0.020	0.103	0.000	0.068	10.520
7	ModelE4	17(+4)	3	0.004	0.080	-0.000	0.066	10.609
8	ModelG5	12(-1)	7	0.007	0.080	0.026	0.136	20.752
9	ModelD2	13(+1)	3	0.001	0.074	-0.001	0.072	11.828
10	ModelA4	7(-5)	1	-0.035	0.068	0.000	0.148	25.637

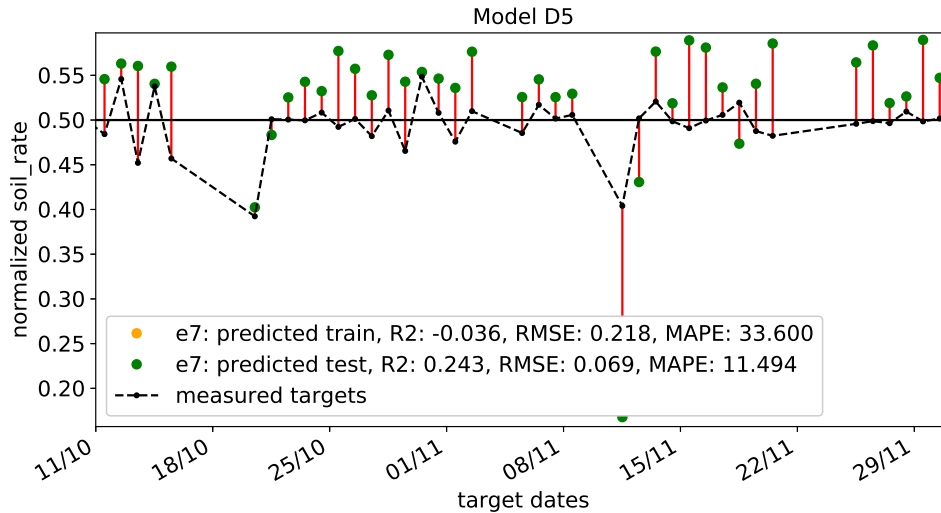


Figure 6.12: The predicted values against their targets for Model D5 using deep layer and unsorted input features. This was best R^2_{test} score of the unsorted deep learning models.

models from R^2_{test} scores on Table 6.5. This indicates the values are closest to the mean of the targets.

Lastly, the predictions from the model B1 on Figure 6.15 have nearly the same values and scores as K5, with a slight tendency to align to the middle score of 0.5.

6.3.2 Model scores with sorted input features

The sorted models were run with the same 11 different node setups with 150 epochs and 2 or 3 layers, with 1 or more feature types respectively. Again the best R^2 score encountered where overall and best from R^2_{test} as seen on Table 6.6.

However, unlike the poor results from sorted input features with SHL, these scores included *the best* score encountered of all tables with Model D4 having $R^2_{\text{test}} = 0.267$. Which is around 10% better than D5's $R^2_{\text{test}} = 0.243$ from the unsorted input features with deep learning. In addition, the other scores from top 10 were somewhat better than the R^2_{test} scores for the unsorted input features. A similar observation on both sorted and unsorted runs are the low presence of the most complex models. K5 were the only one in both top 10 R^2_{test} scores, and has the shortest intervals and all features except wind directions with 576 input nodes. However, the best scores using sorted input features had generally shorter (and thus more) intervals.

From these observations, it may indicate the sorting of input features are more important in non-linear relationships. As deep learning is better on non-linear relationships, and the sorted input features had both general best and highest score. This was also the reason for sorting the input features – to capture the interaction between features *regardless* of time of day they happened.

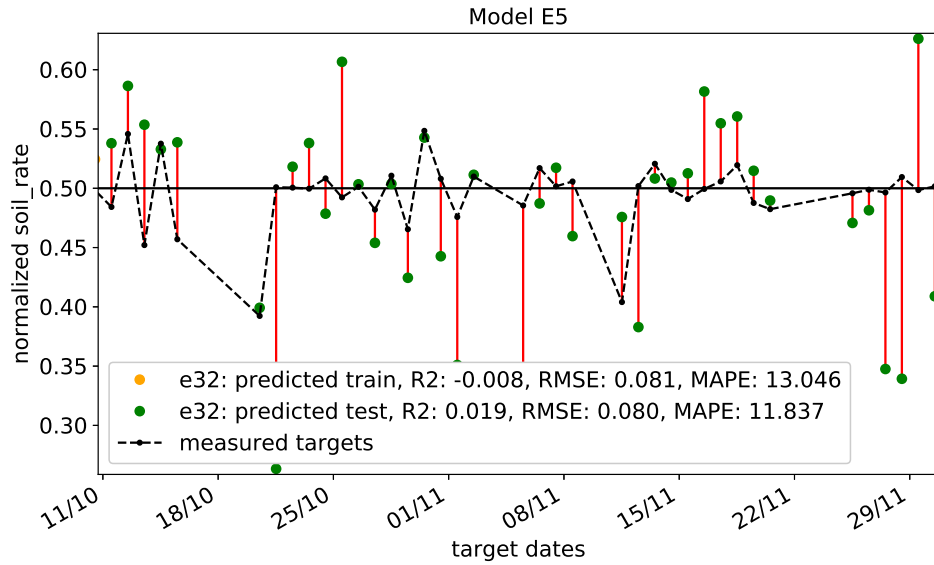


Figure 6.13: The predicted values against their targets for Model E5 using deep layer and unsorted input features. It had the best R_{all}^2 score of the deep learning models.

Prediction plots

Looking at the prediction plots for the top three models from Table 6.6, D4, F4 and D3 in Figures 6.16, 6.17 & 6.18 respectively, the R_{test}^2 score indication corresponds with the prediction values. Both D3 and D4 has RH_{avg} as its only input feature, whereas F4 also includes WS_{avg} together with RH_{avg} . It is difficult to be conclusive, but with 26 targets on the same side of $S_{rate} = 0$ as their targets, F4 may be more aligned its targets than D3 and D4. All three have some points spot on, although, not the same targets. This is expected between model types, but D3 and D4 should be more aligned. An explanation might lie in the neuron weights and the minima they start out from and end up with are different because of the interval lengths and inputs.

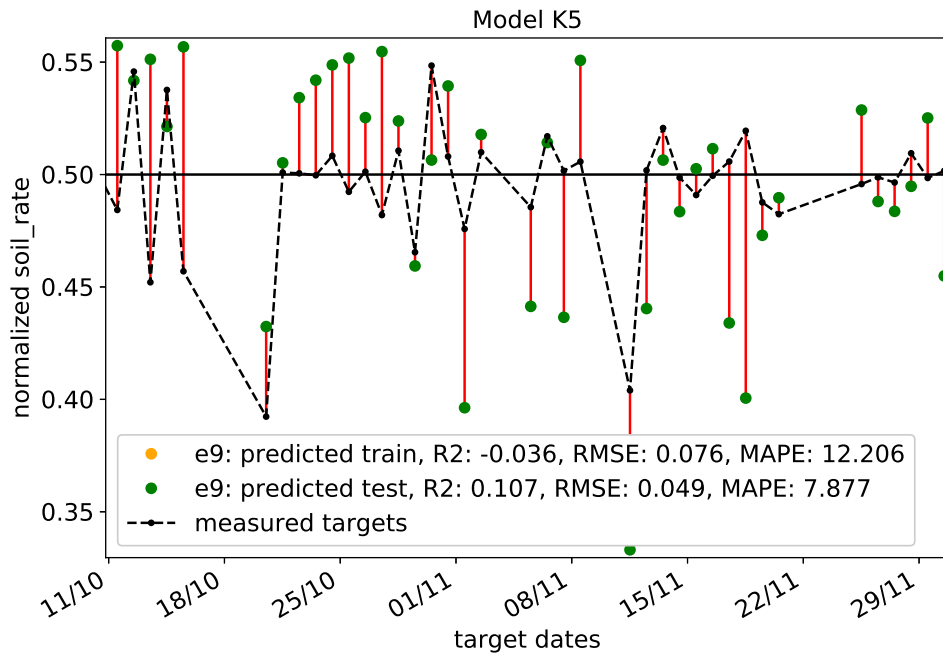


Figure 6.14: The predicted values against their targets for K5, the most complex model in the top 10 R^2_{test} scores.

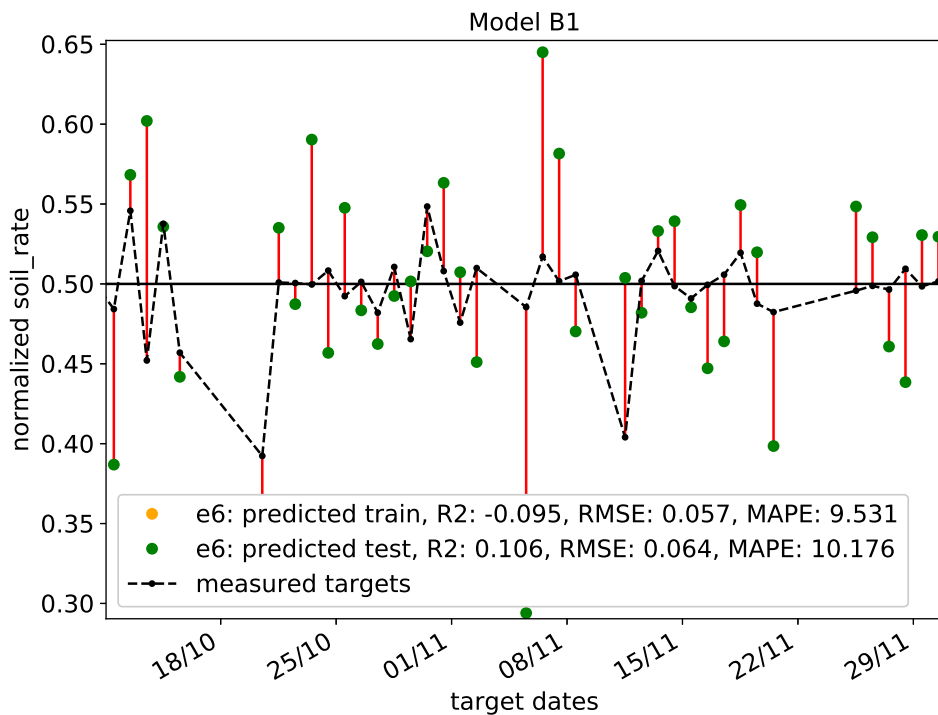


Figure 6.15: The predicted values against their targets for B1, the simplest model in the top 10 R^2_{test} show better potential than K5, even with only 8 input nodes.

Table 6.6: The top 10 best R^2 values encountered on the test data of deep learning models with sorted input features encountered generally higher scores, including the highest before the last run available in appendix..

#	Model	Hidden nodes	epoch	R^2_{train}	R^2_{test}	R^2	RMSE	MAPE
1	ModelD4	17(+5)	3	-0.015	0.267	-0.004	0.065	10.398
2	ModelF4	11(-2)	21	-0.123	0.214	0.012	0.068	11.045
3	ModelD3	14(+2)	5	0.009	0.202	0.002	0.040	6.377
4	ModelF5	8(-5)	55	-0.046	0.141	0.016	0.061	9.914
5	ModelB3	8(-5)	149	-2.121	0.139	-0.180	0.037	5.022
6	ModelK5	13(-1)	2	-0.013	0.127	-0.023	0.042	6.704
7	ModelD5	10(-2)	23	-0.018	0.126	-0.028	0.054	7.765
8	ModelB4	13	13	-0.080	0.125	0.009	0.070	9.849
9	ModelE5	9(-4)	3	-0.069	0.125	-0.044	0.075	11.459
10	ModelG4	16(+3)	33	-0.019	0.094	0.014	0.064	10.556

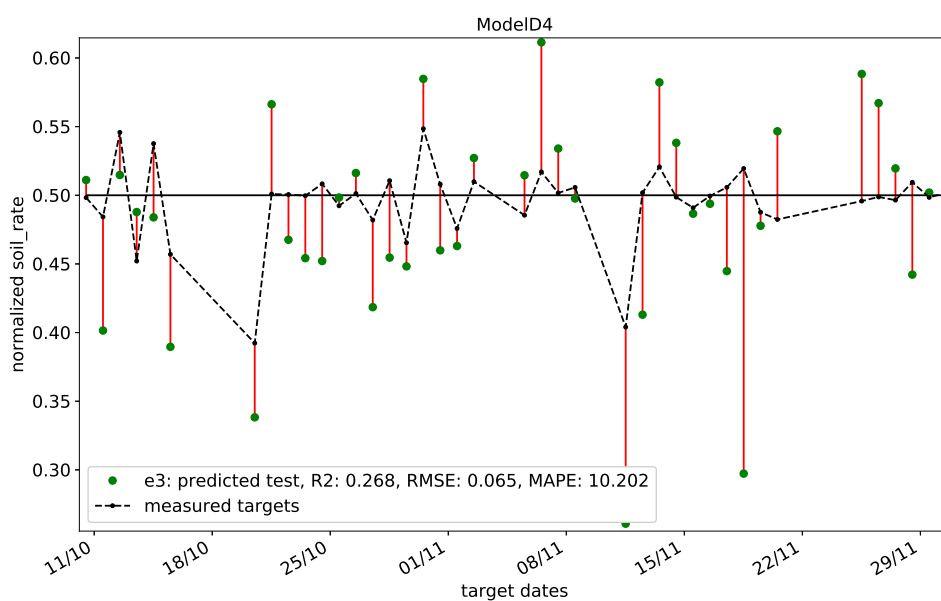


Figure 6.16: Prediction values against their test targets for the best encountered model, D4, with deep learning and sorted input features.

