Optimization of Power Supply for Hydrology- and Meteorology-stations

Optimization of power supply for off-grid hydrology- and meteorology-stations using machine learning and genetic algorithm.

KINE THERESE HONORÉ SALVESEN
KIRSTI HOMSTØL

SUPERVISORS
Anne Gerd Imenes
Joao Leal
Ghali Raja Yakoub

University of Agder, 2020
Faculty of Engineering and Science

Abstract

This thesis consider a case study of a PV/wind/battery hybrid energy system installed at Scanmatic AS headquarters in Arendal, Norway. The energy system is a stand-alone and off-grid hybrid system. It consist of four PV panels of 20 W with different tilt and orientation, a wind turbine of 300 W and a battery of 1.4 kWh. This work consider machine learning and artificial intelligence in Python 3 for prediction and optimization. Machine learning is used to predict the power production from the system components based on the weather data at site. Artificial intelligence is used to optimize the system size based on cost and the ability to produce sufficient power to the load.

Five ANN models are build, one for each of the PV panels and one for the wind turbine. The models are trained and evaluated with the data collected through the case study. The performance of each model are evaluated using multiple evaluation metrics to gain an overall knowledge of the models. The results conclude that all of the models have an ability to predict the power production for the corresponding component, to a varying extend, and with a tendency to underpredict.

A genetic algorithm is implemented to optimize the hybrid system. The objective function of the GA is to minimize the total system cost while always meeting the load demand. The evaluation of the results from the GA show that the algorithm has the ability to optimize the system within given constraints. The GA is simulated for multiple scenario to see how the system performs under different test conditions. The results of the simulation of the GA conclude that for all the scenarios the optimized system include PV panels and battery, but do not include a wind turbine.

Preface

This thesis conclude the Renewable Energy Engineering Masters education at the University of Agder (UiA), campus Grimstad. The thesis is a result of the final course in the master's education, *Master's Thesis Renewable Energy (ENE500)*.

The master thesis correspond to 30 study credits and is written in the period 20.08.2019-10.01.2020. The aim of the thesis is to optimize the power supply for hydrology- and meteorology-stations produced and delivered by Scanmatic AS. The thesis considers prediction of power production from photovoltaic panels and a wind turbine, by the use of machine learning. Artificial intelligence is used to optimize the hybrid renewable energy system based on power reliability and cost, by implementing a Genetic Algorithm. Both the machine learning and genetic algorithm have been implemented in Python 3.

The subject of this thesis was proposed by Scanmatic AS. We chose this subject because of our interest in solar and wind energy. It also proved a opportunity to study a wider range of areas interesting for both parts of the group. Kine has background from a Bachelor Degree in Mechatronics from UiA. Kirsti has a Bachelor Degree in Renewable Energy from UiA. Throughout the project Kirsti has gained competence in hybrid renewable energy systems, energy storage in batteries, and the optimization of renewable energy systems with focus on the genetic algorithm. Kine has gained competence in hybrid renewable energy systems, prediction with machine learning, and optimization with the genetic algorithm. Kine has been responsible for the major part of the python programming. With no background knowledge on artificial intelligence, machine learning, or python programming, we have both gained a lot of new and valuable knowledge throughout this project.

We would like to thank our supervisors Joao Leal and Anne Gerd Imenes. Thank you for your guidance, encouragement and inputs on our thesis. Joao, thank you for your guidance on the subject of machine learning and artificial intelligence. Anne Gerd, thank you for your guidance on the subject of renewable energy systems and the structure of our thesis. We would also like to thank Ghali Raja Yakoub, for his help and guidance on the machine learning.

Finally we would like to thank Scanmatic AS for giving us a interesting (and challenging) subject for our master thesis. A special thank you to Nils Lofstad and Bjørnar Preus-Olsen who gave us knowledge about their hydrology- and meteorology-stations, and provided us with the data used in this thesis.

Individual/Group Mandatory Declaration

The individual student or group of students is responsible for the use of legal tools, guidelines for using these and rules on source usage. The statement will make the students aware of their responsibilities and the consequences of cheating. Missing statement does not release students from their responsibility.

1.	I/We hereby declare that my/our thesis is my/our own work and that I/We have not used any other sources or have received any other help than mentioned in the thesis.	☒
2.	I/we further declare that this thesis: - has not been used for another exam at another department/university/university college in Norway or abroad; - does not refer to the work of others without it being stated; - does not refer to own previous work without it being stated; - have all the references given in the literature list; - is not a copy, duplicate or copy of another's work or manuscript.	☒
3.	I/we am/are aware that violation of the above is regarded as cheating and may result in cancellation of exams and exclusion from universities and colleges in Norway, see Universitets- og høgskoleloven §§4-7 og 4-8 og Forskrift om eksamen §§ 31.	☒
4.	I/we am/are aware that all submitted theses may be checked for plagiarism.	☒
5.	I/we am/are aware that the University of Agder will deal with all cases where there is suspicion of cheating according to the university's guidelines for dealing with cases of cheating.	☒
6.	I/we have incorporated the rules and guidelines in the use of sources and references on the library's web pages.	☒

Publishing Agreement

Authorization for electronic publishing of the thesis.
Author(s) have copyrights of the thesis. This means, among other things, the exclusive right to make the work available to the general public (Åndsverkloven. §2).
All theses that fulfill the criteria will be registered and published in Brage Aura and on UiA's web pages with author's approval.
Theses that are not public or are confidential will not be published.

I hereby give the University of Agder a free right to make the task available for electronic publishing:	YES ☒	NO ☐
Is the thesis confidential? (confidential agreement must be completed)	YES ☐	NO ☒
- if yes: Can the thesis be published when the confidentiality period is over?	YES ☐	NO ☐
Is the task except for public disclosure? (contains confidential information. see Offl. §13/Fvl. §13)	YES ☐	NO ☒

UNIVERSITETET I AGDER
FAKULTET FOR TEKNOLOGI OG REALFAG

By order of Dean of Faculty of Engineering and Science: 30.01.2018

STANDARD AGREEMENT

concerning work on a bachelor's thesis/master's thesis/project assignment (academic work) done in cooperation with a company/external organization (organization).

This is the authoritative agreement that governs academic work by students at the UiA Faculty of Engineering and Science that is carried out in cooperation with an organization.

The involved parties have the responsibility to clarify whether or not a third party (that is not a party to this agreement) may have intellectual property rights to the project background before the latter is used in connection with the academic work.

Agreement between

Student: KINE SALVESEN / Kirsti Homstøl	Date of birth: 29.09.1993 / 31.07.1993

Supervisor(s) at UiA: Anne Gerd Imenes, Joao Leal, Ghali Raja Yakoub

Company/external organization: SCANMATIC AS

and

University of Agder (UiA), represented by the Head of Department

concerning the use and exploitation of the results from a bachelor's thesis/master's thesis/project assignment.

1. Description of the academic work

The student is to carry out

Bachelor's thesis	☐
Master's thesis	☒
Project assignment	☐

(insert cross)

In cooperation with

Scanmatic AS

company/external organization:

SCANMATIC AS	20.08.2019 - 10.01.2020

starting date – completion date (dd-mm-yyyy)

Title of the academic work:

Optimization of Power supply for Hydrology- and Meteorology Stations

The responsible supervisor at UiA has overall academic responsibility for structuring and approving the description of the academic work and the student's learning.

2. Responsibilities of the organization
The organization is to appoint a contact person who has the necessary experience in supervision and will give the student adequate supervision in cooperation with the supervisor at UiA. The contact person at the organization is:

The organization is to appoint a contact person who shall provide the student with the necessary work resources at the organization and, if possible, contribute in supervision in cooperation with supervisor at UiA. The organization's contact person is:

Nils Lofstad

The purpose of completing the academic work is academic training for the student. The academic work is part of a student's course of study and the student is not to receive wages or similar compensation from the organization. The organization agrees to cover the following expenses that are associated with carrying out the academic work:

The components needed for the case study (PV panels, small wind turbine, battery)

3. Rights of the parties
a) The student
The student holds the copyright to his/her academic work. All intellectual property rights to the results of the academic work done by the student alone during the academic work are held by the student with the reservations stated in points b) and c) below.

The student has the right to enter into an agreement with UiA concerning the publication of his/her academic work in UiA's institutional archive on the Internet. The student has also the right to publish his/her academic work or parts of it in other media providing the present agreement has not imposed restriction concerning publication, cf. Clause 4.

b) The organization
If the academic work is based on or develops materials and/or methods (project background) that are owned by the organization, the project background is owned by the organization. If the development work that includes the project background can be commercially exploited, it is assumed that a separate agreement will be drawn up concerning this between the student and the organization.

The organization is to have the right to use the results of the academic work in its own activities providing the commercial exploitation falls within the activities of the organization. This is to be interpreted in accordance with the terminology used in Section 4 of the Act Respecting the Right to Employees' Inventions (Arbeidstakeroppfinnelsesloven). This right is non-exclusive.

The use of the results of the academic work outside of the activities of the organization, cf. the last paragraph above, assumes that a separate agreement will be drawn up between the student and the organization. The agreement between the student and the organization concerning the rights to the results of the academic work produced by the student is to be in writing and the agreement is invalid until UiA has received a copy of the agreement in writing.

If the value of the results of the academic work is considerable, i.e. it is more than NOK 100 000, the student is entitled to receive reasonable compensation. Section 7 of the Act Respecting the Right to Employees' Inventions states how the amount of compensation is to be calculated. This right to compensation also applies to non-patentable results. Section 7 of the Act also states the applicable deadlines.

c) UiA
All copies of the submitted academic work/files containing the academic work and any appendices that are necessary for determining a grade and for the records at UiA, are the property of UiA.
The academic work and any appendices to it can be used by UiA for educational and scientific purposes free of charge, except when the restrictions specified in Clause 4 are applicable.

4. Delayed publication
The general rule is that academic work by students is to be available in the public domain. If there are specific circumstances, the parties can agree to delay the publication of all or part of the academic work for a maximum of 5 years, i.e. the work is not available for other students or organizations during this period.

The academic work is subject to delayed publication for:

one year	☐
two years	☐
three years	☐
five years	☐

(insert cross next to the number of years if this clause applies)

The reasons for delayed publication are as follows:

The parts of the academic work that are not subject to delayed publication can be published in UiA's institutional archive, cf. Clause 3 a) second paragraph.

Even if the academic work is subject to delayed publication, the organization is to make it possible for the student to use all or part of his/her academic work in connection with a job application or follow-up work in connection with doctoral study.

5. General

This agreement takes precedence over any other agreements that are or will be entered into by two of the parties mentioned above. In case the student and the organization are to enter into a confidentiality agreement concerning information the student obtains while he/she is at the organization, UiA's template for a confidentiality agreement is to be used for this purpose. If there is such an agreement, it is to be appended to the present agreement.

Should there be any dispute relating to this agreement, it should be resolved by negotiation. If this does not lead to a solution, the parties agree to the matter being resolved by arbitration in accordance with Norwegian law. Any such dispute is to be decided by Agder District Court or a body appointed by this court.

This agreement is signed in 4 - four - copies, where each party to this agreement is to keep one copy. The agreement comes into effect when it has been approved and signed by UiA represented by the Head of Department.

Note that the Norwegian version of this standard agreement is the authoritative version.

GRIMSTAD 04/01-20	Kine Salvesen	Kirsti Homstøl
place, date (dd-mm-yyyy)	student	

Grimstad 6/1-2020 Anne Gerd Imenes	
place, date (dd-mm-yyyy)	supervisor at UiA

Grimstad 6/1-2020 Geir Granmo	UNIVERSITETET I AGDER Institutt for ingeniørvitenskap
place, date (dd-mm-yyyy)	Head of Department, UiA

ARENDAL 18/12-2019	scanmatic as Bedriftsveien 17 4841 Arendal Org.nr.: 920 353 908 www.scanmatic.no
place, date (dd-mm-yyyy)	for company/organization signed and stamped

Contents

List of Tables

List of Figures

List of Symbols

A_T	Rotor Area
C_p	Power Coefficient
E_{Batt}	Charging Quantity Battery
E_{pv}	PV Generated Energy
E_{wt}	Wind Turbine Generated Energy
G	Irradiance
I_0	Dark Saturation Current
I_L	Light Generated Current
k	Boltzmann's Constant
λ	Velocity Difference
N	Rotational Speed
n	Ideality Factor
η	Learning Rate of Optimizer
η_{bat}	Battery Efficiency
η_{MPPT}	MPPT efficiency
η_{PWM}	PWM efficiency
o_j	Neuron Output
ω	Angular Velocity
q	Charge of an Electron
R	Radius
ρ_a	Air Density
σ	Self-discharge Rate
T	Absolute Temperature
T_a	Air Temperature
T_m	Temperature of Module

U_0	Effect of Radiation
U_1	Effect of Wind Speed
V	Wind Speed
v	Air Volume
V_{oc}	Open Circuit Voltage
φ	Activation Function
w_{ij}	Neuron Weight
x_i	Neuron Input
y_j	Model Output
$\hat{y_j}$	Target Output

Abbreviations

AC	Alternating Current
AI	Artificial Intelligence
AM	Air Mass
ANN	Artificial Neural Network
CNN	Convolutional Neural Network
CS	Cuckoo Search
CV	Constant Voltage
DC	Direct Current
DL	Deep Learning
DOD	Depth of Discharge
E	Nash Sutcliffe Efficiency
FF	Fill Factor
FNN	Feed-Forward Neural Network
GA	Genetic Algorithm
HRES	Hybrid Renewable Energy System
HWT	Wind Turbine Tower Height
LPSP	Loss of Power Supply Probability
MBE	Mean Bias Error
ML	Machine Learning
MLP	Multi-Layer Perceptrons
MPP	Maximum Power Point
MPPT	Maximum Power Point Tracking
NaN	Not a Number
PV	Photovoltaic
PWM	Pulse Width Modulation
ReLU	Rectified Linear Unit
RMSE	Root Mean Square Error
RNN	Recurrent Neural Network
SLP	Single-Layer Perceptrons

SOC	State of Charge
SSE	Sum of Squared Errors
STC	Standard Test Condition
Wp	Peak Watt
WT	Wind Turbine

Chapter 1

Introduction

Scanmatic AS develops and delivers field stations for logging hydrology- and meteorology-data. The hydrology- and meteorology-stations (HydMet-station) gathers data about weather conditions, water levels in reservoirs and the quality of water, as well as the water temperature [37]. The often remote locations for these systems requires a reliable, off-grid power solution. An hybrid renewable PV/wind/battery energy system is suggested. In addition to provide off-grid power, it provides the system with renewable and environmental friendly energy. The main challenge for these type of systems, is to produce adequate energy for the HydMet-station to operate at all circumstances. To have a reliable and cost-effective energy system, the system is optimized based on minimizing the total system cost, while supplying continuous and adequate power to the load.

Since artificial intelligence (AI) was born in the 1950s, multiple definitions have been proposed and used. A concise definition in computer science is *"the effort to automate intellectual tasks normally performed by humans"* as stated by [12]. AI involves learning, reasoning and self-correction. It is being used to solve tasks such as visual perception, understanding human speech, translating between languages, or make intelligent decisions, to mention a few. Machine learning (ML) and deep learning (DL) are some of the most commonly applied subfields of AI, as visualized in Figure 1.0.1.

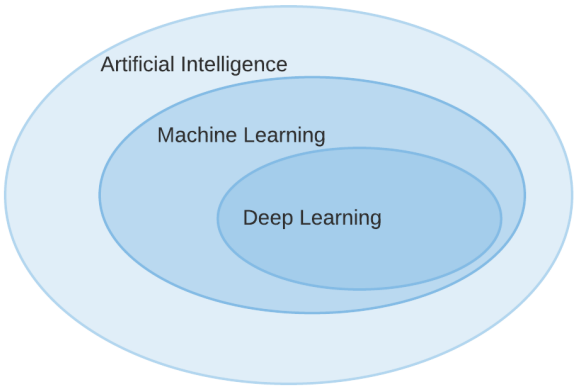


Figure 1.0.1: *How artificial intelligence, machine learning and deep learning are related*

The methodology of ML, DL and Artificial Neural Networks (ANNs) are proposed for predicting the expected energy production from a hybrid renewable energy system at a given site. AI techniques are proposed for solving optimization problems such as optimizing a PV/wind/battery energy system size for a reliable load supply and minimum cost [36]. AI techniques such as the Genetic Algorithm (GA) is capable of solving complex non-linear optimization problems such as the optimization of a hybrid energy system, with better accuracy and convergence than traditional optimization approaches [36].

1.1 Problem Definition

The objective of this project is to study the use of ML, ANNs and the GA for predicting the power production from a hybrid renewable energy system and optimizing the size of the system to minimize the costs while supplying the system load at all times.

ANNs will be used to predict the power produced by the renewable generators (PV panels and wind turbine) at a given location with the weather data as input. Four 20 W PV panels with different tilt and orientation and a 300 W wind turbine will be studied, referenced previous work done in the *Energy Research Project (ENE504)* [21]. Five models will be created, one for each PV panel and one for the wind turbine. All models will be trained and evaluated with data collected at the test HydMet-station installed in cooperation with Scanmatic AS. To optimize the system size and components based on the predicted power production AI will be used. The GA is proposed to optimize the system with the purpose of obtaining the system with the minimum cost that is always able to supply the load demand.

Based on the problem defined, the goal of this project can be divided into multiple sub goals:

- Gain theoretical background in the subjects of ML, AI, hybrid renewable energy systems, solar and wind energy, as well as battery storage.
- Previous work done in optimization of energy systems using ML and AI must also be studied.
- Gather and pre-process (clean, scale, etc.) the data measured at the test HydMet-station
- Select input features for ML and divide the data for training and testing
- Develop and train ANNs for the renewable generators
- Evaluate the ANN models performance
- Implement the GA for optimizing the system cost while still meeting the load demand
- Select the GA parameters to gain the optimum system solution
- Evaluate the final solution and compare to other possible system solutions

1.2 Limitations and Assumptions

To achieve the goal of this thesis several limitations and assumptions were made. The main assumptions and limitations are presented in this subsection, while a deeper explanatory is stated in the methodology section. The limitations and assumptions can be summarized accordingly:

- The data available is within a limited time period from 13.03.2019 to 23.10.2019. The results obtained in this thesis are therefore only valid for this period.
- There may be other weather parameters that could be interesting to study as available input features to the ML that are not included in the case study. But as there is no measured data for these parameters, they are not included in this study.
- The battery model in the GA has been simplified. The battery charging depends only on the system production, load, and the battery self discharge rate (at 20°C), and current state of charge. The battery is also assumed fully charged at the beginning of the simulations.
- It is assumed that if a PV panel is installed, the system also requires a MPPT charge controller. If a wind turbine is installed, a PWM charge controller and a dump load is included.
- The components costs will depend on supplier, market price, exchange rate, shipping, and other possible factors. The minimized system cost will be valid for the selected components with the components costs available to this date. The components costs are generalized to the "cost per installed capacity" (NOK/W or NOK/Wh) to give the selected components range a reasonable cost.
- The available components range available for the GA does not necessarily reflect available component sizes on the market. The components range are generalized within a minimum and maximum limit, to allow the GA to find a optimum solution not constraint by the components available by a given supplier.
- The optimization problem is based on the case study. The available components in the GA are the components installed in the test HydMet-station with their configuration. The optimum system solution for a new site may be a system with different components configurations.
- Both the ML algorithm and the GA have a randomness to the performance which can affect the results obtained from prediction and from the system optimization.

1.3 Thesis Outline

The theoretical background comprise topics related to the objective of this project. In the theoretical background a literature review is included, to give an overview of previous work done in the applicable areas. The literature review will give an understanding of the important contributors to consider when working with ML, AI, and hybrid PV/wind/battery energy systems. The theoretical background covers topics related to PV/wind hybrid energy systems, ML and methods within AI.

The methodology describes the process of the work and the methods used in this project, as well as the case study of the master. The methodology of pre-processing, cleaning, scaling, selecting features and

dividing data for ML is presented, as well as the process of selecting and optimizing the ANNs architecture and structure. The steps of implementing the GA are also included.

The result and discussion chapter presents the results obtained from the work done in this project. The chapter presents the results obtained throughout the process of developing the ANN models, optimizing the system configuration, and evaluation the performance and results obtained.

The conclusion will conclude the masters thesis and the work carried out this semester.

Chapter 2

Theoretical Background

This chapter presents the theory considered important for answering the problem defined for this master thesis. The chapter starts with presenting the literature review. Followed by the theory of solar and wind energy as single energy sources, before providing the advantages of choosing a hybrid energy system over a single source system. In addition to describing the theory of energy storage, the theory of charge controllers is presented. The theory about ML, and how to use and build ANNs are then presented. At least the theory of implementing a GA to optimize the cost of a hybrid energy system is presented.

2.1 Literature Review

The literature review consider previously done on the topics related to the theory of this thesis. It consider the main aspects of machine learning, and optimization techniques for optimizing a PV/wind/battery energy system.

Sinha et al. [36] presents the advantages of using a hybrid system consisting of PV, wind, and battery, instead of using a single energy source system. The advantages is higher reliability and efficiency for a hybrid system. The paper also review the issues of optimizing a hybrid energy system and the sizing methodologies concerning meteorological data and load profile for such a system. This issue is with respect to the interval of the meteorological data, and the importance of having a yearly electrical demand. Belmili et al. [13] conclude that a PV/wind hybrid system offers an adequate solution for remote areas, where the energy reliability is higher.

Faranda et al. [29] explains the main disadvantages and the I-V characteristics of a solar cell. The main disadvantages include the low efficiency for PV systems. [29] explains how the I-V characteristics is affected by irradiation and temperature causing a non-linear characteristics. [29] define the *Maximum Power Point* where the PV system operates at maximum efficiency. This theory is supported by the book *Applied Photovoltaics* by Wenham et al. [34].

The book *Wind energy: fundamentals, resource analysis and economics* by Mathew et al. [32] provides the theory of wind energy production. This theory discuss the factors influences the power production, such as

the wind speed. The book explains the *power curve* of a wind turbine.

Maleki et al. [2] introduce the modeling of the battery bank. The article review the changes regarding the capacity in the battery bank, and conclude that the capacity will constantly change in a hybrid system. This issue is also reviewed by Singh et al. [35]. Both conclude that the battery is in charging state when the total output of the hybrid system is greater than the load demand, and it is in discharging state when the total output is lower than the load demand.

The equations for calculating the charging and discharging needs to be adjusted to the hybrid system studied in this thesis. To do so the efficiency of a PWM and a MPPT charge controller must be included. Qazi et al. [33] and Char et al. [23] provide the knowledge of the function of a MPPT charge controller. They conclude that by using a MPPT charge controller for the PV panels, the lifetime of the battery will improve. Bhatia et al. [31] explains the function of a PWM charge controller used in wind-electrical systems to avoid overcharging the battery.

F. Chollet et al. [12] defines AI as *"the effort to automate intellectual tasks normally performed by human"*, and define how artificial intelligence (AI), machine learning (ML) and deep learning (DL) are related. The two types of ML, supervised and unsupervised learning, are described by J. VaderPlas [19] and MathWorks [24]. Supervised learning use known input and output data to train the ML model, and the unsupervised find patterns in data sets without referencing to labels.

A. Géron et al. [1] describe DL as a subfield of ML based on artificial neural networks (ANNs). ANNs are highly powerful models, and capable of handling highly complex ML tasks. The book describe the neurons in ANN and the structure of the neural networks. When training a ANN it learns to find a set of values for the weights of all network neurons so that the network will learn the relationship between the inputs and their targets, as is also described by F. Chollet et al in [12]. The backpropagation algorithm by D.E Rumelhart et al. [6] is used to optimize the neuron weights. The training of ANNs is explained in more details by A. Géron, [1].

H. R. Maier et al. [14] describe how the ANNs are divided into two main categories, *feed-forward neural networks (FNNs)* and *recurrent neural networks (RNNs)*. FNNs include the *single-layer perceptrons (SLPs)*, *multi-layer perceptrons (MLPs)*, and *convolutional neural network (CNN)*. A. Géron et al. [1] presents the composition of SLPs, MLPs and CNNs, in addition to the composition of RNNs.

The process of developing an ANN model is described by A. Géron et al. [1] and H. R. Maier et al. [14]. The process starts with pre-processing and selection of data, before selecting and optimizing the model architecture and structure. J. Brownlee et al. [16] explains why the data should be corrected and formatted before used in ML. Data cleaning includes detecting outliers and remove irrelevant data.

Sinha et al. [36] provide the advantages of using GA for optimization. The advantages are the possibility to solve multiple solutions problems, and easy implementation is existing simulations. The theory of implementing the GA in python is presented by Stoltz et al. [9]. The implementation is presented as a step by step. [9] include the theory of *selection*, *crossover* and *mutation*, which are the three main operators in the GA.

Maleki et al. [2] and [3] compares multiple AI techniques for optimizing a hybrid system. They conclude that AI techniques for optimization yields a better result than traditional methods such as linear programming. Yang et al. [15] implements the GA in the optimization of a PV/wind hybrid system. The algorithm is used

to find the optimum components combination to obtain the minimum system cost while still being reliable in supplying the load demand. The article conclude that the GA has the ability to attain a global optimum in addition to computational simplicity, and the method will provide a valid and high quality result in optimizing a PV/wind hybrid system.

2.2 Hybrid Renewable Energy System

A hybrid renewable energy system (HRES) is composed of two or more energy sources where at least one is renewable [13]. For remote areas a hybrid PV/wind system offers a preferable solution in terms of efficiency and reliability, compared to a single PV or wind energy system [13][36]. A PV/wind/battery hybrid energy system consists of photovoltaic energy, wind energy, and batteries for energy backup. Solar and wind are promising power generating sources, but depend on climatic conditions and geographical areas [13]. A hybrid system of solar and wind complement each other, as the wind is often stronger in seasons with lower solar radiation, and the wind speed are often low in periods when the solar resource is more available [36].

An important issue concerning an off-grid HRES is the system sizing [3]. It is easy to oversize a hybrid system in order to obtain a reliable system, but this is not cost-effective [3]. The aim of optimizing a PV/wind HRES is to make sure to have a cost-effective system with full use of the renewable energy resources and still satisfy the load requirements [13]. Therefore optimum sizing is defined as the determination of number of system components, or the peak watts of each component, which are able to meet the load demand [3]. Figure 2.2.1 show a schematic overview of a typical PV/wind/battery HRES.

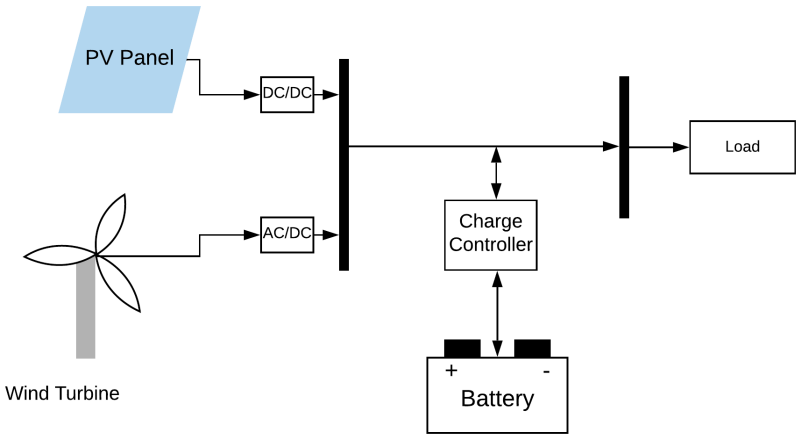


Figure 2.2.1: *Typical components of a PV/wind/battery hybrid energy system*

The wind turbine is connected to an AC/DC converter and the PV panels are connected to a DC/DC converter [39]. These are connected to a DC bus, which are then connected to a charge controller. The charge controller control the charging of the battery. The charge controller is then connected to the load [39].

2.2.1 System Control

The design and control of a hybrid system is more complex than for a single energy source system [36]. For a hybrid system there are more components and factors which have to be taken into account when designing the control of the system. For the PV panel, the irradiance and temperature has to be taken into account, while for the wind turbine, the wind speed is essential. The irradiance, temperature and wind speed are the meteorological data which is measured over a period of time. For an hybrid system there are constraints in the solution of the design and optimization process like resource availability, technology and efficiency [36]. For the optimization process an analysis of the meteorological data for on the location has to be evaluated [36]. Solar and wind energy resources rely on locally available sources in each geographical area [13].

2.2.2 Load

When designing and optimizing a hybrid system it is necessary to know the profile of the electrical load demand [36]. The analyze of the real load demand can be difficult when analyzing data from different seasons, and the system can easily be oversized or undersized. To make sure the system is able to meet the load demand, the PV/wind/battery energy system need to be sized based on the load [36].

2.3 Photovoltaic Solar Energy

Photovoltaic (PV) solar energy is a renewable energy source with great potential. The energy source is clean, and a PV system is easy to maintain and highly reliable [11]. The energy produced by a PV system depend on several aspects such as geographical location, the rated characteristics of the components, and the configuration of the installation [11]. A typical PV system may consist of photovoltaic modules, charge controller, and a battery to store surplus energy [27]. The main disadvantages with PV systems are the low conversion efficiency (9-17%) of electric power generation, and that the electric power generated by a solar array is dependent on the weather conditions [29].

2.3.1 Operational Characteristics

The operational characteristics of a PV system is described by the current-voltage curve (I-V curve), as seen in Figure 2.3.1. The I-V curve give information about the open circuit voltage (V_{OC}), short circuit current (I_{SC}), maximum power point ($P_{mp} = (V_{mp}, I_{mp})$), and fill factor ($FF$) [26]. FF is expressed by Equation 2.3.1.

$$FF = \frac{V_{mp} I_{mp}}{V_{oc} I_{sc}} \tag{2.3.1}$$

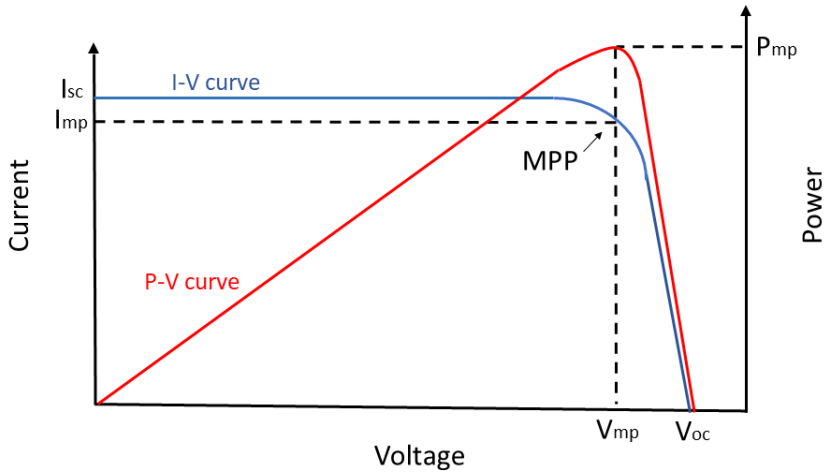


Figure 2.3.1: *I-V curve and power curve of a single solar cell [21]*

For a given irradiance, operating temperature and area, the limiting parameters are V_{oc} and I_{sc} when evaluating the output of solar cells. The open circuit voltage V_{oc} is the maximum voltage at zero current. When the sunlight is increasing, V_{oc} will increase logarithmic. The short circuit current I_{sc} is the maximum current at zero voltage, and is directly proportional to the available sunlight. V_{oc} can be calculated by Equation 2.3.2, where I_0 is the dark saturation current, I_L is the light generated current, q is the charge on an electron, k is Boltzmann's constant, n is the ideality factor and T is absolute temperature.

$$V_{oc} = \frac{nkT}{q} ln(\frac{I_L}{I_0} + 1) \tag{2.3.2}$$

The I-V characteristics of the solar cell is non-linear and varies with irradiation and temperature [29]. For both a I-V curve or a P-V curve there is a *Maximum Power Point (MPP)* where the PV system operates at maximum efficiency to produce maximum output power [29]. The output power at MPP is maximized when the product in Equation 2.3.3 is at its maximum value [34]. V_{mp} is the maximum voltage and I_{mp} is the maximum current.

$$P_{mp} = V_{mp} \cdot I_{mp} \tag{2.3.3}$$

The power output at the MPP under strong sunlight is the "peak power" of the cell. PV panels are usually rated based on their "peak" watts W_p [34]. The voltage at MPP can be calculated by Equation 2.3.4.

$$V_{mp} = V_{oc} - \frac{nkT}{q} ln(\frac{V_{mp}}{nkT/q} + 1) \tag{2.3.4}$$

To maintain the operating point of the PV array at its MPP, a *Maximum Power Point Tracker (MPPT)* is used [29].

2.3.2 Irradiance and Temperature

The theory described in this section is based on [21] if otherwise is not stated. The operation conditions for the PV modules installed outdoors will differ from the STC defined 1000 W/m^2 solar irradiance, module temperature of 25^{o}C, and air mass of 1.5 AM [10]. The real power output will therefore be different from the installed peak power [34].

The solar irradiance can be divided into global and diffuse horizontal irradiance and global tilted irradiance, which are defined by Wenham et al. [34]:

Global Horizontal Irradiance is the summation of diffuse and direct radiation onto a horizontal plane.

Diffuse Horizontal Irradiance is the radiation which is scattered by particles in the atmosphere, or radiation reflected by the surface of the earth. It can be measured by a pyranometer on a horizontal plane where the direct sunlight is blocked.

Global Tilted Irradiance is the irradiance measured in plane with a tilted PV module. This irradiance correspond to the irradiance which the PV module receive.

