Investigating cGAN-Based Modeling for Fault Detection on Wind Turbine Drivetrains

Master Thesis

Pascal Appel

TU Delft

Investigating cGAN-Based Modeling for Fault Detection on Wind Turbine Drivetrains

by

Pascal Appel

to obtain the degree of Master of Science

at the Delft University of Technology,

to be defended publicly on Monday December 16, 2024 at 13:00h.

Student number:	4561023
Project duration:	February 19, 2024 – December 16, 2024
Thesis committee:	Dr. D. Zappalá, Assistant Professor, FPT, Aerospace Engineering, TU Delft
	Prof. dr. S. J. Watson, Chairholder-Wind Energy Systems, Wind Energy Section, Aerospace Engineering, TU Delft
	Dr. A. K. Doan, Assistant Professor, FPT, Aerospace Engineering, TU Delft
Advisors:	Dr. M. A. Mitici, Assistant Professor, Utrecht University
	A. Eftekhari Milani, PhD, FPT, Aerospace Engineering, TU Delft

Cover:	Designed by Klaudia Justyna Kurzacz.
Style:	TU Delft Report Style, with modifications by Daan Zwaneveld

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

TU Delft

Preface

This report was written as part of my master's thesis in Sustainable Energy Technology at the Delft University of Technology. It describes the development and evaluation of a conditional Generative Adversarial Network (cGAN) for fault detection in wind turbine drivetrains, with a focus on the generator bearing. The investigation is motivated by the growing need for enhanced condition monitoring in wind turbines, which can help to speed up the energy transition, thereby contributing to a sustainable future.

I would like to take this opportunity to express genuine gratitude to my supervisor Donatella Zappalá for her support and guidance throughout the entire project. A special thanks to my two advisors Mihaela Mitici and Ali Eftekhari Milani, whose insights and suggestions greatly enriched the quality of this work. Both my supervisor and advisors played a crucial role in my growth throughout this project. Without their support, completing this project would not have been possible, especially as I was new to data-driven modeling at the start. I am also grateful to Energias de Portugal for providing the dataset, an essential component of this study. Finally, I want to thank my family and friends for their support during this intense period of graduation.

The report is intended for researchers, students and professionals interested in machine learning applications for fault detection. It aims to provide a detailed investigation of probabilistic target variable modeling for fault detection in wind turbine drivetrains. While this work particularly investigates monitoring of the generator bearing, the methodology could potentially be extended for monitoring of other wind turbine components. I hope that this work contributes to progressing the research in predictive maintenance, and ultimately supports a sustainable future for our planet.

Pascal Appel
Delft, December 2024

Summary

Accurate fault detection in wind turbines is a key factor for improving their reliability and reducing maintenance cost. This report investigates probabilistic target variable modeling using a conditional Generative Adversarial Network (cGAN) in a Normal Behavior Modeling framework for fault detection on wind turbine drivetrains. The dataset used for the investigation is the open-source Supervisory Control and Data Acquisition (SCADA) dataset provided by Energias de Portugal (EDP). A cGAN and a traditional Recurrent Neural Network (the base model) are developed and trained to model the generator bearing temperature. The performance of the cGAN is compared to that of the base model, to highlight differences associated with the modeling approach. The cGAN's output is a conditional distribution of the target temperature, given the operational state of the turbine. From this distribution, two methods of deriving a Health Indicator are investigated: (1) By reducing the distribution to a point estimate, and (2) by taking into account the whole distribution. From the base model, a third Health Indicator is derived. For all three Health Indicators, anomalies are detected by performing a non-parametric hypothesis test designed to detect deviations from the empirical healthy data distribution.

The findings indicate an enhanced temperature modeling performance of the cGAN on train, validation and test data. Especially the robustness appears to be significantly improved. While interpreting the outcome of applying the Health Indicators for fault detection remains challenging, the results suggest that the cGAN's probabilistic modeling improves fault detection by reducing the number of false responses.

Contents

List of Figures

List of Tables

Nomenclature

Abbreviations

Abbreviation	Definition
AI	Artificial Intelligence
BHI	Base Model Health Indicator
CBM	Condition-Based Maintenance
CCA	Canonical Correlation Analysis
CNN	Convolutional Neural Network
cGAN	Conditional Generative Adversarial Network
CUSUM	Cumulative Sum
DHI	Distributional Health Indicator
EDP	Energias de Portugal
GAN	Generative Adversarial Network
GRU	Gated Recurrent Unit
H0	Null Hypothesis
H1	Alternative Hypothesis
HI	Health Indicator
KCPD	Kernel Change Point Detection
LLN	Law of Large Numbers
LIME	Local Interpretable Model-Agnostic Explanation
LoMST-CUSUM	Local Minimum Spanning Tree and Cumulative Sum
LRP	Layer-Wise Relevance Propagation
LSTM	Long Short-Term Memory
ML	Machine Learning
MLP	Multi-Layer Perceptron
MSE	Mean Squared Error
NBM	Normal-Behavior Model
NBM-Li	Normal-Behavior Model with Lagged Inputs
NN	Neural Network
O&M	Operation and Maintenance
PEHI	Point Estimate Health Indicator
ReLU	Rectified Linear Unit
RNN	Recurrent Neural Network
SCADA	Supervisory Control and Data Acquisition
SHAP	Shapley Additive Explanations
SVM	Support Vector Machine
VAE	Variational Autoencoder
WHC-LOF	Ward Hierarchical Clustering and Local Outlier Factor
WT	Wind Turbine
XAI	Explainable Artificial Intelligence

Symbols

Symbol	Description	Unit
α	Significance level for hypothesis testing	[-]
β_1	First moment decay rate for ADAM optimizer	[-]
β_2	Second moment decay rate for ADAM optimizer	[-]
δ	Dropout rate during training	[-]
δ_G	Generator learning decay rate	[-]
δ_D	Discriminator learning decay rate	[-]
η_G	Generator learning rate	[-]
η_D	Discriminator learning rate	[-]
$\gamma(x, y)$	Coupling of distributions for Wasserstein distance	[-]
μ	Centroid of a cluster	[-]
$\mu_{x,H}$	Mean of the variable x over healthy data	°C
σ	Sigmoid activation function	[-]
τ	Total standard deviation	°C
τ_t	Standard deviation of the training data	°C
τ_{v-h}	Standard deviation of the validation healthy data	°C
$\tau_{x,H}$	Standard deviation of the variable x over healthy data	°C
$\tilde{cs}(t)$	Candidate cell state at timestep t	[-]
$\|d_i - \mu\|^2$	Squared Euclidean distance of a data point from the cluster centroid	[-]
ΔIV	Change in in-cluster variance after merging	[-]
∂L	Gradient of the loss function	[-]
$\partial \mathbf{z}$	Gradient of the neuron activation	[-]
B	Batch size during training	[-]
$\mathbf{c}$	Conditional input to the generator and discriminator	[-]
$cs(t)$	Cell state at timestep t	[-]
D_{steps}	Number of discriminator updates per iteration	[-]
$D(q\|\mathbf{c})$	Discriminator output, the probability that the input q is real given the condition $\mathbf{c}$	[-]
E	Total error	°C
E_t	Training error	°C
E_{v-h}	Error of the validation healthy data	°C
f'	Derivative of the activation function	[-]
$FG(t)$	Forget Gate value at timestep t	[-]
$G(\mathbf{r}\|\mathbf{c})$	Generator output, fake target generated given the noise $\mathbf{r}$ and condition $\mathbf{c}$	°C
H_B	Base Model Health Indicator (BHI)	[-]
H_D	Distributional Health Indicator (DHI)	[-]
H_{PE}	Point Estimate Health Indicator (PEHI)	[-]
HI	Health Indicator	[-]
$IG(t)$	Input Gate value at timestep t	[-]
L_{AD}	Adversarial loss for discriminator	[-]
L_{AG}	Adversarial loss for generator	[-]
L_D	Loss function of the discriminator	[-]
L_G	Loss function of the generator	[-]
$MAD(\mathbf{Y})$	Median Absolute Deviation of sampled set $\mathbf{Y}$	°C
$OG(t)$	Output Gate value at timestep t	[-]
p	Probability distribution (real or generated)	[-]
q	Input to the discriminator	[-]
$\mathbf{r}$	Noise vector input for generator	[-]
R_B	Residual in Base Model Health Indicator (BHI)	°C
R_P	Residual in Point Estimate Health Indicator (PEHI)	°C

Symbol	Description	Unit
ReLu()	Rectified Linear Unit function	[-]
tanh()	hyperbolic tangent function	[-]
T_S	Threshold for consecutive exceedances to trigger alarm	[-]
T_W	Time window size for counting exceedances	[-]
t	Target or true value / Real target value	°C
$\mathbf{U}$	Weight matrix for hidden states	[-]
$v_{healthy}$	Distribution of healthy data	[-]
$\mathbf{W}$	Weight matrix for input features	[-]
$W_1(P, Q)$	L1 Wasserstein distance between distributions P and Q	[-]
$W_{i,j}$	Weight matrix element between layers i and j	[-]
$x_{t,i}$	Input value at timestep t for feature i	[-]
y	Output of the network	[-]
y_B	Base model output	°C
y_D	Discriminator output	[-]
y_G	Generator output	°C
z	Neuron output or hidden state	[-]
Z_r	Robust Z-score	[-]

1

Introduction

Global warming is threatening planet Earth. In the last century, human activities have changed the planet drastically. One major problem is the emission of greenhouse gases. Relying on fossil fuels, the energy sector is a key contributor to these emissions. To combat climate change, the usage of fossil fuels must be reduced, and replaced by carbon-free energy sources.