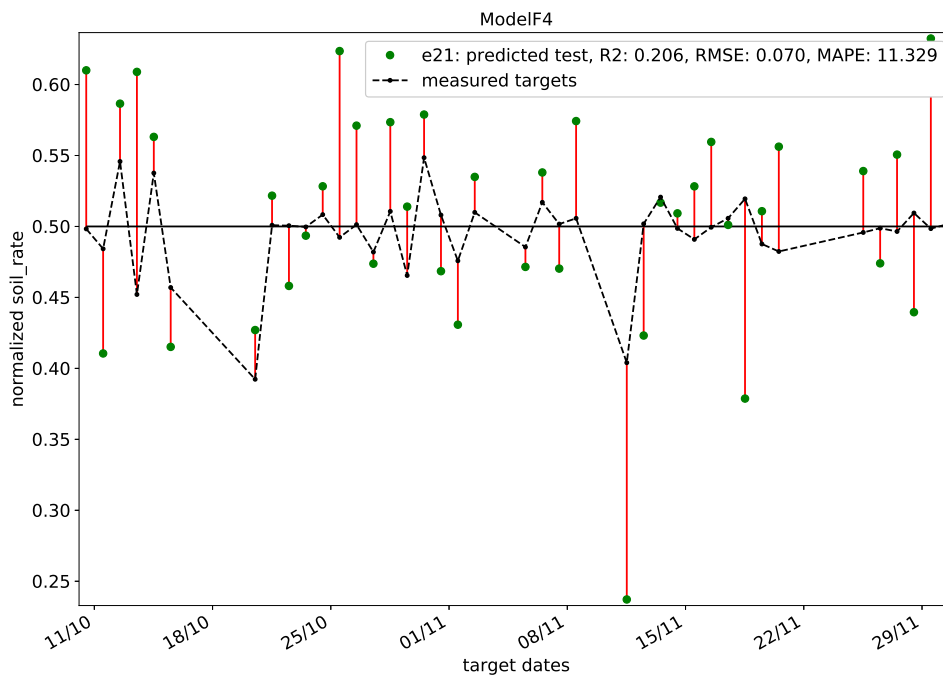


Figure 6.17: The prediction values against their targets for Model F4, the next best encountered model with deep learning and sorted input features. The values are more spread than the best model.

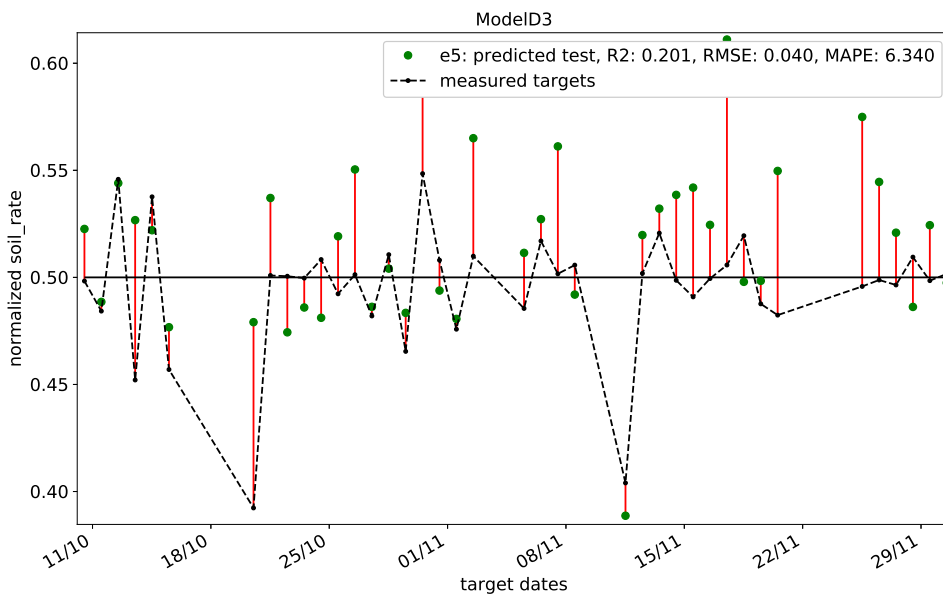


Figure 6.18: The predicted values against their targets for Model D3, the third best encountered score with deep learning and sorted input features. Although some values are spot on, most seem to go too high, suggesting poor predictability.

Chapter 7

Conclusion & recommendations

This chapter try to give som conclusive remarks regarding the results, and suggests recommendations for future work.

7.1 Feature relationship

The final goal of this thesis was to investigate suggested environmental variables relationship with daily soiling, by evaluating with machine learning. Although no conclusive relationship between the features and the defined soiling rate could be established, there are strong indications humidity is an important feature for the daily change yield ratio.

Applying more features together with humidity did not directly improve this indication. However, it did establish some more confidence to the indication of humidity having an influence on the defined S_{rate} . Nearly all models for every top 10 scores had humidity as a part of the feature inputs. Usually the models with the best scores on the table also had either humidity alone, or with few other features like average or maximum wind speed. Among the deep learning scores, humidity still played an important role with the larger models, even when only humidity was input. However, with deep learning more complex and non-linear relationships indicated maximum wind speed also has somewhat influence. Even more so with more (and shorter) intervals. This proved especially true using sorted input features, where average humidity and wind speed seemed like the best candidate of the presented predictions.

In addition, these results comply with what was expected from other studies in soiling. Together with the fact that several predicted values were *spot on* their target values. If more confidence were established that the defined S_{rate} are related to soiling, those targets would be stronger indications. As of now, the defined S_{rate} are likely affected by other factors, making it uncertain if the predicted targets are a result of soiling – or something else.

A conclusion that can be done with the achieved scores are wind directions seem to cause more noise as than valuable data. This is because

none of the top ten models in *any* score table had wind directions as its input features.

Some possible reasons the results are as they are:

- Preprocessing of data to feed a neural network is essential. Nearly half of the available input data were removed due to inconsistencies or clearly not target values because of soiling. In the early stages of predictions, R^2 scores with negative scores in the scale of 1000 were achieved. By removing the erroneous values, the end results showed much more consistent outputs.
- Normalization of the output values may have improved performance. Both larger and lower values were used earlier in the process with worse results. Although this could be due to other factors like update function, layer setup or valid dates or other, because none of the mentioned could be confirmed constant during the change to normalized targets.
- Neural network parameters are important. Another early discovery were the use of correct update function. Using regular SGD as update required several thousand epochs of training to achieve decent R^2 values. With research and discovery of RMSProp and ADAM, the required epochs and time were severely reduced. The change in update function also gave better results.
- The results seems to comply with the neural networks ability to detect non-linear relationships, only when hidden layers are more than one. This is what has been suggested by studies into ANN, and thus why it was included. With a single hidden layer, the lower input and interval models scored better, as was expected. With 2 and 3 hidden layers, more complex models achieved better scores, as expected. Deep learning is thus recommended when trying to investigate several input features or interval relationships.

It is planned to make all code public, if this is not possible or has not happened at the time of reading, but are desired -- contact author to ask for access to the private Git repository.

7.2 Recommendations for future work

Although the results of this thesis cannot conclude a relationship, the indications do provide some direction on where to go next.

7.2.1 Secure more reliable data

The data used in this project were quite unreliable. There were several dates where data was missing, and even wrong in the way it does not make sense. As an example, ambient temperatures above 170°C is not a reliable value to process. The need to assure such values are not part of datasets are

important, but time consuming and in worst case renders input or output values erroneous and useless. It may also be somewhat more accurate to use a two-diode-model instead of single-diode model for calculating maximum power point of the modules. However, given the uncertainties mentioned this is not expected to impact much.

7.2.2 Include data/measurement of airborne particles

A recent study indicated the particle composition of the air could predict soiling levels by around 80% (Micheli and Muller, 2017). Including these features as inputs would be expected to increase prediction accuracy. With better accuracy it could also indicate *how well* this model performed by comparing with and without the particle composition.

7.2.3 Daily cleaning of a reference module

This is likely the single most important recommendation for next step in the investigation of soiling. The uncertainties in this thesis output values could be the reason higher predictions were not achieved. With higher degree of certainty the target values are *only* because of soiling, a relationship may be discovered. The other studies within the same project showed module yield ratio is highly dependent on the magnitude of irradiance, using the defined S_{rate} based upon this irradiance are suspected to be the main source of error in target values. Having the possibility of comparing a dirty module against a clean module on a daily basis would remove this dependency.

7.2.4 Expanding the neural network

As each model only had 11 different setups based upon their input nodes, considering using other update functions, learning rates, momentum values and other machine learning parameters could improve performance of the neural network. Learning rates of 0.015 and 0.01 were done earlier in the thesis. However, they did not improve the performance at that point. With better datasets as mentioned previously, this might change.

7.2.5 Other machine learning techniques

Artificial neural network are good predictors, the one used in this thesis are not the only way to use it. Two other neural network approaches considered were:

- *Recurrent neural network* (RNN) discussed in Chapter 3 are indicated to be the best regression-predicting machine learning algorithm there is. However, it is harder to implement than regular ANN and with this thesis varying date of input, having a recurring loop from previous values would not be as beneficial. If data were more reliable and better with higher resolution – a daily footprint could be captured by the RNN instead of feeded as daily value. RNN would then be recommended.

- *Convolutional neural network* (CNN) were considered. Although CNNs are best at classification tasks, the reason it was considered was by picturing the daily footprint as an image, ie. For an hourly model, all input features would be on the Y-axis, and the hours would be the X-axis, creating a 2-dimensional grid of the footprint. This would enable a more concise footprint of the daily values than the densely connected ANN used in this thesis.

Other machine learning algorithms unrelated to neural networks are available to be used with regression. Some suggestions are Support Vector Machines (SVM) and k-Nearest-Neighbours (kNN).

7.2.6 Increase the dataset

A longer period of study for more data points usually increases machine learning performance on both unique/special conditions and core values. Because the core values would likely be trained more often, and the unique/special conditions are thus observed at least once, influencing the model training.

Appendices

Appendix A

Issues with weather data

Table A.1: First the interpolated dates and their row counts, and then the invalid dates and their row counts. The date with 1441 values had a duplicate that was removed.