The operational temperature of a PV module affect the energy output of the module as it affect the open circuit voltage [34]. The main effect of an increase of temperature of a solar cell, is a decrease in the open circuit voltage, which results in a reduced fill factor (FF) and therefore a reduced energy output of the cell [34].

For a silicon cell, as the temperature increase, the open circuit voltage decrease with about $-2.2\ mV/^oC$ equivalent to -0.30% of V_{oc} per ^oC [34]. This results in a reduced fill factor and also a reduced cell output. The effect of temperature on the maximum power point for a silicon PV cell is about -0.45% of P_{mp} per ^oC. There is also a small increase in the short circuit current as the temperature increase, however the effect is small compared to the decrease in open circuit voltage [34].

The operational temperature of the module T_m is dependent on the irradiance G and the temperature in the air T_a, as described by Equation 2.3.5 [10]. V is the wind speed, and U_0 and U_1 are coefficients describing the effect of radiation and wind on the module temperature.

$$T_m = T_a + G/(U_0 + U_1 V) \tag{2.3.5}$$

How much the module is affected by the temperature is given in the manufacturer data-sheet at STC.

2.4 Wind Turbine Energy

Wind power is an important and promising renewable energy source. It has the advantages of economic efficiency and the absence of greenhouse gas emissions [3].

The available wind energy is the kinetic energy of large masses of air blowing over the Earth's surface. The blades on a wind turbine receive the kinetic energy and use it to produce electrical energy. The available

kinetic energy from an air stream to the turbine is given by Equation 2.4.1, where ρ_a is the air density, v is the volume of the air parcel available to the rotor and V is the wind speed [32].

$$E = \frac{1}{2}\rho_a v V^2 \tag{2.4.1}$$

This air parcel has an area equal to the rotor, and a thickness equal to the wind speed, and can therefore be expressed hence to the theoretical power by Equation 2.4.2 [32], where ρ_a and V is explained for Equation 2.4.1 and A_T is the area of the rotor. P is the maximum theoretical power.

$$P = \frac{1}{2}\rho_a A_T V^3 \tag{2.4.2}$$

The factors which are influencing the produced power are the wind speed, the area of the rotor, and the air density [32]. The air density is influenced by the temperature, atmospheric pressure, elevation and air constituents. A wind turbine is also not able to extract all the theoretical power from 2.4.2 [32]. The actual power produced is determined by the efficiency factor called the power coefficient C_p. C_p is the ratio between the actual power produced and the theoretical production. It is expressed by Equation 2.4.3, where P_T is the power produced by the turbine [32].

$$C_p = \frac{2P_T}{\rho_a A_T V^3} \tag{2.4.3}$$

The power produced by the rotor at a given wind speed rely on the difference in velocity between the wind and the rotor tip. This difference is denoted as λ and is expressed by Equation 2.4.4, where R is the radius of the rotor, Ω is the angular velocity and N is the rotational speed of the rotor [32].

$$\lambda = \frac{R\Omega}{V} = \frac{2\pi N R}{V} \tag{2.4.4}$$

The theory described in this section is based on [21] if otherwise is not stated.

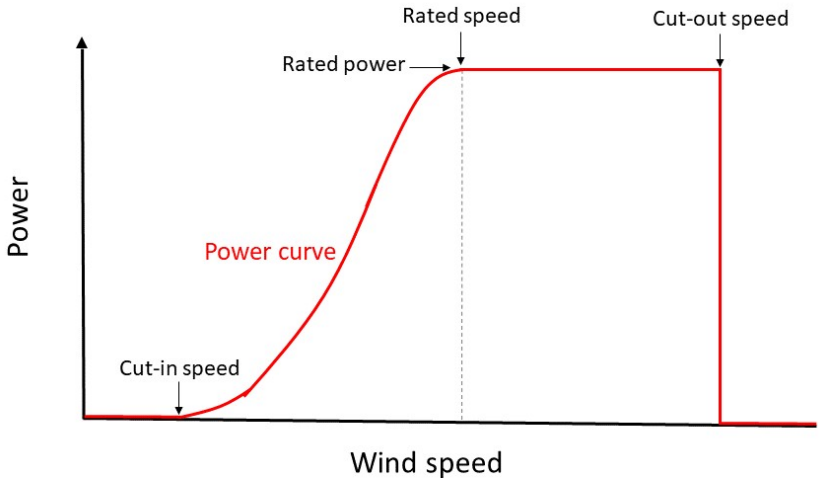


Figure 2.4.1: *Power curve of wind turbine [21]*

A wind turbine has a power performance curve which are characteristic for each wind turbine as shown in Figure 2.4.1. From this curve it is possible to determine how much energy the wind turbine will produce. The curve give the electrical power output as a function of the wind speed [32]. There are three important points on the wind power curve [32]:

Cut-in speed is the minimum wind speed for when the machine will deliver power that can be used.

Rated wind speed is most often the maximum output power reached by the electric generator.

Cut-out speed is the maximum possible wind speed before the turbine stops producing power.

The wind velocity is the most critical factor to the developed power by a wind energy conversion system. A small variation in the wind speed may change the power significantly. The wind speed and wind direction will randomly vary with time and the wind pattern will change. When selecting a turbine for a given site, it is important to know the nature of the wind and how the behaviour may change. If the wind characteristics are known it is possible to assess the generated energy. Knowing the average wind velocity and wind distribution at site can give a preliminary indication of the potential energy. [18][32]

2.5 Energy Storage

A energy storage is often required in a PV/wind energy system to compensate for the fluctuating natures of solar and wind [3]. The energy storage is used to store surplus generated energy. The energy is stored to provide enough energy to supply the load in case of insufficient power generation from the renewable generators [36]. The battery bank capacity will constantly change in a HRES due to the intermittent behavior of PV panels and wind turbine [35].

The battery bank will be in charging state when the total output from the PV panels and the wind turbine is greater than the load energy demand. The battery charging can be obtained by Equation 2.5.1, where $E_{Batt}(t)$ and $E_{Batt}(t-1)$ are the charging quantities at time t and $t-1$ for the battery bank [3]. The hourly self-discharge rate is denoted by σ, and η_{pwm} and η_{pwm} are the efficiency for the *Pulse Width Modulation* (PWM) and MPPT charge controllers [35].

$$E_{Batt}(t) = E_{Batt}(t-1) \cdot (1-\sigma) + [((E_{pv}(t) \cdot \eta_{mppt}) + (E_{wt}(t) \cdot \eta_{pwm}) - E_{Load}(t)] \cdot \eta_{Batt} \tag{2.5.1}$$

E_{pv} and E_{wt} are the energy generated by the solar panels and the wind turbine, and can be calculated by Equation 2.5.2 and 2.5.3 respectively [35].

$$E_{pv}(t) = P_{pv}(t) \cdot \Delta t \tag{2.5.2}$$

$$E_{wt}(t) = P_{wt}(t) \cdot \Delta t \tag{2.5.3}$$

When the total output of the wind turbine and the PV panels is less than the load demand, the battery will be in discharging mode. When the charge efficiency of the battery bank can be assumed 1, the charge quantity of the battery bank is obtained by Equation 2.5.4 [3].

$$E_{Batt}(t) = E_{Batt}(t-1) \cdot (1-\sigma) - [E_{Load}(t) - ((E_{pv}(t) \cdot \eta_{mppt}(+(E_{wt}(t) \cdot \eta_{pwm})] \cdot \eta_{Batt} \tag{2.5.4}$$

2.5.1 Charging and Discharging the Battery

A lead acid battery with a nominal capacity of 120 Ah is commonly used for storing energy in HydMet-stations [30]. For a lead acid battery the capacity will reduce during the storage time of the battery, and after six months time the capacity will be about 86% of the nominal capacity. At rest a battery will loose some of its capacity as a result of *self-discharge* [30]. The self discharge of the battery is depended on the type of battery and the air temperature. For a lead acid battery the commonly self discharge is 4-5% per month [30].

The battery capacity is also determined by the time used to discharge the battery. If the discharging happens rapidly, the capacity will be lower than for a battery where the discharging happen at a slower rate [30]. In charging state, the battery must be supplied with an higher current flow than when discharging the battery.

The *Depth of Discharge* (DoD) is how much the battery can discharge safely before reducing the lifetime of the battery [30]. The DoD is often 50% of the battery capacity [30]. When modeling a battery it is crucial that the battery *State of Charge* (SoC) does not exceed below the minimum value of the battery capacity, and it does not exceed above the maximum value of capacity, as given in Equation 2.5.5. The maximum value of capacity is also the nominal capacity of the battery.

$$E_{batt,min} \leq E_{batt} \leq E_{batt,max} \tag{2.5.5}$$

The minimum capacity of the battery is given by the DoD, and can be calculated by Equation 2.5.6, where $E_{Batt,max}$ is the nominal capacity of the battery.

$$E_{batt,min} = DoD \cdot E_{Batt,max} \tag{2.5.6}$$

The battery efficiency is reduced for each charging cycle (charging and discharging) [30].

2.5.2 Charge Controller

For a small PV/wind/battery hybrid system it is necessary with charge controllers to control the charging of the battery [23]. The efficiency of a PWM and MPPT can be assumed constant [35].

A MPPT charge controller is commonly used for controlling the battery charging from PV panels and to force the PV panels to operate at MPP [33]. The main function of the MPPT is to control the electrical flow

between the PV panels and the battery, depending on the solar radiation available and the battery state of charge [33][23]. A MPPT will extend the lifetime of the battery. It will also control the operation of the PV panels in correspondence to the battery voltage, for the PV panels to operate at maximum power [23]. The MPPT installed must be efficient enough to handle the maximum possible current that can be produced by the PV panels.

A PWM charge controller has the main function of distributing the power generated from the wind turbine between the battery and a dump load to prevent the battery from overcharging. When the battery is fully charged, the charge controller will discard excess energy to the dump load. The PWM charge controller switches on an off the power to the battery at a constant frequency and varied duty ratio to control either the charging voltage or the mean current of the battery [31]. For most wind-electric charge controllers, the AC/DC converter is built within the charge controller [31].

2.6 Machine Learning and Artificial Neural Networks

Machine learning is the subject of implementing computer programs that are able to learn from data how to perform a specified task. The basic concept is to fit the ML model to previously seen data, so the model can be used to predicting and understanding newly observed data [19].

ML is divided into two types: supervised learning and unsupervised learning, as seen in Figure 2.6.1. Supervised learning algorithms, such as classification and regression tasks, use known input and output data to train the ML model to make reasonable predictions as a response to new data. The labels are either discrete categories or continuous quantities. Unsupervised learning algorithms, like clustering and dimensionality reduction, find patterns in datasets without referencing to labels. The model search for distinct groups of data or more succinct representations of the data. [19][24]

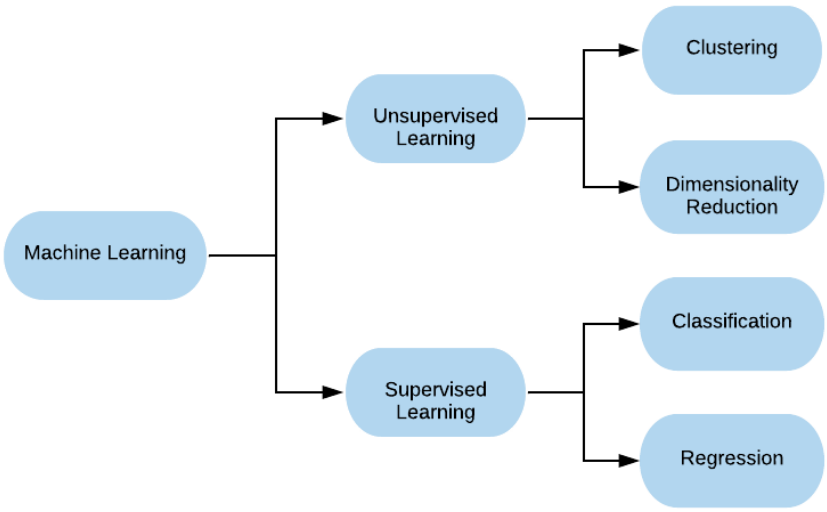


Figure 2.6.1: *Types of machine learning*

Regression

Regression models predict continuous responses in which the labels are continuous quantities. The technique estimates the relationship between variables in multiple dimensions, and is generalized of the well known linear or non-linear regression problem of fitting a line to data with two coordinates. Regression models are used to predict for example changes in temperature or fluctuations in power demand. [19][24]

2.6.1 Deep Learning and Artificial Neural Networks

Traditional ML models tends to focus on one or two layers of data representation, while deep learning models may involves hundreds of successive layers of representations [12]. DL is a subfield of ML based on *artificial neural networks (ANNs)*, where the models are versatile and powerful enough to handle large and highly complex ML tasks [1]. Figure 2.6.2 presents the idea of a deep neural network model. The model combines multiple processing layers, interconnected via *nodes (or neurons)*, where the output of each layer is the input of the next layer [24]. The deep neural network learns through the training data to understand the input object's features and associate them with the corresponding outputs or targets. Each layer in the neural network increase the complexity and amount of details the model learns.

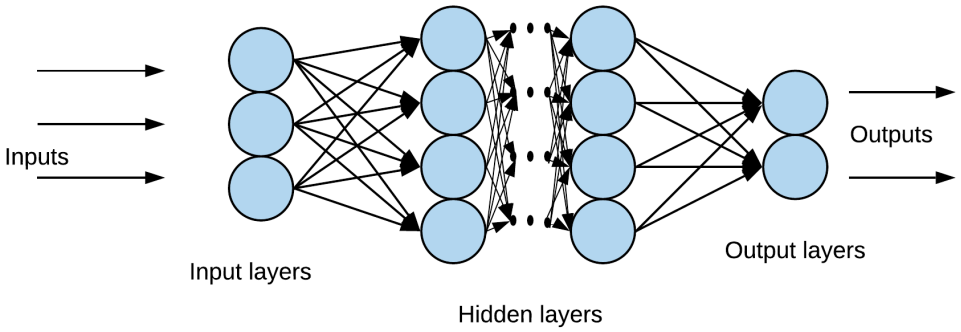


Figure 2.6.2: *A deep neural network*

The *artificial neurons* are the basic unit of an ANN. Each neuron has one or more input connections and one output. Each input connection is associated with a *weight*, which represents the strength of the connection between the neurons. A artificial neuron activates its output if more than a certain number of its inputs are active. The transformation implemented by each network layer is determined by the sum of weights of its inputs ($z = w_1x_1 + w_2x_2 + ... + w_{ij}x_{ij} = \boldsymbol{w^T} \cdot \boldsymbol{x}$). [1]

The output of each neuron is defined according to Equation 2.6.1, where o_j is the neuron output and w_{ij} is the neuron weight. The *activation function* φ sets the output of a neuron by transforming the summed weights input to the neuron. Multiple types of activation function can be used depending on the problem to be solved. [1]

$$o_j = \varphi\left(\sum_{i=1}^{n} w_{ij}o_j\right) \tag{2.6.1}$$

When training, the model learns to find a set of values for the weights of all network neurons, so that the neural network will map the inputs with their targets correctly. [12]

The weights are normally optimized by the *optimizer* which implements a *backpropagation algorithm*. The backpropagation algorithm was presented in [6] in 1986 by D. E. Rumelhart et al, and uses a gradient decent optimization algorithm to train the model. The basic concept is that for each training instance the algorithm makes a prediction, measures the output error, then goes through each network layer in reverse to measure the error contribution of each connection, before it slightly adjusts the weights of each connection to reduce the resulting output error [1]. The *learning rate* of the optimizer controls how much the neurons weights are changed each time they are updated [17]. The weights are updated according to Equation 2.6.2, where w_{ij} is the weight of the neuron connection, η is the *learning rate* of the optimizer, $\hat{y_j}$ is the target output, y_j is the model output, and x_i is the neuron input value [24].

$$w_{ij}^{next\ update} = w_{ij} + \eta(\hat{y_j} - y_j)x_i \tag{2.6.2}$$

The neural network is trained for a number of *instances* where the number of *epochs* is the number of times the entire dataset is processed through the network, and the *batch size* is the number of samples passed to the model at a time.

The performance of the model is evaluated by a *loss function*, reference Equation 2.6.3. Multiple loss functions can be used depending on the problem and type of learning algorithm. When training a neural network, the loss function score can be used as a feedback signal to adjust the weights of the network nodes to optimize the model performance.

$$E = L(\hat{y_j}, y_j) \tag{2.6.3}$$

A network that is trained a sufficient number of times to perform with minimal loss score is called a trained network. [12]

2.6.2 The Architectures of ANNs

The architecture of ANNs are divided into two main categories: *feed-forward neural networks (FNNs)* and *recurrent neural networks (RNNs)*. In FNNs, such as *single-layer perceptrons (SLPs)*, *multi-layer perceptrons (MLPs)*, or *convolutional neural network (CNN)*, the information propagation is in one direction only. In a RNN the output layer nodes may feed-back to the input or hidden layer nodes. [14]

Single-Layer Perceptrons (SLPs)

The perception network is composed of a single layer of neurons, where each neuron is connected to all the inputs. A linear combination of the inputs is calculated and if the result exceeds the threshold, the outputs are activated. A SLP network can be used for simple linear binary classifications, but is incapable of learning a *Exclusive OR* (XOR). A presentation of a SLP network is shown in Figure 2.6.3. [1]

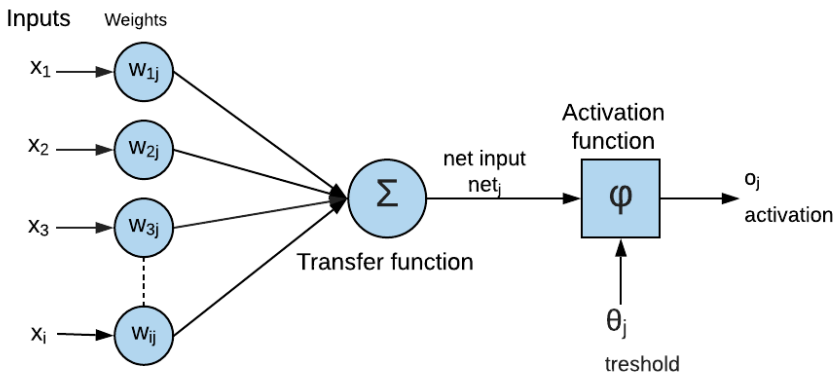


Figure 2.6.3: *Single-layer perceptron network*

Multi-Layer Perceptrons (MLPs)

Stacking multiple perceptrons in a multi-layer perceptron (MLP) network, eliminates the limitations of a SLPs. The network is composed of an input layer, one or more hidden layers, and an output layer, as shown in Figure 2.6.4. [1]

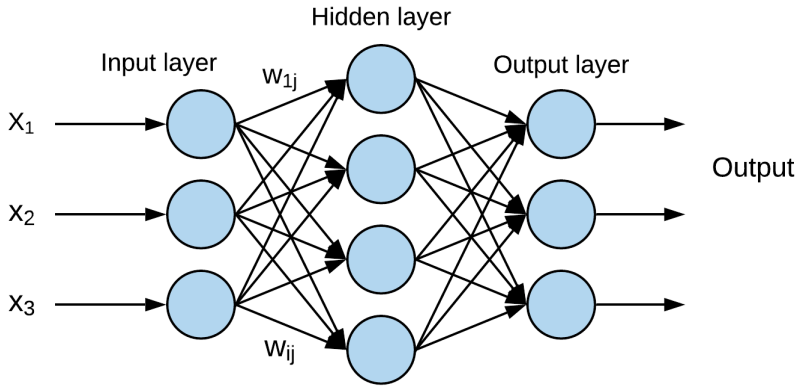


Figure 2.6.4: *Multi-layer perceptron network*

Convolutional Neural Networks (CNNs)

CNNs origins from the study of the brain's visual cortex, and are used for pattern recognition and natural language processing. Each convolutional layer serve as a filter for the output of the previous layer. Between each convolutional layer the features are compressed and generalized in a so called pooling layer. The input to the following convolutional layer is called a feature map. [1]

Recurrent Neural Networks (RNNs)

In a recurrent neural networks (RNNs) each neuron has a feed-back signal in addition to the feed-forward activation. At each frame, the recurrent neuron receives the input signal from the previous frame as well as its own output, as visualized in Figure 2.6.5. The output of a recurrent neuron is a function of all the inputs from the previous frames, giving the neural network memory and knowledge about what it has seen before. RNNs are used to analyze and predict time series data, as the network is capable to learn broader abstractions in the input data. [1]

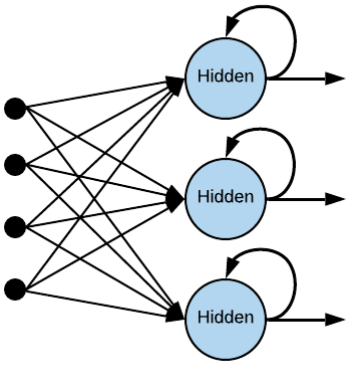


Figure 2.6.5: *Recurrent neural network model*

2.6.3 ANN Model Accuracy and Randomness

ML algorithms are stochastic and use randomness to solve the complex problems. The randomness within the field of ML involves randomness in the collection of the data, the order the data is exposed to the model, the initial state of the model which may be random, randomness in data division, and so on. The stochastic behaviour of the algorithms will affect the performance of the model on different sets of data.

2.7 The Process of Developing an ANN Model

The process of developing an ANN model includes several steps. Depending on the problem to be solved, the available data, and the desired performance, the importance and time spent on each step varies.

The process described in this section is based on [1] and [14]. Figure 2.7.1 shows the flow of the process.

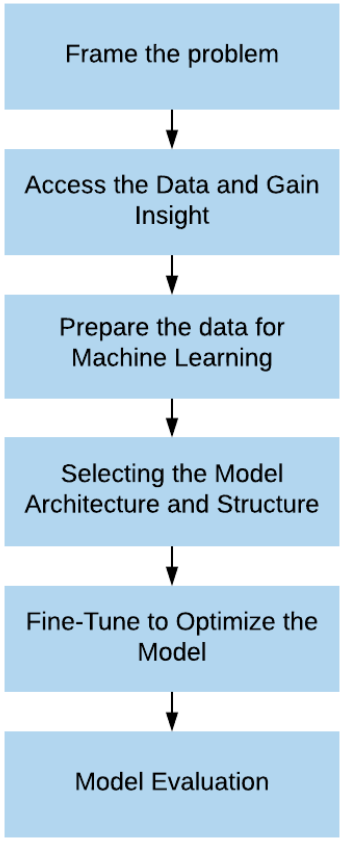


Figure 2.7.1: *Process of developing an ANN model*

2.7.1 Frame the Problem

The first step of developing an ANN model is to define the problem at hand; what is the end goal and how will the model be used? Having a clear definition of the problem will help determine how much time and effort to spend on each step of the development process, decide what algorithms to use, and help select a performance measure for evaluating the model performance.

2.7.2 Access the Data and Gain Insight

A important step in the ANN development process is to gain insight in the data. It is important to identify the available parameters, how much data is available, and how the parameters are related.

The relation between *features* (model inputs) and *targets* (model outputs) is especially important for *input selection*.

2.7.3 Input Selection

Selecting the right and correct number of features is important for the performance of the model. Excluding important features will prevent the model from learning the best possible input-output relationships, while including too many inputs may provide redundant information and increase the likelihood of overtraining the model.

Multiple techniques are used to understand the relationship between the inputs and outputs, and subsequently select the appropriate model inputs. *Correlation* is the most commonly used method for measuring statistical independence, whoever it only measure the linear dependence between the variables. For the purpose of ANN input selection the use of a non-linear dependence method such as *mutual information* is more precise.

Pearson's Correlation Coefficient

Pearson's correlation coefficient (R) measures the linear correlation between features and targets. The correlation is given by Equation 2.7.1, where $cov(X, Y)$ is the covariance of X and Y, $std(X)$ is the standard deviation of X, and $std(Y)$ is the standard deviation of Y. A correlation close to +1 indicate a strong positive relationship between the feature and target, whereas a correlation close to -1 indicate a strong negative relationship. A correlation close to zero (0) indicate no linear relationship between the variables. [17]

$$R = \frac{cov(X, Y)}{std(X) * std(Y)} \tag{2.7.1}$$

Spearman Rank Correlation

Spearman rank correlation (r_s) measures the non-linear relationship between features and targets. The correlation is given by Equation 2.7.2, where $cov(rank(X), rank(Y))$ is the covariance of the rank variables, and $std(rank(X))$ and $std(rank(Y))$ are the standard deviation of the rank variables. The correlation ranges from -1 to +1 for a perfect negatively or positively correlation, as for the Pearson's correlation coefficient. [17]

$$r_s = \frac{cov(rank(X), rank(Y))}{std(rank(X)) * std(rank(Y))} \tag{2.7.2}$$

Mutual Information

Mutual information (MI) is a measure of mutual dependency between two random variables. It quantifies how much of a variable is carried by another variable. The *average mutual information* (AMI) of a set of discrete data, $X = \{x_1, x_2, x_3, ..., x_n\}$ and $Y = \{y_1, y_2, y_3, ..., y_n\}$, is given in Equation 2.7.3, where $p(x, y)$ is the joint probability distribution, and $p(x)$ and $p(y)$ are the marginal distributions of X and Y respectively.

I(X,Y) is equal to zero if the two random variables are independent, and increase as the dependency of the variables increases. [28][8]

$$I(X,Y) = \sum_{y}\sum_{x} P(x,y) log \frac{P(x,y)}{p(x)p(y)} dxdy \tag{2.7.3}$$

2.7.4 Prepare the Data for Machine Learning

Before feeding the data to the ML model it should be processed to ensure the model is based on correct and formatted data. Pre-processing the data includes cleaning the data, handling text and categorical parameters, and scaling the data.

Data cleaning involves taking care of missing samples, detecting outliers, removing irrelevant columns, and other necessary cleanups. Zero (0), the mean, or the median, should replace missing values or "NaN"-values as most ML models are not capable of working with these. This also applies to text labels and categorical attributes, which must be converted to numbers. Many ML algorithms requires consistent input values to learn well from the data and for the central tendencies of the data to be normalized. *Standardization* or *normalization* are commonly used for scaling the attributes and is an important step of the data pre-processing. Standardization does not limit the values to a specific range as normalization does, which can cause problems for some neural network models, however standardization is much less affected by outliers in the data [16].

Standardization

Standardizing the data rescale it so that the mean value of each attribute is zero (0) and the standard deviation is one (1). A value is standardized as given in Equation 2.7.4, where x is the value, $\overline{x}$ is the mean, and STD is the standard deviation of the attribute. [16]

$$z = \frac{x - \overline{x}}{STD} \tag{2.7.4}$$

2.7.5 Data division

The data is split into a *training set*, *validation set*, and *test set*, for training, validating and testing the ANN model respectively.

The model will be trained using the training set, in which it will see and learn from. The validation set will provide a unbiased evaluation of how the model performs, and is used for fine-tuning the model hyper-parameters. The test set will be an unseen dataset for testing how the model performs on new data, and should be used as a final evaluation of the model performance when the model is trained. 80% of the data is commonly used for training and validation, while 20% of the data is held out for testing. To avoid holding out to much of the training data for validation purposes, *cross-validation* is often used for training and validating the model.

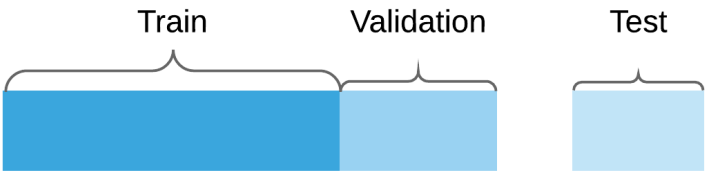


Figure 2.7.2: *A visualization of data division for ML*

Cross-validation

Cross-validation (CV) is a procedure of dividing the training data into k sets. The model is trained by $k-1$ sets of the training data, and validated on the remaining part of the training data. This holdout method is repeated k times where the validation set rotates between the sets, to optimize the model learning. The average error is estimated for all the sets. Figure 2.7.3 describes the principle of k-fold cross validation. [38][1]

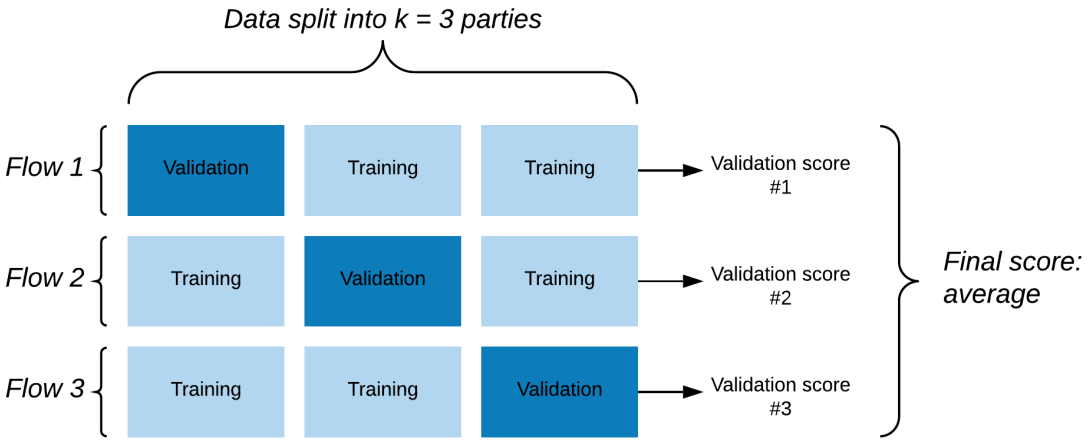


Figure 2.7.3: *Principle of k-fold cross validation*

2.7.6 Selecting the Model Architecture and Structure

The model architecture and structure decides the flow of information in the model and defines how the model relates the input features and the targets. The optimal network architecture and structure balances generalisation ability and complexity, considering the problem to be solved. Deciding on the model structure involves selecting the model hyperparameters such as:

- Number of hidden layers
- Number of neurons in each hidden layer
- Activation function

- Optimizer
- Learning rate
- Weight initializer
- Number of epochs
- Batch size
- Dropout rate

According to [14], MLP is the most popular model architecture. MLP is flexible and can be used for both regression and classification problems [17]. For most problems one or two hidden layers will create a baseline for model accuracy. The number of hidden layers can then be increased to increase the accuracy, until the model begins to *overfit* the training set. The latter also applies to the number of neurons per layer. The simplest practice is to use the same number of neurons per layer, and slowly increase the number until overfitting occurs.

In most cases the *rectified linear unit (ReLU)* activation function can be used in the hidden layers. The activation function in the output layer depends on the model type. For a regression problem the output layer typically has no activation function.

The most popular optimizers are *Gradient Decent*, *Adam*, *RMSProp*, *Momentum optimization*, *Nesterov Accelerated Gradient* and *AdaGrad* [1]. The learning rate should be set to balance learning speed and model performance [17].

The training algorithm requires the weights to have a initial point from which to begin the iterations. The weight initializer determines how the weights will be initialized. Common methods are *random normal*, *random uniform*, *glorot normal (Xavier normal)*, *glorot uniform (Xavier uniform)*, or using *zeros*.

The number of epochs should be optimized to minimize the prediction error while avoiding overfitting. The model is said to overfit when the validation error stops decreasing and start to increase again, as seen in Figure 2.7.4 [1].

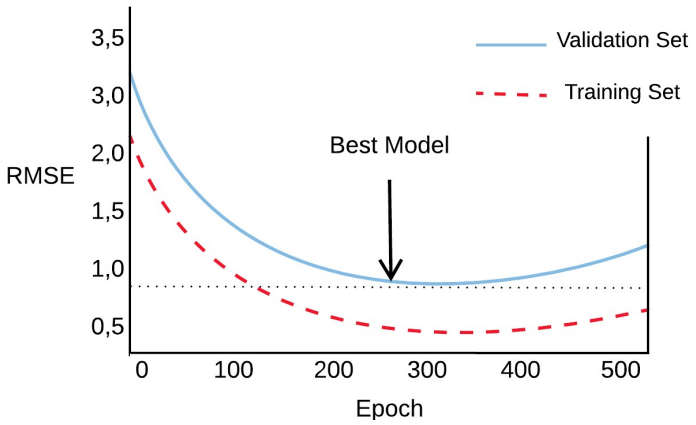


Figure 2.7.4: *Example of model overfitting*

The batch size controls the accuracy of the estimate of the error when training the model. It is normally set between one and a few hundreds, or is typically set to 32 [17].

Dropout is one of the most effectively and common methods to avoid overfitting the model. A dropout layer randomly "drops out" a number of features from the layer, depending on the dropout rate. Usually it is set between 0.2 and 0.5 [12].

2.7.7 Fine-Tune to Optimize the Model

Fine-tuning the model means optimizing the hyperparameters to obtain a good model performance. For fine-tuning a neural network on a large dataset, algorithms such as *randomized search* or *grid search* can be used to efficiently optimize the hyperparameters.

2.7.8 Model Evaluation

The performance of the model should be evaluated by the performance measures substantial for the problem.