Wind energy is playing a dominant role in the ongoing energy transition. In 2023, 77.6 GW have been installed globally bringing the cumulative installed capacity up to 906 GW [1]. In the coming years, the annually installed capacity is expected to grow further, as depicted in Figure 1.1. Next to an increasing rate of onshore installations, the offshore market keeps growing, doubling the annually installed capacity by 2027.

Operation and Maintenance (O&M) costs contribute significantly to a turbines total expenses. The literature suggests a contribution of about 20 - 30 % to total power generation cost for offshore turbines and approximately 10 - 15 % for onshore installations [2].

Wind turbine reliability directly impacts O&M costs. Identifying the most critical components affecting turbine reliability is not an easy task and depends on many factors such as the measurement metrics, turbine age, configuration, and installation location. Artigao et al. [3] conducted an extensive literature study summarizing and comparing the findings of 13 reliability studies related to wind farms globally, on and offshore. They concluded that while electrical and control systems cause the highest failure rate, about 75 % of the downtime is caused by mechanical failures. The findings of Artigao et al. [3] are summarized in Figure 1.2.

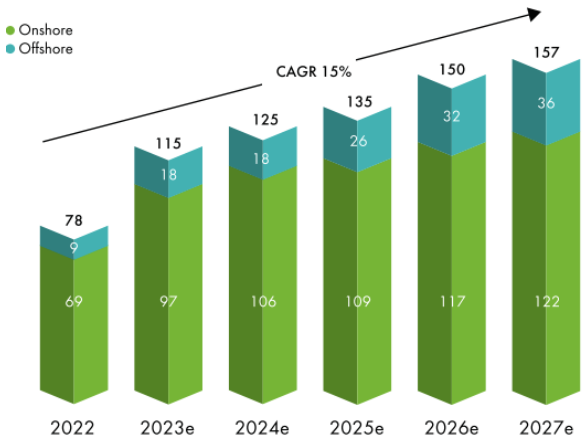


Figure 1.1: Projected global wind capacity growth (GW). Adopted from: *Global Wind Report 2023*, by the Global Wind Energy Council [1]. Copyright 2023 by the Global Wind Energy Council. No further use is allowed.

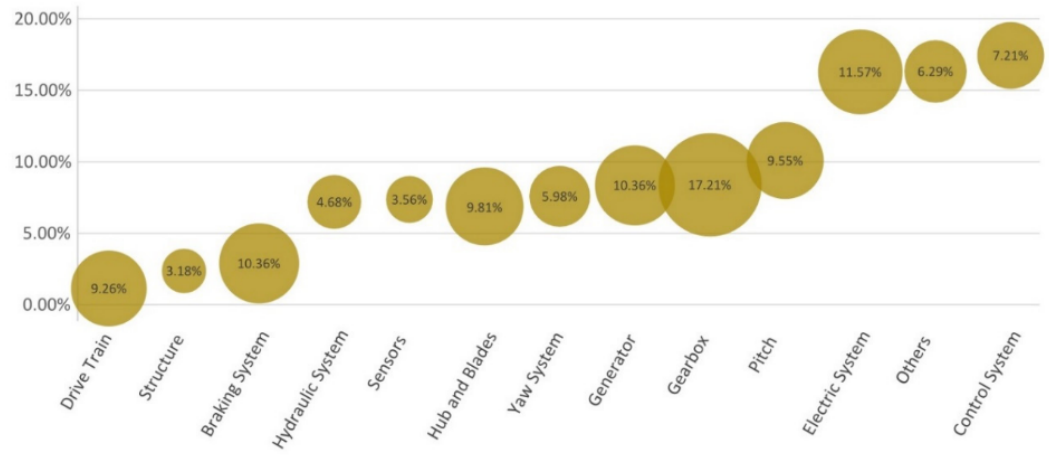


Figure 1.2: Failure rates (Y-axis) and downtime (bubble size) of WT subsystems. Drivetrain refers to main and high-speed shaft, and bearings. Adopted from: *Using SCADA Data for Wind Turbine Condition Monitoring: A Systematic Literature Review* by Maldonado-Correa et al. [2]. Licensed under CC BY.

Offshore installations have advantages over onshore ones but also come with challenges. Typically, these locations experience stronger and more continuous winds, which is beneficial for electricity generation. However, the strong winds and harsh maritime environment also contribute to higher, more fluctuating loads and increased damage from corrosion. It follows, that offshore turbines have higher failure rates than onshore ones [4]. Next to that, offshore turbines are more difficult to access. Not only the distance to shore but also the varying weather conditions make maintenance on these turbines challenging. Additionally, the availability of vessels further complicates the situation. All of these contributions result in higher downtime and O&M cost for offshore turbines, which increase with distance from shore [5]. Nevertheless, also onshore wind farms can be found in remote locations with rather extreme conditions [6]. Artigao et al. [3] emphasized that on and offshore turbines share the same critical components in need of reliability improvement.

Especially offshore, mechanical failures contribute significantly to the O&M cost of a WT. When analyzing the O&M cost of systems, the higher failure rates of electrical components, control, and hydraulic systems are less problematic compared to the difficulties involved in repairing mechanical failures [4]. Mechanical failures typically cause much higher transport costs and loss of production. The biggest savings on offshore O&M cost can be achieved by improving generator, and gearbox (subsequently called drivetrain) reliability [4], [5], [7].

Condition-based maintenance (CBM) can decrease O&M costs of wind turbines. In contrast to traditional maintenance approaches, which typically involve a mix of time-based preventive, and corrective maintenance, CBM strategies aim to intervene only when necessary, but before a severe failure. This becomes increasingly relevant for offshore turbines, due to the logistical challenges and costs associated with unscheduled interventions [8]. Consequently, the industry is increasingly adopting CBM, which relies on condition monitoring to determine the optimal timing for maintenance activities, with the goal of decreasing downtime and repair costs. To apply CBM effectively, a good understanding of the condition of individual components is crucial [8]. For offshore installations, moving from a traditional to a CBM approach can reduce the direct cost (transport, staff, material) of drivetrain failures by up to 8 %. Also, the availability of the turbines is increased which can reduce O&M cost by another 11 % [9].

To take advantage of the benefits of adopting CBM for drivetrain components, a good understanding of their condition is essential. However, monitoring the drivetrain is not an easy task due to fluctuating operating conditions that depend heavily on environmental factors [10]. Traditional methods for wind turbine drivetrain monitoring include vibration analysis [11], acoustic emission analysis [12], electrical signature analysis [13], and oil analysis [14]. Recently, also Supervisory Control and Data Acquisition (SCADA) data analysis is an upcoming topic [2]. SCADA data is collected by various sensors in the turbine, which is necessary for control and monitoring. This includes signals such as ambient conditions,

for example, wind speed, but also operating conditions such as component temperatures or the rotor rotational speed.

The traditional methods can provide accurate and highly sensitive fault detection, but are typically hard to implement. Vibration analysis is one of the most widely applied techniques for gearbox monitoring. It relies on sensitive sensors that require regular maintenance. Next to that, highly skilled engineers are needed to interpret the signals that strongly depend on the specific gearbox geometry [10], [11]. Also, acoustic emission analysis is a sensitive method for monitoring material conditions [15], and applied for fault detection in wind turbine drivetrain components [16]. Similar to vibration analysis, the signals collected by the acoustic sensors are hard to interpret [17]. Another challenge arises due to soundwaves easily interfering with background noise, which can mask damage-related signals [17]. Electrical signature analysis is a sophisticated method capable of detecting various types of faults, such as bearing defects, mechanical misalignments, or electrical faults [18]. Although not as commonly applied as vibration analysis, it still contributes to fault detection in wind turbine drivetrain components [16]. For accurate results, the electromechanical dynamics of the turbine need to be taken into account carefully, which again requires highly trained experts [19]. Oil analysis is a straightforward method to identify wear and contamination in the gearbox [20]. While some parameters like viscosity, particle count, and moisture content can be measured online, a detailed analysis involves sample collection and investigation in a lab. This offline analysis provides more detailed insights into gearbox problems, but typically only takes place every 6 months [20].