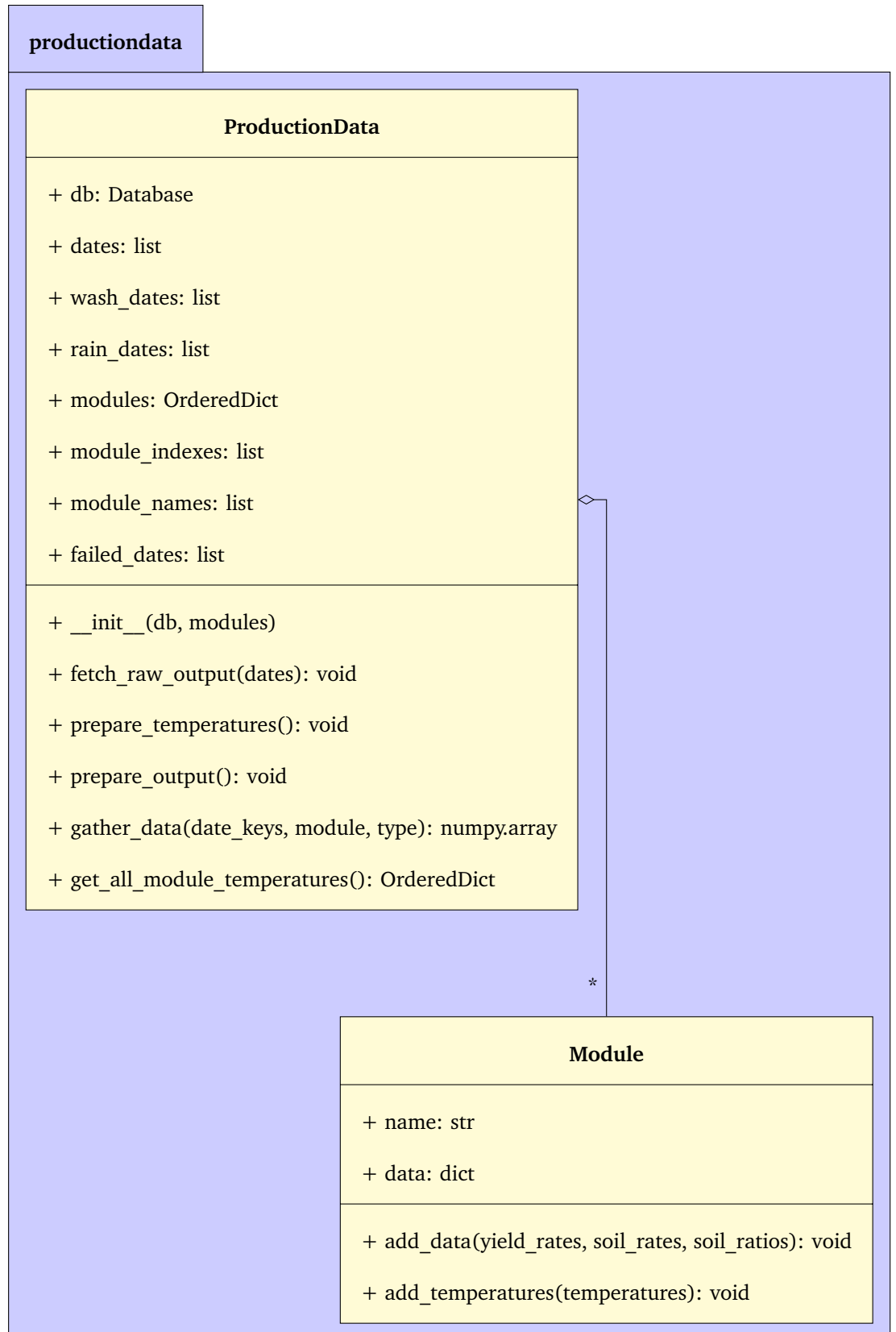
Interpolated dates									
Date	19/05	06/06	19/06	12/08	14/09	11/10	24/11		
rows	1438	1397	1441	1436	1437	1439	1435		
Invalid dates									
Date	08/06	21/06	22/06	24/06	25/06	26/06	27/06	29/06	30/06
rows	866	545	0	0	371	338	516	0	99
Date	01/07	02/07	03/07	04/07	06/07	07/07	09/07	11/07	12/07
rows	377	161	408	293	0	520	679	0	272
Date	13/07	13/08	02/09	13/09					
rows	706	1167	0	0					

Appendix B

UML diagrams

B.1 Data collection

Table B.1: An UML diagram of the 'ProductionData' class and the relevant variables and functions.



B.2 Models

Table B.2: An UML diagram of the 'Model' class and its variables and functions.

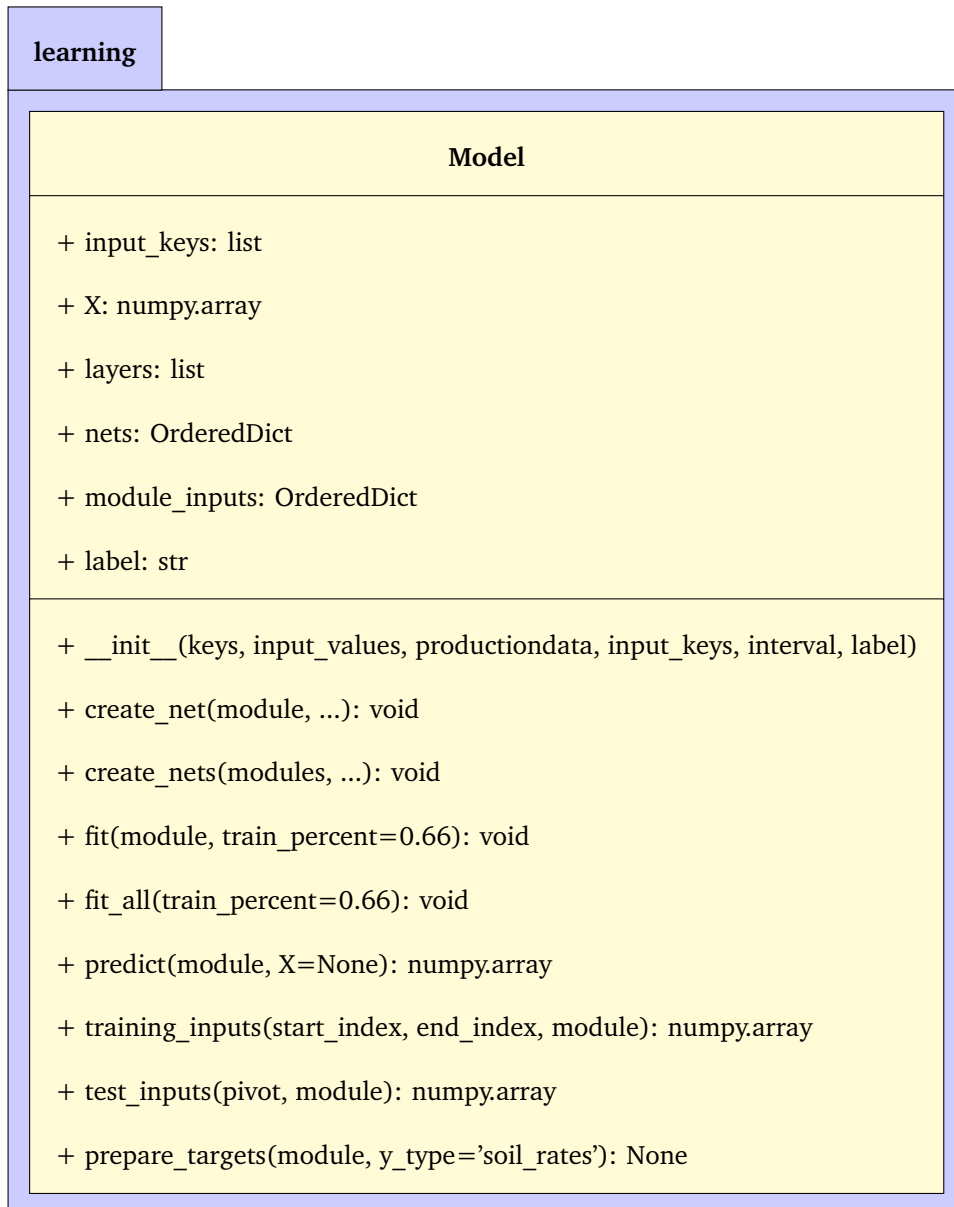


Table B.3: An UML diagram of the subclasses of 'Model'.

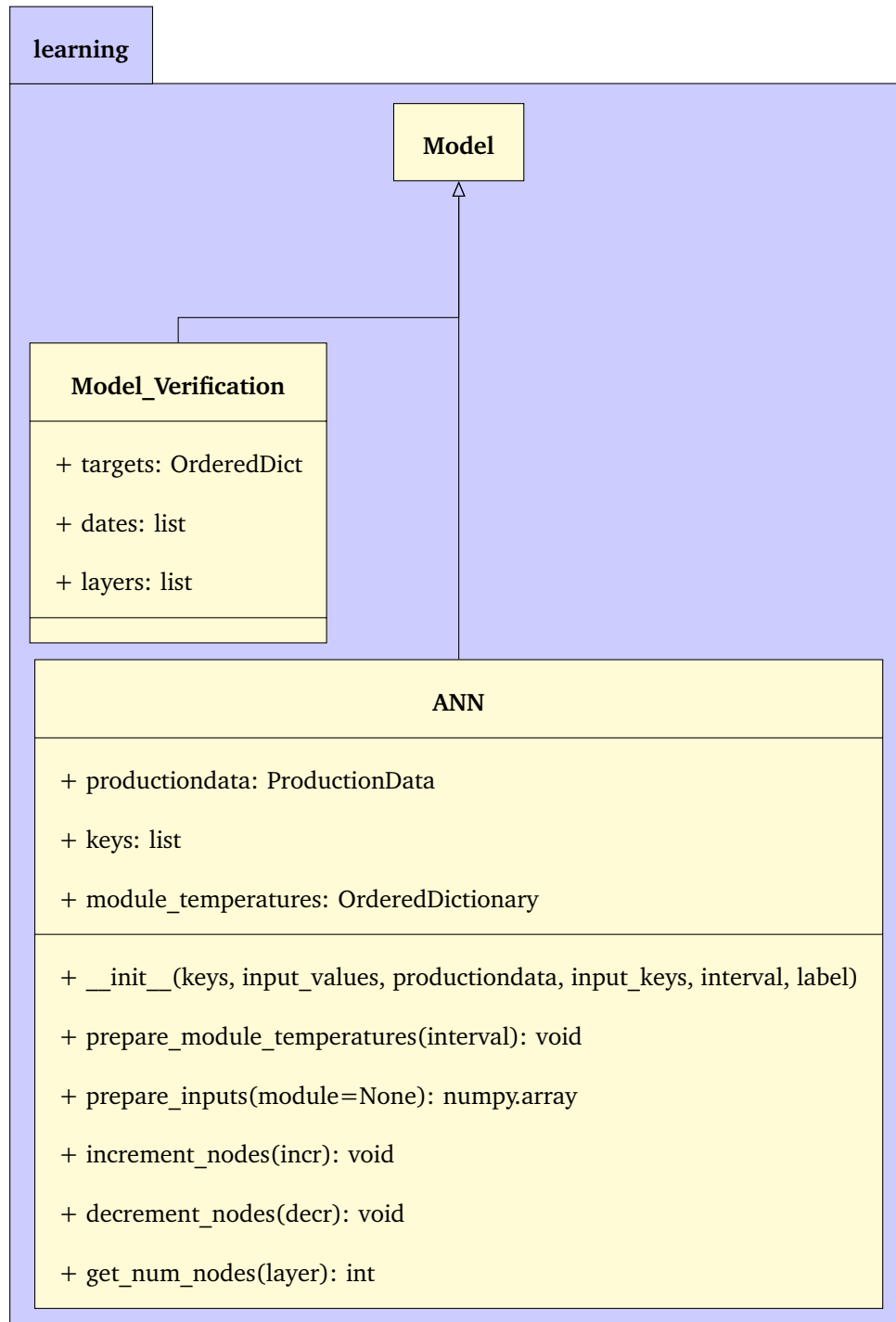


Table B.4: An UML diagram of the two subclasses for 'ANN'. They specify different layer setups with regards to their initialization.