Squared errors, such as the *root mean square error (RMSE)*, quantify the difference between the predicted values and the target values. Squared errors tend to be dominated by errors with high magnitudes. The RMSE is given in Equation 2.7.5 where N is the number of samples measured, y_i is the predicted values and $\hat{y}_i$ is the target values. A lower RMSE is commonly accepted to indicate a better performance. [7]

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2} \tag{2.7.5}$$

Absolute or relative errors are alternatives to squared errors. Absolute errors measures the absolute difference between predicted and target values. They do not however provide information concerning over- or under-prediction. Relative errors allows models with different output magnitudes to be compared. The MAE is given in Equation 2.7.6, where $\hat{y}_i$ is the target values, y_i is the predicted values, and N is the number of samples. A MAE of zero indicate no prediction error. [7]

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|\hat{y}_i - y_i| \tag{2.7.6}$$

To address the issue of under- or overprediction, the *mean bias error (MBE)* should be taken into account to quantify the differences in the predicted values and the target values. MBE is calculated according to Equation 2.7.7, where $\hat{y}_i$ is the target values, y_i is the predicted values, and N is the number of samples. A lower MBE indicate a better model performance [7].

$$MBE = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i) \tag{2.7.7}$$

A measure of correlation is also a conventional performance measure used for ANN models. *Pearson's correlation coefficient* is the most well-known measure of empirical error between the model predicted values and target values. A comparable measure of correlation is the *coefficient of determination* (R^2). Although the correlation metrics are commonly used, they are sensitive to outliers and give no information about proportional differences in the predicted and target values. The correlation coefficient between target values $\hat{y}_i$ and predicted values y_i, can be described by Equation 2.7.8, where N is the number of samples, and $\overline{\hat{y}_i}$ and $\overline{y_i}$ are the average of the target and predicted values respectively. The Pearson's coefficient ranges from -1 to 1, indexing the degree of correlation between the values. If the correlation coefficient is zero, no correlation exists. [7]

$$R = \frac{\sum_{i=1}^{N}(\hat{y}_i - \overline{\hat{y}}) * (y_i - \overline{y})}{\sqrt{\sum_{i=1}^{N}(\hat{y}_i - \overline{\hat{y}})^2 * \sum_{i=1}^{N}(y_i - \overline{y})^2}} \tag{2.7.8}$$

R^2 describes the proportion of variance in the measured data, and is given by Equation 2.7.9, where y_i is the predicted values, $\hat{y}_i$ is the target values and $\overline{\hat{y}_i}$ is the average of the target values, and N is the number of samples. R^2 ranges from 0 to 1, where a higher value indicate a lower error variance. A measure greater than 0.5 is considered acceptable. [7]

$$R^2 = 1 - \frac{\sum_{i=1}^{N}(\hat{y}_i - y_i)^2}{\sum_{i=1}^{N}(\hat{y}_i - \overline{\hat{y}_i})^2} \tag{2.7.9}$$

2.8 The Genetic Algorithm

The Genetic Algorithm (GA) is an advanced search process based on the natural selection [36]. By using techniques inspired by natural evolution such as inheritance and mutation, the GA generates solutions to optimization problems. The advantages of the GA is the possibility to easily solve problems with multiple solutions, and it can easily be implemented in existing simulations [36]. The GA will find the global solutions in multi-objective optimization problems [20]. A disadvantages of the GA is that it can go into premature convergence. A premature convergence is when the objective function converge to early, leading to a suboptimal system [41]. This happens when the parents are not able to generate superior offsprings [41]. The chance of premature convergence can be reduces by finding the optimal parameters for the GA.

There are normally three operators in use when applying the GA as a method; *selection, crossover* and *mutation*, to best imitate the natural evolution processes.

The basic concept of the GA is to initialize a number of chromosomes with random genes. This is the first generation of the population. The individuals in the generation will be evaluated based on their fitness, which determines the individuals ability to be carried on to the next generation or being mated. The individuals selected for mating will produce offsprings for the next generation. The GA will age through a number of generations before settling at a final solution. [9]

A GA is usually implemented by the following steps [25]:

- Define the available genes each chromosome can have.

- Define a chromosome which is a particular solution or an individual.
- Create a population which is a collection of possible solutions (chromosomes/individuals).
- Define the objective function for fitness evaluation.
- Use a selection mechanism to select chromosomes/individuals for the next generation and breeding.
- Use crossover to produce new individuals for next generation.
- Use mutation to introduce randomness into the next generation.
- Iterate through a number of generations to obtain the optimum solution.

Fitness

The individuals in a generation is ranked based on their fitness. The fitness is determined based on how well the individual performs according to the objective function and the system constraints.

Selection

Selection is used to determine which chromosomes are carried on to the next generation and which chromosomes will be mated to create new individuals in the next generation. One of the most common methods for selection is the *fitness proportionate selection*, where the chromosomes in the current generation are evaluated based on their fitness. The individuals with the highest fitness has a higher probability of being selected for the next generation.

Crossover

Crossover is a breeding process for creating the next generation of chromosomes [9]. Multiple methods exists for breeding a population. One method is ordered crossover, where randomly selected subset from the first parent is put together with the remainder from the second parent, as shown in Figure 2.8.1 [9].

Figure 2.8.1: *Random crossover breeding*

Mutation

Mutation is a process which are used to avoid that the objective function will convergence towards the local optimum instead of the global [9].

Mutation assign a low probability of a gene changing. Multiple methods can be used to mutate a gene. In some cases genes in a chromosome are swapped with another gene in the chromosome, or a randomly selected gene will replace the mutated gene. Figure 2.8.2 shows the principle of mutation where two random genes are changed to a random selected value withing the available range.

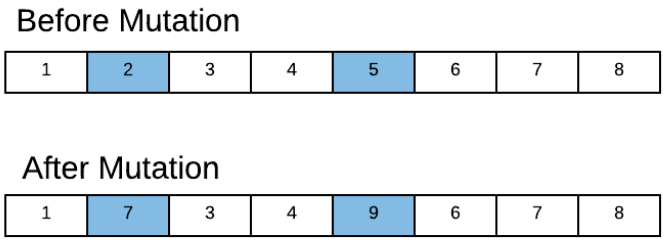


Figure 2.8.2: *Mutation of individuals*

2.8.1 Objective Function and Constraints

When optimizing a PV/wind/battery energy system using the GA it is necessary with an objective function for fitness evaluation within given constraints.

Objective Function

The objective function is defined by Equation 2.8.1, where the goal is to minimize or maximize a numerical value Z [40]. The coefficient c_i of the objective function is corresponding to the *i-th* variable, and X_i is the decision variables.

$$maximize\ or\ minimize\ Z = \sum_{i=1}^{n} c_i \cdot X_i \tag{2.8.1}$$

For a PV/wind/battery energy system the objective function will be to minimize the initial cost of the system, which is the sum of the installed components costs.

Constraints

To be able to meet the load demand needed for the PV/wind/battery energy system it is necessary to include a constraint regarding the load demand in the calculations of the optimization problem. In Equation 2.8.2 the main constraint for meeting the load demand is presented [2]. $E_{pv,actual}(t)$, $E_{wt,actual}(t)$ and $E_{batt}(t)$ are the sums of the energy obtained from the PV panels, wind turbine and the battery. The resulting power by adding these sums must to be bigger than the load demand $E_{Load}(t)$ at all times for the system to supply sufficient energy. [2]

$$E_{pv,actual}(t) + E_{wt,actual}(t) + E_{batt}(t) \geq E_{Load}(t) \tag{2.8.2}$$

The constraints regarding the battery are presented in Section 2.5 in Chapter 2 under *Charging and Discharging the Battery* by Equation 2.5.6 and 2.5.5.

2.8.2 Optimization of a PV/Wind/Battery Hybrid System

The optimization of a PV/wind/battery hybrid energy system is a non-convex and non-linear problem. To effectively solve such an optimization problem there is need for a superior optimization technique such as the GA [2]. The GA can be used to find the combination of system components for a PV/wind/battery hybrid system that should give the most cost-effective solution and still supply the electrical load at all times. [3][15]

The first step of implementing a GA in the optimization problem, is to define the available components and the associated genes for each of the components [15]. The initial population should be a number of system configurations. Each gene should represent the installed capacity of the component. [15] Each system configuration in the population will be evaluated based on how it meets the objective function and the constraints [2].

The system configurations with highest fitness are carried on to the next generation or mated (through crossover) to create new system configurations for the next generation. The GA should move through a number of generations of system configurations in search for the system configuration with the highest fitness i.e. lowest cost. To avoid converging to a local minimum, new individuals should be mutated to introduce randomness in the next generation [15].

Chapter 3

Methodology

The methodology used comprises the use of a case study where an HRES is installed to monitor all relevant system data and weather data at site. The data collected in the case study will be used to train ANN models based on the weather parameters. The models will be used to predict the hourly power production by the PV panels and wind turbine. The results will be used in the optimization problem where the installed capacity for PV, wind power and battery size are the target variables. The battery charge and discharge is also considered, in addition to the load profile. The optimization problem is solved using the GA, ensuring that at all hours the load is ensured in the hybrid system.

3.1 Case Study: Test Hydrology- and Meteorology-station at Scanmatic AS Headquarters

The data used in this project is obtained from the test HydMet-station installed by Scanmatic AS. The system is installed at the headquarters of Scanmatic AS in Arendal, Norway (lat. 58.4757, long. 8.818220). The system is shown in Figure 3.1.1.

The installation consists of four 20 W mono crystalline silicone PV panels, for measuring PV power production. All panels have different tilt and orientations. A horizontally installed pyranometer and temperature sensor is included in the system to measure the global irradiance and operational temperature of one of the PV panels.

A "constant voltage dummy load" is used to measure the actual produced effect of the PV panels. The basic concept is to stepwise measure the current and voltage from the PV panels between 10-20 V in 0.1 V intervals, to determine the maximum effect. The method can be compared to the operation of a MPPT. The maximum effect of each "cycle" is saved to the data with the corresponding voltage.

Figure 3.1.1: *Test HydMet-station at Scanmatic AS headquarters*

The installation also consists of a 300 W wind turbine for measuring wind power production. A wind sensor is installed in line with the wind turbine to give information about the wind speed and direction. The wind speed is calculated from the frequency of the wind sensor multiplied by a factor set for the sensor.

A 1.4 kWh lead-acid battery is installed for storing energy, and a dump load to manage excess power produced by the generators [30]. Due to the remote location for a HydMet-installation, the capacity for the battery is desired to be able to handle three months with no power production from the PV panels and the wind turbine.

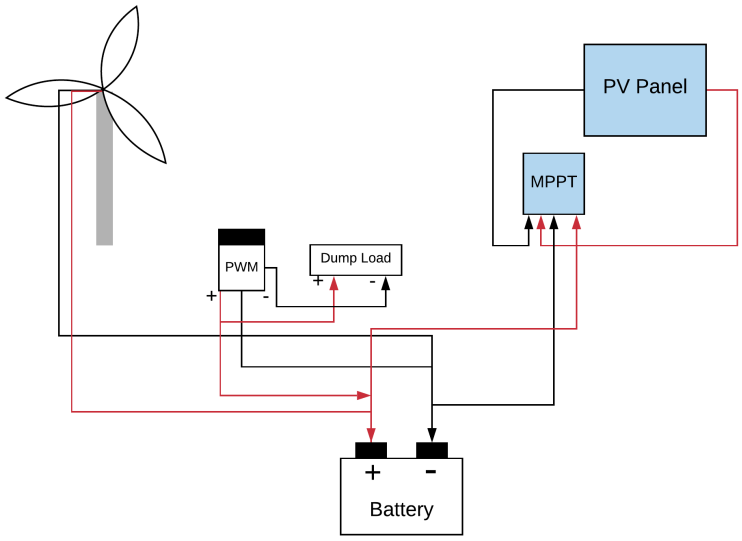


Figure 3.1.2: *Setup for the test HydMet-station*

Figure 3.1.2 shows the schematic of the HRES considered in this thesis. The figure shows the different components (PV panel, wind turbine, and battery) optimized using the GA, as well as the MPPT charge controller, PWM charge controller and dump load.

Table 3.1.1 presents the key information about the test HydMet-station. The datasheet of the main components can be found in Appendix A, B and C.

Table 3.1.1: *Key information of test HydMet-station at Arendal, Norway*

Component	Type/Model	Installed capacity	Configuration	Cost
Wind turbine		300 W	Cut-in: 3 m/s, Rated: 85 W at 8 m/s	9750 NOK[1]
PV panel (no.1)	Mono-Si	20 W	South oriented at 45° tilt	500 NOK[2]
PV panel (no.2)	Mono-Si	20 W	Southeast oriented at 90° tilt	500 NOK[2]
PV panel (no.3)	Mono-Si	20 W	South oriented at 90° tilt	500 NOK[2]
PV panel (no.4)	Mono-Si	20 W	Southwest oriented at 90° tilt	500 NOK[2]
Battery	Lead-acid	1.4 kWh	σ = 4-5% per month, DoD = 50%	2300 NOK[3]
Charge controller	PWM	45/60 A	η=0.80	2250 NOK[1]
Charge controller	MPPT	15 A	η=0.97	1100 NOK[1]
Dump load		500 W		2364 NOK[1]

1. Collected from [22], assumed VAT of 25% and a currency of 11.82 GBP per 1 NOK
2. Collected from [4]
3. Collected from [5], assumed VAT of 25% and a currency of 11.82 GBP per 1 NOK

The data has been logged since March 2019, and is logged continuously throughout the span of this project. An overview of the data collected from the HydMet-station with component ID-number and unit is presented in Table 3.1.2.

All data are collected at one minute timestamps.

Table 3.1.2: *Data logged at the test HydMet-station*

Data No	Data	Unit
1	PV1 MPPT Power	W
2	PV2 MPPT Power	W
3	PV3 MPPT Power	W
4	PV4 MPPT Power	W
5	PV1 MPPT Voltage	V
6	PV2 MPPT Voltage	V
7	PV3 MPPT Voltage	V
8	PV4 MPPT Voltage	V
9	PV1 Power at 14V	W
10	PV2 Power at 14V	W
11	PV3 Power at 14V	W
12	PV4 Power at 14V	W
13	PV Pyranometer (irradiance)	W/m^2
14	PV3 Temperature	°C
15	Wind Current	A
16	Wind Voltage	V
17	Wind Power	W
18	Wind Speed	m/s
19	Wind Direction	degree

3.1.1 Load Requirement/Load Profile

The energy consumption from the HydMet-station is the sum of the energy consumption of all the components installed. The test-station is assumed to draw a constant of 60 mA, which give a daily consumption of 17.28 Wh. For the purpose of this study, the hourly power demand is assumed constant at 0.75 W.

3.2 Machine Learning for Estimating Energy Production

To calculate and predict the power produced by the renewable generators, machine learning in Python 3 is used. The objective of this task is approached as described in Section 2.7 in Chapter 2. First, the logged data is collected and examined to gain insight in the data and its behaviour. Secondly, the data is pre-processed for machine learning, before the ANN are designed and trained with the data. The final step is to test the neural networks and evaluate the models performance. A total of five ANN models are built: one for each PV panel and one for the wind turbine.

3.2.1 Machine Learning in Python

For solving the project issue *Python 3* is used. Python is a popular high-level general-purpose programming language. For implementing machine learning algorithms in Python *Scikit-Learn*, *TensorFlow* and *Keras* are used. Scikit-Learn is a Python library package that provide versions of a large number of common machine learning algorithms. TensorFlow is a more complex open-source framework for solving numerical problems, and is capable of training and running large neural networks. Keras is a high-level neural network API used for fast and easy integration of Tensorflow in Python.

3.2.2 Accessing the Data and Gaining Insight

The raw data measured at the HydMet-station are collected in a four column csv-file. Figure 3.2.1 show the result from printing the first lines of the data.

	Data_No	Timestamp	Status	Value
0	1	2019-03-13 05:01:00.000	0	0.0
1	1	2019-03-13 05:02:00.000	0	0.0
2	1	2019-03-13 05:03:00.000	0	0.0
3	1	2019-03-13 05:04:00.000	0	0.0
4	1	2019-03-13 05:05:00.000	0	0.0

Figure 3.2.1: *Head lines of raw data*

The "Data No" represent the component ID (as shown in Table 3.1.2), "Timestamp" is the time of the sample, "Status" indicates whiter or not there was an error at the time the sample was collected, and "Value" stores the measured value of that component.

Formatting the Raw Data

For the purpose of working with the data, the format of the raw data is changed as shown in Figure 3.2.2.

	Timestamp	PV1_Power	PV2_Power	PV3_Power	PV4_Power	PV_Pyranometer	PV3_Temperature	Wind_Power	Wind_Speed	Wind_Dir
0	2019-03-13 05:01:00.000	0.0	0.0	0.0	0.0	-2.169517	10.504681	0.005107	0.417515	299.548492
1	2019-03-13 05:02:00.000	0.0	0.0	0.0	0.0	-2.086807	10.540688	0.004974	0.950362	278.378418
2	2019-03-13 05:03:00.000	0.0	0.0	0.0	0.0	-2.186663	10.581556	0.004613	0.851066	287.892365
3	2019-03-13 05:04:00.000	0.0	0.0	0.0	0.0	-2.228700	10.620809	0.003906	1.735841	266.965881
4	2019-03-13 05:05:00.000	0.0	0.0	0.0	0.0	-2.251611	10.662459	0.007453	2.329313	257.426666

Figure 3.2.2: *Head of formatted raw data*

The formatting is done by counting the number of timestamps for each component ID, and then extracting the samples of each ID from the dataset and append them to the corresponding column in a nxm dataframe. The script is written so as when the measured dataset increase throughout the project, the method can easily be applied to the new dataset.

Gaining Insight in the Data

Figure 3.2.3 describes all the measured data from the HydMet-station.

	PV1_Power	PV2_Power	PV3_Power	PV4_Power	PV1_Voltage	PV2_Voltage	PV3_Voltage	PV4_Voltage	PV1_Power_at_14V	PV2_Power_at_14V
count	316931.000000	316931.000000	316931.000000	316931.000000	316931.000000	316931.000000	316931.000000	316931.000000	316931.000000	316931.000000
mean	3.216047	1.964899	2.089146	2.314403	13.499778	12.539573	12.563054	12.652210	2.850451	1.802297
std	5.709594	4.008125	3.871209	4.240652	3.112103	2.581474	2.831813	2.700076	5.111260	3.650532
min	0.000000	0.000000	0.000000	0.000000	9.865001	9.844001	9.844001	9.844001	0.000000	0.000000
25%	0.000000	0.000000	0.000000	0.000000	10.210000	10.081000	9.995001	10.081000	0.000000	0.000000
50%	0.288667	0.138750	0.137333	0.161250	13.882251	11.674750	11.100667	11.750251	0.277000	0.084000
75%	2.922250	1.557000	1.660333	1.981750	16.488501	15.092668	15.436667	15.471001	2.526000	1.495000
max	26.043251	32.700249	35.145000	35.233501	19.649750	19.644001	19.572668	19.731001	23.668751	28.618336

PV2_Power_at_14V	PV3_Power_at_14V	PV4_Power_at_14V	PV_Pyranometer	PV3_Temperature	Wind_Current	Wind_Voltage	Wind_Power	Wind_Speed	Wind_Dir
316931.000000	316931.000000	316931.000000	316931.000000	316931.000000	316931.000000	316931.000000	316931.000000	316931.000000	316931.000000
1.802297	1.842620	2.101780	164.012846	15.378742	0.161740	13.230760	2.219891	0.281568	203.473373
3.650532	3.364891	3.819388	278.488706	9.675878	0.435306	0.340735	5.986883	0.723572	68.574609
0.000000	0.000000	0.000000	-11.300942	-6.084168	-0.003067	12.756442	-0.040976	0.000000	39.576393
0.000000	0.000000	0.000000	-2.046397	8.832392	0.000458	12.926030	0.005967	0.095802	167.855766
0.084000	0.056000	0.111000	32.338951	13.396820	0.000612	13.078149	0.007992	0.170880	202.998489
1.495000	1.595667	1.890000	269.203278	20.198290	0.071145	13.600767	0.958755	0.275738	264.652634
28.618336	28.736002	28.781002	11934.000000	53.438782	8.661883	13.870436	118.509064	19.467390	327.772766

Figure 3.2.3: *Measured data described*

The power produced by PV panels have proven to be dependent on the solar irradiance. The operational temperature of the PV panel have also shown to affect the power production. Figure 3.2.4 and 3.2.5 illustrates the relationship between solar irradiance, operational temperature, and the power produced by the four PV panels for two random days.

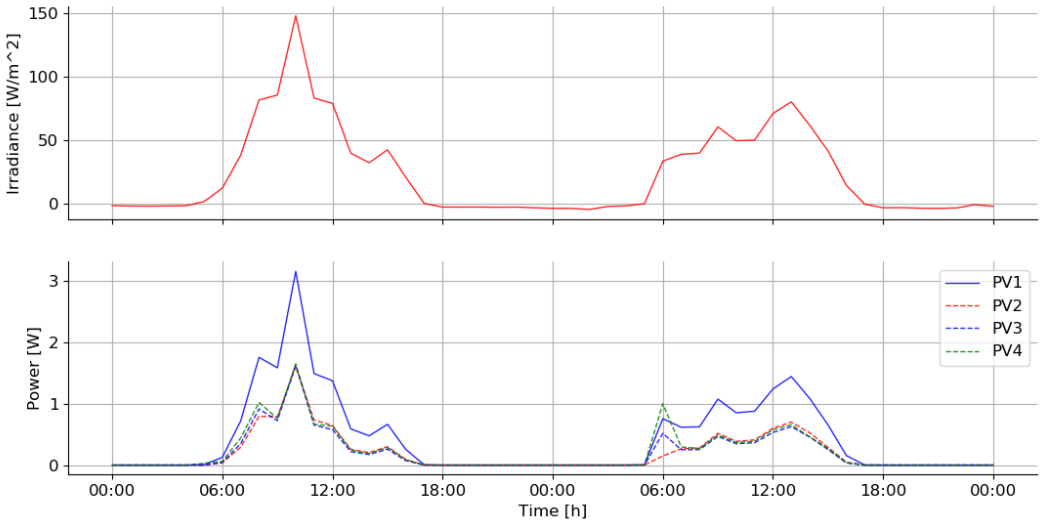


Figure 3.2.4: *Relationship between solar irradiance and PV power production*

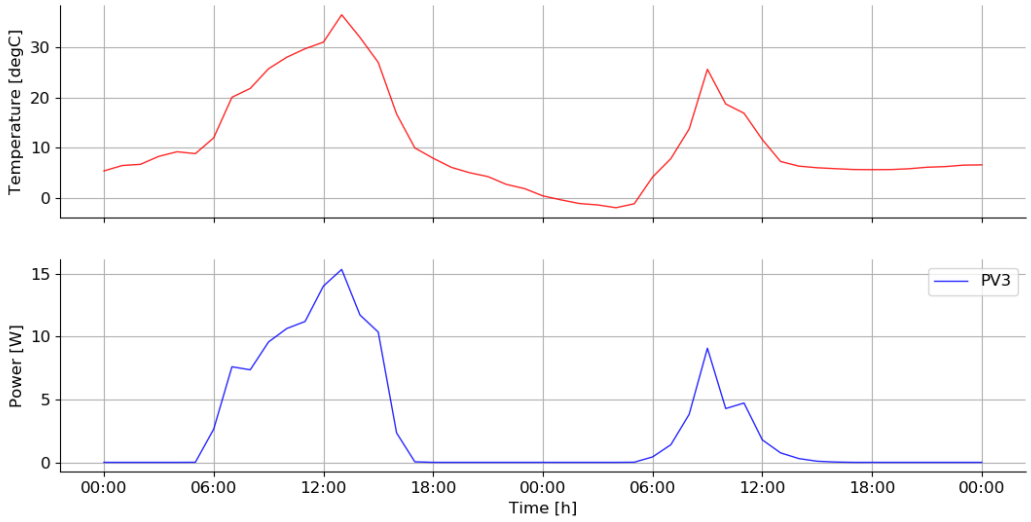


Figure 3.2.5: *Relationship between panel temperature and PV power production*

For the wind turbine power production, the main factor affecting the production is the wind speed. Figure 3.2.6 illustrates the the relationship between wind speed and wind power production for two random days of data.

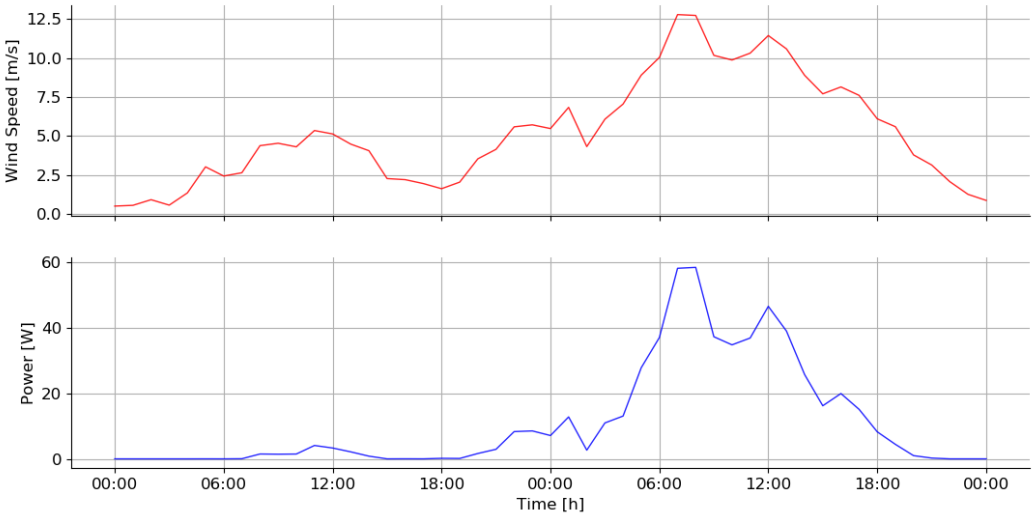


Figure 3.2.6: *Relationship between wind speed and wind power production*

The data focused on throughout the project is presented in Table 3.2.1.

Table 3.2.1: *Data logged at the test HydMet-station included in ML*

Data No	Data	Unit
1	PV1 MPPT Power	W
2	PV2 MPPT Power	W
3	PV3 MPPT Power	W
4	PV4 MPPT Power	W
13	PV Pyranometer (irradiance)	W/m^2
14	PV3 Temperature	°C
17	Wind Power	W
18	Wind Speed	m/s
19	Wind Direction	deg

3.2.3 Pre-Processing the Data for Machine Learning

Cleaning the Data

Data cleaning is included in the pre-processing to correct data-logging faults and handle missing values and anomalies. The data cleaning can be summarized accordingly:

- if PV Pyranometer < 0 $then$ PV Pyranometer = 0
- ifTimestamp < sunrise or Timestamp > sunset $then$ drop the sample
- if PV Pyranometer > mean+std*3 $then$ PV Pyranometer = mean

- if PV Pyranometer < mean-std*3 $then$ PV Pyranometer = mean
- PVx Power$_{scaled} = \frac{\text{PVx Power}_{raw}}{\text{PVx Power}_{installed}}$
- if Timestamp < 2019-03-21 $then$ Wind Speed = (Wind Speed)$_{raw} * \frac{0.0098}{0.166}$
- Wind Speed $=$ (Wind Speed)$_{raw} * 10$
- Wind Power$_{scaled} = \frac{\text{Wind Power}_{raw}}{\text{Wind Power}_{installed}}$

The solar irradiance should not take negative values. The negative measurements for solar irradiance are therefore replaced with zero (0) as it is assumed that no irradiance is present at the timeframes where these anomalies occur.

For the solar ANN model to only consider samples of the data where there is sunlight, the samples between sunset and sunrise are removed. The process of removing the nighttime-samples were done in two steps: 1.) Collect the time of sunrise and sunset at the location for each day within the data timeframe, 2.) Remove the samples between the sunset of one day and the sunrise of the next day. The sunrise and sunset of each day was collected for Oslo city by the use of the Python-package *Astral*. The nighttime data samples were removed by looping through each row of the solar data and comparing the timestamp with the sunrise and sunset times. If the sample was collected before sunrise or after sunset of the given date, the sample is dropped from the data.

Standard deviation is used to detect and replace the observed anomalies in the solar irradiance. Anomalies in the solar irradiance are replaced with the mean value.

The logged wind speed has a decimal error, and is therefore multiplied by a factor of 10 to correct this error. In addition, the wind speed logged before March 20 has a wrong proportionality factor, and is therefore multiplied by $\frac{0.0098}{0.166}$ to correct this error.

To generalize the power produced by the PV panels and the wind turbine, the measured power is divided by the installed power of the generators.

A Python-module containing all necessary functions for cleaning the data is created, to have the ability of repeating the data cleaning for future data sets.

Scaling the Data

To reduce processing time of the ML model, the data size is reduced by scaling the samples to hourly average values instead of minute averages. The hourly average values are calculated by grouping the minute samples by the hour and then calculating the average of the values.

Feature Selection

The input features are selected based on prior knowledge of the data, data correlations and mutual information.

Pearson's correlation coefficient and Spearman rank correlation are calculated for all data to learn how the data is linear and non-linear dependent. Mutual information is also used to gain knowledge of the non-linear correlation between the features and targets in the data.

Data Division

The data is divided with a 80/20 division where 80% of the data is used for training and validation, and 20% is held out for testing. A Scikit-Learn function is used to randomly select samples for the training+validation and test sets. The average and standard deviation of the data sets are calculated to verify that the testing data has a corresponding distribution as the training+validation set.

Standardizing the Data

Standardization is used to scale the training data and the test features, so that each feature will have the same importance in the training of the models. The standardization scales the data so that the mean value is zero (0) and the standard deviation of the data is one (1).

The standardization parameters for both features and targets are saved so they can be used to standardize the test features and also reverse transform the models prediction.

3.2.4 Selecting the Model Architecture

A feed-forward MLP model is used. The model has a input layer, some hidden layers, and a output layer. The fixed architecture of the PV and wind turbine ANNs are presented in Table 3.2.2.

Table 3.2.2: *Models selected architecture*

	Model	
Parameter	**PV1-4 ANN**	**WT ANN**
Network	Feed-forward	Feed-forward
Input layer nodes	2	1
Activation function (input layer)	ReLU	ReLU
Hidden layers	2	2
Hidden nodes	2	2
Activation function (hidden layers)	ReLU	ReLU
Output layer nodes	1	1
Activation function (output layer)	none	none

The number of hidden layers is selected as a starting point for balancing model capability with complexity. If the selected number of hidden layers is not capable of learning the relationship between features and targets well, a increased number of hidden layer should be tested.

The ReLU activation function is used for both the input layers and the hidden layers. The linear behaviour, computational simplicity, and the representational sparsity has made the activation function a default choice for ANNs. No activation function is used in the output layer, as the objective is a regression problem.

The number of nodes in the input layer correspond to the number of input features to the model. The PV ANNs have two input features, i.e the number of nodes in the input layer is two. Likewise, the wind turbine ANN have one input feature and thus one input node. The number of hidden nodes is selected close to the number of input nodes, and as for the number of hidden layers it should be increased if the models are not capable of learning the relationship between the features and targets well. The output layer is fixed to one neuron, as the target for both models is one value.

The model is trained with the backpropagation algorithm, to optimize the neurons weights.

3.2.5 Selecting and Optimizing the Model Structure

To determine the optimum model hyperparameters, a cross-validated grid search methodology is used. The search parameters are set according to Table 3.2.3.

Table 3.2.3: *Models hyperparameters search space*

Hyperparameter	Grid search
Optimizer	SDG, Adam, RMSprop, Adegrad, Adamax, Nadam
Learning rate	0.0001, 0.001, 0.01
Weight initializer	Random Normal, Random Uniform, glorot uniform, normal, uniform, zero
Dropout rate	0.0, 0.1, 0.2, 0.3, 0.4, 0.5
Epocs	10, 20, 30, 40, 50, 60, 70, 80, 90, 100
Batch size	8, 16, 32, 64

The parameters of interest are based on the theory presented in Chapter 2. The grid search algorithm exhaustively considers all parameter combinations, and selects the model that give the highest mean cross-validated score. Dropout layers are included to reduce the likelihood of overfitting.

Due to computer capacity, the grid search has to be divided up into smaller optimization problems; the optimization problem is divided into multiple tests where selected hyperparameters are grid searched and the remaining hyperparameters are fixed. The best result from each test is then combined to a final ANN model.

Table 3.2.4 presents the tests done for each ANN model.

Table 3.2.4: *Hyperparameter search tests*

	Test			
Hyperparameter	**Test 1**	**Test 2**	**Test 3**	**Test 4**
Optimizer	X	Adam	Adam	Adam
Learning rate	X	0.001	0.001	0.001
Weight initializer	glorot uniform	X	glorot uniform	glorot uniform
Dropout rate	0.2	0.2	X	0.2
Epochs	10	10	10	X
Batch size	32	32	32	X

A typical 10-fold cross validation is used.

The best model is saved for later usage, and is used for prediction and evaluation.

3.2.6 Evaluating the Model Performance

The fine-tuned models performance is evaluated using RMSE, MAE, MBE, R, and R^2, to gain knowledge of the models performance on multiple levels.

The selected models are used to predict from both the standardized training+validation and test features. The prediction is inverse transformed before it is compared with the target training-validation and test data.

A Python module is created with functions to calculate all error metrics of the models, so they can be used multiple times.

3.2.7 Pre-Process New Data for Prediction

New data is pre-processed before it is used for prediction. The pre-process for prediction is the same as for the training data, and can be summarized to:

- Clean the data as done with the training, validation and test data
- Scale the data by dividing with the installed capacity of the components
- Scale the data to hourly average timestamps
- Standardize the data with the same parameters as used for training the ML models

3.2.8 Post-Processing the Data From Prediction For the Genetic Algorithm

The results from prediction is post-processed to be ready for the GA. The post-processing of the data can be summarized to:

- Inverse standardize the data with the same parameters as used for training the ML models

- Remove negative prediction values (as it is never assumed that a generator will produce negative power)
- Re-insert the timestamps corresponding to the predicted values
- Fill in missing timestamps and data. This applies to the PV prediction where the nighttime timestamps and values were removed in the pre-process.