Using SCADA data can enhance CBM. The traditional methods share the need of highly specialized experts to setup models, gather data, and interpret the results. Using SCADA data can simplify this, by using sensors that are already available in the turbine [2]. Developing sophisticated methods still requires expert knowledge, but the results can potentially be interpreted easier. Another advantage of using SCADA data is the integration of multiple types of data which allows for a global analysis of the turbine, in particular in the context of its operating conditions. As a result, applying SCADA data to improve wind turbine CBM has gained a lot of attention in recent years [2]. It has the potential to enrich CBM in a cost-efficent way by utilizing available resources [21] and can be implemented non-intrusive, real-time, and online.

Most research on using SCADA data for WT condition monitoring (CM) applies deterministic methods [22]. These methods typically predict a singular outcome for a certain input, which inherently assumes that the relationship between in and output is free from variability. However, as WTs operate in a highly dynamic environment, this assumption is often violated in a real-world scenario. As a result, deterministic models are sensitive to noise and may struggle to generalize to unseen data. In the context of WT CM, this can be particularly problematic as models are often developed based on finite datasets, which are insufficient to capture all conditions a turbine may encounter during its life. Many of these limitations can be addressed by employing probabilistic modeling techniques. In contrast to deterministic methods, probabilistic approaches are able to account for uncertainty in the model outputs. By modeling the range of possible outcomes, probabilistic methods have the potential to improve generalization capabilities [23], and enhance trustworthiness in the model's predictions [22]. While probabilistic approaches are used in many other domains, their application for WT CM has not been researched much.

This leads to the goal of this project: to investigate the potential of data-driven probabilistic modeling for early-stage WT fault detection using SCADA data. The focus is on analyzing the impact of probabilistic modeling, compared to traditional deterministic models. To achieve this, a novel probabilistic method and a deterministic base model are developed, and compared. The base model is designed similarly to the probabilistic model, to allow for a comparison that is focused on the modeling approach. As drivetrain problems have a large impact on O&M cost, the developed techniques aim to monitor a key drivetrain component: the generator bearing. For model development and comparison, the open-source SCADA dataset from Energias de Portugal (EDP) [24] is used. The report is structured as follows: In Chapter 2, a literature study is conducted to identify the state of the art in data-driven fault detection, as well as shortcomings and possibilities for improvement. Thereafter, some theoretical background on machine learning (ML) and data analysis is given in Chapter 3. It follows a description of the dataset and data-preprocessing approach in Chapter 4. Chapter 5 gives insights into the methodology, while Chapter 6 presents the results. After that, the method and results are discussed critically

in Chapter 7. Finally, in Chapter 8, the report is completed with conclusions and recommendations for further research.

2

Literature Review

This chapter begins with a brief overview of how to differentiate between ML problem setups in the context of WT fault detection. Then, some background information on the most relevant ML techniques for WT drivetrain monitoring is given. After that, explainable AI (XAI) methods are introduced and several methods are discussed in more detail. Next, relevant recent literature related to probabilistic and deterministic anomaly detection, as well as XAI methods, is discussed. Next to techniques applied to WTs, also relevant approaches in other domains are recapped. Methods using the EDP dataset are described in detail. Then, the research gap is identified and the detailed goal of the report is presented by formulating the research questions. The chapter concludes with a brief analysis of this project in a socio-economical context.

2.1. Machine Learning Problem Setups

In data-driven techniques, the user uses an algorithm to learn from the data. This allows the user to uncover patterns in the data or model relationships that are described by the data. The latter often involves adjusting the algorithm's parameters, similar to fitting a polynomial to a set of data points.

Before diving into how ML methods can be applied for condition monitoring, the two main goals of condition monitoring [25] are introduced:

- Condition monitoring for diagnosis, which aims to identify and locate a fault or failure when it happens.
- Condition monitoring for prognosis, which aims to identify early-stage faults that are indicative of a failure in the near future.

In this context, a fault refers to a malfunction while the turbine is still operational. In contrast, a failure implies a major problem that causes the turbine to stop.

The field of machine learning is wide and contains many different approaches that can be tailored to solve a variety of problems. Regarding WT CM, supervised and unsupervised methods have been applied [25], [26]. In contrast to an unsupervised method, a supervised approach requires knowledge about the true relationship to be modeled.

Condition monitoring techniques are used to continuously assess the health and performance of a machine. In the case of WT CM, we are typically interested in distinguishing a healthy turbine from a faulty turbine. In many datasets, component failures are indicated, but early-stage faults are not clearly labeled. The distinction between healthy and faulty datapoints must be made by domain knowledge, heuristics, or other unsupervised techniques.

Assuming the dataset either provides information about healthy conditions or healthy data has been identified successfully, the problem is typically approached using classification or regression techniques [25].

Classification aims to describe the turbine's health condition by a categorical variable. Imbalanced datasets, which contain many instances of normal operation, and few instances of faulty operation, pose a challenge for classification. This is due to the model learning that normal operation is much more likely, so a model output of normal operation is correct for the vast majority of times. To avoid a biased model towards healthy operation [25], additional measures need to be taken to balance the datasets before training a classification model.

A common, regression-based approach is to create a Normal-behavior model (NBM) of the components of interest [25], [26]. This involves the prediction of a fault-indicative variable representing the normal state, as a function of other turbine parameters. The predicted variable is compared to the measured one, and consecutive significant deviations trigger an alarm. Often temperatures are used as fault indicative variables [10], but also the expected performance of the turbine in terms of power output is a typical choice [27], [28]. As NBMs are only trained on healthy data, they work well for imbalanced datasets. While temperature-based fault detection is a reliable and cost-effective solution, it can struggle with identifying early-stage faults due to high inertia and strong sensitivity to environmental conditions [20].

2.2. Machine Learning Techniques for Wind Turbine Drivetrain Monitoring

Drivetrain monitoring based on SCADA data has received substantial attention in the research community [10], [20], [25]. Traditional techniques, such as support vector machines (SVM) or decision trees are still applied in this field [25]. In addition, neural networks (NNs) are an emerging approach that leverages large datasets and can achieve enhanced performance when provided with substantial amounts of data [25]. Due to the performance advantage of NNs on large datasets, they are discussed in more detail in the following sub-section.

2.2.1. Neural Networks for Condition Monitoring in WTs

NNs for CM have been applied in both regression and classification setups. The most commonly used NNs for CM in WTs include multi-layer perceptron, autoencoder, convolutional, and recurrent NNs [26]. Recently, also transformers and generative models have been applied increasingly for such tasks [29]. While traditional NNs are set-up in a deterministic way, generative models, such as Variational Autoencoders (VAE) and Generative Adversarial Networks (GANs) are designed to approximate distributions. A deterministic model in this context outputs a single value for a certain input. In contrast, a generative model outputs different values for a certain input. This can potentially improve their performance, as they are able to model the uncertainty associated with a prediction. The different types of NNs applied for WT prognostics are briefly discussed below.

Multi-Layer Perceptron

A Multi-Layer Perceptron (MLP) is a feedforward neural network consisting of an input layer, one or more hidden layers, and an output layer. They are rather simple NNs, which are typically applied for basic fault detection tasks.

Autoencoder

An autoencoder compresses input data into a lower-dimensional representation and then reconstructs it. Similarly to an NBM, the reconstructed signal is compared to the input. As the autoencoder is trained to reconstruct healthy data, large deviations indicate unknown conditions and therefore potential faults.

Convolutional Neural-Network

Convolutional Neural Networks (CNNs) wereoriginally designed for processing grid-like data such as images. By applying convolutional filters to the input data, they are able to extract spatial features in the data. However, they still find applications in time-series analysis. Typically, they are well suited for identifying short-term dependencies while struggling with longer-term dependencies.

Recurrent Neural-Network

Recurrent Neural Networks (RNNs) process data sequentially. They are inherently designed to handle sequences, such as text data or time series. This enables them to establish relationships between

data points over the whole input sequence, making them particularly powerful for identifying long-term dependencies in time-series data. Long-short-term memory (LSTM) cells are a type of RNN neuron that performs exceptionally well in analyzing long-term dependencies.

Transformer

Transformers are advanced neural networks that process data in parallel, while still being able to identify relationships in the complete input sequence. Typically, these models require large amounts of data for accurate results. However, if provided with sufficient data, they can achieve outstanding performance.

Generative Adversarial Network

Generative Adversarial Networks (GANs) consist of two neural networks, a generator and a discriminator, that compete against each other to improve their performance. The generator creates synthetic data samples, while the discriminator evaluates whether the samples are real (from the dataset) or generated. GANs are widely applied in image generation tasks but have also proven very useful for creating synthetic data (data augmentation). This synthetic data could then be used to balance datasets. Conditional GANs (cGANs) are an extension of the classical GAN framework that takes conditional information into account when generating data instances. Next to image generation and data augmentation, the generative modeling approach of GANs, and especially cGANs have also been applied for anomaly detection. For simplicity, the distinction between GAN and cGAN is not made in the remainder of this literature review as their fundamental concepts are very similar.