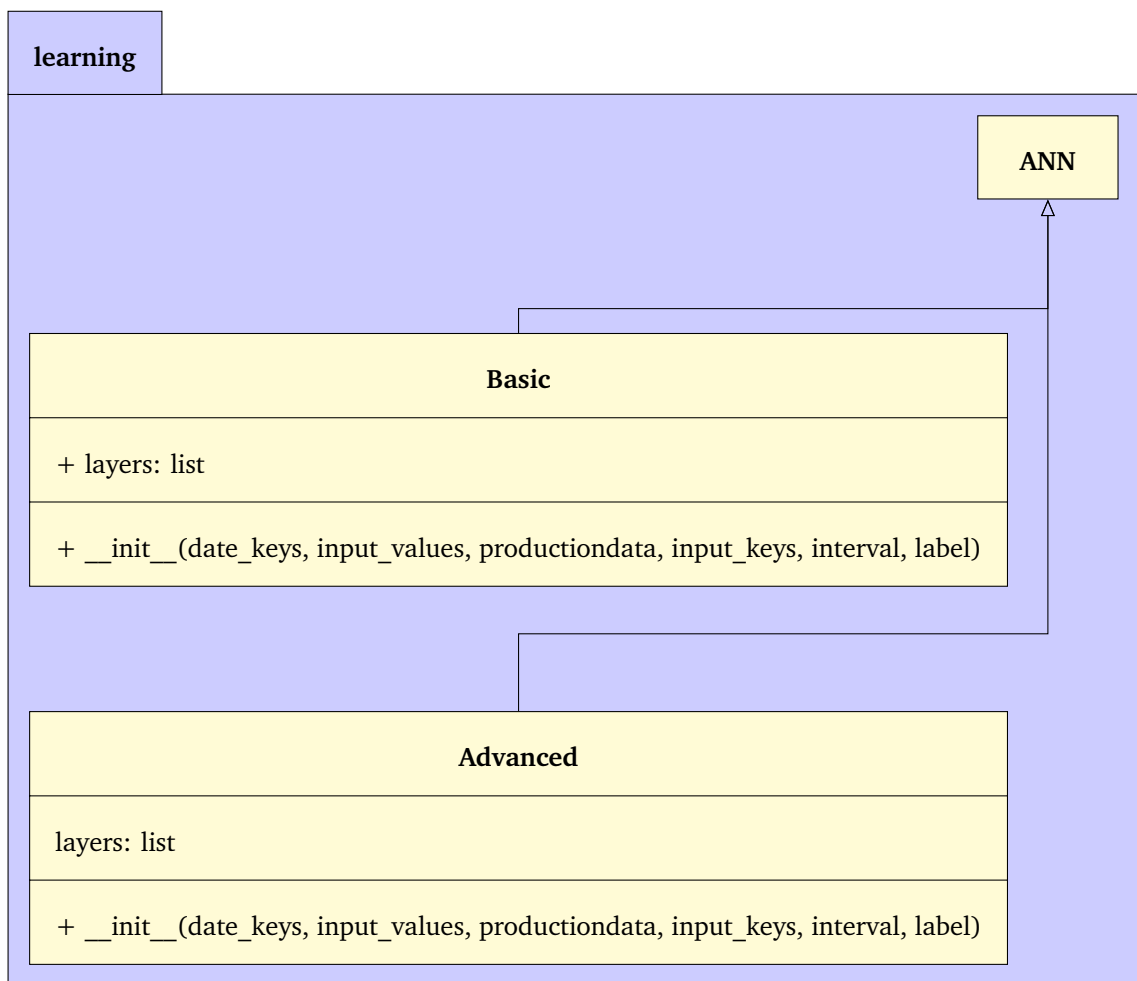
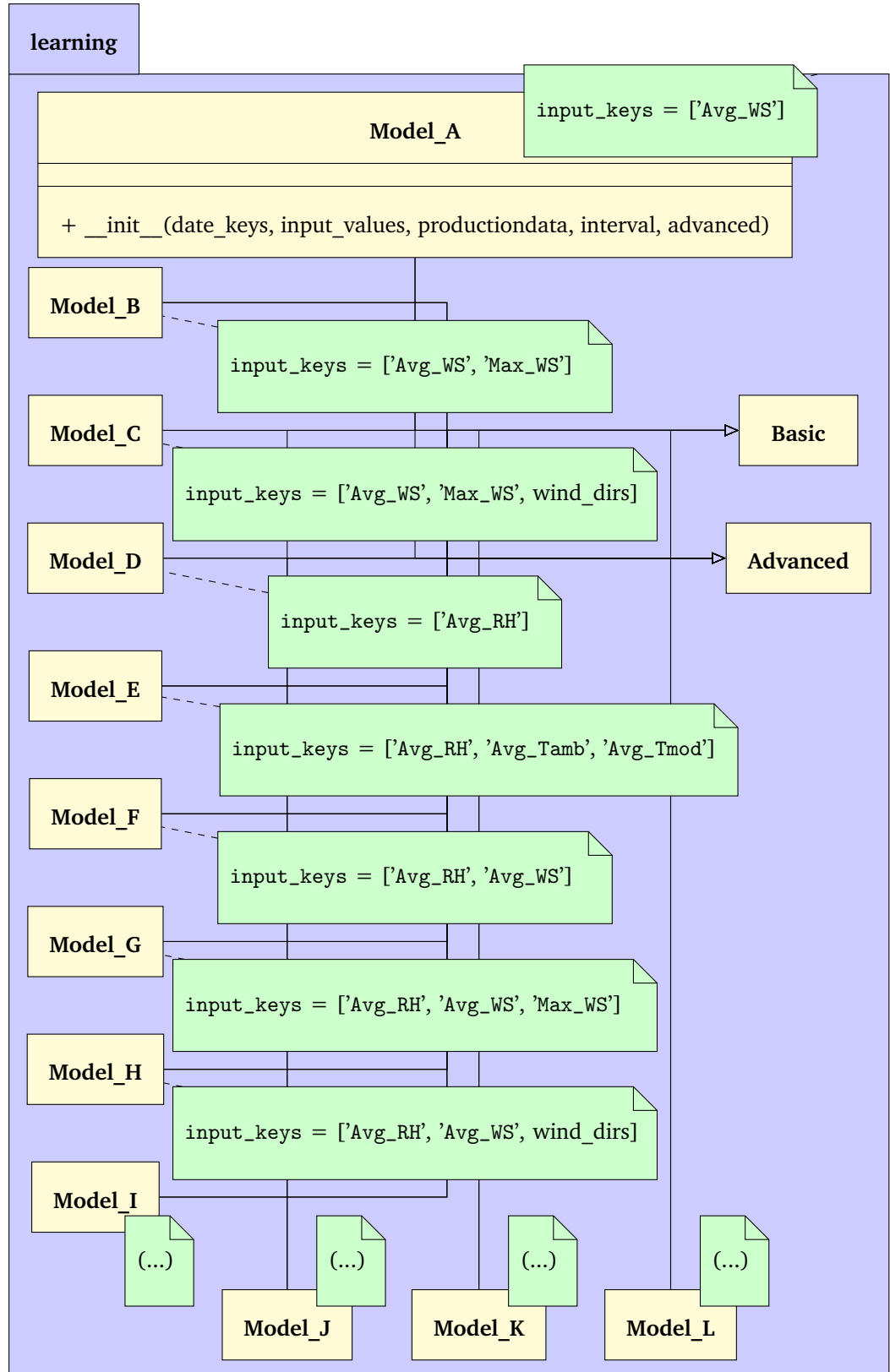


Table B.5: An UML diagram of all subclasses for both 'Basic' and 'Advanced'. These are the model classes that are initialized, and have predefined 'input_keys' as shown in notes for most classes.



Appendix C

Additional result tables and plots

C.1 Unsorted inputs SHL second run and plots

First are the R^2_{train} and R^2 scores for SHL unsorted input features on Table C.1

Because results were lost, a second run with SHL on unsorted input features were run. The scores are presented on Table C.3. The scores comply almost directly with the previous achieved scores. This gives further confidence on the models that gives the best scores, as once again Model D achieves new highscore, with $R^2_{\text{test}} = 0.286$.

Table C.1: The top 10 best R^2 values encountered with SHL on whole data for unsorted input features.

#	Model	Hidden nodes	epoch	R^2_{train}	R^2_{test}	R^2	RMSE	MAPE
1	ModelE3	14(+1)	0	0.059	0.024	0.059	0.121	21.188
2	ModelE5	11(-2)	0	0.077	-0.035	0.052	0.085	13.380
3	ModelD3	7(-5)	5	0.039	0.040	0.039	0.122	20.711
4	ModelF4	9(-4)	0	0.033	0.067	0.035	0.104	18.048
5	ModelD5	7(-5)	21	0.005	0.165	0.035	0.067	10.341
6	ModelD4	7(-5)	0	0.042	0.034	0.033	0.153	27.680
7	ModelI3	14	0	0.010	0.001	0.027	0.048	7.859
8	ModelG3	17(+4)	3	-0.021	0.055	0.026	0.181	27.986
9	ModelF2	11(-2)	4	0.012	0.084	0.025	0.054	9.014
10	ModelF3	11(-2)	0	0.044	-0.028	0.024	0.050	7.640

Table C.2: The top 10 best R^2 values encountered with SHL on the training data for unsorted input features.

#	Model	Hidden nodes	epoch	R^2_{train}	R^2_{test}	R^2	RMSE	MAPE
1	ModelE3	13	2	0.082	-0.017	0.036	0.162	27.662
2	ModelE5	11(-2)	0	0.077	-0.035	0.052	0.085	13.380
3	ModelD4	7(-5)	11	0.065	0.010	0.030	0.122	21.741
4	ModelD5	15(+3)	0	0.050	0.021	0.034	0.161	26.538
5	ModelD3	10(-2)	68	0.049	-0.147	-0.014	0.056	9.135
6	ModelE2	11(-2)	6	0.047	-0.191	0.016	0.052	8.391
7	ModelF3	11(-2)	0	0.044	-0.028	0.024	0.050	7.640
8	ModelF4	9(-4)	0	0.033	0.067	0.035	0.104	18.048
9	ModelF5	13	20	0.022	-0.087	0.001	0.095	15.951
10	ModelD2	17(+5)	10	0.021	-0.026	-0.022	0.113	22.094

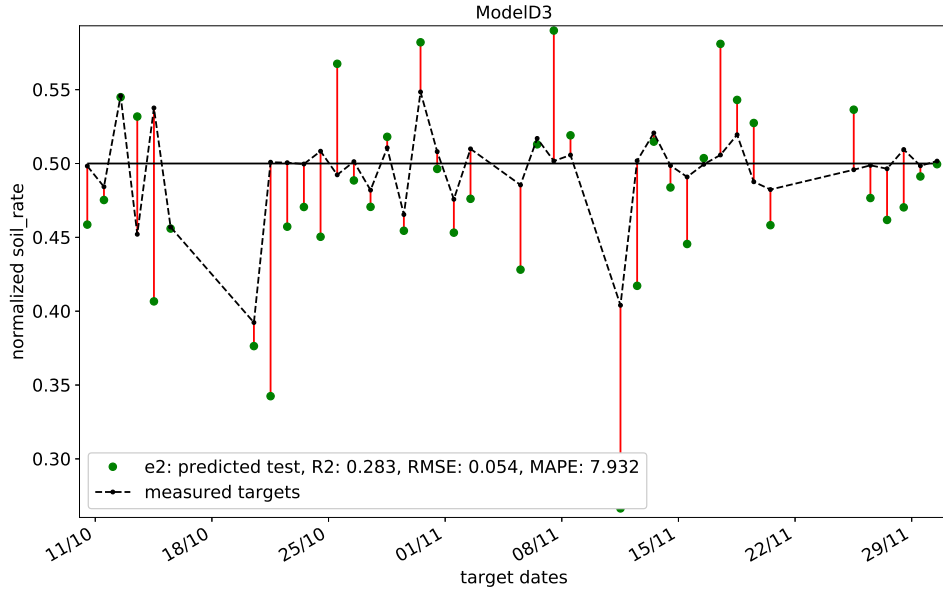


Figure C.1: The test predictions values against their target values for Model D3, #1 from last runs R^2_{test} with SHL and unsorted inputs.

Table C.3: The top 10 best R^2 values encountered on the test dataset after a second run with unsorted inputs on SHL.

#	Model	Hidden nodes	epoch	R^2_{train}	R^2_{test}	R^2	RMSE	MAPE
1	ModelD3	7(-5)	2	-0.008	0.286	0.006	0.055	7.932
2	ModelD2	11(-1)	11	0.002	0.245	-0.004	0.041	6.802
3	ModelD5	7(-5)	10	0.006	0.216	0.003	0.064	9.856
4	ModelG5	18(+5)	9	0.002	0.199	-0.004	0.055	9.985
5	ModelG2	16(+3)	27	-0.053	0.170	0.016	0.069	11.609
6	ModelD4	15(+3)	17	0.006	0.144	0.004	0.102	15.723
7	ModelF3	8(-5)	3	-0.077	0.138	-0.025	0.055	9.418
8	ModelK1	12(-2)	14	-0.062	0.123	0.031	0.065	10.996
9	ModelE2	14(+1)	2	0.007	0.095	-0.002	0.061	10.353
10	ModelG3	11(-2)	30	-0.182	0.086	-0.063	0.045	7.212

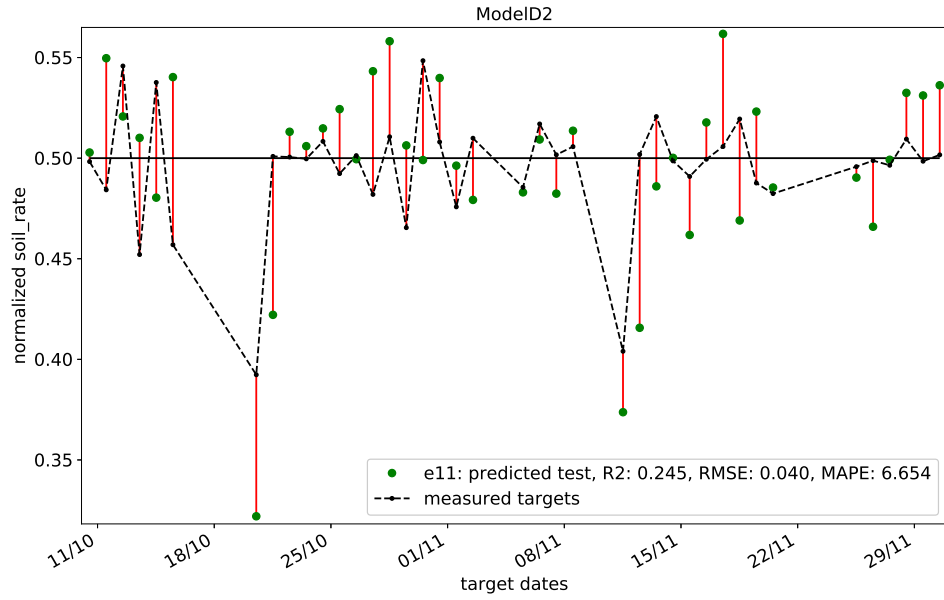


Figure C.2: The test predictions values against their target values for Model D2, #2 from last runs R_{test}^2 with SHL and unsorted inputs.