3.3 Genetic Algorithm for System Optimization

The genetic algorithm is implemented in Python to optimize the hybrid system. The purpose of the GA is to select the optimum system components to minimize the system cost and to supply the load with sufficient power at all times. The algorithm is based on the *Traveling Salesman Problem* (TSP), which is traditionally used to teach the GA. The TSP is defined as *"Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city once and return to the origin city?"* [9].

For the objective of this thesis, to optimize a hybrid renewable energy system, the problem transfers to: *"Given a list of components with their capacity, what is the system configuration which minimizes the total system cost and is still able to meet the load demand at all times?"*.

The GA is implemented as a step-by-step process as followed:

1. Define the available genes (component configurations) for each component.
2. Create a individual (system) as an alternative solution for the problem.
3. Create the initial population/generation.
4. Rank the individuals based on their fitness.
5. Select the elite (best scoring individuals) through elitism to be carried on to the next generation. Select individuals through fitness proportionate selection for mating.
6. Extract the selected individuals from the population to the mating pool.
7. Use random point crossover to create children with the parents genes. The parents are selected from the mating pool.
8. Mutate the children genes to introduce randomness and avoid the algorithm to converge to a local solution.
9. Create a new generation with the individuals selected through elitism and the newly created children.
10. Iterate through a number of generations in search for the optimum solution.

3.3.1 Define the Components Range and a Individual

Each gene is defined to be a number between 0-10, where each number correspond to an installed capacity of each type of component. The available range of installed capacity for each component is presented in Table 3.3.1.

Table 3.3.1: *Definition of genes and the components range*

Genes range	PV panels range	Wind turbine range	Battery range
Lower limit	0 W	0 W	0 Wh
Upper limit	50 W	500 W	3500 Wh
Resolution	5 W	50 W	350 Wh

The installed capacity of each component is selected by multiplying the gene number with the corresponding component resolution. The components resolution is calculated by Equation 3.3.1, where U is the upper limit, L is the lower limit, and B is the number of available genes (here $B = 10$).

$$R = \frac{U - L}{B} \tag{3.3.1}$$

The upper limit of the battery capacity is calculated from Equation 3.3.2, for being able to supply the load for a three month period with no production from the generators. The three month period is multiplied by a factor of 1.5 to account for the battery minimum DoD which is not available for supplying the load. 94% of the battery storage is assumed to be left after three months due to the battery self discharge rate.

$$3500 \text{ Wh} = 24 \text{ hours} \cdot 30.5 \text{ days} \cdot 3 \text{ months} \cdot 1.5 \cdot \frac{1}{0.94} \tag{3.3.2}$$

The self-discharge rate is commonly given as a *per month*-value in the datasheet. The self-discharge rate should be adjusted to a *per hour*-value, as the timestamps of the data used in ML and the GA are at hourly basis. The calculation of self-discharge rate for each hour is done according to Equation 3.3.3 and 3.3.4, where $Batt_{max}$ is the battery fully charged and $Batt_{SoC}(1month)$ is the state of charge of the battery after one month. The difference between $Batt_{max}$ and $Batt_{SoC}(1month)$ is divided by the total hours of one month. σ is the self-discharge.

$$capacity\ reduction = \frac{Batt_{max} - Batt_{SoC}(1month)}{24 \cdot 30,5} \tag{3.3.3}$$

$$\sigma = \frac{capacity\ reduction}{Batt_{max}} \tag{3.3.4}$$

Table 3.3.2 shows how the range of each component is partitioned on each gene number.

Table 3.3.2: *Definition of genes and the components range in details*

Components range	PV panels range	Wind turbine range	Battery range
0	0 W	0 W	0 Wh
1	5 W	50 W	350 Wh
2	10 W	100 W	700 Wh
3	15 W	150 W	1050 Wh
4	20 W	200 W	1400 Wh
5	25 W	250 W	1750 Wh
6	30 W	300 W	2100 Wh
7	35 W	350 W	2450 Wh
8	40 W	400 W	2800 Wh
9	45 W	450 W	3150 Wh
10	50 W	500 W	3500 Wh

Each individual correspond to a system configuration. A individual has six genes, one for each available system component. Equation 3.3.5 presents how a individual is configured, where G_i is a gene corresponding to an installed capacity of each component (PV no.1, PV no.2, PV no.3, and so on).

$$\text{individual} = [\text{G}_1, \text{G}_2, \text{G}_3, \text{G}_4, \text{G}_5, \text{G}_6] = [\text{PV}_1, \text{PV}_2, \text{PV}_3, \text{PV}_4, \text{WT}, \text{BAT}] \tag{3.3.5}$$

3.3.2 Create the Initial Generation of a Population

An initial population of individuals is created as as starting point for the algorithm. A set number of individuals with randomly selected genes form the initial generation.

3.3.3 Rank the Individuals Based on Their Fitness

All individuals in a generation are evaluated and ranked by their fitness. The fitness function evaluates the individuals ability to supply the electrical load at all times, in addition to rating the individuals based on the total system cost.

The individuals ability to produce sufficient power is determined in multiple steps. The power produced by each component is calculated, based on the installed capacity and expected power production from the ML models. The system production is calculated for each hour within the data time frame, and compared to the load requirements at each hour. If the individual is able to supply the electrical load at all times, the system is ranked based on the total system cost. The fitness score is the inverse of the system cost: a higher cost give a lower fitness score and vise versa. If the individual is not able to supply the load at all times, the

system fitness is set to zero. Equation 3.3.6 explains how the fitness is calculated for each individual.

$$\begin{aligned} if\ \mathrm{E}_{produced}(t) > \mathrm{E}_{load}(t)\ then\ \text{Fitness} = \frac{1}{\text{System cost}} \\ else\ \text{Fitness} = 0 \end{aligned} \tag{3.3.6}$$

Calculation of System Production

The total system production each hour is calculated by Equation 3.3.7, where $P_{PV1-PV4}$ is the power produced by each PV panel, P_{WT} is the power produced by the wind turbine, and E_{BAT} is the available battery supply. dt is set to 1 hour.

$$E_{system} = (P_{PV1} + P_{PV2} + P_{PV3} + P_{PV4} + P_{WT}) \cdot dt + E_{BAT} \tag{3.3.7}$$

The production per hour for the PV panels and the wind turbine is calculated by Equation 3.3.8, where $P_{installed}$ is the installed capacity of the component, ML_{factor} is the results from ML for that component, and n_{factor} is the efficiency of the charge controller corresponding to the component. For the PV panels a MPPT charge controller is used with $n_{charge\ controller} = 0.97$, and for the wind turbine a PWM charge controller with $n_{charge\ controller} = 0.80$ is used.

$$P_{produced}(t) = P_{installed} \cdot ML_{factor}(t) \cdot n_{charge\ controller} \tag{3.3.8}$$

The battery SoC is modelled based on the theory presented in Section 2.5 in Chapter 2, and programmed with a set of conditions.

If the production from the system is higher than the load, and the SoC is below maximum, the battery will charge. The battery will charge until it reaches its maximum or the load becomes larger or equal to the system production. If the battery is already full charged, the maximum SoC will be maintained.

If the production is smaller than the load, and SoC is above the minimum, the battery will discharge while supplying the load. The battery will never discharge below its minimum SoC. At rest, the battery will self discharge.

Calculation of System Cost

The total system cost is calculated by summing the cost of the system components, as described by Equation 3.3.9.

$$System\ cost = \sum Component\ costs \tag{3.3.9}$$

The cost of the PV panels, wind turbine and battery components are calculated according to Equation 3.3.10, where the cost factor is given by Equation 3.3.11.

$$Component\ cost\ (PV/WT/BAT) = Installed\ capacity \cdot Cost\ factor \tag{3.3.10}$$

$$Cost\ factor = \frac{Cost\ of\ component}{Installed\ capacity\ of\ component} \tag{3.3.11}$$

The cost of the charge controllers are included if the component is installed, and is set as a constant cost for the component. If a PWM charge controller is installed, the cost of a dump load is also included as a constant component cost.

3.3.4 Select Individuals by Elitism and for Mating

The individuals with the highest fitness score are selected through elitism to be carried on the the next generation. Fitness proportionate selection is used to assign a probability to the remaining individuals of being selected to the mating pool. The probability of being selected is distributed based on the fitness score. An individual with higher fitness has a higher probability of being selected for mating.

3.3.5 Mate Individuals Through Random Point Crossover

The individuals selected for mating are used as parents for the next generation. Randomly selected parents are mated using random point crossover, to create an offspring (child) with genes from both parents. The crossover point is randomly selected within the number of genes. The first part of genes from *Parent 1* is then combined with the last part genes from *Parent 2*, to create the offspring.

3.3.6 Mutate Individuals for the Next Generation

Each individual created through mating has a chance of mutation. The mutation is done by introducing a novel configuration which will allow the algorithm to explore new parts of the solution space. The mutation is assigned to each gene of the individual, so that each gene has a small probability of mutation. If the gene is selected as a mutant, a randomly selected gene within the allowable range is selected as replacement.

3.3.7 Iterate Through a Number of New Generations for the Solution

The individuals selected through elitism and the offsprings created through mating forms the next generation of the population.

The GA will iterate through a set number of generations in search for the optimum solution for the problem.

3.3.8 Optimize the GA Parameters

A sensitivity analysis is carried out to find the optimal parameter configuration for the GA. The goal of the sensitivity analysis is to obtain the parameters which give the best system solution, in addition to prevent premature convergence.

The sensitivity analysis is done by changing one parameter at time to see how it affect the other parameters and the total configuration. The tests carried out in the sensitivity analysis are presented in Table 3.3.3. A total of *5* main tests were carried out.

- Each main test is executed a total of five times.
- Each of the 5 subtests are executed five times for each of the main test.
- For each of the main tests, n = $[1, 2, 3, 4, 5]$, the population size is given a value constant $Z^\star$. $Z^\star$ = $[Z_1, Z_2, Z_3, Z_4, Z_5]$ = $[10, 20, 50, 100, 200]$
- For each of the subtests, the number of generations is given a constant value $X^\star$. $X^\star$ = $[X_1, X_2, X_3, X_4, X_5]$ = $[10, 20, 50, 100, 200]$. This means that for each subtest the value of population size and elite size is constant, while the number of generations changes.
- Each of the subtests are evaluated to see which give the most stable results.
- The mutation rate of GA can be adjusted if the system converge prematurely.

Table 3.3.3: *Sensitivity analysis of genetic algorithm parameters*

	Test n				
Parameter	**Test 1**	**Test 2**	**Test 3**	**Test 4**	**Test 5**
Population size	$Z^\star$	$Z^\star$	$Z^\star$	$Z^\star$	$Z^\star$
Elite size	5	10	15	20	30
Number of Generations	$X^\star$	$X^\star$	$X^\star$	$X^\star$	$X^\star$

3.3.9 Optimum System Solution From the Genetic Algorithm

To compensate for the randomness of the GA, the multiple tests were performed with the data and the optimized GA parameters. The data used as input to the GA are gained from the ML through prediction based on the weather data available at site. Table 3.3.4 shows the three case scenarios for simulating the system with optimized parameters. The system case scenarios are for *0 month with no production*, *1 month with no production*, and *3 month with no production*.

Table 3.3.4: *Test scenarios explored for the GA*

Test	Scenario
1	0 month with no production
2	1 month with no production
3	3 month with no production

The testing is done for these three scenarios due to see how big the battery needs to be in case of no production. For each of the three cases, ten tests are executed to compensate for the random behaviour of the GA. By running the ten tests the tendency for the GA to stabilize should increase.

Chapter 4

Results and Discussion

This chapter comprise the results obtained from the ANN models, the optimized system configurations and an evaluation of the results. The results of the five ANN models are evaluated using multiple evaluation metrics. The models are evaluated based on the training+validation set and the test set. The results of the predicted data are used in the GA. The optimal parameters of the GA are found by a sensitivity analysis. By using the predicted data and the optimal parameters, the optimal system solution is found for three scenarios studied.

4.1 Input Selection

Feature selection for the ANN models were based on available data on site, prior knowledge of the data, data correlation and mutual information. In the following subsections, the results from feature selection will be presented.

4.1.1 Correlation

The Pearson's correlation coefficient and Spearman rank correlation coefficient reveals if there is a linear or non-linear correlation in the data respectively. Figure 4.1.1 shows the linear Pearson's correlation of the data, and Figure 4.1.2 shows the non-linear Spearman correlation of the data.

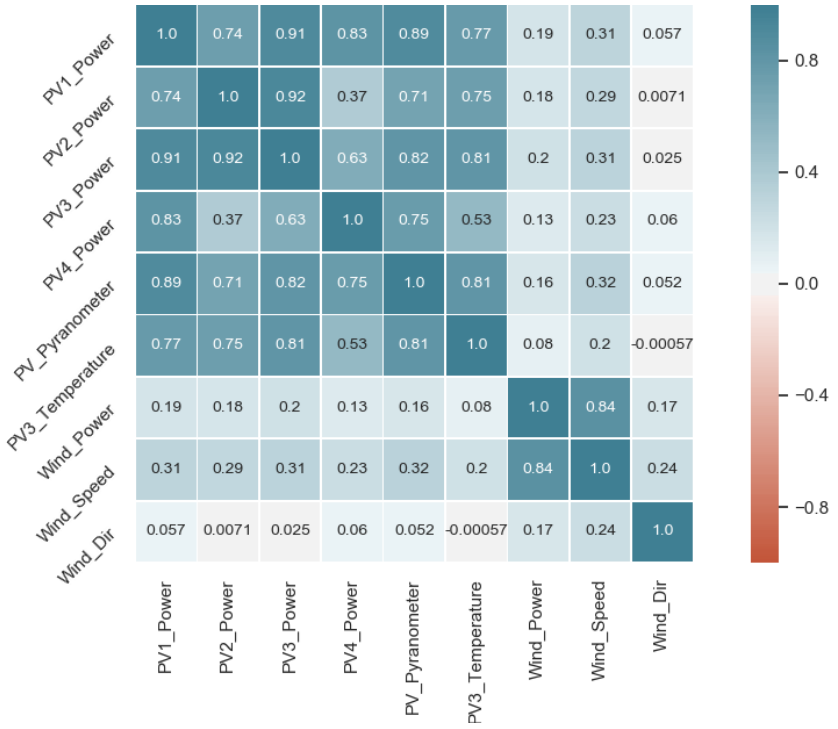


Figure 4.1.1: *Pearson's correlation coefficient of the data*

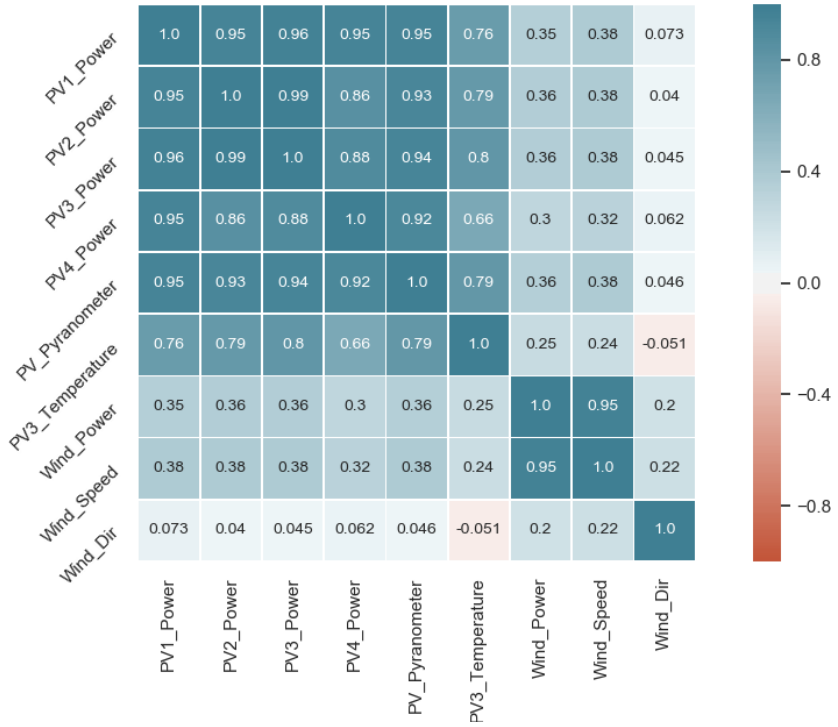


Figure 4.1.2: *Spearman rank correlation of the data*

The power production from all PV panels are highly correlated with the solar irradiance, as the Pearson's correlation coefficients are all above 71% and Spearman rank coefficient are all above 92%. The power production from the PV panels is also closely correlated with the operation temperature of PV panel no.3, with a linear correlation above 75% and non-linear correlation above 76% for PV panel no.1-3. The correlation for fourth PV panel deviate from the others, and only has a linear correlation of 53% and non-linear correlation of 66%. The power production from the PV panels are not highly correlated to the wind speed or wind direction.

The variation in linear and non-linear correlation between the PV panels power production and the solar irradiance may be explained by several factors. The tilt and orientation of the panels affects how much and when the solar irradiance is absorbed by each PV panel. The placement of the panels in comparison to the pyranometer may also affect the correlation. The correlation with the PV panel temperature is of course highest for PV panel no. 3 as the temperature sensor is installed on that panel.

The main factor important for the power produced by a wind turbine is the wind speed. The correlation study proves this theory. The wind power production has a Pearson's correlation of 84% and Spearman correlation of 92% with the wind speed, indicating a high correlation between the data. The wind direction is not highly correlated with the wind power production as the correlation coefficients are both below 20%.

4.1.2 Mutual Information

Mutual information quantifies how much of a variable is carried by another variable. Table 4.1.1 presents the MI between the features and targets.

Table 4.1.1: *Mutual information of the data*

	PV1 Power	**PV2 Power**	**PV3 Power**	**PV4 Power**	**Wind Power**
PV Pyranometer	1.6672	1.4096	1.4123	1.4057	0.0774
PV3 Temperature	0.5061	0.5575	0.6097	0.3466	0.0644
Wind Speed	0.1096	0.1140	0.1165	0.0963	1.6351
Wind Dir	0.0961	0.1057	0.1057	0.0896	0.1674

The power production from the PV panels has a higher correlation with the solar irradiance than the temperature of PV panel no.3, wind speed and wind direction. The correlation with the temperature of PV panel no.3 is higher for PV panel no.3 than for the rest of the PV panels. PV panel no.4 has the lowest correlation with the temperature.

The wind power has the highest correlation with the wind speed. The correlation between power production and the wind direction is low compared to the wind speed. MI also indicate a low correlation between the wind power and the solar irradiance and the operational temperature of the PV panel.

4.1.3 Final Feature Selection

Table 4.1.2 presents the final feature selection for the ANN models.

Table 4.1.2: *Final features selected*

Model	Features	Target
PV1 ANN	PV Pyranometer, PV3 Temperature	PV1 Power
PV2 ANN	PV Pyranometer, PV3 Temperature	PV2 Power
PV3 ANN	PV Pyranometer, PV3 Temperature	PV3 Power
PV4 ANN	PV Pyranometer, PV3 Temperature	PV4 Power
Wind Turbine ANN	Wind Speed	Wind Power

The correlation and MI between solar irradiance (PV Pyranometer), operation temperature of PV panel no.3, and power production of all PV panels, indicate that the irradiance and temperature should be included in the input features of all PV ANN models. The other available inputs show little coherence with the target, and are therefore left out. Figure 4.1.3 and 4.1.4 shows the cleaned input features for the PV ANN models.

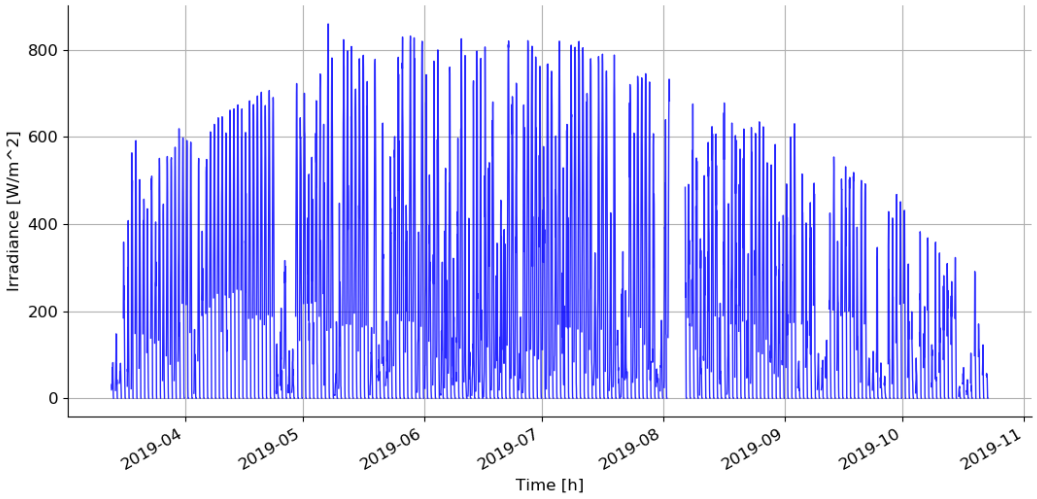


Figure 4.1.3: *Input feature for ML: Solar irradiance*

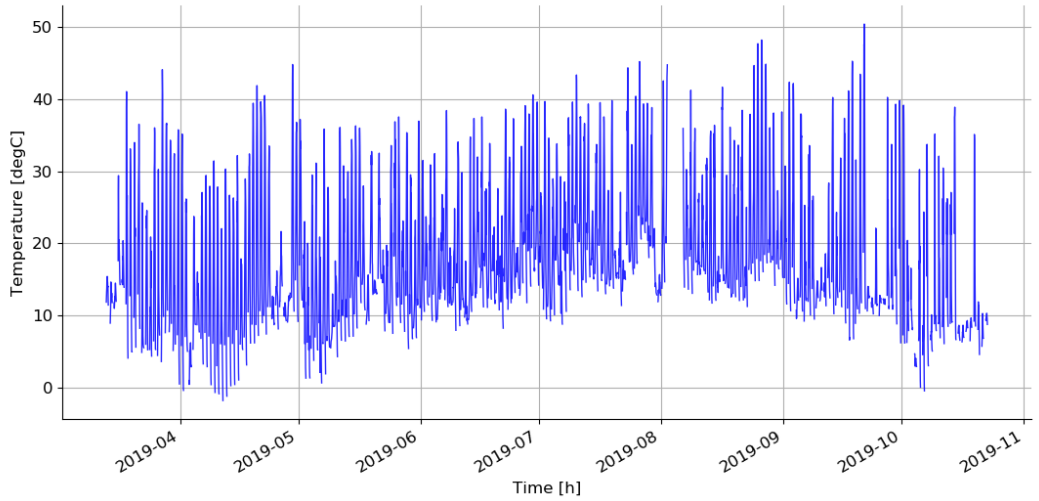


Figure 4.1.4: *Input feature for ML: Operational temperature of PV panel no.3*

The wind speed is the only available input feature that show a significant correlation and MI with the wind turbine power production. The wind speed is therefore selected as the only input feature for the wind ANN model. Figure 4.1.5 shows the cleaned feature used for training and validation of the wind turbine ANN model.

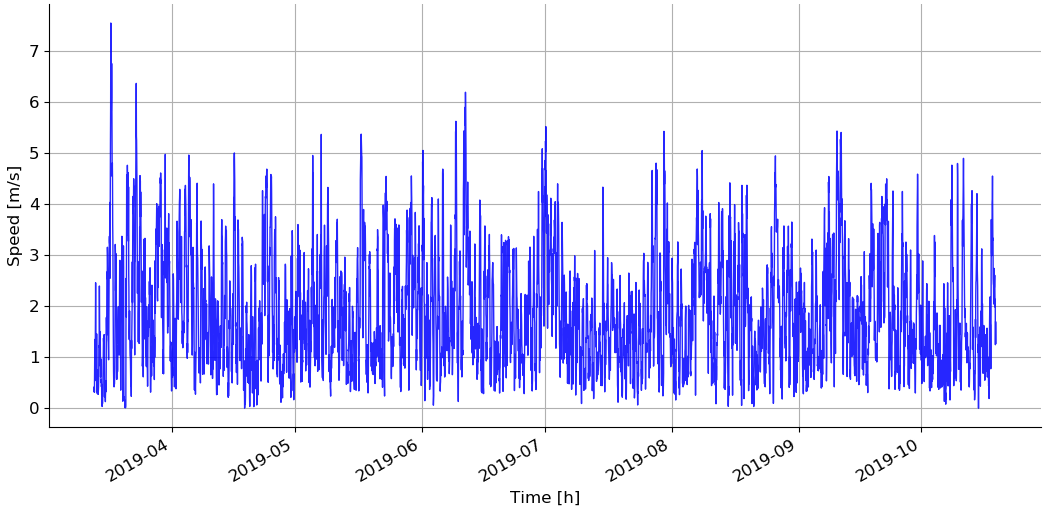


Figure 4.1.5: *Input feature for ML: Wind speed*

4.2 Data Division

The data was divided so that 80% of the data was used for training and validation, and 20% of the data was held back for testing. To verify that the models were trained and tested on data with approximately the same distribution, the mean and standard deviation of the data sets were calculated. The statistics of the data division is shown in Table 4.2.1.

Table 4.2.1: *Statistics of data division*

	Model							
	PV1-4 ANN				**WT ANN**			
	Training/Validation set		**Test set**		**Training/Validation set**		**Test set**	
Data	**Mean**	**Std**	**Mean**	**Std**	**Mean**	**Std**	**Mean**	**Std**
PV Pyranometer	240.38	233.72	235.33	232.19				
PV3 Temperature	19.04	9.14	18.43	9.42				
Wind Speed					1.89	1.13	1.89	1.14

The results show a little variance in the datasets for the PV ANN models, and very little variance in the datasets used for training, validation and testing the wind turbine ANN.

Figure 4.2.1 shows how the PV Pyranometer data is divided for training/validation and testing. The concept transfers to the other models and features.

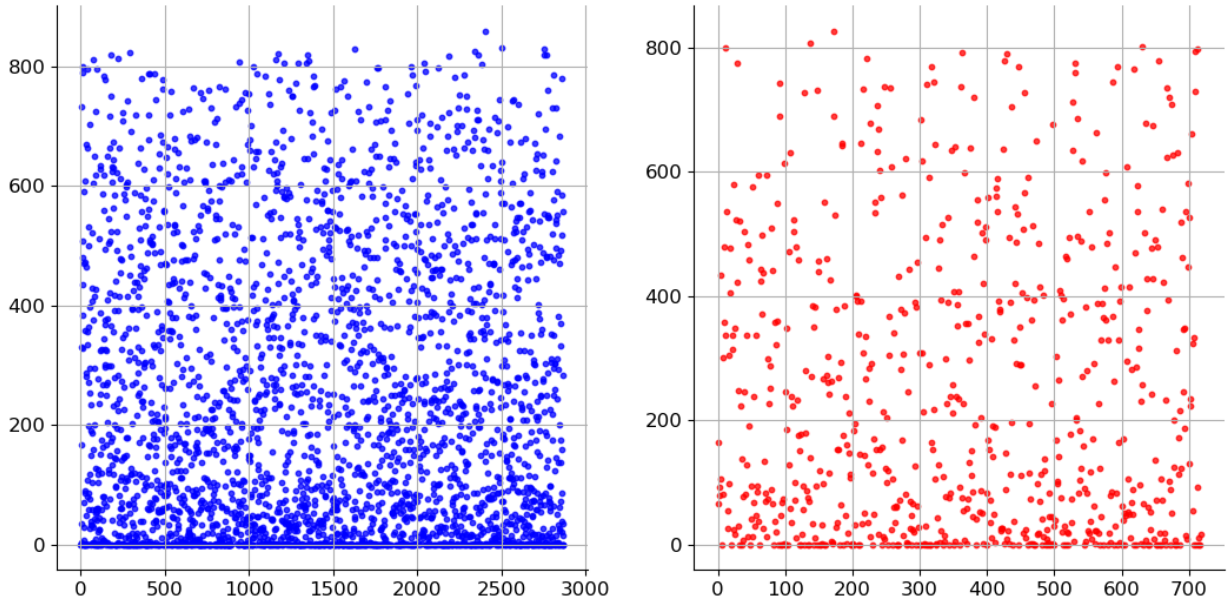


Figure 4.2.1: *Data division, Left: Training and validation data, Right: Test data*

4.3 Model Structures

The model architecture and hyperparameters were selected based on the theory presented in Chapter 2 and through a grid search methodology. Due to computer capability, the grid search was divided into multiple tests. Each model was trained with several different configurations for each test and evaluated using k-fold cross-validation. The optimum hyperparameters from each test was combined to the final model. The selected model architectures and hyperparameters are presented in Table 4.3.1.

Table 4.3.1: *ANN model structures*

	Model				
Hyperparameters	**PV1 ANN**	**PV2 ANN**	**PV3 ANN**	**PV4 ANN**	**WT ANN**
Optimizer*	Nadam	SGD	SGD	Nadam	Nadam
Learning rate*	0.0001	0.0001	0.01	0.001	0.01
Input layer nodes	2	2	2	2	1
Input layer act.func.	ReLU	ReLU	ReLU	ReLU	ReLU
Hidden layers	2	2	2	2	2
Hidden nodes	2	2	2	2	2
Hidden layer act.func.	ReLU	ReLU	ReLU	ReLU	ReLU
Output layer nodes	1	1	1	1	1
Output layer act.func.	none	none	none	none	none
Weight initializer	uniform	RandomUniform	glorot uniform	glorot uniform	RandomNormal
Dropout rate*	0	0	0	0.1	0
Epochs*	80	100	100	60	40
Batch size*	32	32	16	32	8

* Hyperparameter selected through grid search

Nadam and SGD proves to give good results for both the PV and wind turbine ANNs.

The learning rate controls how much the model weights are changed for each epoch due to the evaluated error. The learning rate varies between the models, but has a tendency to a slower learning. The lower learning rate is likely to provide a more stable learning, but will also slow down the learning process. However, for the amount of data feed to the models in this project, the learning speed will likely not be a restricting factor.

Different types of uniform weight initalizers dominates the results. A normal distribution also proves to give better results than initializing the weights with zeros.

For most models, the relationship between features and targets are best learned without adding dropouts.

The number of epochs varies between 40 and 100, with a batch size between 8 and 32. In some cases, the number of epochs approaches the maximum available configuration, which could mean that the optimum epoch number lies beyond this limit.

A batch number of 32 is often used as a fixed value for training ANNs, and as seen in the table a batch size of 32 dominates the model structures.

4.4 Model Evaluations

The performance of the models on the training+validation and test set were evaluated using multiple evaluation metrics to gain overall knowledge of the models performances. The performance was studied for the prediction inverse transformed and compared to the cleaned raw data. The following subsections presents the evaluation of each ANN model.

4.4.1 PV1 ANN

Table 4.4.1 presents the performance of the ANN for PV panel no.1.

Table 4.4.1: *ANN model evaluation on training+validation and test set for PV1 ANN*

PV1 ANN		
Performance	**Training/Validation set**	**Test set**
RMSE	0.1296	0.1291
MAE	0.0737	0.0719
MBE	0.0032	0.0021
R	0.9061	0.9032
R^2	0.8206	0.8156

On the test set the model gave a R of 0.9032 and R^2 of 0.8156 which indicate a reasonable good correlation between the predicted power and the target values. The RMSE of 0.1291 is in the higher range compared to the other models studied. MAE imply an average error of 0.0719, and MBE indicate that on average the model tends to underpredict. This means that the predicted values are in general a bit lower than the target values.

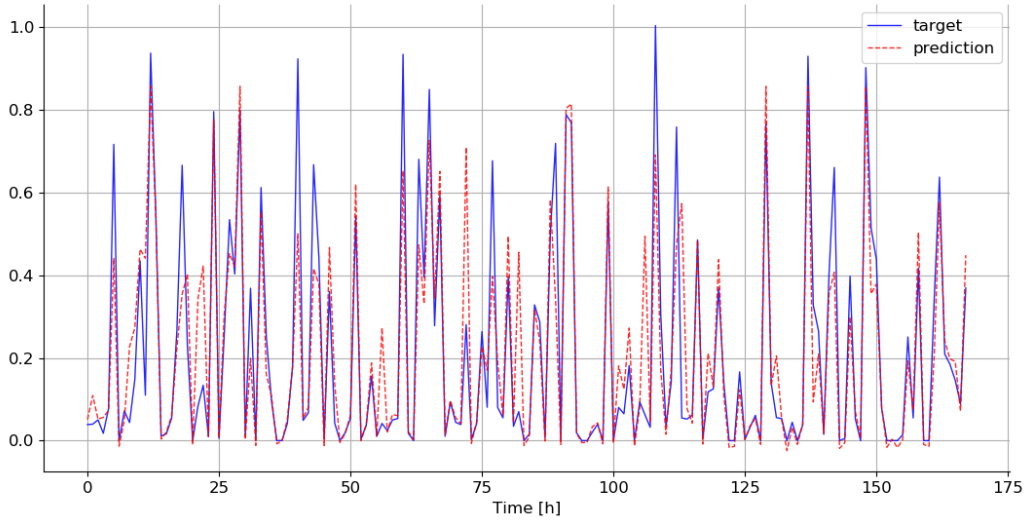


Figure 4.4.1: *Prediction versus target on 1 random week of test data for PV1 ANN model*

The same indications can be seen in Figure 4.4.1, where one random week of prediction and target are shown. The prediction follow the trend of the target, but is rarely able to peak as high. Some miscalculations can also be seen in the prediction, where the target values are low and the model predict a much higher value.

Figure 4.4.2 and 4.4.3 shows the prediction versus target for the PV no.1 ANN model on the training+validation and test set respectively. The model behaves correspondingly on the training+validation set and the test set, with little deviation in the performance measures.