2.2.2. Trustworthiness of AI Models

For AI models to be used for decision-making, trust is essential in many domains like medical, engineering, or economics [30]. However, many data-driven methods fail to provide reasons for their predictions. As a result, trust is limited and AI models are not widely deployed in these critical applications. Also in the wind industry AI models are not widely applied as operators need to understand the predictions made by the model to have confidence in its accuracy and reliability [25], [28]. This lack of trust is in particular problematic for NN's. The limitations can be addressed by applying explainable artificial intelligence (XAI) methods [31] and guidelines, which can provide reasoning for black-box model outputs, or improve a model's transparency. Consequently, the research community has recently focused more attention on this area [32].

2.3. Explainable Artifical Intelligence

Explainable AI models are machine learning models, which, in contrast to black-box models, provide reasoning for their decision [31]. Before diving into different methods and approaches, the nuanced distinction between interpretability and explainability in the context of AI models is introduced. An interpretable model is inherently understandable by humans. This means that the model's structure is transparent enough for humans to follow the cause-and-effect relationship without needing additional tools. In contrast, explainability is not that closely related to the internal structure of the model. Any kind of human understandable reasoning for a models decision improves its explainability. Explanations are thus answers to the question '*Why did the model make this decision*' [30]. Complex models are often not interpretable, but with the use of additional tools, or following XAI design guidelines, they can be explainable to some extent.

There are many ways to improve the explainability of ML models. Some general guidelines provided by Saranya et al. [31] are summarized below. One should use:

- Simple models: Simple models are often interpretable to some extent.
- Meaningful features: Use features that are understandable and relevant to the problem.
- Modular design: Break down complex models into smaller, interpretable parts.

An obvious possibility to improve explainability is incorporating a non-complex, interpretable model, like for example rule-based systems or decision trees. Neural networks, however, are inherently non-interpretable, due to the vast amount of parameters involved in decision-making. Additional measures can be taken to improve the explainability of such non-interpretable models. Consider the following XAI-related terms [31], which will be used to describe the different methods:

- Post-Hoc vs Ante-Hoc: Post-hoc techniques are applied after the model has made a prediction, while ante-hoc approaches aim to make the model itself more explainable.
- Model-Agnostic vs Model-Specific: Model-agnostic methods can be applied to any kind of ML model, whereas model-specific techniques can only be used with a certain type of model.
- Local vs Global: Local explanations focus on individual predictions, providing insights into why a specific prediction was made. Global methods aim to provide an understanding of the overall model behavior across all predictions.

The outputs provided by XAI methods can vary significantly. The most common types include:

- Feature relevance score: Identify which parts of the input were most relevant for the output.
- Surrogate models: Simplify the model by a less-complex, interpretable one.
- Leveraging structure: Provide explanations about the internal structure of the model.

As the field of XAI is broad, this list is not complete, and many other strategies exist. For more detailed information on the taxonomy of XAI methods, please refer to the works by Saranya et al. [31] and Marcinkevi et al. [33].

Choosing a suitable technique depends on many factors including problem domain, and data type. From now on, we will focus the discussion on methods that can be applied for NNs and time-series data. While numerous techniques are available, the following have been examined more thoroughly: Shapley Additive Explanations, Local Interpretable Model-Agnostic Explanation, Layer-Wise Relevance Propagation, DeepLift, Activation Maximization, Gradient-Weighted Class Activation Mapping, Attention Mechanism, [31], [34].

Shapley Additive Explanations (SHAP):
SHAP is one of the most used model-agnostic methods. It is a post-hoc technique that can be used to provide local and global feature relevance scores [31].

Local-Interpretable Model-Agnostic Explanation (LIME):
LIME is a popular model-agnostic technique used for explaining individual predictions of machine learning models. It works by approximating the complex model locally around the prediction to be explained with a simpler, interpretable model. Thus it creates a surrogate model. The basic idea is to perturb the input data, observe the resulting changes in the predictions, and then fit a simple model to these observations [31].

Layer-Wise Relevance Propagation (LRP):
Layer-Wise Relevance Propagation is model-specific to NNs. It works by backpropagating the prediction through the network layers to attribute the relevance of each input feature [34].

Activation Maximization:
Activation Maximization aims to understand what type of input maximally activates a particular neuron or layer within a neural network. Hence, it is a model-specific method for NNs. By optimizing the input to maximize the activation, this technique helps visualize the features that the network has learned to recognize [34]. By that it provides information about the leveraging structure.

Gradient-Weighted Class Activation Mapping (Grad-Cam):
Grad-CAM is a post-hoc visualization technique specifically designed for convolutional neural networks (CNNs). It produces a heatmap highlighting the regions of the input data that are most influential for the network's prediction [31], [34], which is equivalent to feature relevance scores. This is particularly useful for image data but can also be adapted for time-series data by visualizing important time steps.

Attention Mechanism:
Attention Mechanisms are initially invented to improve model performance in RNNs. They allow the model to dynamically focus on different parts of the input sequence. By visualizing the attention weights for a certain prediction, one can draw conclusions about which parts of the sequence the model focused on [34]. This method is ante-hoc and provides local timestep, or feature relevance scores, depending on the type of attention mechanism employed.

Li et al [34] analyzed the performance of 5 different XAI methods as a function of the neural network architecture and quality of training data. They identified LRP and SHAP as the best-performing methods in their model setup. Further, they found that SHAP is less sensitive to architectural modifications compared to LRP.

2.4. Generative Models for Fault Detection

The possibility of combining both NNs and XAI methods or individual elements has led to a vast number of research papers, each investigating a slightly different approach. This section first provides a brief summary of the most recent deterministic methods applied to WT fault detection. Next, the use of generative models for fault detection is explored in more detail. These types of models are capable of capturing the distribution of operational data, allowing them to address the problem from a probabilistic perspective. This is believed to provide a more nuanced understanding of the machine's condition, allowing for improved fault detection.

2.4.1. Deterministic AI Methods in WT Fault Detection

X. Xiao et al. [35] have proposed a method using stacked LSTM cells and an MLP regression layer to forecast the temperature of the WT main bearing. Liu et al. [36] used a three-layer convolutional autoencoder to detect anomalies in SCADA data. To improve its performance, they used information about fault instances during training. Further, they also used an advanced GAN-based data augmentation method to create additional fault instances. Jankauskas et al. [37] analyzed several RNN setups to predict the gearbox bearing temperature, and identified a 2-layer LSTM structure as best best-performing architecture.

2.4.2. Generative AI Methods in WT Fault Detection

Fu et al. [38] used synthetic data generated by a GAN to enhance the detection of main bearing faults in wind turbines. The GAN helped balance the vibration data, which improved the accuracy of the fault classifier. Fan et al. [39] proposed a convolutional variational autoencoder in a GAN setup to detect bearing faults, on, amongst others, WTs. They use the discriminator's output on the high-frequency, reconstructed vibration data as a health indicator for the bearing. A similar model setup was suggested by Zhang et al. [40]. They also use a GAN variational autoencoder, however, instead of convolutions, they use LSTMs. Using SCADA data enables them to apply the method to multiple systems. Another distinction lies in anomaly detection: Zhang et al. employ a traditional threshold approach using the reconstruction error as a health indicator.

2.4.3. Generative AI Methods in other Domains

Hoh et al. [41] have proposed a GAN-based model for anomaly detection of high-frequency audio data, with a generator consisting of a convolutional autoencoder. Another GAN-based anomaly detection approach was suggested by Zhao et al. The novel method consists of 2 GANs that are trained to generate anomalous and normal data. This synthetic data is used to train a classifier to distinguish between normal and abnormal data. Inference is done taking into account the outputs of the 2 discriminators, and the classifier. Ezeme et al. [42] have proposed a similar method. AD-GAN is a two-stage data augmentation method that is used to train a detector that learns to distinguish normal and abnormal samples. A different GAN-based approach has been employed by Qiu et al [43]. They use historical values of the targets to predict future timesteps. The anomaly is detected by comparing the discriminator output of the real sequence with the discriminator output of the fake sequence.

2.5. Model Setups incorporating XAI Methods

The need for improved trust in ML methods has drawn the research community's attention toward incorporating XAI methods. This section summarizes recent advancements in XAI, both within the domain of wind turbines and in a broader scope.

2.5.1. XAI Methods in WT Fault Detection

Wang et al. [44] used a transformer-like architecture to predict several fault-indicative temperatures. Next to that, they suggested a method to identify the relative impact of each feature on a detected

anomaly using the attention weights. Oliveira-Filho [45] proposed a variational autoencoder to detect anomalies on WT main bearing temperature. From the compressed latent space representation of the input data, they were able to derive WT diagnostic information. Further, they visualized the latent space for improved interpretability. Roelofs et al. [46] presented an autoencoder-based anomaly detection method, enhanced with an anomaly root cause analysis.

2.5.2. XAI Methods in other Domains

Park et al. [47] have combined a GRU-based autoencoder with a Light Gradient Boosting Machine to identify anomalies in nuclear power plants. To provide reasons for the outputs, the authors calculated Shapley values and employed an additional rule-based system. Roshan et al. [48] suggested an autoencoder-based model for computer network anomaly detection, including Shapley values to improve explainability. However, the authors also used the Shapley values for feature selection and provided a refined approach using only relevant features. Al-Kaf et al. [49] have used a random forest model with LIME and SHAP for accurate and explainable fault detection on inverters. Lee et al. [50] implemented a CNN and Grad-Cam to detect faults in a welding process.