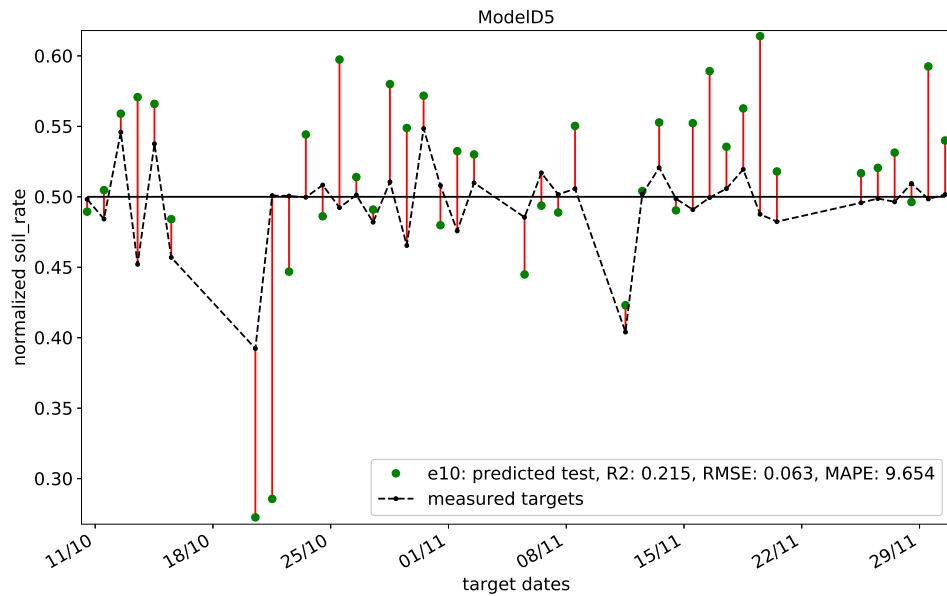


Figure C.3: The test predictions values against their target values for Model D5, #3 from last runs R_{test}^2 with SHL and unsorted inputs.

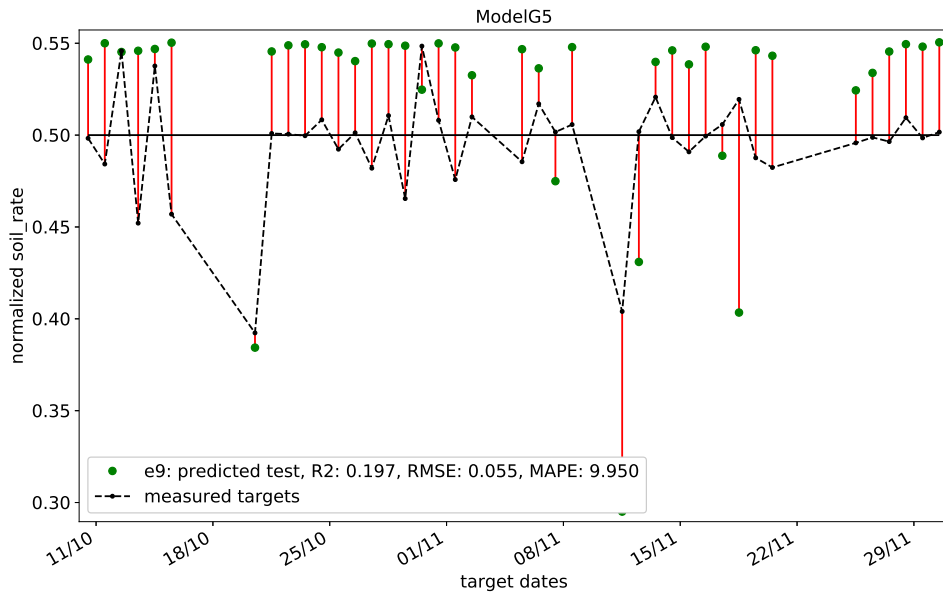


Figure C.4: The test predictions values against their target values for Model G5., #4 from last runs R^2_{test} with SHL and unsorted inputs.

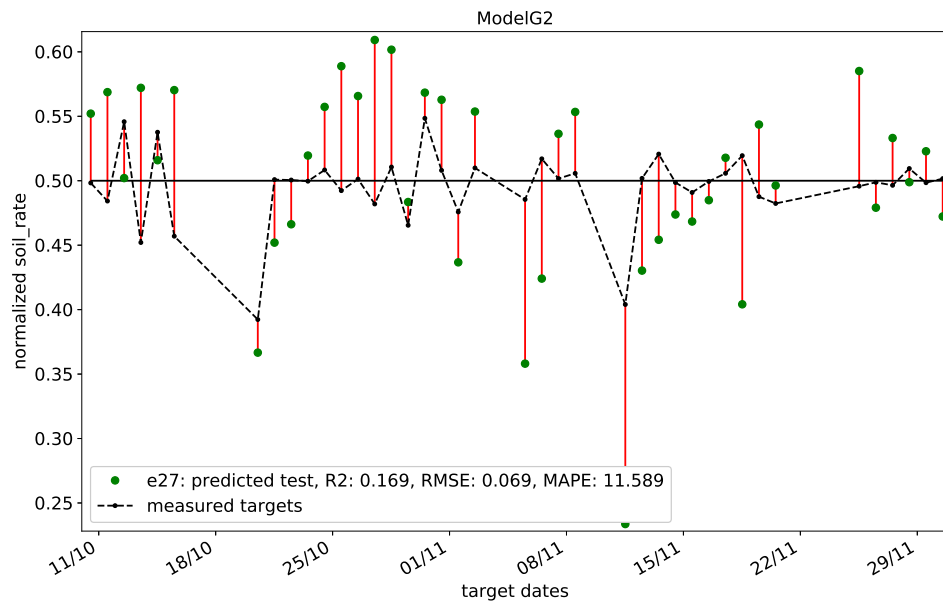


Figure C.5: The test predictions values against their target values for Model G2, #5 from last runs R^2_{test} with SHL and unsorted inputs.

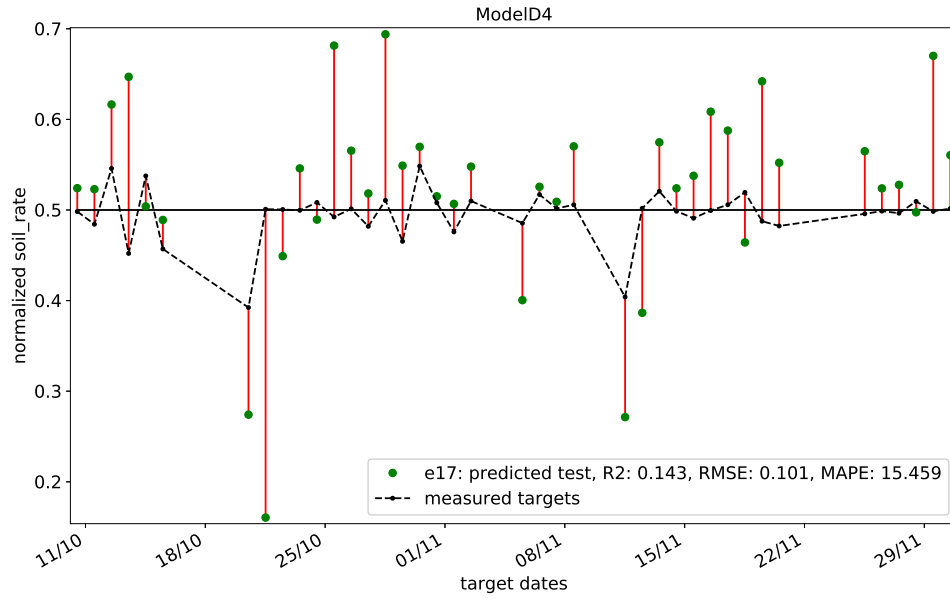


Figure C.6: The test predictions values against their target values for Model D4, #6 from last runs R^2_{test} with SHL and unsorted inputs.

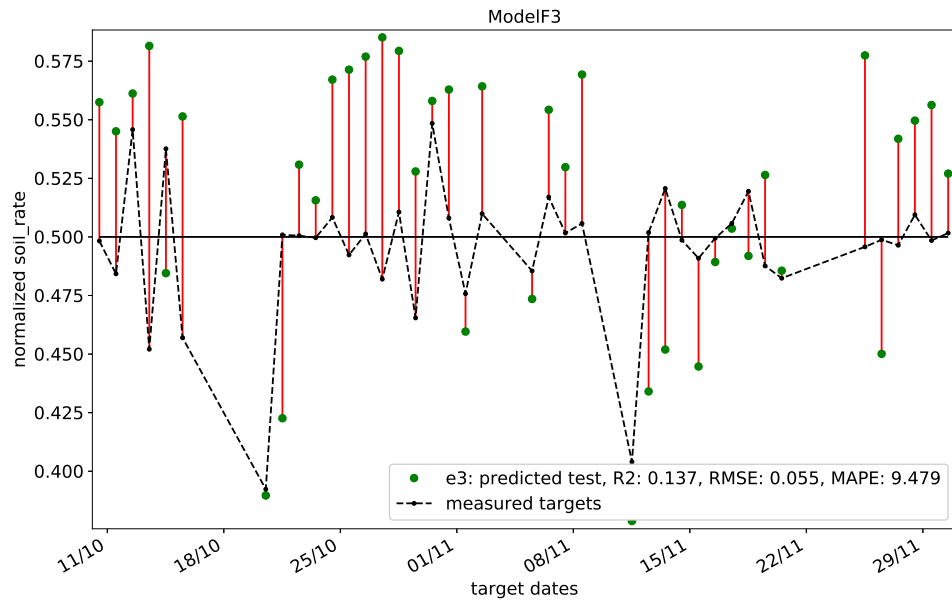


Figure C.7: The test predictions values against their target values for Model F3, #7 from last runs R^2_{test} with SHL and unsorted inputs.

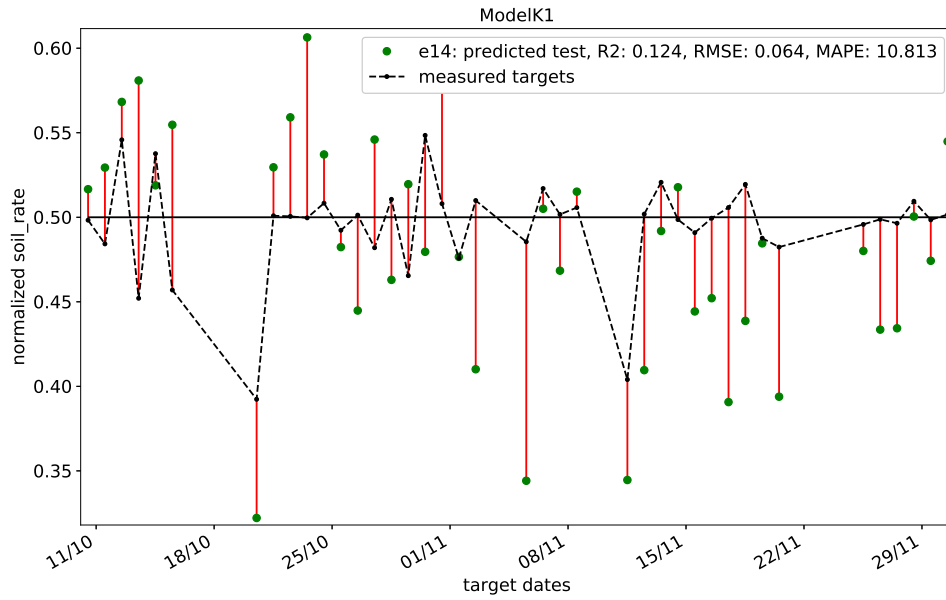


Figure C.8: The test predictions values against their target values for Model K1, #8 from last runs R^2_{test} with SHL and unsorted inputs.