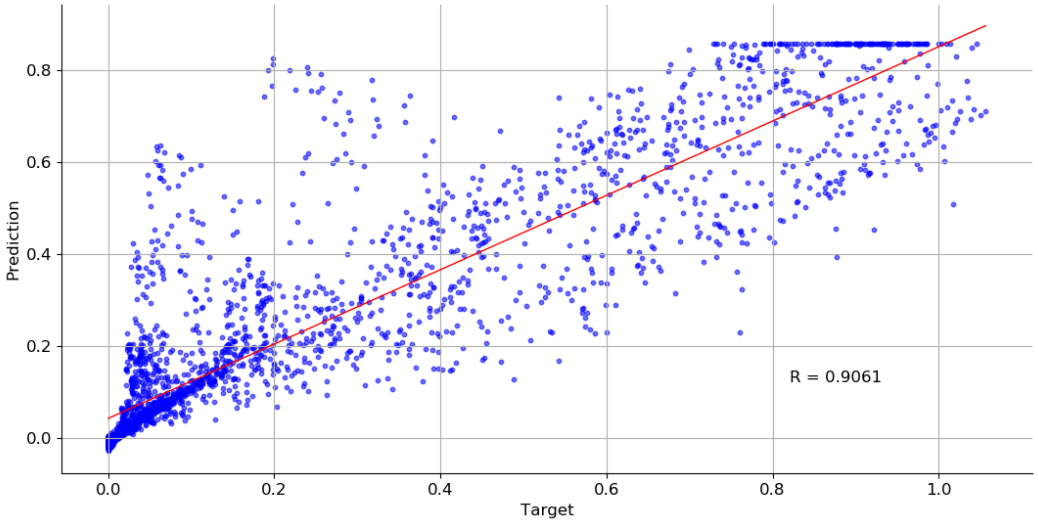


Figure 4.4.2: *Prediction versus target on training+validation data for PV1 ANN model*

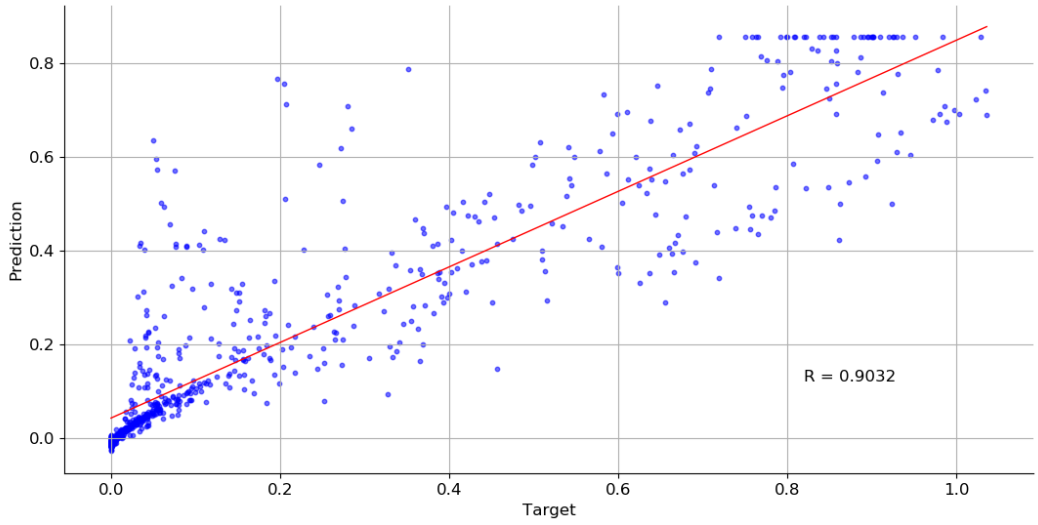


Figure 4.4.3: *Prediction versus target on test data for PV1 ANN model*

4.4.2 PV2 ANN

Table 4.4.2 presents the performance of the ANN for PV panel no.2.

Table 4.4.2: *ANN model evaluation on training+validation and test set for PV2 ANN*

PV2 ANN		
Performance	**Training/Validation set**	**Test set**
RMSE	0.1305	0.1203
MAE	0.0708	0.0640
MBE	0.0050	0.0053
R	0.8005	0.8196
R^2	0.6401	0.6710

The RMSE of 0.1203 on the test set indicate that the model has a good prediction ability, while the MAE imply an average error of 0.0640 in the prediction. The model has a small tendency to underpredict, implied by the MBE of 0.0053. The correlation factor between the prediction and target is at 0.8196 on the test set, a bit higher than on the training+validation data. R^2 indicate that 67% of the prediction data can be explained by the target data.

Figure 4.4.4 show the prediction versus targets on the test set for a random week. As indicated by the performance evaluation, the prediction follow the trend of the target data. In most cases, the prediction is unable to peak as high as the target values. The prediction also show a tendency to predict a higher production when the target has a lower peak.

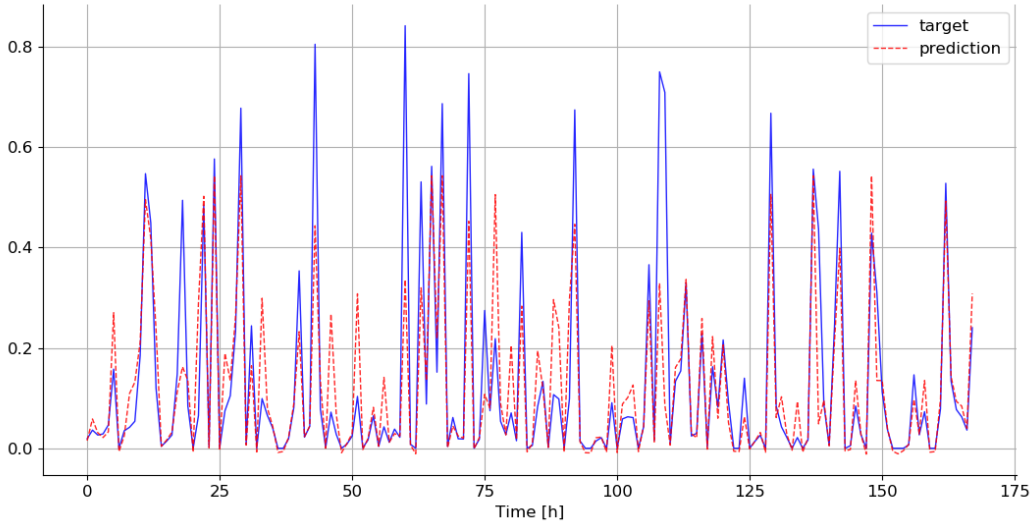


Figure 4.4.4: *Prediction versus target on 1 random week of test data for PV2 ANN model*

The prediction versus target for the ANN model of PV no.2 for the training+validation and test set is shown in Figure 4.4.5 and 4.4.6 respectively.

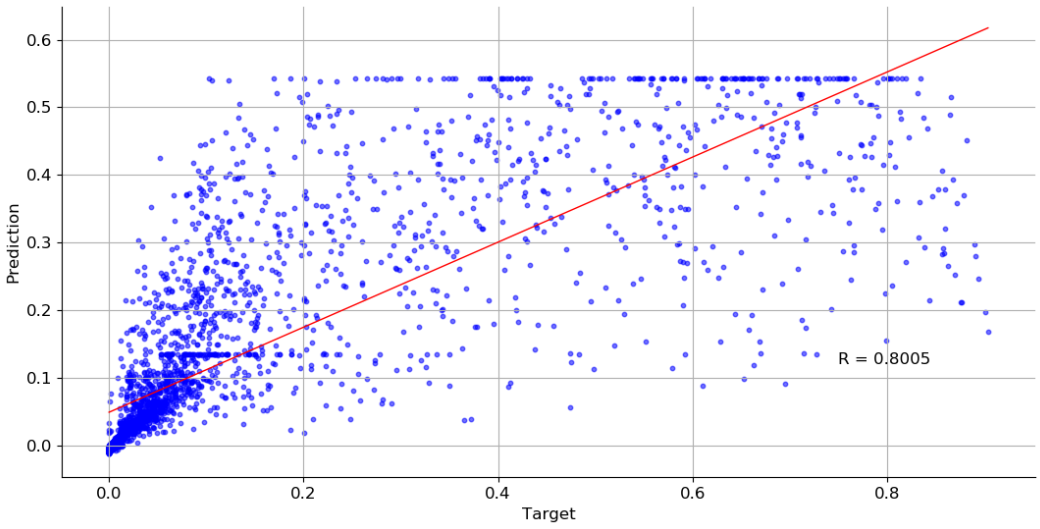


Figure 4.4.5: *Prediction versus target on training+validation data for PV2 ANN model*

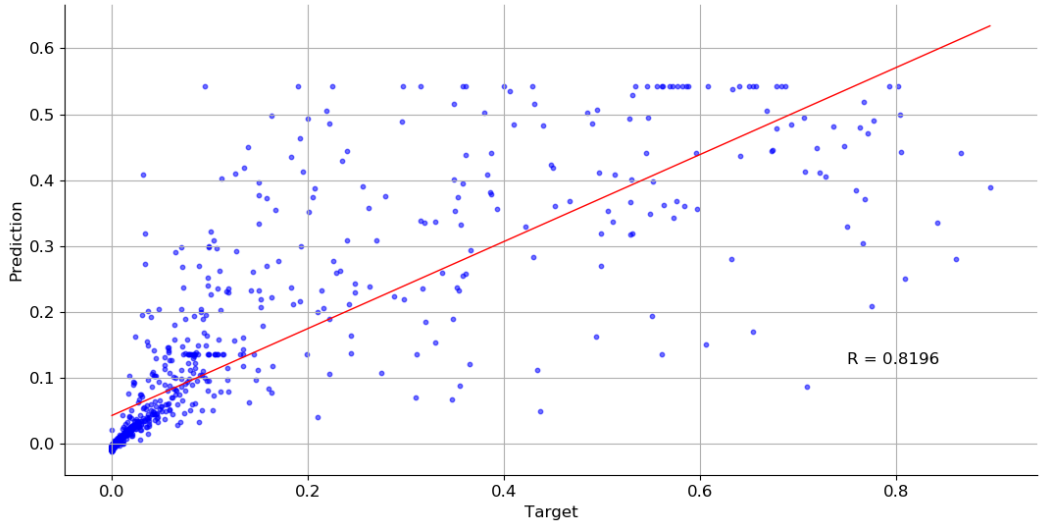


Figure 4.4.6: *Prediction versus target on test data for PV2 ANN model*

The same tendency is seen in the training+validation set and the test set. There is only a little deviation in the performance measures, with the highest performance in the test data.

4.4.3 PV3 ANN

Table 4.4.3 presents the performance of the ANN model for PV panel no.3 on the training+validation set and test set.

Table 4.4.3: *ANN model evaluation on training+validation and test set for PV3 ANN*

PV3 ANN		
Performance	**Training/Validation set**	**Test set**
RMSE	0.0983	0.0927
MAE	0.0579	0.0517
MBE	0.0012	0.0021
R	0.8827	0.8924
R^2	0.7786	0.7954

The model appear to have a reasonable good ability to predict. The correlation between prediction an targets are measured to 0.8924, and the R^2 to 0.7954 on the test set. This indicate a high correlation between the data. The RMSE is 0.0983 on the test set, and MAE indicate a average error between the prediction and target values of 0.0517. The MBE of 0.0021 show that the model has a small tendency to on average predict below the target values.

The tendency of the model performance can also be seen in Figure 4.4.7, which shows the prediction versus target for one random week of data.

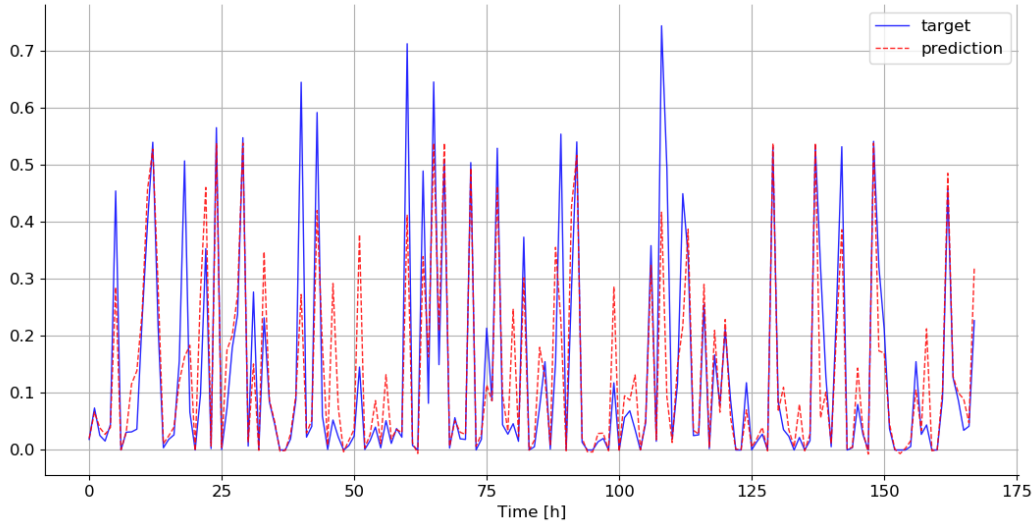


Figure 4.4.7: *Prediction versus target on 1 week of test data for PV3 ANN model*

The model show a capability to predict the trend of the target data. However, as for the previous discussed models, the prediction is not able to peak as high as the target in many cases. The tendency to overpredict where the target peaks are lower is also clear.

The prediction versus target for the training+validation set and the test set is shown in Figure 4.4.8 and 4.4.9 respectively.

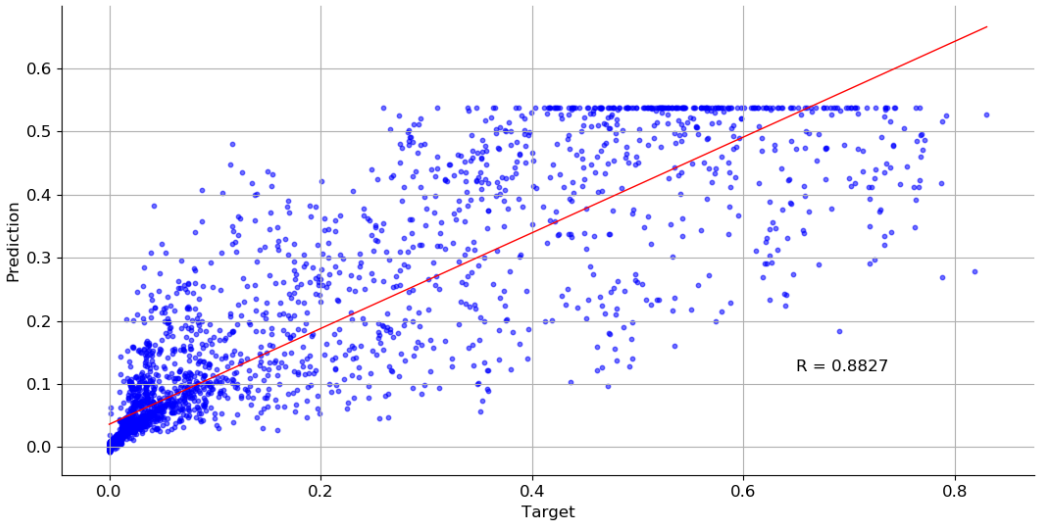


Figure 4.4.8: *Prediction versus target on training+validation data for PV3 ANN model*

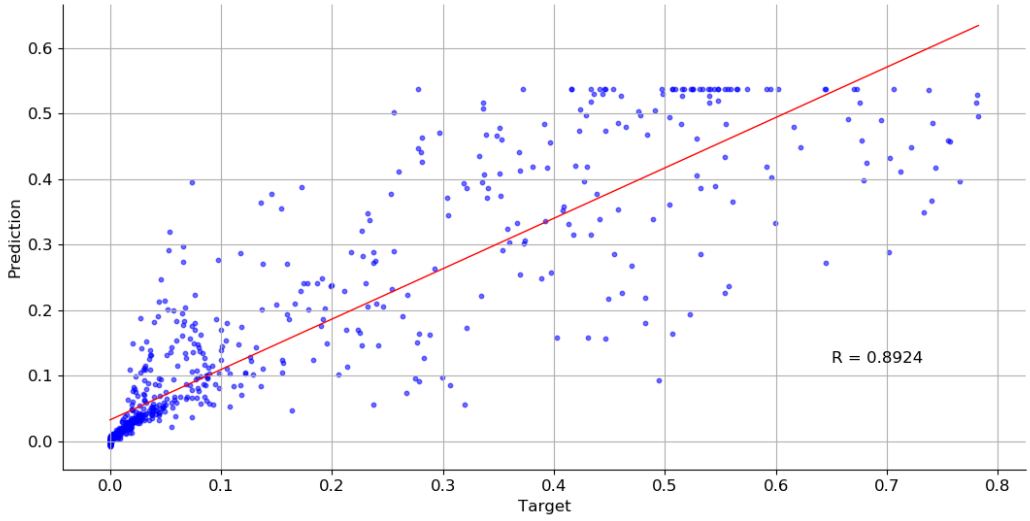


Figure 4.4.9: *Prediction versus target on test data for PV3 ANN model*

The results are similar between the sets of data. The model does however show a better performance on the test set.

4.4.4 PV4 ANN

The model performance on the training+validation and test set for the ANN model of PV panel no.4 is presented in Table 4.4.4.

Table 4.4.4: *ANN model evaluation on training+validation and test set for PV4 ANN*

PV4 ANN		
Performance	**Training/Validation set**	**Test set**
RMSE	0.1521	0.1526
MAE	0.1031	0.1014
MBE	0.0068	0.0074
R	0.7720	0.7807
R^2	0.5579	0.5651

The RMSE of 0.1526 and MAE of 0.1014 on the test set is the highest for all models. The measure of difference between the prediction and target values is thus the highest for this model compared to the other models. MBE indicate a average error between the prediction and target of 0.0074. The correlation factor of 0.7807 indicate that the model is capable of predicting the pattern in the target data. R^2 do however indicate that only 56% of the prediction can be explained by the targets, which is low compared to the other models.

Figure 4.4.10 show the prediction versus target for a random week of test data. It is clear how the model perform worse than the other PV models. The prediction follows the trend of the target data but is both unable to peak corresponding to the target or drop as low as the target. The result is in general a prediction lower than the target tops and higher than the target lows.

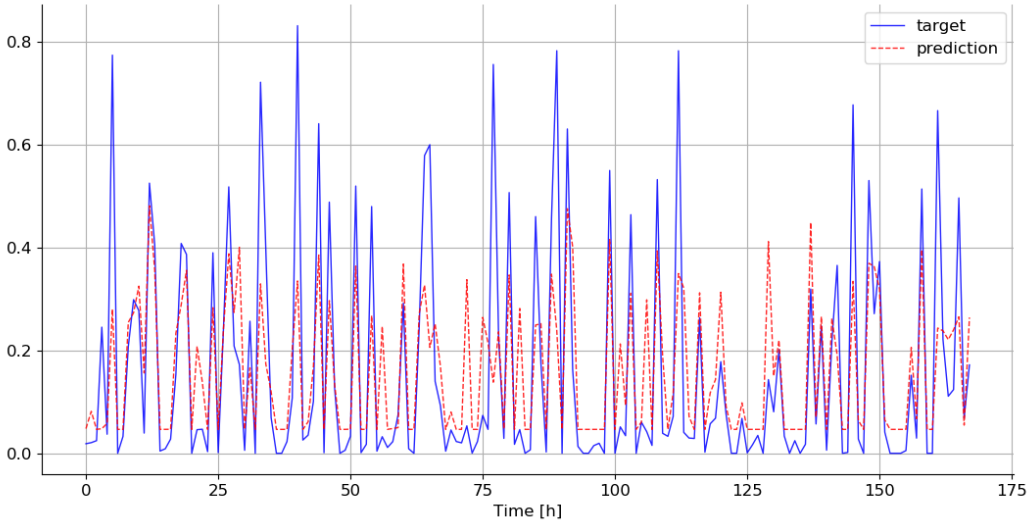


Figure 4.4.10: *Prediction versus target on 1 week of test data for PV4 ANN model*

The prediction versus target for the training+validation set and test set for is presented in Figure 4.4.11 and 4.4.12 respectively.

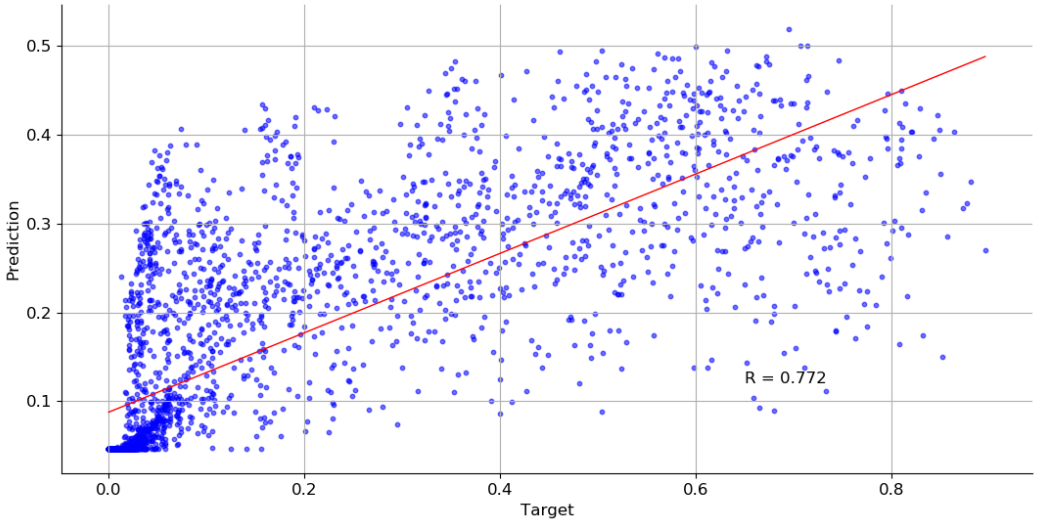


Figure 4.4.11: *Prediction versus target on training+validation data for PV4 ANN model*

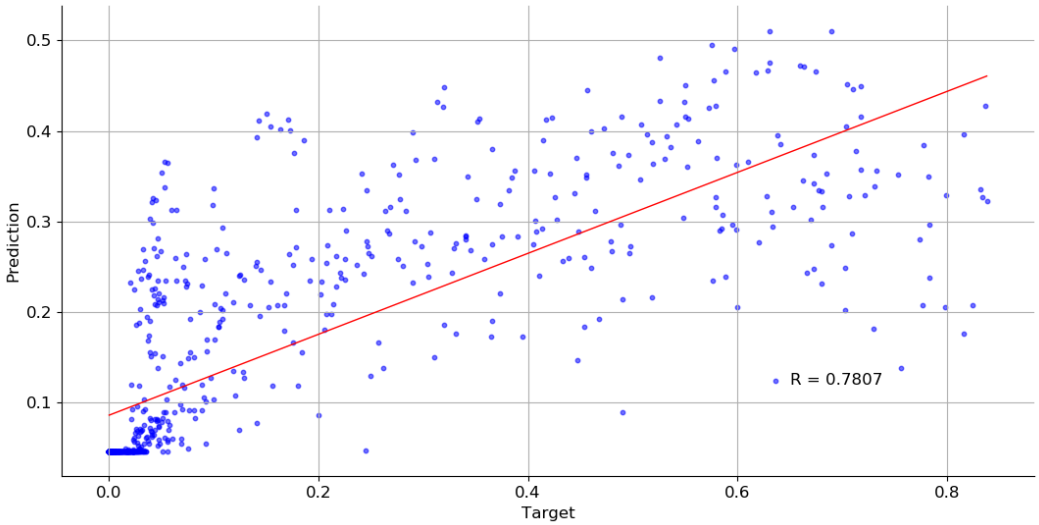


Figure 4.4.12: *Prediction versus target on test data for PV4 ANN model*

The results indicate a corresponding performance in the two data sets, however the model performance is the best for the test set.

4.4.5 WT ANN

The performance of the ANN for the wind turbine on the training+validation set and test set is presented in Table 4.4.5.

Table 4.4.5: *ANN model evaluation on training+validation and test set for WT ANN*

WT ANN		
Performance	**Training/Validation set**	**Test set**
RMSE	0.0031	0.0030
MAE	0.0019	0.0019
MBE	-0.0004	-0.0003
R	0.9805	0.9826
R^2	0.9600	0.9649

The performance results presented in the table indicate that the model has a good ability to predict. The RMSE is 0.0030 on the test set, and the MAE indicate that the average error is 0.0019. The MBE of -0.0003 imply that the model has a small tendency to overpredict. This means that the prediction is on average a bit higher than the target values. The R of 0.9826 show a high correlation between the prediction and target data. The prediction data can also be well explained by the target data, which is implied by the R^2 of 0.9649.

The performance of the model can also be viewed in Figure 4.4.13, which shows the prediction versus target for one random week of test data.

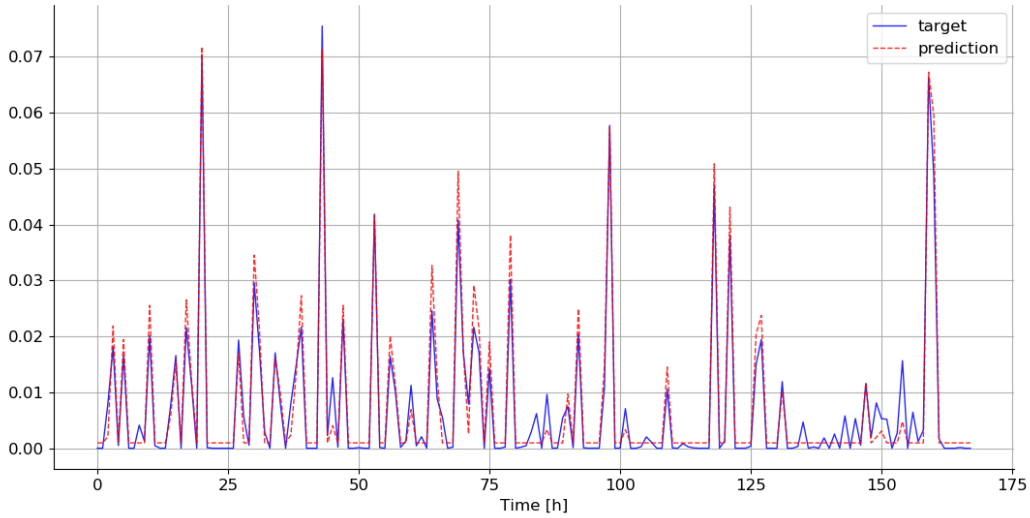


Figure 4.4.13: *Prediction versus target on 1 week of test data for WT ANN model*

The model predicts close to the target, but peaks a bit higher than the target values. The model tends however to not be able to predict the small peaks in the target data. The prediction tends to overall be a bit

higher than the target.

Figure 4.4.14 and 4.4.15 show the prediction versus target on the training+validation and test set respectively.

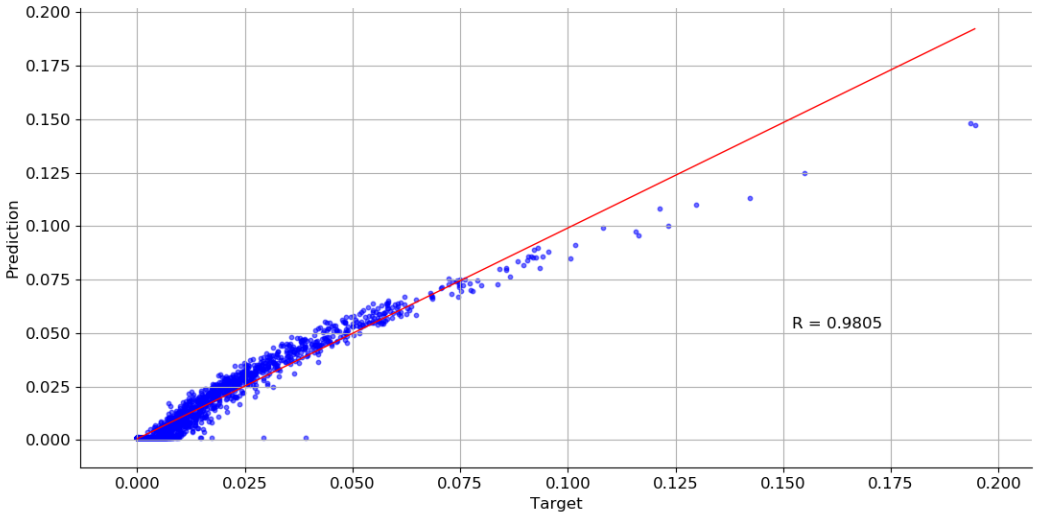


Figure 4.4.14: *Prediction versus target on training+validation data for WT ANN model*

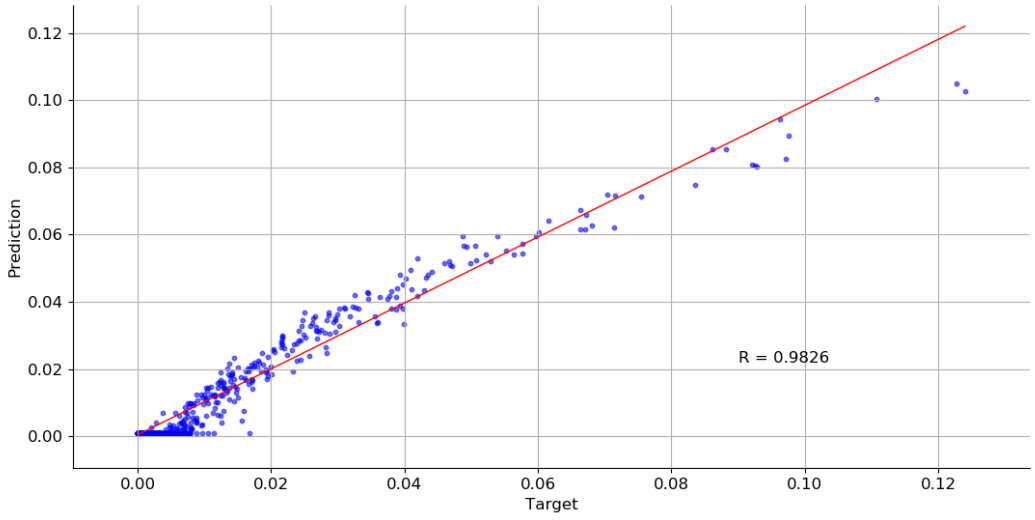


Figure 4.4.15: *Prediction versus target on test data for WT ANN model*

The same tendency can be seen in both the data sets. The model performs similar on both sets of data, with the test set proving a little increase in the performance.

4.5 Prediction on "New" Data

The ANN models ability to predict were tested on all the data available by combining all the data to one set, cleaning and scaling the data with the written functions, and then use the data for prediction. The performance of the models on the "new" data is presented in Table 4.5.1.

Table 4.5.1: *ANN model evaluation on "new" data*

	Model				
Performance	**PV1 ANN**	**PV2 ANN**	**PV3 ANN**	**PV4 ANN**	**WT ANN**
RMSE	0.4231	0.5484	0.4862	0.5119	0.1980
MAE	0.2397	0.3732	0.3268	0.3884	0.1223
MBE	0.0097	-0.2793	-0.2655	-0.1918	-0.0259
R	0.9055	0.8042	0.8846	0.7737	0.9809
R^2	0.8197	0.3960	0.4890	0.5331	0.9610

The ANN for PV panel no.1 performs well according to the R and R^2. The RMSE, MAE and MBE is however higher for all the data than for the test data alone. For the ANN for PV panel no.2 the correlation between the prediction and target data for all the data is close to the correlation in the test data only. The R^2 is however even lower for all the data, and the RMSE and MAE has increased. The MBE implies a overprediction. The same changes applies to the ANN for PV panel no.3 and PV panel no.4.

Figure 4.5.1-4.5.4 show the prediction versus target on the "new" data for the PV models.