2.6. AI Methods Using the EDP Dataset

This project uses the EDP dataset to develop, analyze and test a novel approach to WT fault detection. Therefore, research conducted using the same dataset is particularly relevant. This section aims to provide a summary of ML methods demonstrated using the EDP dataset.

In 2021, EDP hosted a competition on wind turbine fault detection. Barber et al. [51] summarized the findings of the competition in great detail. The participants were asked to come up with innovative solutions to reduce the corrective maintenance cost of five major wind turbine components: Gearbox, generator, generator bearing, transformer and hydraulic system. Two years of SCADA data (2016 - 2017) from five offshore turbines was provided by EDP. The performance of the proposed methods was measured in terms of savings. EDP supplied information on the associated cost for true positives, false positives and false negatives. Six solutions were submitted:

1. NBM for selected turbine temperatures: The residual error was normalized and an alarm triggered when the error rises above +- 3 standard deviations repeatedly in one week.
2. Combined local minimum spanning tree and cumulative sum of multivariate time series data (LoMST-CUSUM): The minimum spanning tree was used as anomaly score. Subsequently, small changes in average anomaly score were detected using the cumulative sum of this score [52]
3. Combined ward hierarchical clustering and novelty detection with local outlier factor (WHC-LOF): This method compares the turbines with each other, and associates anomalous conditions when one turbine's state differs significantly from the others.
4. NBM with lagged inputs (NBM-Li): Lagged data was added to the dataset. Normal turbine slip ring temperature was predicted using a random forest decision tree. Errors larger than 15 times their standard deviation were flagged as anomalies.
5. Canonical Correlation Analysis (CCA): The correlation between 2 sets of variables is analyzed. Significant changes in that correlation are detected and flagged as anomalies.
6. Kernel change point detection (KCPD): Changes in a single turbine signal are detected. The method works offline and is therefore only suitable for identifying anomalies in existing training datasets.

Please refer to Barber et al. [51] for more detailed information on the competition, or one of those methods.

Outside of the EDP competition, the dataset has also been used for the development and demonstration of fault detection algorithms. For example, Jankauskas et al. [37] compared different RNN-based architectures in an NBM setup of gearbox bearing temperature.

2.7. Project Contributions and Research Question

To reduce O&M cost of WTs, improved condition monitoring is an important factor. Substantial savings can be realized through high-performance, early-stage fault detection in drivetrain systems. From the literature study, it has become clear that trust is a major barrier to AI deployment in decision-making for critical applications, such as O&M decisions in the wind industry. While the research community is turning attention to using XAI techniques in many domains, the application for WT fault detection has not been researched much.

The need for explainable, data-driven methods led to a review of the use of generative models for fault detection. By addressing the problem in a probabilistic way, the model output can potentially be better understood by humans. While the model itself remains a black-box, a probabilistic output could be inherently more interpretable in human terms, as it aligns with the way humans naturally think about risks and uncertainties. Additionally, the explainability could be enhanced by XAI techniques. With this idea in mind, generative models, particularly GANs for fault detection, were reviewed. It turned out that GANs have received substantial attention in the research community in many domains. However, GANs are a relatively new method in the field of WT fault detection and require further investigation to understand their potential. Next to that, generative models are mainly used for data augmentation and rarely directly for anomaly detection. Researchers who use GANs directly for anomaly detection often use the discriminator to identify abnormal data. Probabilistic modeling of a target variable in a normal behavior framework has yet to be researched, particularly in the domain of WTs.

This project seeks to contribute to this gap by developing a cGAN-based drivetrain fault detection method that leverages probabilistic modeling of a target variable. Conditional information is provided to enable the model to generate samples specific to a certain operating condition. While the output is expected to be inherently better understandable to humans, the interpretability and explainability of the model are not part of this research project. In contrast, the focus is on analyzing the impact of probabilistic target variable modeling, compared to a deterministic model setup. For that, the cGAN is compared to a deterministic RNN (the base model). Both the cGAN and the base model are developed to model the generator bearing temperature.

The main research question to be answered in this report is formulated as:

> **What impact does probabilistic target variable modeling of the generator bearing temperature have on drivetrain monitoring and fault detection?**

This question is answered by addressing the following sub-questions:

1. *How does probabilistic modeling compare to a deterministic approach in representing the generator bearing temperature? What implications does this have concerning the goal of fault detection?*
2. *What is the impact of adopting probabilistic modeling on the detection sensitivity?*
3. *How does probabilistic modeling influence the robustness of fault detection in varying operational conditions?*
4. *What are the main advantages and disadvantages of deterministic vs. probabilistic approaches in this context?*

2.8. Socio-Economical Impact of Improved Drivetrain Monitoring

The implications of this research project in a socio-economical context are briefly discussed. Improved condition monitoring of WT drive trains has the potential to benefit economy and society in a variety of ways:

Mitigate Climate Change Impacts
During the literature review, it became evident that O&M cost are a major part of the total power generation costs of wind turbines. Improved condition monitoring can potentially reduce these costs, paving the way for wind energy to become an even more important player in the energy transition, which has the potential to speed up the energy transition. A quick and reliable energy transition does help society on the long run by mitigating climate change impact. This improves overall quality of life on planet earth by reducing the amount of natural disasters, water scarcity, heat waves and other direct or indirect consequences of climate change [53].

Extend the Turbine's Life
Drivetrain monitoring helps to maintain turbine availability and performance over its lifetime. This contributes to using the turbine in an optimal way, which ensures that resources are used as good as possible. Resources in this context include the raw materials and energy to build the turbine, but also vessels and machinery to place it. The benefits of this include reduced environmental impact of fabricating and placing new turbines, and accelerated energy transition due to higher availability of construction machinery.

Provide Stable and Reliable Power
A major challenge of the energy transition is to ensure that the grid is able to reliably provide power to the consumers, at all times. With an energy system relying on power generated by WTs, the reliability of individual turbines becomes a key factor in the overall reliability of the grid. This is in particular important for drivetrains, as these have a major impact on the overall downtime of turbines [3]. Therefore, improved condition monitoring of WT drivetrains helps to provide stable power to the consumers, which is beneficial for both, society and economy.

Economic Growth
The wind industry is a growing sector. Advancements in drivetrain monitoring can help wind energy become more competitive, which helps the wind industry grow even faster. The Global Wind Energy Council has estimated in 2021 that the wind industry creates 3.3 million new jobs until 2026 [54]. These jobs are created not only directly in the wind industry, but also along the whole supply chain. Improved drivetrain monitoring benefits society and economy by contributing to this economic growth.

3

Theoretical Background

To understand the techniques applied in this research project, some background knowledge is essential. This Chapter aims to introduce the main concepts used in the remainder of the report. Section 3.1 is dedicated to the introduction of ML basic concepts. After, the discussion is turned to NNs in Section 3.2, as these are closely related to GANs and form the basis of understanding the latter. It follows an extension to RNNs in Section 3.3. Next, the GAN itself is introduced in Section 3.4, highlighting the differences with the classical NN. In Section 3.5 some statistical concepts and metrics, which are used to validate and post-process the output of the models, are explained. Finally, the Chapter concludes with a discussion of Hierarchical Clustering in Section 3.6.

3.1. Machine Learning Basics

The field of ML is large and consists of an enormous number of different methods. While the exact purpose and setup of a method can change significantly, the broader goal is typically the same: to learn from data. In other words, to reveal relationships in a dataset that are not immediately visible.

3.1.1. Normal Behaviour Modeling

Normal behaviour modeling first aims to estimate the value of a fault-indicative variable in a healthy machine condition. In the second step, this estimated value is compared to the measured value. This comparison results in an anomaly score: a quantitative measure of how anomalous the observation is with respect to the healthy machine condition. The three steps of NBM are discussed in a bit more detail below.

Step 1: Healthy Machine Modeling

First, a relevant target variable t must be selected, that describes the machine's (or component's) condition. As mentioned already in the introduction of the report, component temperatures are a common choice. Next, corresponding suitable predictors $\mathbf{x}$ for the target t must be selected, also called features. The modelling problem can be described by the following equation:

$$t = f(\mathbf{x}) + \epsilon \tag{3.1}$$

The goal is to find the function $f(\mathbf{x})$ that minimizes the error ϵ. As the healthy machine condition is modeled, the error should be minimal over all healthy data points available. For that, the dataset is divided into healthy and faulty data points. Since a continuous variable is modeled, this is also referred to as regression.

Step 2: Comparison of Estimated and Measured Target

The comparison between the estimated and measured target variables can be performed in various ways. The outcome of this comparison is used to quantify how anomalous an observation is. Let the estimated target value be denoted as y, such that:

$$y = f(\mathbf{x}) \tag{3.2}$$

The residual R is defined as the difference between the measured and estimated target:

$$R = t - y \tag{3.3}$$

Note that this is the same as the error ϵ mentioned before. However, it is now used in the context of anomaly quantification.