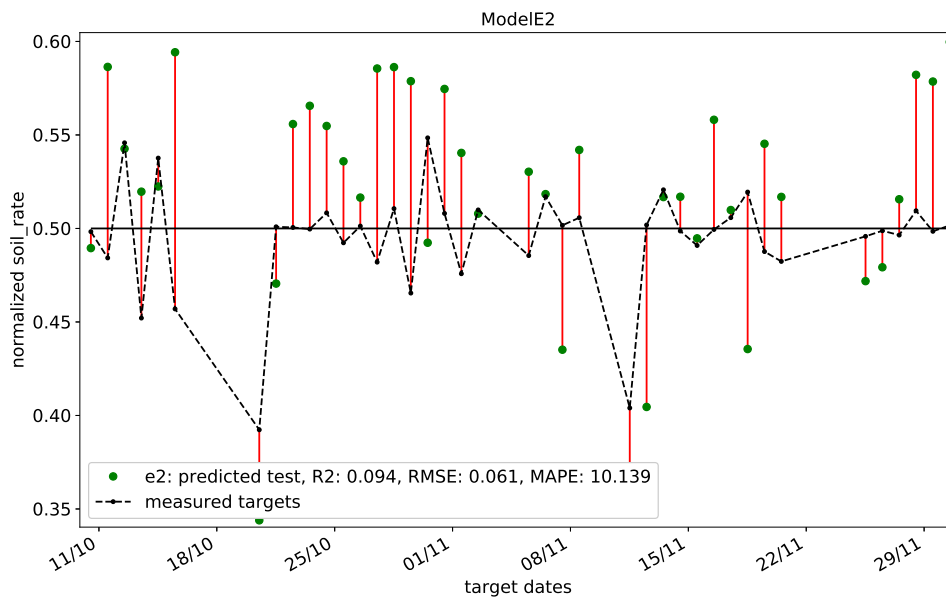


Figure C.9: The test predictions values against their target values for Model E2, #9 from last runs R^2_{test} with SHL and unsorted inputs.

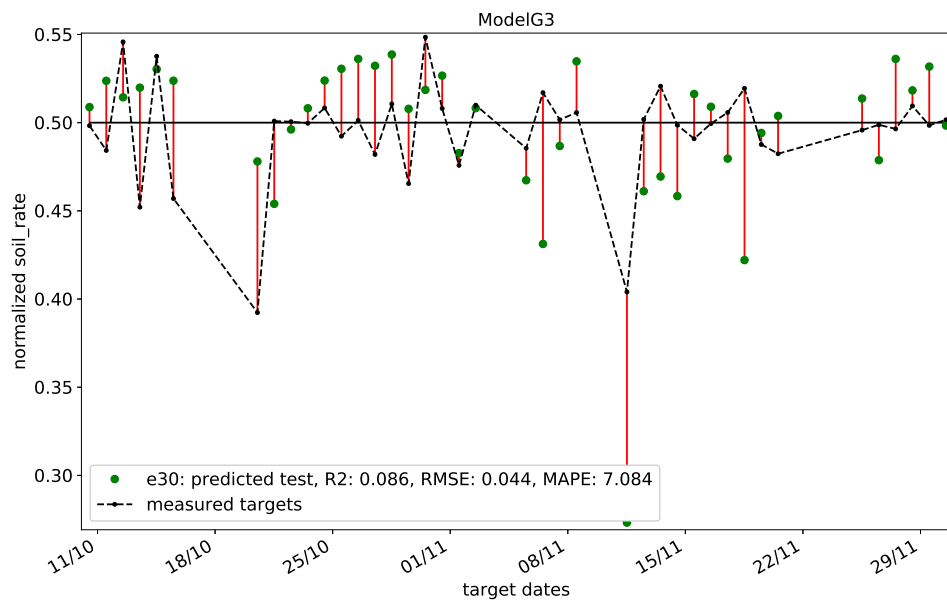


Figure C.10: The test predictions values against their target values for Model G3, #10 from last runs R^2_{test} with SHL and unsorted inputs.

C.2 Sorted inputs SHL plots

Table C.4: The top 10 best R^2 values encountered on the whole dataset for sorted inputs and SHL.

#	Model	Hidden nodes	epoch	R^2_{train}	R^2_{test}	R^2	RMSE	MAPE
1	ModelG3	13	8	-0.024	0.132	0.032	0.070	11.207
2	ModelI2	9(-5)	6	-0.002	-0.012	0.021	0.139	24.985
3	ModelE3	15(+2)	2	0.017	-0.008	0.018	0.150	26.133
4	ModelD4	7(-5)	3	0.022	0.005	0.015	0.218	38.840
5	ModelD5	11(-1)	1	0.023	0.005	0.015	0.202	35.645
6	ModelF3	9(-4)	3	0.005	0.037	0.009	0.180	34.157
7	ModelD3	15(+3)	18	0.028	0.002	0.009	0.144	23.091
8	ModelI3	17(+3)	2	-0.034	-0.008	0.009	0.200	38.411
9	ModelE5	8(-5)	7	0.022	-0.057	0.009	0.151	23.384
10	ModelG5	13	3	0.016	0.009	0.007	0.215	42.610

Table C.5: The top 10 best R^2 values encountered on the train data with sorted inputs and SHL.

#	Model	Hidden nodes	epoch	R^2_{train}	R^2_{test}	R^2	RMSE	MAPE
1	ModelI5	19(+5)	25	0.062	-0.170	-0.014	0.083	15.190
2	ModelE4	15(+2)	40	0.050	-0.043	0.006	0.161	25.268
3	ModelG4	16(+3)	26	0.037	-0.207	-0.051	0.065	12.405
4	ModelE5	8(-5)	8	0.036	-0.150	0.007	0.118	21.361
5	ModelD5	14(+2)	34	0.034	-0.191	-0.015	0.061	10.357
6	ModelF4	12(-1)	47	0.031	-0.430	-0.032	0.062	10.207
7	ModelD3	15(+3)	18	0.028	0.002	0.009	0.144	23.091
8	ModelF5	13	56	0.025	-1.306	-0.100	0.038	5.394
9	ModelD4	7(-5)	3	0.022	0.005	0.015	0.218	38.840
10	ModelE3	15(+2)	2	0.017	-0.008	0.018	0.150	26.133

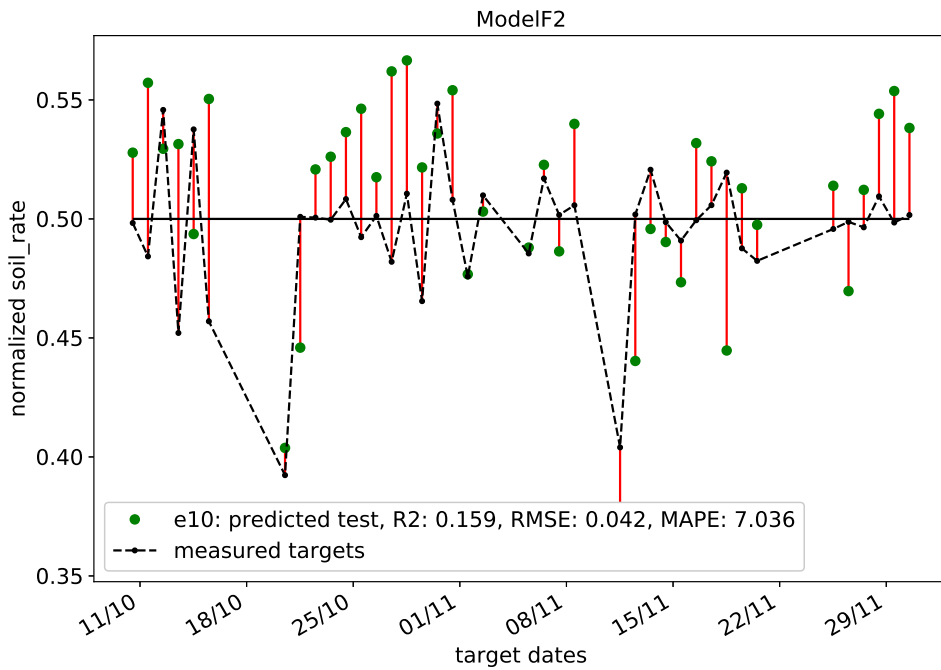


Figure C.11: The predictions on test data against their targets for Model F2 ranked #3 on SHL with sorted input features.

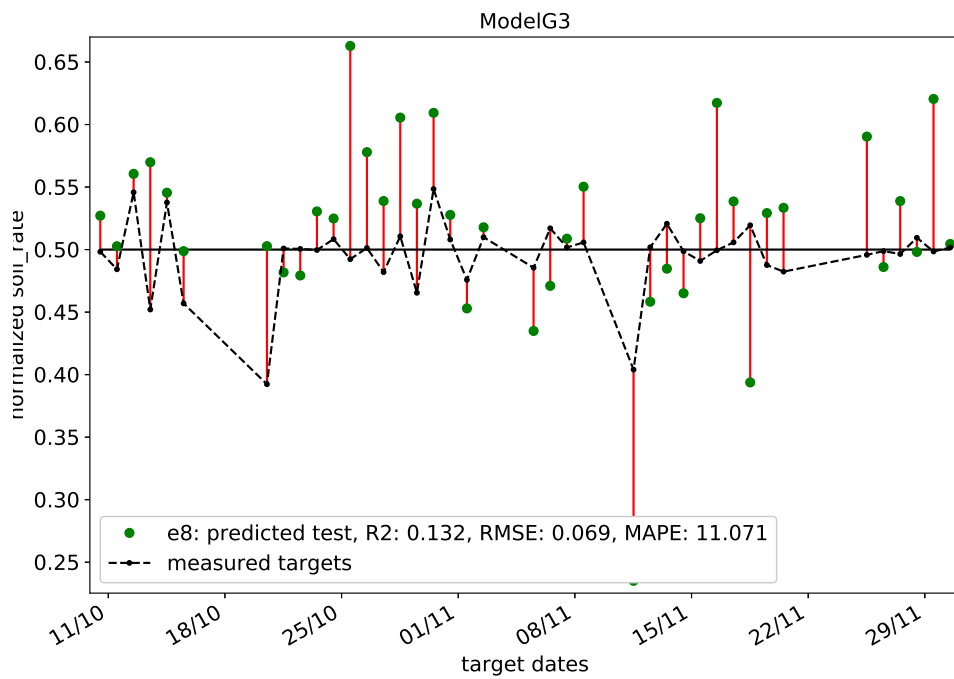


Figure C.12: The predictions on test data against their targets for Model G3 ranked #4 on SHL with sorted input features.

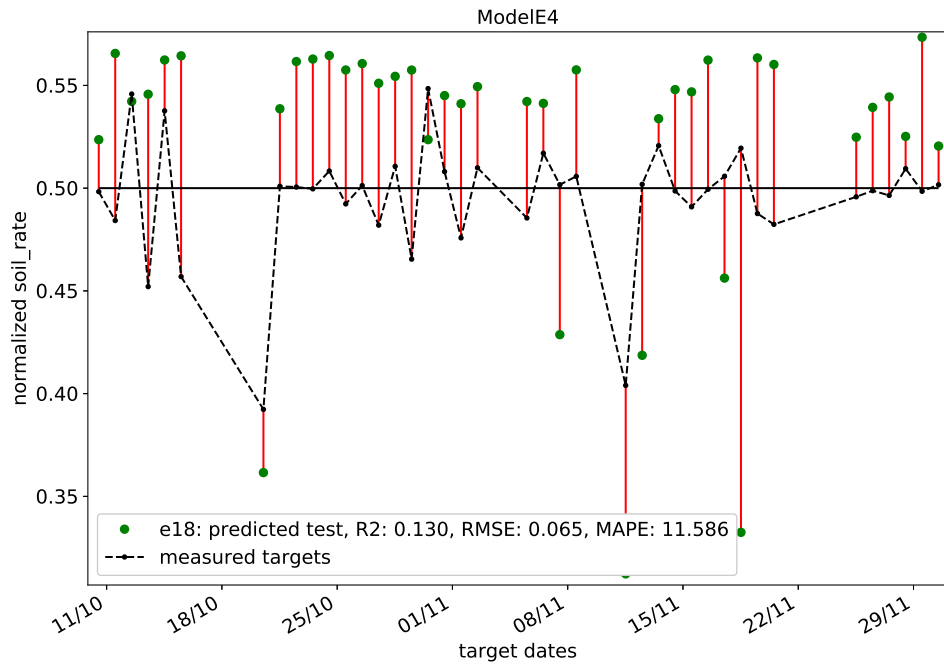


Figure C.13: The predictions on test data against their targets for Model E4 ranked #5 on SHL with sorted input features.