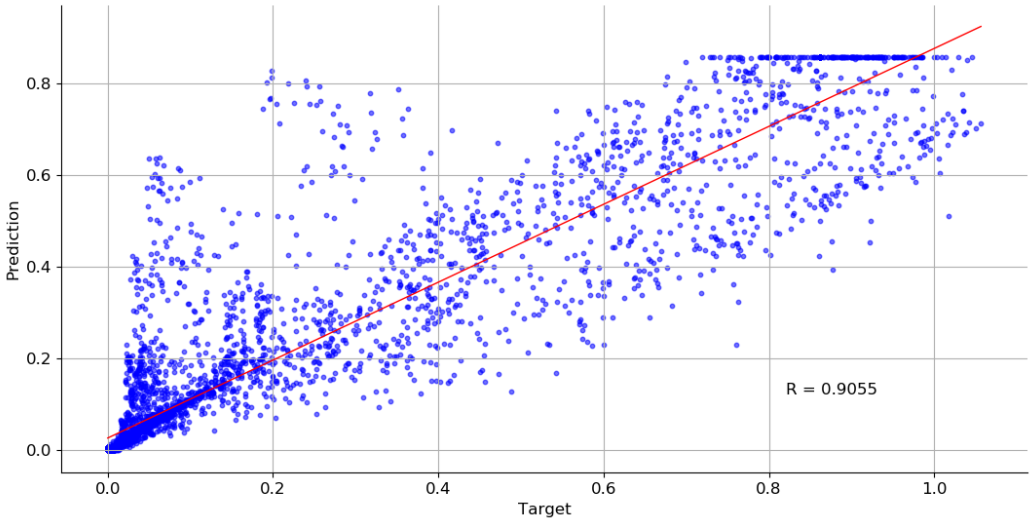


Figure 4.5.1: *Prediction vs target on "new" data for PV1*

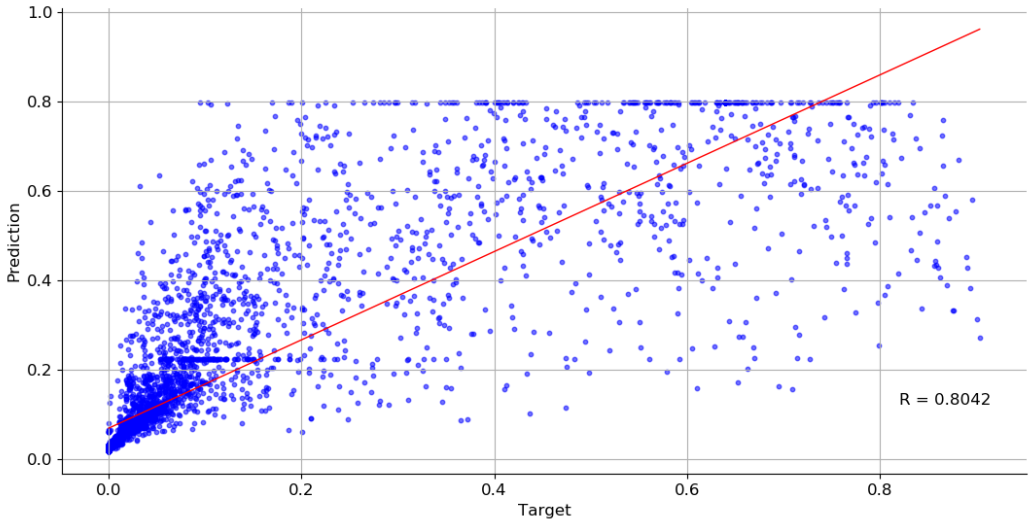


Figure 4.5.2: *Prediction vs target on "new" data for PV2*

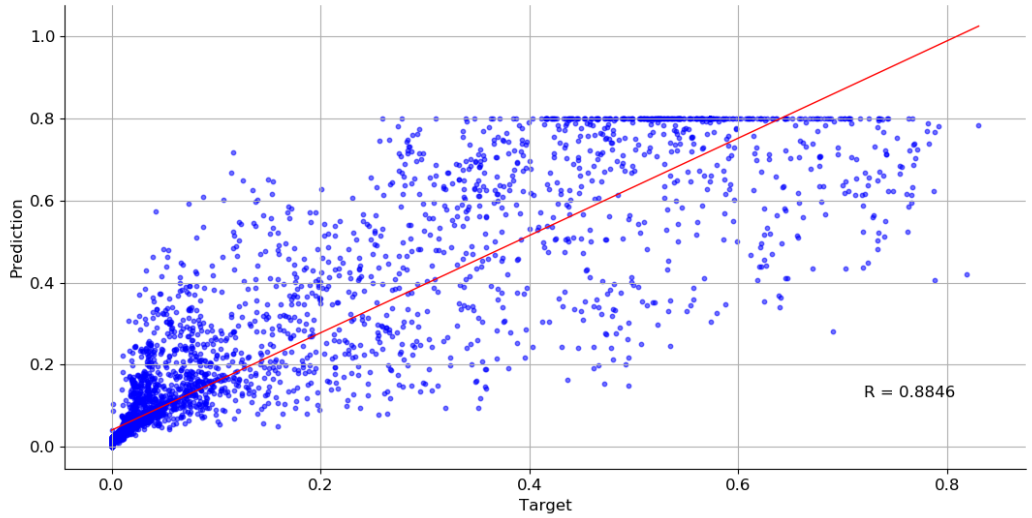


Figure 4.5.3: *Prediction vs target on "new" data for PV3*

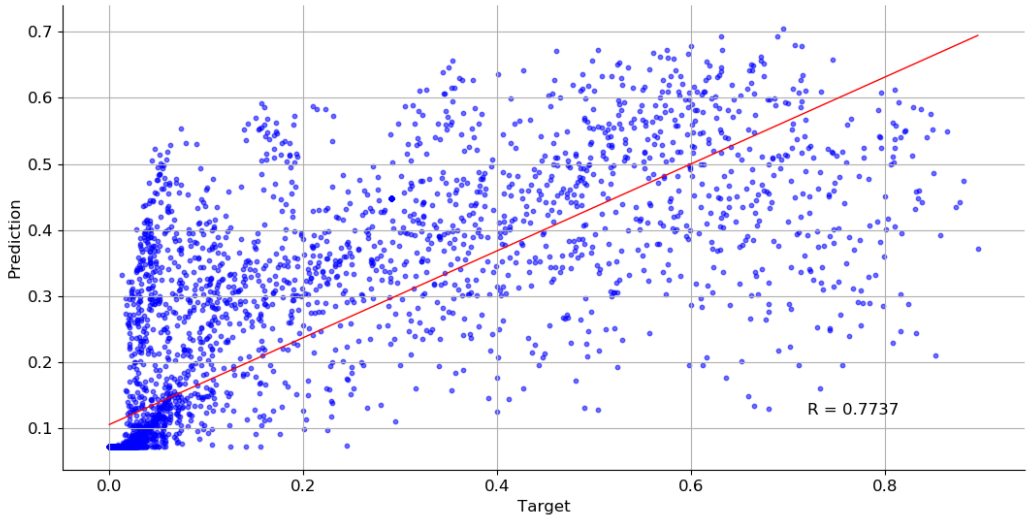


Figure 4.5.4: *Prediction vs target on "new" data for PV4*

The ANN for the wind turbine still performs well. This is implied by the R and R^2 which is also close to the performance on the test set alone. The RMSE and MAE has increased, and the MBE implies a higher overprediction than for the test set. Figure 4.5.5 show the prediction versus target on the "new" data for the wind turbine model.

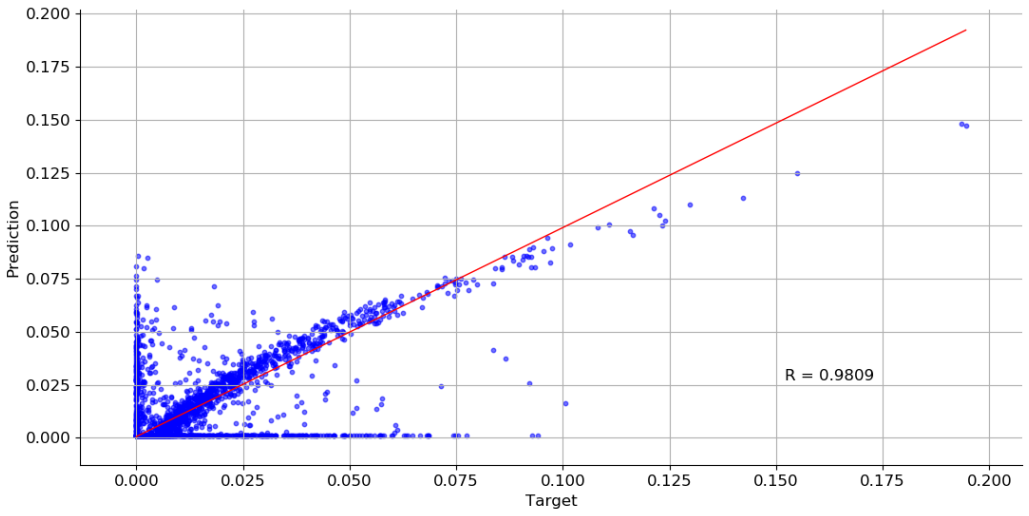


Figure 4.5.5: *Prediction vs target on "new" data for WT*

A performance close to the performance on the training+validation and test set is expected as the models have already seen and learned from most of the data. The varying results may be a result of randomness in the prediction both for the training, test and "new" data.

4.6 Genetic Algorithm Parameters

The sensitivity analysis was performed on the population size, elite size, and number of generations, and the best combination of these three was further investigated. The mutation rate was changed for the best combination of population size, elite size and number of generations. This was done to see if the combination could be improved even further. The optimal parameters found by the sensitivity analysis are presented in Table 4.6.1.

Table 4.6.1: *Final parameters for the Genetic Algorithm*

Parameter	Configuration
Population size	50
Elite size	10
Mutation rate	0.15
Number of generations	30

The performance of the sensitivity analysis shows that the GA model has a tendency to not stabilize the results of the objective function if the population is small (pop_{size} < 10). If the number of generations is small, in addition to a small population, the results will not stabilize. When increasing the population and keeping the number of generations below 50, the system will stabilize, have a faster convergence, while avoiding premature convergence. An increased number of generations increase the possibility of premature convergence for the system. In addition, the increased number of generations slows the convergence time of the GA. The best parameter for the elite size should be a small value compared to the population size. When the elite size increase, the possibility of a higher cost of the objective function increase. The mutation rate is a small value, but if it is to small the tendency of premature convergence increase. If the mutation rate is to high, the system loose the ability to stabilize and the total cost of the system increases.

4.7 Optimum System Solution From the Genetic Algorithm

The objective function is to minimize the cost, within the constraint to always meet the load demand. If the result of the objective function gets better for each of the generations, then the fitness value will increase.

The optimum system solution from the genetic algorithm is simulated for three different scenarios. The first simulation is from the power prediction results and there is no constraint on the power production. The results obtained from the first simulation is presented in Section 4.7.1. In the second simulation, one month of power from the power prediction is removed. The results obtained from the second simulation is presented in Section 4.7.2. The third simulation is for three months without power production in the power prediction results. Section 4.7.3 presents the results from the third simulation.

4.7.1 Optimum System Solution for Case 1

The results from the first simulation shows that from the predicted power used in the genetic algorithm, the optimum system solution consists of PV panels. This system solution is able to meet the load demand of 0.75 W due to power production from the PV panels. The cost of this system solution is low, and the fitness is high. From the ten tests there are low variation within the results. The fitness from the ten tests is either 0.00063 or 0.00068, and the cost is either 1600.0 NOK or 1475.0 NOK. Table 4.7.1 presents the results from Case 1.

Table 4.7.1: *Genetic algorithm test results with 0 months of no production*

	Installed capacity						**Results**				
Test	**PV1**	**PV2**	**PV3**	**PV4**	**WT**	**BAT**	**System production**			**Fitness**	**Cost**
	[W]	[W]	[W]	[W]	[W]	[Wh]	Min	Mean	Max		NOK
Test 1	0.0	5.0	0.0	10	0.0	0.0	0.79	2.63	10.22	0.00068	1475.0
Test 2	0.0	0.0	0.0	15.0	0.0	0.0	1.06	2.68	10.25	0.00068	1475.0
Test 3	5.0	5.0	0.0	10.0	0.0	0.0	0.8	3.48	14.38	0.00063	1600.0
Test 4	0.0	5.0	0.0	10.0	0.0	0.0	0.79	2.63	10.22	0.00068	1475.0
Test 5	0.0	0.0	5.0	15.0	0.0	0.0	1.06	3.52	13.74	0.00063	1600.0
Test 6	0.0	0.0	0.0	15.0	0.0	0.0	1.06	2.68	10.25	0.00068	1475.0
Test 7	0.0	0.0	0.0	15.0	0.0	0.0	1.06	2.68	10.25	0.00068	1475.0
Test 8	0.0	0.0	5.0	15.0	0.0	0.0	1.06	3.52	13.74	0.00063	1600.0
Test 9	0.0	5.0	0.0	10.0	0.0	0.0	0.79	2.63	10.22	0.00068	1475.0
Test 10	0.0	5.0	0.0	10.0	0.0	0.0	0.79	2.63	10.22	0.00068	1475.0

Figure 4.7.1 shows the overall system production for the optimal system solution from *Test 1*.

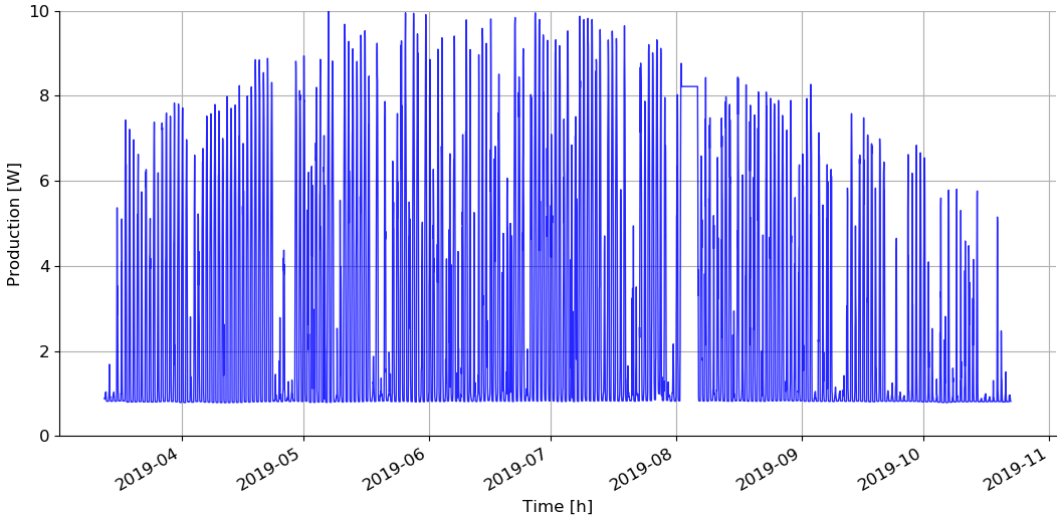


Figure 4.7.1: *System production for no month without production*

Test 1 is chosen since this is an optimal result obtained by the ten test. The fitness and cost for this test is 0.00068 for fitness and 1475.0 NOK for cost. From Figure 4.7.1, the overall produced power varies between 0-10 W in the time period of the available data from the power prediction. The maximum power produced by PV panels will naturally occur in the summer months. The results proves the theory with the maximum production occurring in July and August.

Figure 4.7.2 show how the cost change during the generations, and Figure 4.7.3 show how the fitness change during the generations. At first the cost is higher and during the generations it stabilizes around 1475.0 NOK. The corresponding fitness starts low, when the costs are high, and slowly increase as the system cost decrease.

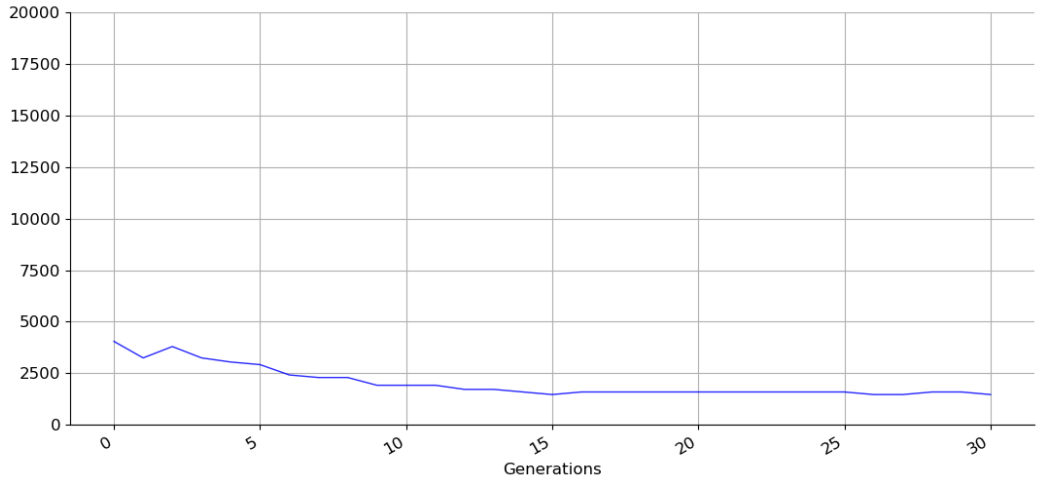


Figure 4.7.2: *The cost of the system for no month without production*

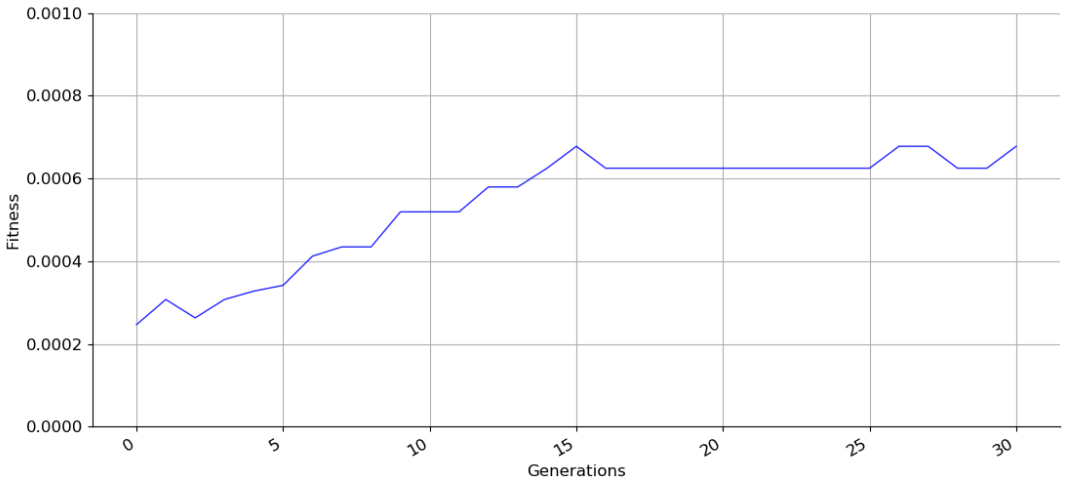


Figure 4.7.3: *The overall fitness for no month without production*

4.7.2 Optimum System Solution for Case 2

From the results of the second simulation presented in Table 4.7.2, the optimum system solution is a combination of PV panels in addition to a battery. From the GA, this system is able to meet the load demand even when there is one month without any power production. The installed capacity of a PV panel is 5.0 W and 1400.0 Wh for the battery. From the ten tests there are no variation in the total installed capacity, the fitness nor the cost. The fitness from the ten tests is 0.00014, and the cost is 3525.0 NOK.

Table 4.7.2: *Genetic algorithm test results with 1 months of no production*

	Installed capacity						**Results**				
Test	**PV1**	**PV2**	**PV3**	**PV4**	**WT**	**BAT**	**System production**			**Fitness**	**Cost**
	[W]	[W]	[W]	[W]	[W]	[Wh]	Min	Mean	Max		NOK
Test 1	0.0	5.0	0.0	0.0	0.0	1400.0	112.5	611.55	703.87	0.00028	3525.0
Test 2	5.0	0.0	0.0	0.0	0.0	1400.0	27.55	613.58	704.15	0.00028	3525.0
Test 3	0.0	0.0	5.0	0.0	0.0	1400.0	90.63	615.53	703.88	0.00028	3525.0
Test 4	5.0	0.0	0.0	0.0	0.0	1400.0	27.55	613.58	704.15	0.00028	3525.0
Test 5	0.0	5.0	0.0	0.0	0.0	1400.0	112.52	611.55	703.87	0.00028	3525.0
Test 6	0.0	0.0	0.0	5.0	0.0	1400.0	69.10	639.72	703.42	0.00028	3525.0
Test 7	5.0	0.0	0.0	0.0	0.0	1400.0	27.55	613.58	704.15	0.00028	3525.0
Test 8	0.0	0.0	0.0	5.0	0.0	1400.0	69.10	639.72	703.42	0.00028	3525.0
Test 9	5.0	0.0	0.0	0.0	0.0	1400.0	27.55	613.58	704.15	0.00028	3525.0
Test 10	0.0	5.0	0.0	0.0	0.0	1400.0	112.52	611.52	703.87	0.00028	3525.0

Figure 4.7.4 shows the system power production over the time for the optimal system solution from *Test 1*.

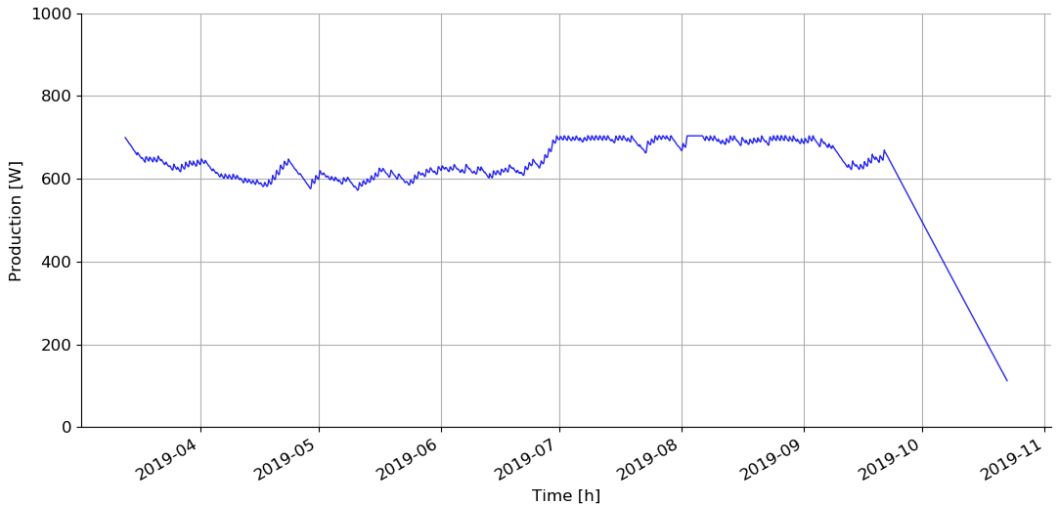


Figure 4.7.4: *System production for one month without production*

Test 1 is chosen since the results from all the tests gives the same fitness of 0.00028 and the same cost of 3525.0 NOK. For the month without any power production from the PV panel, the system power production linearly decrease from around 600.0 W to around 100.0 W. The installed capacity of the PV panel is just a fraction of the production compared to the battery capacity installed.

Figure 4.7.5 and 4.7.6 show how the cost and fitness vary throughout the generations. Figure 4.7.5 shows that the cost starts at a higher value and then stabilizes around 3500.0 NOK. Figure 4.7.6 shows a lower fitness in the beginning before it increase as the cost decrease. The cost is in these results higher than the results from the first simulation due to the additional battery capacity, and thus the fitness is also lower.

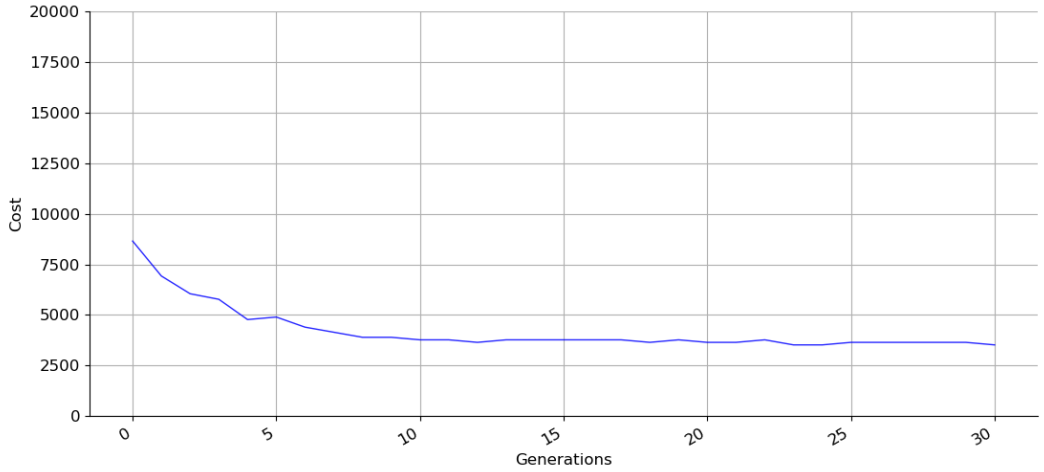


Figure 4.7.5: *Cost of system for second scenario for test 1*

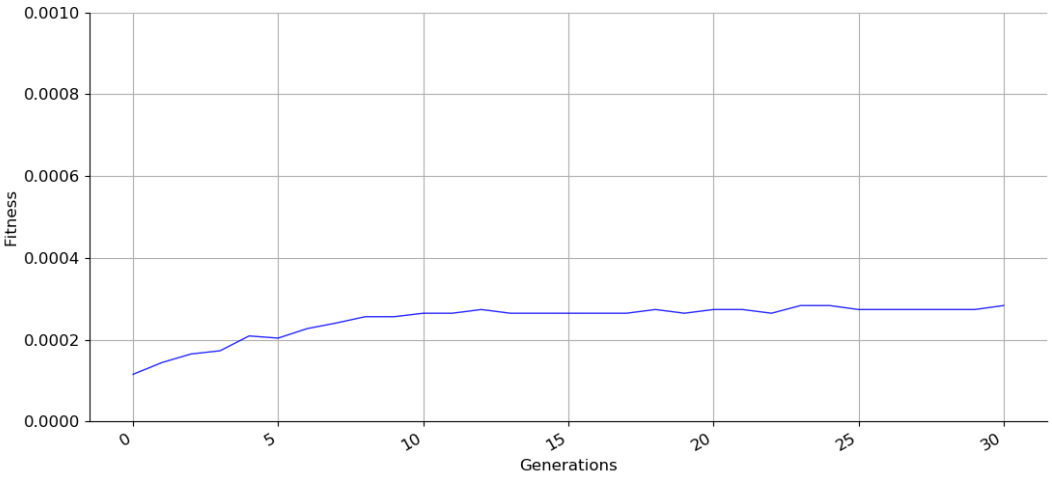


Figure 4.7.6: *Fitness of system for second scenario for test 1*

4.7.3 Optimum System Solution for Case 3

Table 4.7.3 shows the optimal system solutions for the system when there are three months without any power production. When there is three months without any power production the capacity of the battery rises to 3500 Wh. With the results from the GA, the system solution is able to meet the load demand when the battery capacity is at the maximum available capacity. The fitness from the ten tests is stable at 0.00014, but the cost differ and is either 7100.0 NOK, 7225.0 NOK or 7350.0 NOK.

Table 4.7.3: *Genetic algorithm test results with 3 months of no production*

	Installed capacity						**Results**				
Test	**PV1**	**PV2**	**PV3**	**PV4**	**WT**	**BAT**	**System production**			**Fitness**	**Cost**
	[W]	[W]	[W]	[W]	[W]	[Wh]	Min	Mean	Max		NOK
Test 1	5.0	0.0	10.0	0.0	0.0	3500.0	2.70	1384.73	1761.91	0.00014	7225.0
Test 2	0.0	5.0	0.0	5.0	0.0	3500.0	6.07	1387.03	1756.97	0.00014	7100.0
Test 3	0.0	0.0	15.0	0.0	0.0	3500.0	5.41	1386.14	1761.63	0.00014	7225.0
Test 4	0.0	0.0	0.0	10.0	0.0	3500.0	9.85	1389.77	1756.83	0.00014	7225.0
Test 5	5.0	0.0	0.0	10.0	0.0	3500.0	11.57	1391.25	1760.99	0.00014	7225.0
Test 6	0.0	0.0	15.0	0.0	0.0	3500.0	5.41	1386.14	1761.63	0.00014	7225.0
Test 7	0.0	5.0	0.0	5.0	0.0	3500.0	6.07	1387.03	1756.97	0.00014	7100.0
Test 8	0.0	0.0	0.0	10.0	0.0	3500.0	9.85	1389.77	1756.83	0.00014	7100.0
Test 9	0.0	0.0	0.0	10.0	0.0	3500.0	9.85	1389.77	1756.83	0.00014	7100.0
Test 10	15.0	0.0	0.0	5.0	0.0	3500.0	8.63	1389.55	1765.89	0.00014	7350.0

Figure 4.7.7 shows the system power production over the time for the optimal system solution from *Test 2*.

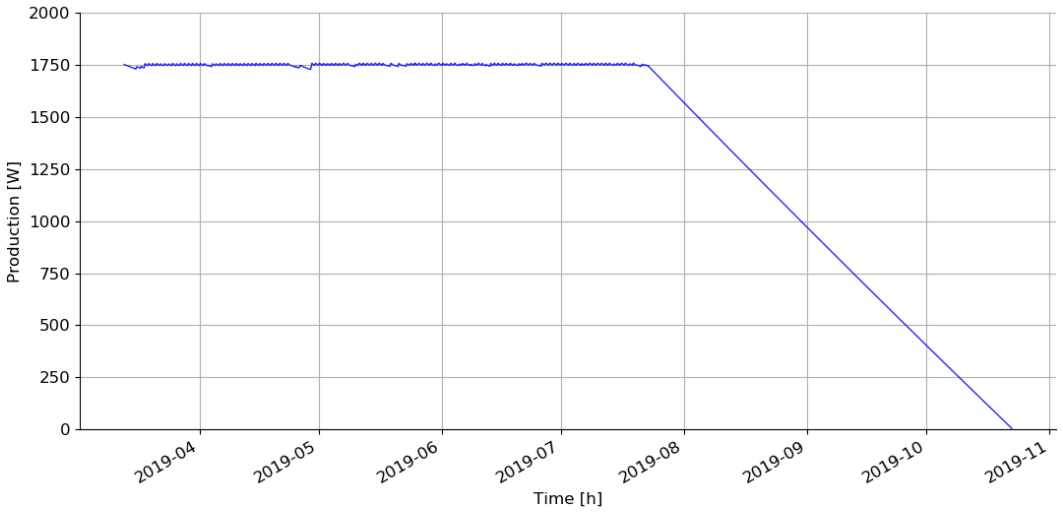


Figure 4.7.7: *System production for three months without production*

Test 2 is chosen due to that this is the result which gives the lowest cost during the run of the ten tests. The system production is stable during the first months with power production from the PV panels. When the power production from the PV panels stop, the system power production linearly decrease when only the battery is used to meet the load demand. The battery capacity goes to zero at the end of the three months, and due to this, the system is not able to be without power production longer than the three months.

Figure 4.7.8 and 4.7.9 shows how the cost and fitness vary throughout the generations. Figure 4.7.8 shows that the cost starts at a higher value and then stabilizes around 7000 NOK. The cost for this test ends at total 7100 NOK. Figure 4.7.9 shows a lower fitness in the beginning before it increase when the cost decrease. The cost is in these results higher than the results from the first and second simulation due to the additional battery, and thereby the fitness is lower.

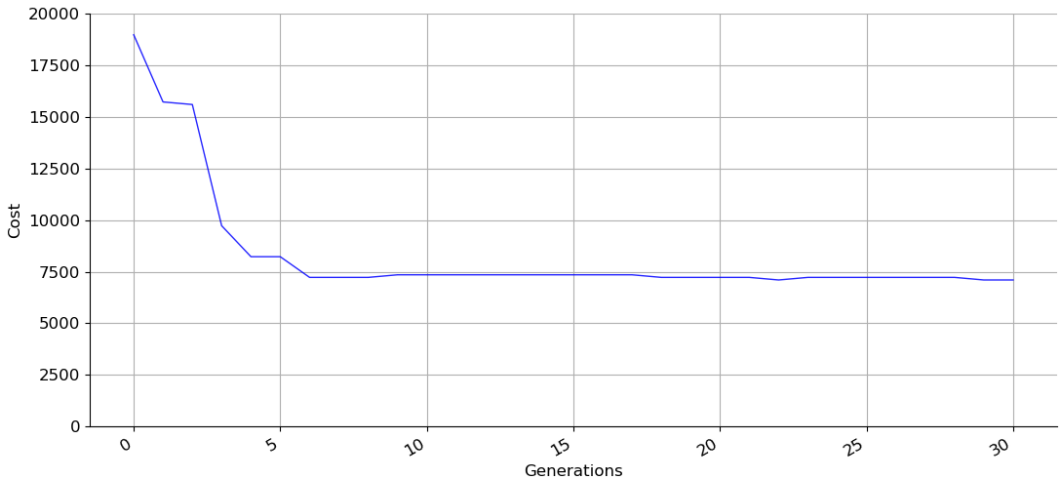


Figure 4.7.8: *Cost of system for third scenario for test 2*

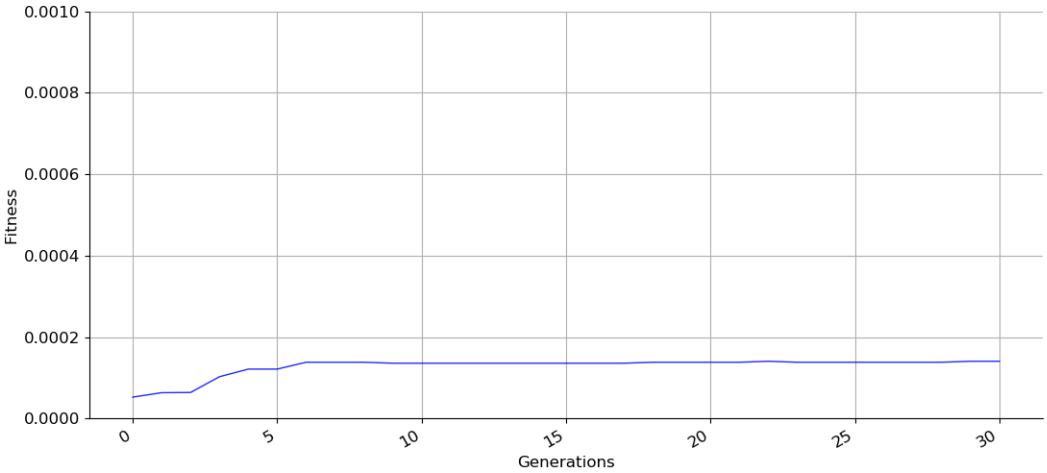


Figure 4.7.9: *Fitness of system for third scenario for test 2*

4.7.4 Evaluation of the GA Results

When comparing the results from the three simulation scenarios there are some correlation between them. The optimal system solution for all of the three simulation never include a wind turbine, but they all include PV panels. This indicate that as a result of using the predicted power in the GA, the optimum solution for the system is a PV/battery energy system.