Typically, every new observation of R is normalized with the healthy mean μ_R and healthy standard deviation σ_R. Healthy in this context means that this is a value obtained considering only healthy datapoints. The obtained value is called the Z-score, Z_R, of the residual.

$$Z_R = \frac{R - \mu_R}{\sigma_R} \tag{3.4}$$

It quantifies, how many standard deviations the residual deviates from its healthy mean. Large values of Z_R can then be interpreted as anomalies since they reflect unusual large deviations from the residual's mean. This approach was also applied to the EDP dataset [51].

Step 3: Anomaly detection
Since the final output of the algorithm should be a binary value (healthy, not healthy), the results from Step 2 need to be processed further. Typically, this step involves filtering anomaly scores based on a threshold or accumulating the anomaly scores. The techniques used in this project are explained in detail in Chapter 5.

3.1.2. Training, Validating and Testing Machine Learning Models

To properly evaluate the performance of an ML model, it needs to be validated and tested on unseen data. Therefore, the whole dataset is usually divided into train-, validation-, and test-sets. The train set is used to learn from data, while the validation and test sets are used to fine-tune the model and evaluate the performance, respectively. During the validation phase, model-defining parameters (hyperparameters) can be adjusted based on some performance metrics on the validation data set. In contrast, the test set remains untouched until all model tuning is finished. It serves as a true measure of the final model performance.

3.2. Neural-Networks

A neural network is a mathematical model designed to map inputs to corresponding outputs, as for example described in Equation 3.1. It consists of interconnected layers of nodes that process and transform data from input to output. Nodes in this context are also referred to as neurons. The model setup is inspired by the human brain, where information flows through the synaptic connection of neurons. In this project, a neural network is used for step 1 of the NBM to model the target variable t. In this section basic concepts of NNs are explained. For more details, please refer to *Deep Learning* by Goodfellow et al. [55].

Figure 3.1 shows a simple setup of a neural network.

The input vector $\mathbf{x}$ is fed into the network's input layer on the left. In the hidden layer, each element of $\mathbf{x}$ is first multiplied by three weights $\mathbf{W}_{i,j}^{(h)}$, which are represented by the arrows. Then the result of those multiplications is summed up to form the input a_j to the hidden neurons:

$$a_j = \sum_{i=1}^{n} x_i W_{i,j}^{(h)} \tag{3.5}$$

This input is transformed by a nonlinear function f and then passed on to the following layer. In the output layer, the data coming from the previous layer is first multiplied by the weights $\mathbf{W}_j^{(o)}$. This is

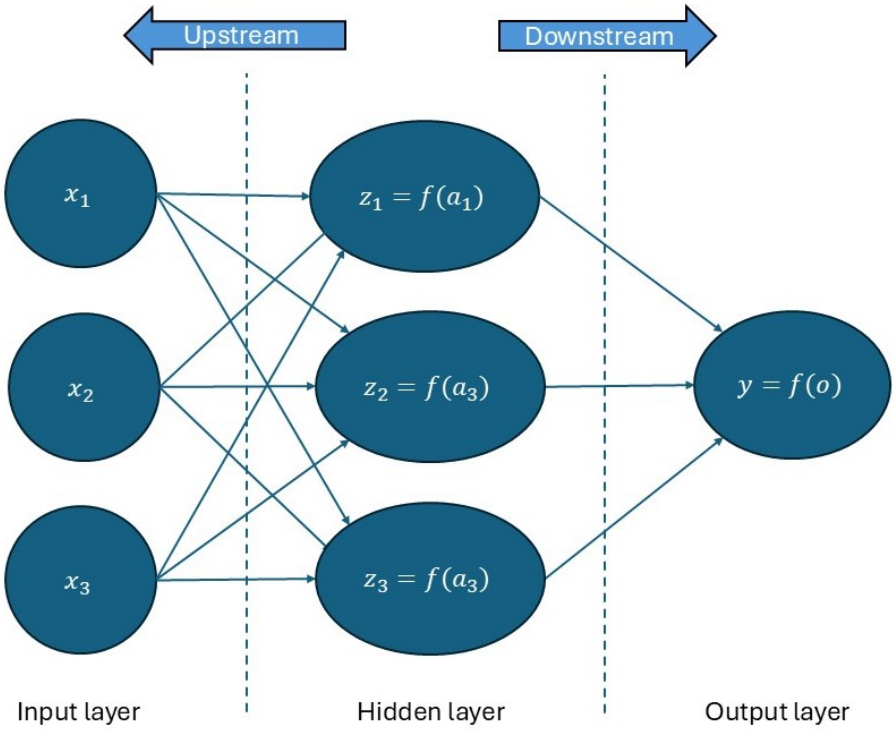


Figure 3.1: A neural network with 1 input layer, 1 hidden layer and 1 output layer. Input- and hidden layers have 3 neurons, while the output layer has a single neuron.

then summed up and passed through the nonlinear activation function f, to form the final output y. It follows that the output y can be described as a nested function of the input $\mathbf{x}$:

$$y = f(\sum_{j=1}^{m} f(a_j) W_j^{(o)}) = f(\sum_{j=1}^{n} f(\sum_{i=1}^{n} x_i W_{i,j}^{(h)}) W_j^{(o)}) = \mathcal{F}(\mathbf{x}, \mathbf{W}) \tag{3.6}$$

The purpose of the activation function is to introduce non-linearity in the mapping. Multiple different functions can be used, such for example rectified linear unit (ReLU), leaky ReLU, sigmoid, or hyperbolic tangent:

$$\text{ReLU}(a) = \max(0, a) \tag{3.7}$$

$$\text{Leaky ReLU}(a) = \begin{cases} a & \text{if } a > 0 \\ \gamma a & \text{if } a \leq 0 \end{cases} \tag{3.8}$$

$$\sigma(a) = \frac{1}{1 + e^{-a}} \tag{3.9}$$

$$\text{Tanh}(a) = \frac{e^a - e^{-a}}{e^a + e^{-a}} \tag{3.10}$$

In principle, any non-linear, differentiable function could be used as activation function. By increasing the number of layers or neurons of the network, the mapping described in Equation 3.6 can approximate highly complex relationships between input $\mathbf{x}$ and output y. This approximation is tuned by changing the weights $\mathbf{W}^{(l)}$ of each layer, to fit the training data as good as possible.

3.2.1. Training

The process of updating the weight matrices $\mathbf{W}^{(l)}$ is referred to as training the NN. In the remainder of this chapter, the weight matrix $\mathbf{W}$ will denote a single matrix that contains all the elements from each $\mathbf{W}^{(l)}$.

Examples of real data $(\mathbf{x}, t)$ are fed to the network to adjust its parameters in a way that it outputs accurate y for an input $\mathbf{x}$. Note that y refers to the model output, while t refers to the true value of the target variable.

Loss Function

For the model to learn the mapping in Equation 3.6, we need a measure to judge how good an output y for a given input $\mathbf{x}$ is. In other words, how close y is to the true target t. This measure is referred to as the loss function L. Many loss functions exist, and the choice depends on the task to be solved by the network. In this project, the output y is a continuous variable. For this kind of task, a common choice of loss function is the mean squared error (MSE), defined in Equation 3.11.

$$\text{MSE}(y, t) = \frac{1}{B} \sum_{i=1}^{b} (y_i - t_i)^2 \tag{3.11}$$

The MSE is the average of the squared distance of the true and generated target over the considered dataset with size B. During training of the network this loss function is minimized. Typically the loss function does have multiple local minima, due to the vast amount of parameters involved.

Backpropagation

To minimize the Loss function L, it is necessary to find the gradients of the loss with respect to all weights of the network $\mathbf{W}$. Once the gradients are known, a step is taken in their opposite direction. The gradient matrix is denoted by $\mathbf{G}$, where each element G_{ij} represents a partial derivative $\frac{\partial L}{\partial W_{ij}}$.

$$\mathbf{G} = \frac{dL}{d\mathbf{W}}$$

These gradients can be calculated using backpropagation. It consists of two main steps: A forward pass and a backward pass.

The forward pass is equivalent to applying Equation 3.6 to an input $\mathbf{x}$. Data flows from left to right in Figure 3.1. Each neurons' activation z and final output y are determined. Then the loss is calculated using for example the MSE as in Equation 3.11.