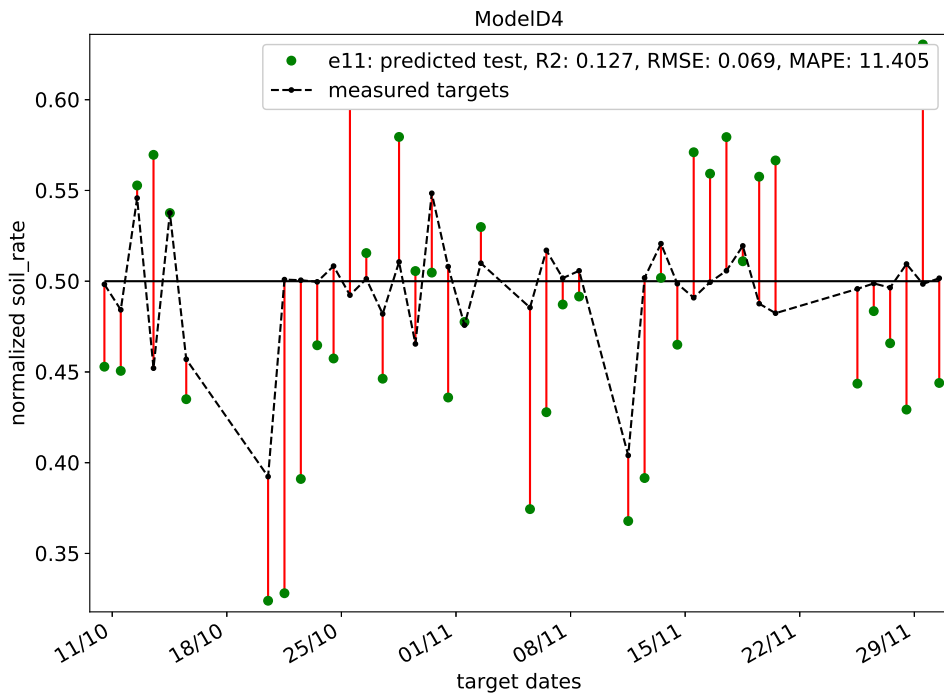


Figure C.14: The predictions on test data against their targets for Model D4 ranked #6 on SHL with sorted input features.

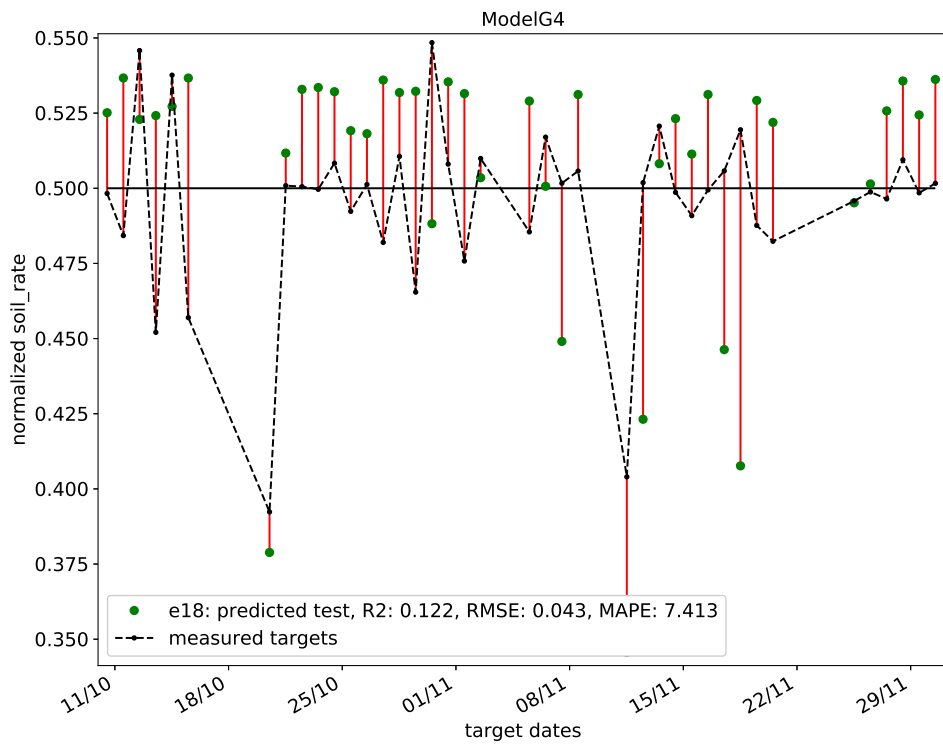


Figure C.15: The predictions on test data against their targets for Model G4 ranked #7 on SHL with sorted input features.

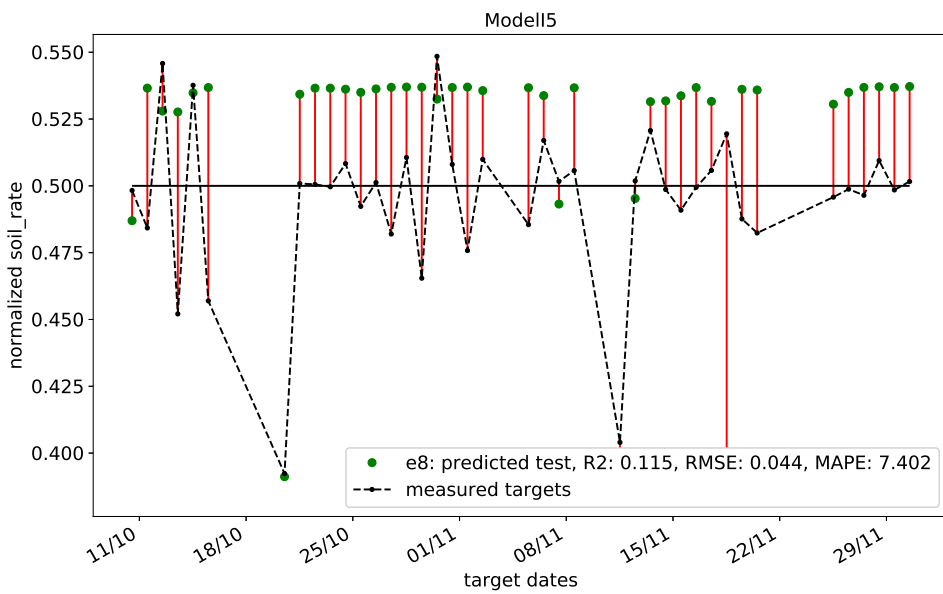


Figure C.16: The predictions on test data against their targets for Model I5 ranked #8 on SHL with sorted input features.

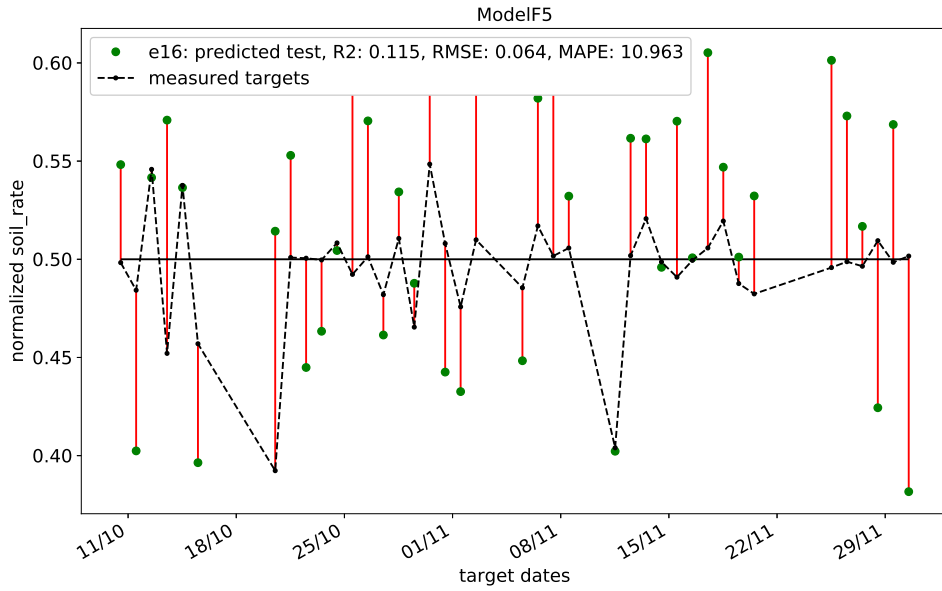


Figure C.17: The predictions on test data against their targets for Model F5 ranked #9 on SHL with sorted input features.

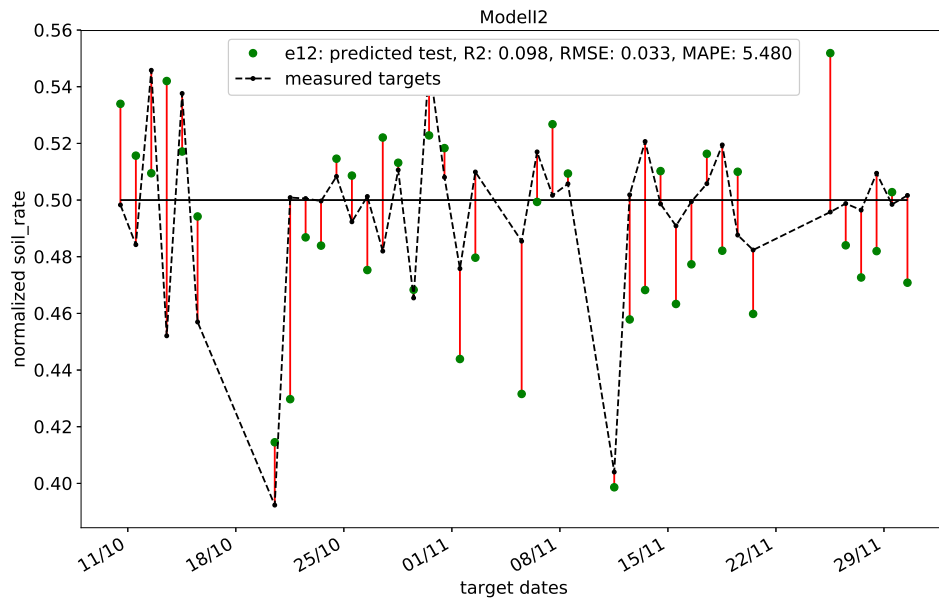


Figure C.18: The predictions on test data against their targets for Model I2 ranked #10 on SHL with sorted input features.

C.3 Unsorted inputs deep learning scores

Table C.6: The top 10 best R^2 values encountered on all data of the deep learning models with unsorted input features.