The cost of a wind turbine is high and the additional cost of a PWM charge controller and dump load must also be included. The load is 0.75 W, so with continuous power production, the installed capacity is able to meet the load demand as seen from *Simulation 1*, Table 4.7.1. This indicate that since the PV/battery system is able to meet the load without a wind turbine as seen in the results from Table 4.7.1, 4.7.2, and 4.7.3. Including the wind turbine will give a higher total cost of the system. The results indicate that including the wind turbine will never be cost effective, if the system is able to meet the load demand without the turbine. However, if there are longer periods without any sun and thus no power production from the PV panels, a wind turbine could be necessary, unless the battery is oversized. The fitness figures from the three scenarios, show that the fitness decrease throughout the generations. Correspondingly, the figures of the cost show that they decrease throughout the same generations. The correspondingly behaviour of the cost and fitness indicate that the GA algorithm function as it should. The GA is by this capable of finding the solution of the minimization problem.

An assumption when simulating the three scenarios, is that when there is no power production, then there are no power production from neither the PV panels nor the wind turbine. This is done to give an indication of how big the battery needs to be in case there is no power production. If the loss of production was only for the PV panels the results could have been different and the wind turbine could have been included in the system.

The battery size in the three different scenarios varies. For the first scenario the optimal battery capacity is zero. The power produced by the PV panels should be enough to meet the load demand. For the second scenario the optimal battery capacity is 1400 Wh. This is the capacity of one battery from the case study presented in Section 3.1 in Chapter 3. From the results of this scenario, one battery of 1400 Wh in addition to 5.0 W installed PV panel is able to meet the load demand. When the time without any power production increase to three months, then the optimal system needs at least 3500 Wh to meet the load demand. This correspond to minimum three of the batteries presented in the case study. Included in the calculation of the needed capacity for the battery is DoD, saying that the maximum DoD should be 50% of the maximum capacity of the battery. This constraint is used to prolong the lifetime of the battery, but a decrease in this value could affect the size of the battery needed. When reducing the DoD, more of the capacity could be used before the charge controller disconnect the battery from the load. However, this would also reduce the lifetime of the battery.

4.7.5 Comparison of the GA Results Versus the Typical HydMet-Installation

The typical HydMet-installation, based on the case study presented in Section 3.1 in Chapter 3, consists of a PV panel with installed capacity of 20 W, a wind turbine with installed capacity of 300 W, and a battery of 3.5 kWh (scaled for three months backup if no power production). In addition to the wind turbine, a PWM

charge controller and a dump load is also installed. A MPPT controller is also installed for the PV panel. The total cost and fitness of the HydMet-installation can be seen in Table 4.7.4. The total system cost is calculated to 21964.0 NOK and the fitness to 0.00046.

Table 4.7.4: *A typical installed system of a HydMet-installation*

	Installed capacity						**Result**				
Test	**PV1**	**PV2**	**PV3**	**PV4**	**WT**	**BAT**	**System production**			**Fitness**	**Cost**
	[W]	[W]	[W]	[W]	[W]	[Wh]	Min	Mean	Max		NOK
HydMet	0.0	0.0	20.0	0.0	300.0	3500.0	4.28	105.15	3325.6	0.000046	21964.0

When comparing the results obtained from the GA to the HydMet-installation the difference in cost and fitness is noticeable. The results from the first simulation in Table 4.7.1, the fitness is 0.00068, nearly 15 times higher than the fitness from the HydMet-installation. Correspondingly the cost of the HydMet-installation is 15 times higher than the optimal system solution obtained from the first simulation scenario. However, the results from the first scenario has no safety net in case of periods without any power production from the PV panels. Including the battery with one month power backup, the cost of this optimal system is 3525 NOK, still 6 times cheaper than the HydMet-installation. This optimal system provides a bigger safety net in case of periods without any power production. An even bigger safety net is when the optimal system is sized for being capable of meeting the load demand with no power production for three months. This system solution is three times cheaper than the HydMet-installation.

Chapter 5

Conclusion

The objective of this project have consisted of two major parts: 1.) to predict the power production from photovoltaic panels and a wind turbine using machine learning, and 2.) optimize the size of a hybrid renewable energy system to have the least cost while still being able to supply the load at all times by using artificial intelligence. The objective was based on a case study which included a set of typical HydMet-station components with their configurations.

The input features eventually selected for the PV ANNs were as expected from prior knowledge of the data: the solar irradiance and operational temperature of the panels showed a significant relationship with the power production from the PV panels. For the wind turbine, only the wind speed showed a significant importance to the power production. A relationship between wind direction and power production was expected to see in the data, however the correlation and mutual information showed no significant correlation. The wind speed was therefore the only selected input to the wind turbine ANN.

The structure of the ANNs were selected based on gained knowledge on the models and through grid searching for the hyperparameters which allowed the models to best learn the relationship between the features and targets. The selected parameters proved to allow the ANNs to learn the relationship in the data with different hyperparameters. The optimum hyperparameters showed to be similar for the models, as would be expected when the models are feed similar data. The selected optimizers were Nadam and SGD. The learning rate tends to be slow for all models, with a higher number of epochs for the model to learn effectively. The typical batch size of 32 dominates the results. The best relationship between features and targets also seem to be learned with a type of uniform or random normal weight initialization.

All models were evaluated with different performance measures to gain insight in the model performance on multiple levels. The evaluation were based on inverse transformed predictions and cleaned target data. All models show a reasonable ability to predict, and have a corresponding performance on the training+validation set and the test set. The R ranges between 77% and 91% for the PV panels ANN. The performance could be improved, especially for the models performing in the lower range, as all the PV ANNs tends to underpredict. The ANN of PV panel no.4 show the lowest ability to learn from the data. The behaviour may be explained by the data correlations where the PV panel show a lower correlation to the solar irradiance than most PV panels and only show a low correlation to the operational temperature of

PV panel no.3. Removing the temperature from the input features to this model could maybe improve the model performance. The wind turbine ANN show a R of 98% and a small tendency to overpredict.

The selected models were used to predict the power production from the PV panels and wind turbine from "new" seen data. The performance was measured as for the training, validation and test data; on inverse transformed prediction and clean target data. All models have a lower ability to predict from the unseen data. Larger errors are expected when feeding the models with unseen data, as the behaviour of the data may not be exactly the same as the data the models have learned from. As the "new" data feed to the model in this thesis is the same as used for training, validation and testing combined, some higher ability to predict from the data were expected. The results my be explained by the randomness and inaccuracy the ANNs are subjected to during both training and validation, testing and prediction. Running the models multiple times to maybe find a pattern in their behaviour could improve the performance of the models especially on the training, validation and testing sets.

The optimization using GA with the predicted data from ML gave the optimal solution for the PV/wind/battery energy system. The results obtained from the GA indicate that it is possible to decrease the total cost by optimizing the system in the GA based on the predicted data.

For the GA a sensitivity analysis was carried out to find the GA parameters witch were able to give the best system solution and not be restricted by premature convergence. It was found that the number of generations should be below 50 to stabilize the system and avoid premature convergence. For a population size below 10 the system were not able to stabilize. Elite size should be a fraction of the population size, and was found to give the best result if it was $\frac{1}{5}$ of the population size. The mutation rate is small, and was found to be best at 0.15. If it is below this the tendency to premature convergence for the system increase, and if it is to high the ability to stabilize the total cost decrease.

The optimal system solution were found for three case scenarios. For the first scenario, when there was no months of no production, the result of the GA is that the system was capable of meeting the load demand with only PV panels 15-20 W. The second scenario with one month of no production, the result of GA is the combination of a 1400 kWh battery and 5 W PV panels. In the third scenario, with three months of no production, the result is a 3500 Wh battery and PV panels of 15 W. A comparison of these scenarios give the result of scenario one is the cheapest, with only the cost of PV panels. However, this system are not able to meet the load demand in case of no power production. For a system with the ability to meet the load demand in case of no production over a longer period, scenario two or three are fitted. With one battery of capacity 1400 Wh the system would increase the cost by the cost of the battery. In the third scenario the system was optimized with the constraint of always meeting the load demand with three months of no production. The result of GA for this scenario is that the system is always able to meet the load demand, if the battery capacity is at maximum.

Comparing these system with a typical HydMet-installation by the results of GA, the results show that for the optimized systems, they do not include a wind turbine. The wind turbine is expensive with the additional cost of a PWM controller and dump load. By the results of the GA, the system with only PV panels and batteries are able to meet the load demand without the wind turbine. Since the component cost of the wind turbine is high, including the wind turbine will not be the best solution by the definition of the objective function in the problem definition. However, including the wind turbine could overcome the obstacles in periods with no sun, but the presence of wind.

Artificial neural networks and artificial intelligence have proven able to optimize the hybrid system size to minimize the system cost while still proving reliability to the system load. They have also proven capable of predicting the power production from the available PV panels and wind turbine based on the meteorological data at site. However, the performance of the methods may still be improved to predict and optimize with higher accuracy and reliability.

5.1 Further Work

The abilities and potential that lies within the area of artificial intelligence and machine learning are major. The limiting factor in this thesis have been the lack of time to explore more areas of the subjects, as well as the time to dig deeper into the details of the methods and truly explore the potential that lies within them. This means that there still remains work to do, both in the areas of exploring other potential solutions to the problem, and to improve the performance and accuracy of the work already done. The interesting matters to study further can be summarized accordingly:

- Machine Learning:
  - Grid search all parameters together to find the actual best hyperparameters. These tests could also be done multiple times to hopefully find a trend in the results.
  - More data could be feed to the ANNs for the models to better learn the relationship between the features and targets.
  - Run the ANNs n times to study the accuracy and randomness of the prediction results for both the training/validation, test, and "new" data sets.
  - Test the ANNs on unseen data from a new site to validate the models ability to predict from other data.
- Optimization:
  - Further work for the optimization should include how much capacity the battery needs at the end of the discharging period. Including both the *one month* and *three months* constraint.
  - Test the optimization with one and three months with either just power production from PV panels or the wind turbine. This is to see how the optimized system will behave when the only power production is from the wind turbine or the PV panels.
  - Study other potential AI optimization techniques to optimize the hybrid system.
- Other:
  - Generalize the system to a higher extend by including more components, component configurations, costs, and so on.
  - Study other potential methods within artificial intelligence and machine learning for solving the problem.

Bibliography

[1] A. Géron. *Hands-On Machine Learning with Scikit-Learn and TensorFlow*. 1st ed. O'Reilly Media, 2017.

[2] A. Maleki and A. Askarzadeh. "Comparative study of artificial intelligence techniques for sizing of a hydrogen-based stand-alone photovoltaic/wind hybrid system". In: *international journal of hydrogen energy* 39.19 (2014), pp. 9973–9984.

[3] A. Maleki and F. Pourfayaz. "Optimal sizing of autonomous hybrid photovoltaic/wind/battery power system with LPSP technology by using evolutionary algorithms". In: *Solar Energy* 115 (2015), pp. 471–483.

[4] Alternativ Energi AS. *20W DC-Solar solcellepanel 64x29cm*. Available at `https://www.alternativenergi.no/202_-_20w_dc-solar_solcellepanel_64x29cm.html` (2019-04-25).

[5] ApexBattery. *Haze Batteries HZB12-110 Battery*. Available at `https://www.batts.com/haze-batteries-hzb12-110-battery.html` (2019-12-18).

[6] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. *Learning internal representations by error propagation*. Tech. rep. University of California, San Diego, Institute for Cognitive Science, 1985.

[7] D. N. Moriasi, J. G. Arnold, M. W. Van Liew, R. L. Bingner, R. D. Harmel, T. L. Veith. "Model Evaluation Guidelines for Systematic Quantification of Accuracy in Watershed Simulations". In: *Transactions of the ASABE* 50 (May 2007). DOI: 10.13031/2013.23153.

[8] E. G. Learned-Miller. *Entropy and Mutual Information*. Available at `https://people.cs.umass.edu/~elm/Teaching/Docs/mutInf.pdf` (2019-10-06).

[9] E. Stoltz. *Evolution of a salesman: A complete genetic algorithm tutorial for Python*. Available at `https://towardsdatascience.com/evolution-of-a-salesman-a-complete-genetic-algorithm-tutorial-for-python-6fe5d2b3ca35` (2019-11-10).

[10] European Commission. *Overview of PVGIS data sources and calculation methods*. Available at `http://re.jrc.ec.europa.eu/pvg_static/methods.html` (2019-05-03). 2017.

[11] F. Almonacid, C. Rus, P. Pérez-Higueras and L. Hontoria. "Calculation of the energy provided by a PV generator. Comparative study: Conventional methods vs. artificial neural networks". In: *Energy* 36.1 (2011), pp. 375–384.

[12] F. Chollet. *Deep Learning with Python*. Version 6. Manning, 2017.

[13] H. Belmili, M. Haddadi, S. Bacha, M. F. Almi, and B. Bendib. “Sizing stand-alone photovoltaic–wind hybrid system: Techno-economic analysis and optimization”. In: *Renewable and Sustainable Energy Reviews* 30 (2014), pp. 821–832.

[14] H. R. Maier, A. Jain, G. C. Dandy and K. P. Sudheer. “Methods used for the development of neural networks for the prediction of water resource variables in river systems: Current status and future directions”. In: *Environmental Modelling Software* 25 (2010).

[15] H. Yang, W. Zhou, L. Lu and Z. Fang. “Optimal sizing method for stand-alone hybrid solar–wind system with LPSP technology by using genetic algorithm”. In: *Solar energy* 82.4 (2008), pp. 354–367.

[16] J. Brownlee. *Deep Learning with Python*. Version 1.7. Self-published. 2016.

[17] J. Brownlee. *Machine Learning Mastery (Website)*. Available at `https://machinelearningmastery.com` (2019-10-18).

[18] J. F. Manwell, J. G. McGowan and A. L. Rogers. *Wind energy explained: theory, design and application*. John Wiley & Sons, 2010.

[19] J. VaderPlas. *Python Data Science Handbook*. 1st ed. O’Reilly Media, 2017.

[20] K. Anoune, M. Bouya, A. Astito and A. B. Abdellah. “Sizing methods and optimization techniques for PV-wind based hybrid renewable energy system: A review”. In: *Renewable and Sustainable Energy Reviews* 93 (2018), pp. 652–673.

[21] K. Homstøl and K. T. H. Salvesen. *Optimization of power supply for hydrology- and meteorology stations*. Energy Research Project (ENE504), University of Agder, Faculty of Engineering and Science. 2019.

[22] Leading Edge. *Leading Edge Power*. Available at `https://www.leadingedgepower.com/` (2019-12-18).

[23] M. H. Rashid and L. Chaar. *Power Electronics Handbook (Second Edition)*. Academic Press, 2007.

[24] MathWorks. *Machine Learning with MATLAB (E-book)*. Available at `https://se.mathworks.com/campaigns/offers/machine-learning-with-matlab.html` (2019-09-13). MathWorks, 2016.

[25] O. Abdoun and J. Abouchabaka and C. Tajani. “Analyzing the performance of mutation operators to solve the travelling salesman problem”. In: *arXiv preprint arXiv:1203.3099* (2012).

[26] P. Bajpai and V. Dash. “Hybrid renewable energy systems for power generation in stand-alone applications: A review”. In: *Renewable and Sustainable Energy Reviews* 16.5 (2012), pp. 2926–2939.

[27] P. G. V. Sampaio, and M. O. A. González, “Photovoltaic solar energy: Conceptual framework”. In: *Renewable and Sustainable Energy Reviews* 74 (2017), pp. 590–601.

[28] R. D. Thomas, N. C. Moses, E. A. Semple and A. J. Strang. “An efficient algorithm for the computation of average mutual information: Validation and implementation in Matlab”. In: *Journal of Mathematical Psychology* 61 (2014), pp. 45–59.

[29] R. Faranda and S. Leva. “Energy comparison of MPPT techniques for PV Systems”. In: *WSEAS transactions on power systems* 3.6 (2008), pp. 446–455.

[30] R. Vader. “Energy Unlimited”. In: *Victron Energy* (2011).

[31] S. C. Bhatia. *Advanced renewable energy systems,(Part 1 and 2)*. WPI Publishing, 2014.

[32] S. Mathew. *Wind energy: fundamentals, resource analysis and economics*. Vol. 1. Springer, 2006.

[33] S. Qazi. *Standalone Photovoltaic (PV) Systems for Disaster Relief and Remote Areas*. Elsevier, 2016.

[34] S. R. Wenham, M. A. Green, M. E Watt, R. Corkish and A. Sproul. *Applied photovoltaics*. Routledge, 2013.

[35] S. S. Singh and E. Fernandez. "Modeling, size optimization and sensitivity analysis of a remote hybrid renewable energy system". In: *Energy* 143 (2018), pp. 719–731.

[36] S. Sinha and S. Chandel. "Review of recent trends in optimization techniques for solar photovoltaic–wind based hybrid energy systems". In: *Renewable and Sustainable Energy Reviews* 50 (2015), pp. 755–769.

[37] Scanmatic AS. *Hydrologistasjoner*. Available at `https://www.scanmatic.no/energi/hydrologistasjoner/` (2019-12-18).

[38] Scikit-Learn. *Cross-validation: evaluating estimator performance*. Available at `https://scikit-learn.org/stable/modules/cross_validation.html` (2019-09-25).

[39] T. Khatib and A. Mohamed and K. Sopian. "A Software Tool for Optimal Sizing of PV Systems in Malaysia". In: *Modelling and Simulation in Engineering* 2012 (May 2012). DOI: 10.1155/2012/969248.

[40] The Pennysylvania State University. *What is the objective function?* Available at `https://www.courses.psu.edu/for/for466w_mem14/Ch11/HTML/Sec1/ch11sec1_ObjFn.htm` (2019-12-06).

[41] Y. Leung, Y. Gao and Z. Xu. "Degree of population diversity-a perspective on premature convergence in genetic algorithms and its markov chain analysis". In: *IEEE Transactions on Neural Networks* 8.5 (1997), pp. 1165–1176.

Appendices

Appendix A

Haze 120 VRLA Battery

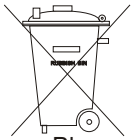
HZB12-120 Valve Regulated Lead Acid battery.
12 year design life for stand by power applications.
12 Volts 120 Ah

Innovative Features

- Completely maintenance free, sealed construction eliminates the need for watering
- Fully tank formed plates
- Analytical Grade electrolyte
- Spill proof / leak proof
- Valve regulated Max internal pressure 2.5 psi
- Multi-position usage
- ABS Case and cover - V0 on request
- Low self discharge
- FAA and IATA approved as non-hazardous
- Built to comply with IEC 896-2, DIN 43534, BS 6290 Pt4, Eurobat.

Specifications

Nominal Voltage	12 Volts
Nominal Capacity	120Ah (C20 @ 20 ˚C)
Design Life	12 Years
Operating Temperature	-20 ˚C to 50 ˚C
Grid alloy	Calcium / Tin lead alloy
Plates	Flat Pasted
Separator	Absorbant Glass Mat
Active material	Very high purity lead
Case and cover	ABS (VO on request)
Charge Voltage	Float 2.25 - 2.30 VPC @25 ˚C Cycliing 2.35 @25 ˚C Max. 2.4 VPC Max ripple 0.05C (A)
Electrolyte	Sulphuric acid Analytical grade purity
Venting Valve	EPDM Rubber 1.5 to 2 psi (10.5 - 14 KPa) release pressure. Resealing at 1 psi (7 KPa)
Terminal	Insert 12mm Dia M5 thread. Epoxy sealed by extended mechanical paths
Torque setting	The recommended torque value for all types is 5-7 Nm
Cables	Connectors, cables, terminal covers on request.



Haze Battery Company keenly encourages environmental awareness; PLEASE follow guidelines for the recycling /disposal of lead.

Website: www.hazebattery.com
E mail : sales@hazebattery.com

Sealed Lead Acid 12 Volt Bloc AGM Range
PRODUCT SHEET HZB12-120

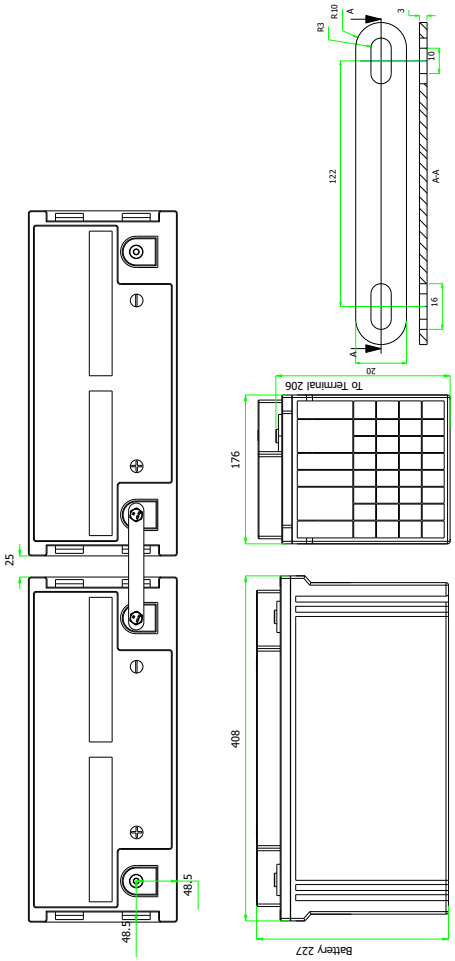
12V AGM

Specifications

Nominal Voltage	12V		
Nominal Capacity	120 Ah		
Dimensions	Total Height	227 mm	8.94 inches
	(Inc. terminals)	n/a mm	n/a inches
	Length	408 mm	16.06 inches
	Width	176 mm	6.93 inches
	Weight	35 Kg	77.35 lbs

Characteristics

Capacity 20 °C (68 °F) To 1.7 volts	20 hour rate	120.0 Ah
	10 hour rate	107.5 Ah
	5 hour rate	95.1 Ah
	1 hour rate	80.7 Ah
	15 min rate	55.4 Ah
	Internal Resistance	3 mOhms
	Impedance	S
Capacity correction for Temperature Variations (C_{20})	40 °C (104 °F)	102%
	20 °C (68 °F)	100%
	0 °C (32 °F)	85%
	-15 °C (5 °F)	65%
Self-Discharge 20 °C (68 °F)	Capacity after 1 months storage	98%
	Capacity after 3 months storage	94%
	Capacity after 6 months storage	86%
Short Circuit Current 20 °C (68 °F)	3300	
Terminal	Standard	14mm Insert M6 thread
	Optional	Cu/Lead Flag - J Type - Auto
Charging (Constant Voltage)	Cyclic	2.35 - 2.40 VPC (20-25 °C)
	Float	2.27 - 2.30 VPC (15-25 °C)



Constant Power Discharge - Watts per Cell @20 °C

End V per Cell	5M	10M	15M	20M	25M	30M	35M	40M	45M	60M	90M	2 hr	3 hr	4 hr
1.85	493	406	341	292	261	233	209	193	176	144	105	81.4	56.7	43.1
1.80	603	478	382	322	278	246	220	201	185	149	107	83.4	57.6	44.1
1.75	613	502	394	330	285	251	224	203	188	150	107	84.1	58.0	44.2
1.70	641	507	403	333	289	254	227	207	191	153	109	84.6	58.5	44.9
1.65	669	514	407	336	292	257	229	209	193	155	110	85.0	58.7	-
1.60	700	533	420	343	294	260	232	211	194	156	111	85.6	58.8	-

Constant Amps Discharge - Amps @20 °C

End V per Cell	5M	10M	15M	20M	25M	30M	35M	40M	45M	60M	90M	2 hr	3 hr	4 hr	5 hr	8 hr	10 hr	12 hr	20 hr
1.85	265	217	182	155	138	123	110	102	92.2	75.2	54.3	42.0	29.1	22.0	17.9	12.1	10.0	8.60	5.59
1.80	329	260	207	173	149	131	117	106	97.9	78.0	55.7	43.3	29.7	22.6	18.4	12.5	10.4	8.90	5.80
1.75	339	276	215	179	154	135	119	108	99.8	78.9	56.0	43.8	30.0	22.8	18.6	12.6	10.4	8.94	5.82
1.70	357	281	222	182	157	137	122	111	102	80.7	57.1	44.2	30.3	23.2	19.0	12.9	10.8	9.21	6.00
1.65	374	286	224	184	159	139	123	112	103	82.1	58.0	44.5	30.5	-	-	-	-	-	-
1.60	393	297	232	188	160	141	125	113	104	82.7	58.7	44.8	30.6	-	-	-	-	-	-

Ampere Hour @20 °C

End V per Cell	2 hr	3 hr	4 hr	5 hr	8 hr	10 hr	12 hr	20 hr
1.85	84.0	87.2	87.8	89.6	97.0	100	103	112
1.80	86.5	89.0	90.4	92.0	100	104	107	116
1.75	87.5	90.0	91.0	93.0	101	104	107	116
1.70	88.3	91.0	92.6	95.1	103	108	111	120
1.65	88.9	91.4	-	-	-	-	-	-
1.60	89.6	91.7	-	-	-	-	-	-

UL Recognised Component MH28512

EMC tested

Appendix B

LE-300 Wind Turbine Manual

Leading Edge

Local Power Worldwide

LE-300

Designed for industrial applications, **durability** and **reliability** are at the heart of the LE-300

Designed & manufactured in the UK

Industrial & residential off-grid

LE-300 Features

Features

High outputs
85W at 8m/s (17.8mph), 300W max

Quiet
Just 6dB(A) above background

Reliable
Precision engineered in the UK with only two moving parts

Rugged
Withstands storm force winds up to 27m/s, 60mph

Light
Weighs just 6.5Kg, reducing the loading on the support structure and making it easy to install in difficult locations

Piece of mind
Industry leading 5-year warranty

Renowned for being quiet thanks to our Whispower™ blade design

This compact and quiet horizontal axis turbine is considered to be the most rugged and reliable small wind turbine in its class. All at an affordable price.

The LE-300 is designed around a unique low inertia axial flux generator which utilises Neodymium rare earth magnetic materials. This alternator has zero 'cogging' which, together with its highly efficient and low 'TSR' Whispower blades, allow the turbine to generate power at very low wind speeds and deliver a higher output than the competition in high wind speeds.

The LE-300 survives winds up to 27m/s (60mph) by means of a simple passive aerodynamic design that reduces turbine RPM and power output at a certain threshold.

The robust aluminium alloy chassis and stainless steel components are protected from the elements with aerospace grade coatings and anodising.

Leading Edge

Local Power Worldwide

www.leadingedgepower.com

LE-300 Technical Overview

Rotor diameter - 1 metre

Rotor Type - 3-Blade upwind

Blade Material - Glass Reinforced, UV resistant Nylon

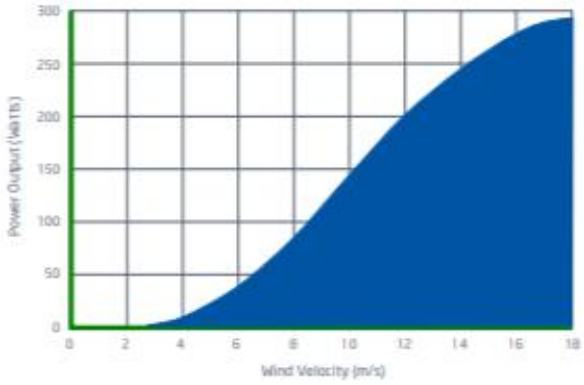
Rated output - 85W at 8m/s (18mph)

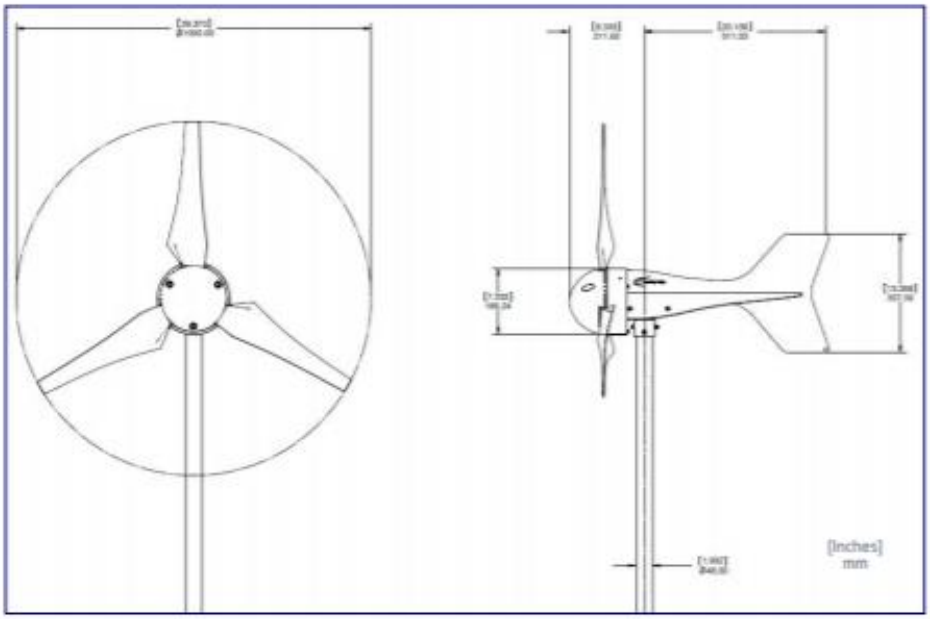
Peak output - 300W

Cut-in speed - 3m/s (6.7mph)

Weight - 6.5Kg

DC output voltage - 12V, 24V or 48V





In a typical stand alone system, the turbine sits on a tower (see our Guyed Tower Kit) and is connected to a battery bank via a maintenance (run/stop) switch. A charge controller is used to divert excess power to a dump load when your batteries are full.

The LE-300 is often combined with Solar PV panels in 'power hungry' off-grid renewable energy systems. Both systems are connected directly onto the batteries and each has their own controller.

Leading Edge

Local Power Worldwide

www.leadingedgepower.com

Industrial & residential off-grid

LE-300 Applications

- Data communications
- Telemetry
- Security
- LED Lighting systems
- Road signage
- Off-grid residences

Our LE-300 horizontal axis turbine is a quiet and powerful yet lightweight 300 watt wind generator for battery charging applications. It is used extensively in a variety of industrial applications.

Wind turbine performance is subject to many factors. All output data contained in this document is indicative and actual turbine outputs will depend on the prevailing site and installation conditions.

Your local distributor

Leading Edge

Leading Edge Turbines
Skyrrid Farm | Pontrilas
Herefordshire | HR2 0BW
www.leadingedgepower.com

Call +44 (0)1981 241668
Email sales@leadingedgepower.com

Appendix C

20W PV Module Datasheet

20W DATASHEET

ITEM NO	SNM20M-36
Cell Type	Mono-crystalline Silicon Solar Cells
Maximum power (Wp)	20wp
Maximum power voltage (V)	18V.
Maximum power current (A)	1.14A
Open circuit voltage (V)	22V
Short circuit current (A)	1.27A
Number of cells (Pcs)	36 PCS
Size of module (mm)	640×290×20mm
Maximum system voltage (V)	800
Temperature coefficients of Isc (%)	0.065+/-0.015%/ ℃
Temperature coefficients of Voc (%)	-（2.23+/-0.1）mv/ ℃
Temperature coefficients of Pm (%)	-（0.5+/-0.05）%/ ℃
Temperature coefficients of Im (%)	+0.1/ ℃
Temperature coefficients of Vm (%)	-0.38/ ℃
Temperature Range	-40℃～+80℃
Glass Type	Tempered Glass(3.2mm)
Surface Maximum Load Capacity	60m/s(200kg/sq.m)
Allowable Hail Load	steel ball fall down from 1m height
Weight per piece (kg)	2.5kg (6.0lbs)
Output tolerance (%)	+/-3%
Frame (Material, Corners, etc.)	Aluminum
Standard Test Conditions	AM1.5 100mW/cm2 25'C
Warranty	5 years product warranty and 10 years 90% of power
FF (%)	73%
Packing	1PCS/CTN
Remarks:	

Appendix D

Python Scripts

The main python scrips are included in this appendix. The "Main" combines the machine learning and the genetic algorithm to do the prediction and the optimization in one process. The "Machine Learning" is the script used for developing the ANN models which are used for prediction in the Main-script. The "Genetic Algorithm" is the module used for implementing the optimization of the energy system in the Main-script.