#	Model	Hidden nodes	epoch	R^2_{train}	R^2_{test}	R^2	RMSE	MAPE
1	ModelE5	16(+3)	32	0.071	0.019	0.060	0.080	11.837
2	ModelE3	16(+3)	2	0.042	0.009	0.042	0.131	22.301
3	ModelE4	8(-5)	19	0.055	-0.024	0.037	0.127	22.405
4	ModelG4	13	30	0.024	0.069	0.028	0.062	10.276
5	ModelG5	12(-1)	7	0.007	0.080	0.026	0.136	20.752
6	ModelI5	15(+1)	0	0.014	0.009	0.024	0.142	26.228
7	ModelD5	16(+4)	143	0.114	-0.555	0.023	0.051	8.176
8	ModelK4	11(-3)	59	-0.006	0.006	0.022	0.093	15.635
9	ModelK5	12(-2)	26	-0.028	0.030	0.021	0.102	14.576
10	ModelD4	12	7	0.002	0.081	0.019	0.126	23.913

Bibliography

- Agboola, AH et al. (2013). “Development of a fuzzy logic based rainfall prediction model.” In: *International Journal of Engineering and Technology* 3.4 (cit. on p. 23).
- Bayer, Michael (2016). *SQLAlchemy*. 1.0.1. <http://www.sqlalchemy.org/> (cit. on p. 46).
- Bonsignore, Luca et al. (2014). “Neuro-Fuzzy Fault Detection Method for Photovoltaic Systems.” In: *Energy Procedia* 62, pp. 431–441. ISSN: 1876-6102. DOI: <http://dx.doi.org/10.1016/j.egypro.2014.12.405>. URL: <http://www.sciencedirect.com/science/article/pii/S1876610214034365> (cit. on p. 23).
- BP (2016). “BP Energy Outlook 2016 edition.” In: URL: /energyoutlook (cit. on p. 1).
- Caron, J Riley and Bodo Littmann (2013). “Direct monitoring of energy lost due to soiling on first solar modules in California.” In: *IEEE Journal of photovoltaics* 3.1, pp. 336–340 (cit. on p. 19).
- Dieleman, Sander et al. (2015). *Lasagne: First release*. DOI: 10.5281/zenodo.27878. URL: <http://dx.doi.org/10.5281/zenodo.27878> (cit. on p. 69).
- Duchi, John, Elad Hazan, and Yoram Singer (2011). “Adaptive subgradient methods for online learning and stochastic optimization.” In: *Journal of Machine Learning Research* 12, Jul, pp. 2121–2159 (cit. on p. 31).
- Energy Yield and Performance Ratio of Photovoltaic Systems* (n.d.). http://www.greenrhinoenergy.com/solar/technologies/pv_energy_yield.php. Accessed: 2017-04-30 (cit. on pp. 5, 6).
- Fraunhofer-ISE (2016). *Photovoltaics report*. Tech. rep. Fraunhofer Institute for Solar Energy Systems, ISE (cit. on p. 3).
- García, Miguel et al. (2011). “Soiling and other optical losses in solar-tracking PV plants in navarra.” In: *Progress in Photovoltaics: Research and Applications* 19.2, pp. 211–217. ISSN: 1099-159X. DOI: 10.1002/pip.1004. URL: <http://dx.doi.org/10.1002/pip.1004> (cit. on p. 19).
- Gostein, Michael (2014). *Update on Edition 2 of IEC 61724: PV System Performance Monitoring*. Accessed: 2016-04-30. URL: https://www.nrel.gov/pv/assets/pdfs/2014_pvmrw_84_gostein.pdf (cit. on p. 9).
- Guo, B. et al. (2015). “Effect of dust and weather conditions on photovoltaic performance in Doha, Qatar.” In: *2015 First Workshop on Smart Grid and Renewable Energy (SGRE)*, pp. 1–6. DOI: 10.1109/SGRE.2015.7208718 (cit. on pp. 19, 21).

- Heaton, Jeff (2015). *Neural Networks and Deep Learning*. 1.0. Vol. 3. Artificial Intelligence for Humans. Heaton Research, Inc. (cit. on pp. 26, 27, 29–31).
- IEA. (2015a). “Key world energy statistics.” In: URL: <https://www.iea.org/publications/freepublications/publication/key-world-energy-statistics-2015.html> (cit. on p. 1).
- (2015b). “World Energy Outlook 2015.” In: DOI: <http://dx.doi.org/10.1787/weo-2015-en>. URL: [/content/book/weo-2015-en](http://content/book/weo-2015-en) (cit. on p. 1).
- IEA (n.d.). “World Energy Outlook 2016.” In: DOI: <http://dx.doi.org/10.1787/weo-2016-en>. URL: [/content/book/weo-2016-en](http://content/book/weo-2016-en) (cit. on p. 2).
- IEC. (1998). *IEC 61724 Std. Photovoltaic System Performance Monitoring-Guidelines for Measurement, Data Exchange and Analysis*. IEC. (cit. on p. 9).
- ipcc-contributors (2014). *Climate Change 2014: Mitigation of Climate Change*. Report 5. Intergovernmental panel on climate change (cit. on p. 1).
- ISO8601 (2000). *Data elements and interchange formats – Information interchange – Representation of dates and times*. Standard 8601:2000. International Organization for Standardization (cit. on p. 52).
- Japenga, Robert (2016). *How to write a software requirements specification*. <http://www.microtoolsinc.com/Howsrs.php> (cit. on p. 44).
- Jones, R. K. et al. (2016). “Optimized Cleaning Cost and Schedule Based on Observed Soiling Conditions for Photovoltaic Plants in Central Saudi Arabia.” In: *IEEE Journal of Photovoltaics* 6.3, pp. 730–738. ISSN: 2156-3381. DOI: 10.1109/JPHOTOV.2016.2535308 (cit. on p. 4).
- Kantardzic, Mehmed (2011). *Data mining: concepts, models, methods, and algorithms*. John Wiley & Sons (cit. on pp. 24, 27, 28).
- Kingma, Diederik and Jimmy Ba (2014). “Adam: A method for stochastic optimization.” In: *arXiv preprint arXiv:1412.6980* (cit. on p. 32).
- Kurokawa, Kosuke et al. (1998). “Sophisticated verification of simple monitored data for Japanese field test program.” In: KURO LAB. URL: <http://www.kurochans.net/english/kurochan/e-papers.htm> (cit. on p. 11).
- LeCun, Yann et al. (1998). “Gradient-based learning applied to document recognition.” In: *Proceedings of the IEEE* 86.11, pp. 2278–2324 (cit. on p. 35).
- Lorenz, Elke et al. (2004). “PVSAT-2: Intelligent performance check of PV system operation based on satellite data.” In: *ResearchGate*. URL: https://www.researchgate.net/publication/46658279_PVSAT-2_Intelligent_performance_check_of_PV_system_operation_based_on_satellite_data (cit. on p. 18).
- MATLAB Engine API for Python* (2016). Accessed: 2017-04-30. MathWorks. <http://se.mathworks.com/help/matlab/matlab-engine-for-python.html> (cit. on p. 46).
- Mejia, Felipe A and Jan Kleissl (2013). “Soiling losses for solar photovoltaic systems in California.” In: *Solar Energy* 95, pp. 357–363 (cit. on pp. 4, 18).

- Micheli, Leonardo and Matthew Muller (2017). "An investigation of the key parameters for predicting PV soiling losses." In: *Progress in Photovoltaics: Research and Applications* 25.4, pp. 291–307 (cit. on p. 97).
- Naeem, M. and G. Tamizhmani (2015). "Climatological relevance to the soiling loss of photovoltaic modules." In: *2015 Saudi Arabia Smart Grid (SASG)*, pp. 1–5. DOI: 10.1109/SASG.2015.7449280 (cit. on pp. 4, 19).
- Ndapuka, Andreas Tangeni (2015). "Design and development of a monitoring station for the long-term investigation of dust pollution effects on the performance of PV panels." PhD thesis. Stellenbosch: Stellenbosch University (cit. on p. 42).
- Nouri, Daniel (2014). *nolearn: scikit-learn compatible neural network library*. URL: <https://github.com/dnouri/nolearn> (cit. on p. 69).
- Plessis, Armand A. du (2016). "The Influence of Dust Soiling on the Performance of Photovoltaic Modules in the Semi-Arid Areas of South Africa." MA thesis. University of Stellenbosch (cit. on pp. 21, 39, 40, 52, 53, 56).
- Pulipaka, Subrahmanyam, Fani Mani, and Rajneesh Kumar (2016). "Modeling of soiled PV module with neural networks and regression using particle size composition." In: *Solar Energy* 123, pp. 116–126 (cit. on p. 24).
- PV Modeling Collaborative* (2016). URL: <https://pvpmc.sandia.gov/> (visited on 04/30/2016) (cit. on p. 11).
- Ramli, Makbul AM et al. (2016). "On the investigation of photovoltaic output power reduction due to dust accumulation and weather conditions." In: *Renewable Energy* 99, pp. 836–844 (cit. on p. 20).
- Al-Rfou, Rami et al. (2016). "Theano: A Python framework for fast computation of mathematical expressions." In: *arXiv e-prints* abs/1605.02688. URL: <http://arxiv.org/abs/1605.02688> (cit. on p. 69).
- Sarver, Travis, Ali Al-Qaraghuli, and Lawrence L. Kazmerski (2013). "A comprehensive review of the impact of dust on the use of solar energy: History, investigations, results, literature, and mitigation approaches." In: *Renewable and Sustainable Energy Reviews* 22, pp. 698–733. ISSN: 1364-0321. DOI: <http://dx.doi.org/10.1016/j.rser.2012.12.065>. URL: <http://www.sciencedirect.com/science/article/pii/S136403211300021X> (cit. on p. 4).
- Serrano-Luján, Lucia et al. (2016). "Case of study: Photovoltaic faults recognition method based on data mining techniques." In: *Journal of Renewable and Sustainable Energy* 8.4, p. 043506. DOI: 10.1063/1.4960410. eprint: <http://dx.doi.org/10.1063/1.4960410>. URL: <http://dx.doi.org/10.1063/1.4960410> (cit. on p. 23).
- Silvestre, Santiago, Aissa Chouder, and Engin Karatepe (2013). "Automatic fault detection in grid connected {PV} systems." In: *Solar Energy* 94, pp. 119–127. ISSN: 0038-092X. DOI: <http://dx.doi.org/10.1016/j.solener.2013.05.001>. URL: <http://www.sciencedirect.com/science/article/pii/S0038092X13001849> (cit. on pp. 15–18).
- Spataru, Sergiu et al. (2015). "Diagnostic method for photovoltaic systems based on light I–V measurements." In: *Solar Energy* 119, pp. 29–44. ISSN: 0038-092X. DOI: <http://dx.doi.org/10.1016/j.solener.2015.06.020>. URL:

- www.sciencedirect.com/science/article/pii/S0038092X1500328X (cit. on p. 24).
- TC 82 *Solar photovoltaic energy systems (61724)* (1998). URL: http://www.iec.ch/dyn/www/f?p=103:38:0:::FSP_ORG_ID,FSP_APEX_PAGE,FSP_LANG_ID,FSP_PROJECT:1276,23,25,PNW/TS%5C%2082-1061%5C%20Ed.%5C%201.0# (cit. on p. 9).
- Walt, Stéfan van der, Chris Colbert, and Gaël Varoquaux (2011). *The NumPy Array: A Structure for Efficient Numerical Computation*. [Online; accessed 2017-04-21]. DOI: <http://dx.doi.org/10.1109/MCSE.2011.37>. URL: <http://scitation.aip.org/content/aip/journal/cise/13/2/10.1109/MCSE.2011.37> (cit. on p. 69).
- Weldemariam, Ashenafi (2016). “Analyzing the effect of soiling on the performance of a photovoltaic system of different module technologies in Kalkbult, South Africa.” MA thesis. Dalarna University (cit. on p. 41).
- Woyte, Achim et al. (2014). *IEA-PVPS T13-D2 3 Analytical Monitoring of PV Systems Final*. Tech. rep. International Energy Agency (IEA) (cit. on pp. 9, 10).
- Yadav, Amit Kumar and S.S. Chandel (2014). “Solar radiation prediction using Artificial Neural Network techniques: A review.” In: *Renewable and Sustainable Energy Reviews* 33, pp. 772–781. ISSN: 1364-0321. DOI: <http://doi.org/10.1016/j.rser.2013.08.055>. URL: <http://www.sciencedirect.com/science/article/pii/S1364032113005959> (cit. on p. 24).
- (2017). “Identification of relevant input variables for prediction of 1-minute time-step photovoltaic module power using Artificial Neural Network and Multiple Linear Regression Models.” In: *Renewable and Sustainable Energy Reviews*. ISSN: 1364-0321. DOI: <http://dx.doi.org/10.1016/j.rser.2016.12.029>. URL: <http://www.sciencedirect.com/science/article/pii/S136403211631084X> (cit. on pp. 32, 71).
- Zorrilla-Casanova, J et al. (2011). “Analysis of dust losses in photovoltaic modules.” In: *World Renewable Energy Congress-Sweden; 8-13 May; 2011; Linköping; Sweden*. 057. Linköping University Electronic Press, pp. 2985–2992 (cit. on p. 20).
- Øgaard, Mari Benedikte (2016). “Effect of Soiling on the Performance of Photovoltaic Modules in Kalkbult, South Africa.” MA thesis. Norges miljø- og biovitenskaplige universitet (cit. on pp. 20, 21, 42, 52, 58).