D.1 Main

```
import pandas as pd
import numpy as np
import tensorflow as tf
from sklearn.externals import joblib
import matplotlib.pyplot as plt

from hydmetopt.preprocessing import datacleaning, scale, prediction
from hydmetopt.genalg import genetic_algorithm


if __name__ == '__main__':

    ### Load the data
    ------------------------------------------------------------------------------
    df = pd.read_csv('data/UiA-data-export-20.08.2019_and_23.10.2019_formatted.csv', sep=';'
    )

    # Copy the original dataframe
    df_solar = df.copy()
    df_wind = df.copy()

    # Select columns for each dataframe
    df_solar = df_solar[['Timestamp', 'PV1_Power', 'PV2_Power', 'PV3_Power', 'PV4_Power', '
    PV1_Voltage', 'PV2_Voltage', 'PV3_Voltage', 'PV4_Voltage', 'PV1_Power_at_14V', '
    PV2_Power_at_14V', 'PV3_Power_at_14V', 'PV4_Power_at_14V', 'PV_Pyranometer', '
    PV3_Temperature']]
```

```
    df_wind = df_wind[['Timestamp', 'Wind_Current', 'Wind_Voltage', 'Wind_Power', '
    Wind_Speed', 'Wind_Dir']]


    ### Pre-process the data for prediction
    ------------------------------------------------------
    # Clean the solar and wind data for prediction
    df_solar = prediction.clean_solar_data_for_prediction(dataframe=df_solar)
    df_wind = prediction.clean_wind_data_for_prediction(dataframe=df_wind)

    # Scale the solar and wind data for prediction
    df_solar = scale.hourly_average(dataframe=df_solar)
    df_wind = scale.hourly_average(dataframe=df_wind)

    # Select the features for prediction
    df_solar_features = df_solar[['PV_Pyranometer', 'PV3_Temperature']]
    df_wind_features = df_wind[['Wind_Speed']]

    # Standardize features for prediction
    PV_X_scaler = joblib.load('models/PV_X_scaler_dump.save')
    WT_X_scaler = joblib.load('models/WT_X_scaler_dump.save')

    df_solar_features_std = prediction.standardize_data_for_prediction(scaler=PV_X_scaler,
    data=df_solar_features)
    df_wind_features_std = prediction.standardize_data_for_prediction(scaler=WT_X_scaler,
    data=df_wind_features)


    ### Predict from ML models
    ------------------------------------------------------------------
    # Load the ML models
    ML_model_PV1 = tf.keras.models.load_model('models/PV1_ML_model.h5')
    ML_model_PV2 = tf.keras.models.load_model('models/PV2_ML_model.h5')
    ML_model_PV3 = tf.keras.models.load_model('models/PV3_ML_model.h5')
    ML_model_PV4 = tf.keras.models.load_model('models/PV4_ML_model.h5')
    ML_model_WT = tf.keras.models.load_model('models/WT_ML_model_new.h5')

    # Predict
    PV1_pred_std = ML_model_PV1.predict(df_solar_features_std)
    PV2_pred_std = ML_model_PV2.predict(df_solar_features_std)
    PV3_pred_std = ML_model_PV3.predict(df_solar_features_std)
    PV4_pred_std = ML_model_PV4.predict(df_solar_features_std)
    WT_pred_std = ML_model_WT.predict(df_wind_features_std)


    ### Post-process data from ML models
    ------------------------------------------------------
    # Inverse standardize the prediction
    PV_y_scaler = joblib.load('models/PV_y_scaler_dump.save')
    WT_y_scaler = joblib.load('models/WT_y_scaler_dump.save')

    PV1_pred = prediction.inverse_standardize_data_from_prediction(scaler=PV_y_scaler, data=
    PV1_pred_std)
```

```
PV2_pred = prediction.inverse_standardize_data_from_prediction(scaler=PV_y_scaler, data=
PV2_pred_std)
PV3_pred = prediction.inverse_standardize_data_from_prediction(scaler=PV_y_scaler, data=
PV3_pred_std)
PV4_pred = prediction.inverse_standardize_data_from_prediction(scaler=PV_y_scaler, data=
PV4_pred_std)
WT_pred = prediction.inverse_standardize_data_from_prediction(scaler=WT_y_scaler, data=
WT_pred_std)

# Post-process the prediction data (clean negative values and reinsert timestamps)
df_start_time = df.Timestamp.min()
df_end_time = df.Timestamp.max()
solar_timestamp = df_solar['Timestamp']
wind_timestamp = df_wind['Timestamp']

PV1_pred_post = prediction.post_process_prediction(data=PV1_pred, dataname='
PV1_prediction', timestamp=solar_timestamp)
PV2_pred_post = prediction.post_process_prediction(data=PV2_pred, dataname='
PV2_prediction', timestamp=solar_timestamp)
PV3_pred_post = prediction.post_process_prediction(data=PV3_pred, dataname='
PV3_prediction', timestamp=solar_timestamp)
PV4_pred_post = prediction.post_process_prediction(data=PV4_pred, dataname='
PV4_prediction', timestamp=solar_timestamp)
WT_pred_post = prediction.post_process_prediction(data=WT_pred, dataname='WT_prediction'
, timestamp=wind_timestamp)

# Save the prediction data vs target
df_solar_filled = prediction.fill_empty_timestamps(df_solar)

prediction.save_pred_data(y_true=df_solar_filled['PV1_Power'], y_pred=PV1_pred_post['
PV1_prediction'], save_to_file=True, model_name='PV1')
prediction.save_pred_data(y_true=df_solar_filled['PV2_Power'], y_pred=PV2_pred_post['
PV2_prediction'], save_to_file=True, model_name='PV2')
prediction.save_pred_data(y_true=df_solar_filled['PV3_Power'], y_pred=PV3_pred_post['
PV3_prediction'], save_to_file=True, model_name='PV3')
prediction.save_pred_data(y_true=df_solar_filled['PV4_Power'], y_pred=PV4_pred_post['
PV4_prediction'], save_to_file=True, model_name='PV4')
prediction.save_pred_data(y_true=df_wind['Wind_Power'], y_pred=WT_pred_post['
WT_prediction'], save_to_file=True, model_name='WT')


### Save the results to GA
-----------------------------------------------------------------
ML_results = pd.concat([PV1_pred_post, PV2_pred_post['PV2_prediction'], PV3_pred_post['
PV3_prediction'], PV4_pred_post['PV4_prediction'], WT_pred_post['WT_prediction']], axis
=1)
ML_results = ML_results.dropna(axis=0)
print('ML results: \n {}'.format(ML_results))
ML_results.to_csv('ML_results.csv', sep=';')


### Pre-process data for GA
-----------------------------------------------------------------
# Prepare ML results to the GA
```

```
    ML_timestamps = ML_results['Timestamp']
    ML_results = [ML_results['PV1_prediction'].values,
                  ML_results['PV2_prediction'].values,
                  ML_results['PV3_prediction'].values,
                  ML_results['PV4_prediction'].values,
                  ML_results['WT_prediction'].values]



    ### Find the optimum system solution using GA
    ---------------------------------------------
    # Set the GA parameters
    NUM_COMPONENTS = 6
    POP_SIZE = 50
    ELITE_SIZE = 10
    MUTATION_RATE = 0.15
    NUM_GENERATIONS = 30
    LOAD = 0.75
    component_range = list(range(0B0000, 0B1011))

    GA_solution = genetic_algorithm.genetic_algorithm(pop_size=POP_SIZE, ML_results=
    ML_results, load=LOAD, elite_size=ELITE_SIZE, component_range=component_range,
    num_components=NUM_COMPONENTS, mutation_rate=MUTATION_RATE, generations=NUM_GENERATIONS)
    genetic_algorithm.present_final_solution(final_solution=GA_solution, ML_results=
    ML_results, load=LOAD)
    genetic_algorithm.save_final_solution(final_solution=GA_solution, ML_timestamps=
    ML_timestamps, ML_results=ML_results, load=LOAD)
```

D.2 Machine Learning

```
"""
    Machine Learning with Tensorflow and Keras.

    Resources:
    https://pythonprogramming.net/introduction-deep-learning-python-tensorflow-keras/
    https://pythonprogramming.net/tensorboard-analysis-deep-learning-python-tensorflow-keras
    /?completed=/convolutional-neural-network-deep-learning-python-tensorflow-keras/
    https://pythonprogramming.net/tensorboard-optimizing-models-deep-learning-python-
    tensorflow-keras/?completed=/tensorboard-analysis-deep-learning-python-tensorflow-keras/
"""

import tensorflow as tf
from tensorflow.keras.callbacks import TensorBoard
import datetime
import os

import pandas as pd
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
```

```
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.externals import joblib
import scipy

import evaluation_metrics
import save_data


def build_ANN_model(optimizer='sgd', learning_rate=0.01, init_mode='glorot_uniform',
    hidden_neurons=2, dropout_rate=0.0):
    # Build the Feed-Forward network
    model = tf.keras.models.Sequential()

    # Add layers to the model:
    model.add(tf.keras.layers.Dense(2, input_dim=2, activation=tf.nn.relu)) # Input layer
    model.add(tf.keras.layers.Dense(2, kernel_initializer=init_mode, activation=tf.nn.relu))
     # 1 Hidden layer
    model.add(tf.keras.layers.Dropout(dropout_rate))
    model.add(tf.keras.layers.Dense(2, kernel_initializer=init_mode, activation=tf.nn.relu))
     # 2 Hidden layer
    model.add(tf.keras.layers.Dropout(dropout_rate))
    model.add(tf.keras.layers.Dense(1)) # Output layer

    # Set parameters of optimizers
    tf.keras.optimizers.SGD(learning_rate=learning_rate)
    tf.keras.optimizers.RMSprop(learning_rate=learning_rate)
    tf.keras.optimizers.Adam(learning_rate=learning_rate)

    # Compile the model
    model.compile(optimizer=optimizer,
                  loss='mean_squared_error',
                  metrics=['accuracy'])

    return model


if __name__ == '__main__':
    # Load the data and extract features and targets
    df = df = pd.read_csv('../../data/UiA-data-export-20.08.2019_and_23.10.2019
    _formatted_clean_hourly_average_wind.csv', sep=';')
    X = df[['PV_Pyranometer','PV3_Temperature']].values
    y = df[['PV1_Power']].values

    # Split the data into train+validate and test sets (80/20 division)
    X_train_val, X_test, y_train_val, y_test = train_test_split(X, y, test_size=0.2,
    random_state=42)
    print('Train and validate on {} samples and test on {} samples'.format(len(X_train_val),
     len(X_test)))
    print('Train/val set (features): mean={}, std={}'.format(X_train_val.mean(axis=0),
    X_train_val.std(axis=0)))
    print('Train/val set (targets): mean={}, std={}'.format(y_train_val.mean(axis=0),
    y_train_val.std(axis=0)))
    print('Test set (features): mean={}, std={}'.format(X_test.mean(axis=0), X_test.std(axis
    =0)))
```

```
print('Test set (targets): mean={}, std={}'.format(y_test.mean(axis=0), y_test.std(axis
=0)))


# Standardize the training+validation data and test features
X_scaler = StandardScaler() # create scaler for features
X_scaler.fit(X_train_val) # compute the mean and std to be used for scaling
X_train_val_transformed = X_scaler.transform(X_train_val) # standardize the traning+
validation features

y_scaler = StandardScaler() # create scaler for targets
y_scaler.fit(y_train_val) # compute the mean and std to be used for scaling
y_train_val_transformed = y_scaler.transform(y_train_val) # standardize the training+
validation targets

# Save the scalers
joblib.dump(X_scaler, '../../data/WT_X_scaler_dump.save')
joblib.dump(y_scaler, '../../data/WT_y_scaler_dump.save')


# Build the model using a KerasRegressor
keras_reg = tf.keras.wrappers.scikit_learn.KerasRegressor(build_ANN_model)

# Define the grid search space
optimizers = ['sgd','adam','rmsprop', 'adagrad', 'adamax', 'nadam']
learning_rates = [0.0001, 0.001, 0.01]
initializers = ['RandomNormal', 'RandomUniform', 'glorot_uniform', 'uniform', 'normal',
'zero']
dropout_rates = [0, 0.1, 0.2, 0.3, 0.4, 0.5]
epochs = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
batch_sizes = [8, 16, 32, 64]

MODEL_NAME = 'WT_model_new'
parameters_search_space = dict(optimizer=optimizers,
                               learning_rate=learning_rates,
                               init_mode=initializers,
                               dropout_rate=dropout_rates,
                               epochs=epochs,
                               batch_size=batch_sizes)

# Train the model and grid search for the best hyperparameters
grid_search = GridSearchCV(estimator=keras_reg, param_grid=parameters_search_space,
n_jobs=-1, cv=10)
grid_search.fit(X_train_val_transformed, y_train_val_transformed, verbose=0)

# Print all the results
means = grid_search.cv_results_['mean_test_score']
stds = grid_search.cv_results_['std_test_score']
params = grid_search.cv_results_['params']
print('All grid search results:\n')
for mean, stdev, param in zip(means, stds, params):
    print('Mean={}, Std={}, with: {}'.format(mean, stdev, param))

# Print and save the best result
```

```
    print('Best score={} using {}'.format(grid_search.best_score_, grid_search.best_params_)
    )
    optimized_model = grid_search.best_estimator_
    optimized_model.model.save('{}_{}.h5'.format(MODEL_NAME, datetime.datetime.now().
    strftime("%Y%m%d-%H%M%S")))


    # Standardize the test features for model evaluation
    X_test_transformed = X_scaler.transform(X_test)

    # Predict with the optimum model
    y_pred_train_val_transformed = optimized_model.predict(X_train_val_transformed, verbose
    =0)
    y_pred_test_transformed = optimized_model.predict(X_test_transformed, verbose=0)

    # Inverse the standardization of the prediction for validation
    y_pred_train_val = y_scaler.inverse_transform(y_pred_train_val_transformed)
    y_pred_test = y_scaler.inverse_transform(y_pred_test_transformed) # scale back the data
    to the original representation


    # Evaluate the model performance and save the results
    evaluation_results = evaluation_metrics.evaluate_all(y_true_train_val=y_train_val,
    y_pred_train_val=y_pred_train_val, y_true_test=y_test, y_pred_test=y_pred_test,
    save_to_file=False, model_name=MODEL_NAME)
    print('Performance results:\n {}'.format(evaluation_results))

    # Save the train/test data and the predictions
    save_data.save_used_data(X_train_val=X_train_val, X_test=X_test, y_train_val=y_train_val
    , y_test=y_test, save_to_file=True, model_name=MODEL_NAME)
    save_data.save_pred_data(y_train_val=y_train_val, y_pred_train_val=y_pred_train_val,
    y_test=y_test, y_pred_test=y_pred_test, save_to_file=True, model_name=MODEL_NAME)
```

D.3 Genetic Algorithm

```
"""
    Genetic Algorithm for selecting the optimum system configuration, based on "The
    travaling
    salesman problem" (TSP). TSP is described as: "Given a list of cities and the distance
    between each pair of cities, what is the shortest possible route that visits each city
    and returns to the original city?"

    Resources:
    https://towardsdatascience.com/evolution-of-a-salesman-a-complete-genetic-algorithm-
    tutorial-for-python-6fe5d2b3ca35
"""

import numpy as np
import pandas as pd
import random, operator
import matplotlib.pyplot as plt
```

```
from hydmetopt.genalg import save_data


class Components:
    def __init__(self):
        self.PV1 = self.Component(capacity=20, cost=500, upper_bound=50, lower_bound=0)
        self.PV2 = self.Component(capacity=20, cost=500, upper_bound=50, lower_bound=0)
        self.PV3 = self.Component(capacity=20, cost=500, upper_bound=50, lower_bound=0)
        self.PV4 = self.Component(capacity=20, cost=500, upper_bound=50, lower_bound=0)
        self.WT = self.Component(capacity=300, cost=9750, upper_bound=500, lower_bound=0)
        self.BAT = self.Component(capacity=1400, cost=2300, upper_bound=3500, lower_bound=0)
        self.PWM = self.PWM_Charge_Controller(capacity=120, cost=4864, n_controller=0.80)
        self.MPPT = self.MPPT_Charge_Controller(capacity=120, cost=1100, n_controller=0.97)

    class Component():
        def __init__(self, capacity, cost, upper_bound, lower_bound, bit_length=10):
            self.capacity = capacity # Installed capacity
            self.cost = cost # Cost of installed capacity
            self.upper_bound = upper_bound # Upper bound of component search space
            self.lower_bound = lower_bound # Lower bound of component search space
            self.bit_length = bit_length # Length of component range

        def cost_factor(self):
            # Calculate the cost factor as the cost per W/Wh => kr/W or kr/Wh
            cost_factor = self.cost/self.capacity
            return cost_factor

        def resolution(self):
            # Calculate the resolution for each component within the search space
            resolution = (self.upper_bound-self.lower_bound)/self.bit_length
            return resolution


    class Battery():
        def __init__(self, capacity, initial_charge, self_discharge_rate, n_bat,
    depth_of_discharge):
            self.capacity = capacity # Battery capacity as Wh
            self.self_discharge_rate = self_discharge_rate # Discharge rate of battery (as
    efficiency)
            self.n_bat = n_bat # Battery efficiency
            self.depth_of_discharge = depth_of_discharge # As percentage
            self.minimum_SoC = self.capacity*(self.depth_of_discharge/100) # Minimum battery
     charge level
            self.maximum_SoC = self.capacity # Maximum battery charge
            self.initial_charge = initial_charge # initial charge of battery
            self.charge = initial_charge

        def battery_charge(self, gen_prod, load):
            ''' Simulates the behaviour of the installed battery.
            '''
            if gen_prod > load:
                # If E_prod > E_load check if it is possible to charge the battery
                if self.charge < self.maximum_SoC:
```

```
                # Charge the battery
                battery_charge = self.charge*(1-self.self_discharge_rate) + (gen_prod-
load)*self.n_bat
                # Stop charging the battery when the SoC reaches SoC_max
                if battery_charge > self.maximum_SoC:
                    battery_charge = self.maximum_SoC
                else:
                    pass
            else:
                # Maintain the battery charge if the charge is higher than charge_max
                battery_charge = self.maximum_SoC
        elif gen_prod < load:
            # If E_prod < E_load check if there is available battery charge
            if self.charge > self.minimum_SoC:
                # Use the battery
                battery_charge = self.charge * (1-self.self_discharge_rate) - (load-
gen_prod)*self.n_bat
                # Stop discharging the battery when SoC reaches SoC_min
                if battery_charge < self.minimum_SoC:
                    battery_charge = self.minimum_SoC
                else:
                    pass
            else:
                # Maintain the battery charge but loose self discharge
                battery_charge = self.charge * (1-self.self_discharge_rate)
        else:
            battery_charge = self.charge * (1-self.self_discharge_rate)

        self.charge = battery_charge
        return self.charge


class PWM_Charge_Controller():
    def __init__(self, capacity, cost, n_controller):
        self.capacity = capacity
        self.cost = cost # Cost of PWM and Dump load
        self.n_controller = n_controller
        self.installed = 0

    def is_installed(self, WT_inst):
        if WT_inst > 0:
            PWM_is_installed = 1
        else:
            PWM_is_installed = 0

        self.installed = PWM_is_installed
        return self.installed


class MPPT_Charge_Controller():
    def __init__(self, capacity, cost, n_controller):
        self.capacity = capacity
        self.cost = cost
        self.n_controller = n_controller
```

```
            self.installed = 0

        def is_installed(self, PV_inst):
            if PV_inst > 0:
                MPPT_is_installed = 1
            else:
                MPPT_is_installed = 0

            self.installed = MPPT_is_installed
            return self.installed


class Fitness:
    def __init__(self, system, ML_results, load):
        self.system = system # System = [1, 2, 3, 4, 5, 6]
        self.ML_results = ML_results # ML_results = [PV1_Power/Inst, PV2_Power/Inst,
    PV3_Power/Inst, PV4_Power/Inst, WT_Power/Inst]
        self.load = load # system requirements
        self.installed = self.installed_per_component()
        self.produced = self.produced_per_component()
        self.cost = 0
        self.production = 0
        self.fitness = 0

    def installed_per_component(self):
        # Calculate the installed capacity of each system component
        components = Components()

        P_pv1_inst = self.system[0]*components.PV1.resolution()
        P_pv2_inst = self.system[1]*components.PV2.resolution()
        P_pv3_inst = self.system[2]*components.PV3.resolution()
        P_pv4_inst = self.system[3]*components.PV4.resolution()
        P_wt_inst = self.system[4]*components.WT.resolution()
        E_bat_inst = self.system[5]*components.BAT.resolution()

        PWM_inst = components.PWM.is_installed(WT_inst=P_wt_inst)
        MPPT_inst = components.MPPT.is_installed(PV_inst=(P_pv1_inst+P_pv2_inst+P_pv3_inst+
    P_pv4_inst))

        installed_per_component = [P_pv1_inst, P_pv2_inst, P_pv3_inst, P_pv4_inst, P_wt_inst
    , E_bat_inst, PWM_inst, MPPT_inst]

        self.installed = installed_per_component
        return self.installed


    def produced_per_component(self):
        # Calculate the production for each system component
        components = Components()
        battery_model = Components.Battery(capacity=self.installed[5], initial_charge=self.
    installed[5], self_discharge_rate=0.00003, n_bat=0.95, depth_of_discharge=50)

        if self.installed[6] == 1:
            n_PWM = components.PWM.n_controller
```

```
        else:
            n_PWM = 1

        if self.installed[7] == 1:
            n_MPPT = components.MPPT.n_controller
        else:
            n_MPPT = 1

        P_pv1_prod = self.installed[0]*self.ML_results[0]*n_MPPT
        P_pv2_prod = self.installed[1]*self.ML_results[1]*n_MPPT
        P_pv3_prod = self.installed[2]*self.ML_results[2]*n_MPPT
        P_pv4_prod = self.installed[3]*self.ML_results[3]*n_MPPT
        P_wt_prod = self.installed[4]*self.ML_results[4]*n_PWM
        gen_production = P_pv1_prod + P_pv2_prod + P_pv3_prod + P_pv4_prod + P_wt_prod

        E_bat_SoC = []
        for i in gen_production:
            E_bat_SoC_t = battery_model.battery_charge(gen_prod=i, load=self.load)
            E_bat_SoC.append(E_bat_SoC_t)
        E_bat_SoC = np.array(E_bat_SoC)
        E_bat_Wh = E_bat_SoC-battery_model.minimum_SoC # The available SoC does not include
    below the minimum SoC

        produced_per_component = [P_pv1_prod, P_pv2_prod, P_pv3_prod, P_pv4_prod, P_wt_prod,
     E_bat_Wh]

        self.produced = produced_per_component
        return self.produced


    def system_cost(self):
        # Get the costs per W/Wh for each system component
        components = Components()
        cf = [components.PV1.cost_factor(),
              components.PV2.cost_factor(),
              components.PV3.cost_factor(),
              components.PV4.cost_factor(),
              components.WT.cost_factor(),
              components.BAT.cost_factor(),
              components.PWM.cost,
              components.MPPT.cost]

        # Calculate the system cost
        if self.cost == 0:
            system_cost = self.installed[0]*cf[0] + self.installed[1]*cf[1] + self.installed
    [2]*cf[2] + self.installed[3]*cf[3] + self.installed[4]*cf[4] + self.installed[5]*cf[5]
    + self.installed[6]*cf[6] + self.installed[7]*cf[7]

        self.cost = system_cost
        return self.cost


    def system_production(self):
        dt = 1
```

```
        # Calculate system production
        if self.production == 0:
            system_production = self.produced[0]*dt + self.produced[1]*dt + self.produced
    [2]*dt + self.produced[3]*dt + self.produced[4]*dt + self.produced[5]

        self.production = system_production
        return self.production


    def system_fitness(self):
        # calculate the fitness of the system as iverse of the system cost
        system_production = self.system_production()

        if self.fitness == 0:
            if (all(system_production > self.load)):
                self.fitness = 1/float(self.system_cost())
            else:
                self.fitness = 0

        return self.fitness


def create_system(component_range, num_components):
    ''' Generate a random system within our components range.

        component_range = bits from 0000 to 1111 for all components configuration
        num_components = number of components in the system
    '''
    system = random.choices(component_range, k=num_components) # seleck k random values from
     the component list

    return system


def initial_population(pop_size, component_range, num_components):
    ''' Create the first generation of the population.

        pop_size = number of individuals/chromosomes in the population
        component_range = range for components configuration
        num_components = number of components in the system
    '''
    population = []

    for i in range(0, pop_size):
        population.append(create_system(component_range, num_components)) # add a new
    individual to the population

    print('initial population: {}'.format(population))
    return population


def rank_systems(population, ML_results, load):
    ''' Rank the individuals of the population.
```

```
        population = population of systems
        ML_results = results from Machine Learning as nparray
        load = hourly system required energy
    '''
    fitness_results = {}

    for i in range(0, len(population)):
        fitness_results[i] = Fitness(population[i], ML_results, load).system_fitness() #
    calculate the fitness of each system in the population
    system_ranked = sorted(fitness_results.items(), key=operator.itemgetter(1), reverse=True
    )

    return system_ranked


def selection(pop_ranked, elite_size):
    ''' "Fitness proportionate selection" use the fitness of each individual relative to the
     population
        to assign a pobability of selection.
        "Elitism" carry the best performing individuals from the population over to the next
     generation.

        Use the output from rank_systems() to determine which individuals to select, and use
     a random
        drawn number to select our mating pool.
        Returns the IDs of the systems that will make up our mating pool.

        pop_ranked = ranking of the individuals
        elite_size = number of individuals selected for elitism
    '''
    selection_results = []

    # Set up a rulette wheel by calculating a relative fitness weight for each individual (
    next 3 lines)
    df = pd.DataFrame(np.array(pop_ranked), columns=['Index', 'Fitness'])
    df['cum_sum'] = df.Fitness.cumsum()
    df['cum_prec'] = 100*df.cum_sum/df.Fitness.sum()
    print('df: {}'.format(df))

    # Select the best systems through elitism
    for i in range(0, elite_size):
        selection_results.append(pop_ranked[i][0])

    # Compare a random drawn number to the fitness weights to select the mating pool
    for i in range(0, len(pop_ranked)-elite_size):
        pick = 100*random.random()
        for i in range(0, len(pop_ranked)):
            if pick <= df.iat[i, 3]:
                selection_results.append(pop_ranked[i][0])
                break

    return selection_results
```

```
def mating_pool(population, selection_results):
    ''' Extracts the selected individuals from out population.

        population = population to be breed
        selection_results = results from selection()
    '''
    mating_pool = []
    for i in range(0, len(selection_results)):
        index = selection_results[i]
        mating_pool.append(population[index])

    return mating_pool


def breed(parent1, parent2):
    ''' Create a new generation using crossover.

        Selects a random crossover point and creates a child from the parents genes.

        parent1, parent2 = individuals used for breeding
    '''
    child = []

    # Select random crossover point
    crossover_point = random.randint(1, len(parent1)-1)

    geneA = parent1[0:crossover_point]
    geneB = parent2[crossover_point:]

    # Create child with the parents genes
    child = np.ndarray.tolist(np.hstack((geneA, geneB)))

    return child


def breed_population(mating_pool, elite_size):
    ''' Create a offspring population by first selecting the elites then breed to fill the
    rest of
        the next generation.

        mating_pool = selected mating pool
        elite_size = number of individuals for elitism
    '''
    children = []
    length = len(mating_pool)-elite_size
    pool = random.sample(mating_pool, len(mating_pool))

    # Use elitism to retain the best systems
    for i in range(0, elite_size):
        children.append(mating_pool[i])

    # Breed to create the rest of the generation
    for i in range(0, length):
```

```
        child = breed(pool[i], pool[len(mating_pool)-i-1])
        children.append(child)

    return children


def mutate(individual, component_range, mutation_rate):
    ''' Mutate individuals to avoid local convergence by introducing novel configurations
    that will
        allow us to explore other parts of the solution space. Mutation assign a low
    probability of
        a gene changing.

        If the random generation for each gene in the system is smaler than the mutation
    rate
        then a random configuration from the component range is selected as mutant for that
    gene.

        individual = a chromosome
        component_range = the available component range used to create a new random gene (
    mutation)
        mutation_rate = probability of a gene changing
    '''
    individual_orig = individual.copy()
    for gene in range(len(individual)):
        if(random.random() < mutation_rate):
            mutant = random.choices(component_range, k=len(individual)) # seleck k random
    values from the component list
            individual[gene] = mutant[gene]

    return individual


def mutate_population(population, component_range, mutation_rate):
    ''' Extend the mutate function to run through the new population.

        population = current population of chromosomes
        component_range = the available component range used to create a new random gene (
    mutation)
        mutation_rate = probability of a gene changing
    '''
    mutated_pop = []

    for ind in range(0, len(population)):
        mutated_ind = mutate(population[ind], component_range, mutation_rate)
        mutated_pop.append(mutated_ind)

    return mutated_pop


def next_generation(current_gen, ML_results, load, elite_size, component_range,
    mutation_rate):
    ''' Generate a new population:
            rank_systems() ranks the systems in the current generation
```

```
            selection() determine the potential parents
            mating_pool() creates the mating pool
            breed_population() create our new generation
            mutate_population() applies mutation to the new generation

        current_gen = current generation
        ML_results = results from Machine Learning as nparray
        load = hourly system required energy
        elite_size = number of individuals selected for elitism
        component_range = the available component range used to create a new random gene (
    mutation)
        mutation_rate = probability of a gene changing
    '''
    pop_ranked = rank_systems(current_gen, ML_results, load)

    selection_results = selection(pop_ranked, elite_size)

    matingpool = mating_pool(current_gen, selection_results)

    children = breed_population(matingpool, elite_size)

    next_generation = mutate_population(children, component_range, mutation_rate)

    return next_generation


def genetic_algorithm(pop_size, ML_results, load, elite_size, component_range,
    num_components, mutation_rate, generations):
    ''' Loop through generations until the best solution.
    '''
    progress = []
    fitness_progress = []

    pop = initial_population(pop_size, component_range, num_components)
    progress.append(1/rank_systems(pop, ML_results, load)[0][1]) # Save the initial rank
    fitness_progress.append(rank_systems(pop, ML_results, load)[0][1]) # Save the initial
    rank

    for i in range(0, generations):
        pop = next_generation(pop, ML_results, load, elite_size, component_range,
    mutation_rate)
        progress.append(1/rank_systems(pop, ML_results, load)[0][1])
        fitness_progress.append(rank_systems(pop, ML_results, load)[0][1]) # Save the
    initial rank

    final_solution_index = rank_systems(pop, ML_results, load)[0][0]
    final_solution = pop[final_solution_index]

    # Plot and save the results
    plot_GA(progress=progress)
    plot_GA(progress=fitness_progress)
    save_data.save_generation_progress(progress=progress, save_to_file=True)
    save_data.save_generation_fitness_progress(progress=fitness_progress, save_to_file=True)
```

```
    return final_solution


def plot_GA(progress):
    plt.plot(progress)
    plt.ylabel('Cost')
    plt.xlabel('Generations')
    plt.show()


def present_final_solution(final_solution, ML_results, load):
    print('FINAL SYSTEM SOLUTION:')

    print('System: {}'.format(final_solution))

    components_installed = Fitness(final_solution, ML_results, load).installed_per_component
    ()
    print('Installed per component: {} W/Wh/N'.format(components_installed))

    components_production = Fitness(final_solution, ML_results, load).produced_per_component
    ()
    print('Produced per component:')
    print('- PV1: min={}W, mean={}W, max={}W'.format(round(min(components_production[0]),2),
     round(np.mean(components_production[0]),2), round(max(components_production[0]),2)))
    print('- PV2: min={}W, mean={}W, max={}W'.format(round(min(components_production[1]),2),
     round(np.mean(components_production[1]),2), round(max(components_production[1]),2)))
    print('- PV3: min={}W, mean={}W, max={}W'.format(round(min(components_production[2]),2),
     round(np.mean(components_production[2]),2), round(max(components_production[2]),2)))
    print('- PV4: min={}W, mean={}W, max={}W'.format(round(min(components_production[3]),2),
     round(np.mean(components_production[3]),2), round(max(components_production[3]),2)))
    print('- WT: min={}W, mean={}W, max={}W'.format(round(min(components_production[4]),2),
    round(np.mean(components_production[4]),2), round(max(components_production[4]),2)))
    print('- BAT: min={}Wh, mean={}Wh, max={}Wh'.format(round(min(components_production[5])
    ,2), round(np.mean(components_production[5]),2), round(max(components_production[5]),2))
    )
    plt.plot(components_production[0], label='PV1 power', linestyle='--')
    plt.plot(components_production[1], label='PV2 power', linestyle='--')
    plt.plot(components_production[2], label='PV3 power', linestyle='--')
    plt.plot(components_production[3], label='PV4 power', linestyle='--')
    plt.plot(components_production[4], label='WT power', linestyle=':')
    plt.plot(components_production[5], label='BAT power', linestyle=':')
    plt.legend()
    plt.ylabel('Component production [W/Wh]')
    plt.xlabel('Time [h]')
    plt.show()

    fitness = Fitness(final_solution, ML_results, load).system_fitness()
    print('Fitness: {}'.format(fitness))

    cost = Fitness(final_solution, ML_results, load).system_cost()
    print('Cost: {} NOK'.format(cost))

    system_production = Fitness(final_solution, ML_results, load).system_production()
    system_load = [load for i in range(0, len(system_production))]
```

```
    print('System production: min={}W, mean={}W, max={}W'.format(round(min(system_production
    ),2), round(np.mean(system_production),2), round(max(system_production),2)))
    plt.figure()
    plt.plot(system_production, label='system production')
    plt.plot(system_load, label='system load')
    plt.legend()
    plt.ylabel('System production')
    plt.xlabel('Time')
    plt.show()


def save_final_solution(final_solution, ML_timestamps, ML_results, load):

    components_installed = Fitness(final_solution, ML_results, load).installed_per_component
    ()
    fitness = Fitness(final_solution, ML_results, load).system_fitness()
    cost = Fitness(final_solution, ML_results, load).system_cost()

    components_production = Fitness(final_solution, ML_results, load).produced_per_component
    ()
    save_data.save_components_production(timestamps=ML_timestamps, components_production=
    components_production, save_to_file=True)

    system_production = Fitness(final_solution, ML_results, load).system_production()
    save_data.save_system_production(timestamps=ML_timestamps, system_production=
    system_production, save_to_file=True)

    system_load = [load for i in range(0, len(system_production))]
    save_data.save_system_load(timestamps=ML_timestamps, system_load=system_load,
    save_to_file=True)

    save_data.save_final_system_data(components_installed=components_installed,
    components_production=components_production, system_production=system_production,
    fitness=fitness, cost=cost, save_to_file=True)
```