The backward pass starts at the output layer of the network: First the gradient of the loss with respect to the output is calculated. In the example of a network structure from Figure 3.1, and the MSE loss this breaks down to:

$$\frac{\partial L}{\partial \mathbf{y}} = \frac{\partial}{\partial \mathbf{y}} \left(\frac{1}{B} \sum_{i=1}^{B} (y_i - t_i)^2 \right) = \frac{2}{B} (\mathbf{y} - \mathbf{t}) \tag{3.12}$$

The gradients of the output layer can then be calculated with the chain rule:

$$\frac{\partial L}{\partial \mathbf{W}^{(o)}} = \frac{\partial L}{\partial \mathbf{y}} \frac{\partial \mathbf{y}}{\partial \mathbf{o}} \frac{\partial \mathbf{o}}{\partial \mathbf{W}^{(o)}} \tag{3.13}$$

where:

- $\mathbf{y}$ is the final output for all batches
- o is the input of the layer: A weighted sum of the upstream layers' outputs
- $y = f(z)$ is a nonlinear, differentiable activation function

This process can then be repeated for the upstream layer (in this example, the hidden layer). First, it is necessary to compute the gradient of the loss with respect to this layers' output $\mathbf{z}$:

$$\frac{\partial L}{\partial \mathbf{z}} = \frac{\partial L}{\partial \mathbf{y}} \frac{\partial \mathbf{y}}{\partial \mathbf{z}} \tag{3.14}$$

With this, the gradient of the loss with respect to the layers' weights can be calculated similarly to Equation 3.13. Following this approach, it is possible to calculate the gradient of the loss with respect to all weights of the network.

Optimization Strategies

Calculating all gradients is the first step for optimizing the network. After, a step is taken towards the opposite direction of the gradient, a technique known as gradient descent. All weights are changed according to Equation 3.15

$$\mathbf{W}^{(t+1)} = \mathbf{W}^{(t)} - \eta \cdot \mathbf{G} \tag{3.15}$$

where:

- $\mathbf{W}^{(t)}$ is the weight matrix at iteration t,
- $\mathbf{W}^{(t+1)}$ is the updated weight matrix at the next iteration,
- η is the learning rate,
- $\mathbf{G}$ is the gradient of the loss function L with respect to the weights $\mathbf{W}$.

The process of backpropagating and optimizing must be repeated for many iterations to find the minimum of the loss function. Typically, the network processes the entire dataset multiple times, with each complete pass through the dataset referred to as an epoch.

In the context of neural networks, this is usually done in a stochastic way. That means, for one update step, only a subset of the dataset, known as a batch, is used. This approach has several advantages. Next to better use of computational resources, it can also help reduce the likelihood of getting stuck in local minima.

The previously treated stochastic gradient descent (SGD) forms the basis of many optimization strategies for NNs. In this project, a more advanced technique is applied: To smooth out fluctuations of the gradient, the ADAM optimization algorithm is employed.

For each element $G_{i,j}$ of the gradient matrix, ADAM estimates the mean and variance as in Equation 3.16 and 3.17, respectively. The value of the estimate at time t is the weighted sum of the value of the estimate at time $t-1$ and the value of the gradient at time t. The parameters β_1 and β_2 are referred to as the decay rate. They control the influence of the past estimates on the current estimate.

$$\mu_{G_{ij}}^{(t)} = \beta_1 \cdot \mu_{G_{ij}}^{(t-1)} + (1 - \beta_1) \cdot G_{ij}^{(t)} \tag{3.16}$$

$$\tau_{G_{ij}}^{2\,(t)} = \beta_2 \cdot \tau_{G_{ij}}^{2\,(t-1)} + (1 - \beta_2) \cdot (G_{ij}^{(t)})^2 \tag{3.17}$$

This algorithm is very memory efficient since it requires only two values to estimate the mean and standard deviation of the gradients. However, as the estimates are initialized at zero, there is a bias towards zero during the early stages of training.

To counteract this behavior, the estimates are bias-corrected using Equation 3.18 and 3.19.

$$\hat{\mu}_{G_{ij}}^{(t)} = \frac{\mu_{G_{ij}}^{(t)}}{1 - \beta_1^t} \tag{3.18}$$

$$\hat{\tau}_{G_{ij}}^{2\,(t)} = \frac{\tau_{G_{ij}}^{2\,(t)}}{1-\beta_2^t} \tag{3.19}$$

During early timesteps, this approach effectively removes the bias towards zero of the estimates. The elements of the weight matrix are finally updated with Equation 3.20. A small number δ is added to the denominator to avoid division by zero.

$$W_{ij}^{(t+1)} = W_{ij}^{(t)} - \eta \cdot \frac{\hat{\mu}_{G_{ij}}^{(t)}}{\sqrt{\hat{\tau}_{G_{ij}}^{2\,(t)} + \delta}} \tag{3.20}$$

Overfitting & Regularization

The phenomenon of overfitting can be described as a model fitting too well to the dataset used for training. The model has memorized the patterns in the dataset, instead of approximating the true relationship to be modeled. As a result, it performs well on the data used for training, but fails to generalize to unseen data. Flexible models, such as NNs, can be subjected to overfitting, if not monitored carefully. The more flexible a model is, the higher the risk of overfitting. A commonly used indicator for overfitting is the difference in error of the train-, and validation set. A combination of low train and large validation error is a strong indicator of overfitting. More on this can be found in *Pattern Recognition and Machine Learning* [56].

Regularization techniques are applied to combat overfitting. Several methods are available, which make use of different strategies to improve model performance. The strategies can be categorized as follows:

1. Restrict model flexibility: Methods that penalize the model for large weights.
2. Add randomness or noise: Methods that introduce stochasticity during training, stimulating the model to learn more robust patterns.
3. Early Stopping: To stop the training process when the model starts to overfit.

While this list is not complete, it gives a broad idea of the available methods. For more detailed information on regularization strategies, please refer to *Deep Learning* [55] or *Pattern Recognition and Machine Learning* [56]. In this project, two different regularization techniques are applied, which fall in the second and third category:

- Dropout (Category 2)
- Early Stopping (Category 3)

Dropout is a method that deactivates a portion of the neurons in the network during training. The dropout rate δ determines how big this portion is. During every training step, a set of different neurons is deactivated, which is determined randomly. For each layer, the amount of activated neurons is equal to:

$$N_{\text{active}} = (1-\delta) \times N \tag{3.21}$$

- N_{active}: The number of neurons activated in the layer after applying dropout.
- δ: The dropout rate
- N: The total number of neurons in the layer before dropout.

Early Stopping concludes the training process when the models' performance on the validation set is not improving anymore. By that, it effectively prevents the model from memorizing the training data and combats overfitting. The approach used in this project is discussed in detail in Section 5.1.2.

Normalization

Normalization of the data used for training and inference is crucial for model performance [56]. It tackles the issue of different data ranges and spreads among the in and outputs. Normalization facilitates that every feature can contribute equally to the model's output. Several different normalization strategies exist. The most common ones include:

1. Standardization (Z-score Normalization), which transforms the data to have a mean of zero and unit variance:

$$X_{\text{normalized}} = \frac{X - \mu_X}{\tau_X} \tag{3.22}$$

Where:

- X: The original data value.
- μ_x: The mean of the data.
- τ_X: The standard deviation of the data.

2. Min-Max Normalization, transforms the data into a fixed range [0,1]:

$$X_{\text{normalized}} = \frac{X - X_{\min}}{X_{\max} - X_{\min}} \tag{3.23}$$

Where:

- X: The original data value.
- $X_{\min}$: The minimum value of the data.
- $X_{\max}$: The maximum value of the data.

A more detailed discussion about Normalization strategies can be found in *Deep Learning* by Goodfellow et al. [55].

3.3. Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are a type of NN designed to handle sequential data, such as text or time-series data. In sequential data, the order of the data points matters. In contrast to classical feedforward neural networks, RNNs employ a mechanism to transfer information across timesteps. This makes them better suited for tasks that contain temporal dependencies. RNNs process data sequentially: For a given input tensor $\mathbf{X}$ with dimensionality (T, F), where T represents time steps and F denotes features, the network processes slices $\mathbf{x}$ of size $(1, F)$ one at a time. Consider the RNN in Figure 3.2.

The input Tensor $\mathbf{X}$ has dimensionality $(3, 1)$, meaning it consists of 3 time steps with 1 feature per step:

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_{1,1} \\ \mathbf{x}_{2,1} \\ \mathbf{x}_{3,1} \end{bmatrix}$$

At first, the slice $\mathbf{x}_{1,1}$ is fed into the network. It is multiplied by the weight $W_{1,1}$, which is depicted as an arrow, to form the argument for the activation function in the hidden cell. After the activation function is applied, the hidden cell state z_1 at $T = 1$ is obtained:

$$z_1 = f(W_{1,1}\mathbf{x}_{1,1}) \tag{3.24}$$

Next, the subsequent timestep is fed into the network. It is again multiplied by the weight $W_{1,1}$. However, since the cell state z_1 is nonzero, it contributes to the new cell state z_2. Thus, z_1 is multiplied by the weight $W_{2,1}$. The argument to the activation function f is the sum of these 2 inputs. The new hidden cell z_2 can be computed:

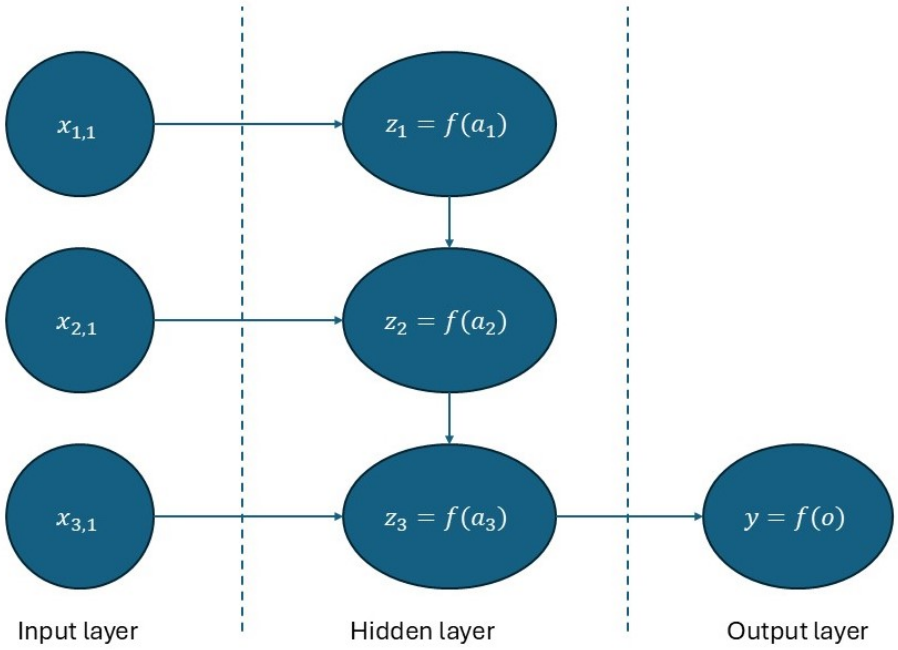


Figure 3.2: A RNN with 1 input, 1 hidden layer and 1 output layer. All layers have a single neuron.

$$z_2 = f(W_{1,1}\mathbf{x}_{2,1} + W_{2,1}z_1) \tag{3.25}$$

Then the process is repeated for the last timestep $T = 3$:

$$z_3 = f(W_{1,1}\mathbf{x}_{3,1} + W_{2,1}z_2) \tag{3.26}$$

To obtain the output y, the last timestep hidden state z_3 is again multiplied by a weight $W_{1,2}$. Finally, the activation function is applied to receive the estimated target y:

$$y = f(W_{1,2}z_3) \tag{3.27}$$

This example describes a sequence-to-1 modeling task since a single target value is modeled using a sequence as an input. It is worth noting that in RNNs the weights $\mathbf{W}$ stay the same across timesteps.

3.3.1. Vanishing and Exploding Gradients

As with a standard feedforward NN, the gradients are a key element necessary to train RNNs. However, due to the recurrent nature of the computations, the gradients can be very unstable, which can be an issue for the training process of the RNN. This is particularly problematic for longer sequences. The problem is illustrated using the previous example depicted in Figure 3.2.

By substituting Equation 3.24 - 3.26 in Equation 3.27, the output y can be described as a nested function of the input tensor $\mathbf{X}$:

$$y = f\left(W_{1,2}f\left(W_{1,1}\mathbf{x}_{3,1} + W_{2,1}f\left(W_{1,1}\mathbf{x}_{2,1} + W_{2,1}f\left(W_{1,1}\mathbf{x}_{1,1}\right)\right)\right)\right) \tag{3.28}$$

The target variable is a continuous variable. Therefore, the MSE loss as in Equation 3.11 is used as loss function. The goal is to find the gradient of the loss with respect to the weights of the network, using backpropagation in time. To focus on the mechanism of vanishing and exploding gradients, the example is demonstrated focusing on a single output y, instead of the whole batch $\mathbf{y}$ as used before. The gradient of the loss L with respect to the output y is:

$$\frac{\partial L}{\partial y} = 2(y - t) \tag{3.29}$$

Using the chain rule, the gradient of the loss with respect to the last weight $W_{1,2}$ can be calculated:

$$\frac{\partial L}{\partial W_{1,2}} = 2(y - t) \cdot f'(W_{1,2} z_3) \cdot z_3 \tag{3.30}$$

To calculate gradients of the hidden layer, the first step is to calculate the gradient of L with respect to z_3:

$$\frac{\partial L}{\partial z_3} = \frac{\partial L}{\partial y} \cdot \frac{\partial y}{\partial z_3} = 2(y - t) \cdot f'(W_{1,2} z_3) \cdot W_{1,2} \tag{3.31}$$

To find the gradient of z_3 with respect to the hidden layers' weights $\mathbf{W}^{(h)}$ the dependencies to $\mathbf{x}_{3,1}$ and z_2 need to be taken into account:

$$\frac{\partial z_3}{\partial W_{1,1}} = f'(z_3) \cdot \mathbf{x}_{3,1} + f'(z_3) \cdot W_{2,1} \cdot \frac{\partial z_2}{\partial W_{1,1}} \tag{3.32}$$

Since z_2 depends on the previous hidden state z_1, a similar expression is obtained:

$$\frac{\partial z_2}{\partial W_{1,1}} = f'(z_2) \cdot \mathbf{x}_{2,1} + f'(z_2) \cdot W_{2,1} \cdot \frac{\partial z_1}{\partial W_{1,1}} \tag{3.33}$$

z_1 is only dependent on the input $\mathbf{x}_{1,1}$:

$$\frac{\partial z_1}{\partial W_{1,1}} = f'(z_1) \cdot \mathbf{x}_{1,1} \tag{3.34}$$

Combining the above it becomes evident that the gradient of the loss with respect to the first weight $W_{1,1}$ can be described with:

$$\frac{\partial L}{\partial W_{1,1}} = \sum_{t=1}^{3} \frac{\partial L}{\partial z_t} \cdot \frac{\partial z_t}{\partial W_{1,1}} \tag{3.35}$$

Since multiple products of weights and derivatives are involved, the value of $\frac{\partial L}{\partial W_{1,1}}$ can easily become very large or very small, which makes the training unstable. The choice of activation function strongly influences the stability, since the value of the gradient enormously depends on the derivative of the activation function. Another disadvantage of a classical RNN is the reduced ability to capture long-term dependencies, which originates from the problem of transporting a gradient signal over long-sequences. For a deeper discussion of these issues, the reader is referred to *Deep Learning* [55].

Note: For simplicity, bias terms are not included in any of the previous equations in this section, as they do not significantly impact the overall concept being demonstrated. However, in practice typically bias terms are added to the argument of the activation function of each neuron. Calculating the output of the hidden layer in Figure 3.1 then becomes:

$$\mathbf{z} = f(\mathbf{W}^{(l)} \mathbf{x} + \mathbf{b})$$

From now on, bias terms will be included to provide a more detailed explanation of the concepts.

3.3.2. Long Short-Term Memory Cells

The previously mentioned drawbacks of RNNs can be effectively diminished by using Long Short-Term Memory (LSTM) cells instead of classical RNN neurons. The LSTM was first introduced by Hochreiter and Schmidhuber in 1997 [57] as a solution to the vanishing gradient problem. Today LSTMs are widely applied, due to their improved memory and training stability.

The basic idea is that the cell is able to keep information over multiple timesteps. To do that, every cell has a cell state $cs^{(t)}$, which acts as a memory of the cell. Note that this is different from the hidden state $z^{(t)}$, which is the value seen by the other cells in the network (the output of the cell). The flow of information in the cell is regulated by 4 mechanisms:

- Input Node $IN^{(t)}$: Calculates a new candidate cell state $\tilde{cs}^{(t)}$.
- Forget Gate $FG^{(t)}$: Determines how much of the old cell state $cs^{(t-1)}$ to keep.
- Input Gate $IG^{(t)}$: Determines how much of the new candidate cell state $\tilde{cs}^{(t)}$ should influence the new cell state $cs^{(t)}$.
- Output Gate $OG^{(t)}$: Determines to what extent the new cell state influences the new hidden state $z^{(t)}$.

The value of these 4 parameters depends on the hidden state of the previous timestep $\mathbf{z}^{(t-1)}$ and the input $\mathbf{x}^{(t)}$ at the current timestep. A weighted sum of those values is passed through an activation function:

$$\tilde{cs}^{(t)} = \tanh\left(\mathbf{W}_c\mathbf{x}^{(t)} + \mathbf{U}_c\mathbf{z}^{(t-1)} + \mathbf{b}_c\right) \tag{3.36}$$

$$FG^{(t)} = \sigma\left(\mathbf{W}_{FG}\mathbf{x}^{(t)} + \mathbf{U}_{FG}\mathbf{z}^{(t-1)} + \mathbf{b}_{FG}\right) \tag{3.37}$$

$$IG^{(t)} = \sigma\left(\mathbf{W}_{IG}\mathbf{x}^{(t)} + \mathbf{U}_{IG}\mathbf{z}^{(t-1)} + \mathbf{b}_{IG}\right) \tag{3.38}$$

$$OG^{(t)} = \sigma\left(\mathbf{W}_{OG}\mathbf{x}^{(t)} + \mathbf{U}_{OG}\mathbf{z}^{(t-1)} + \mathbf{b}_{OG}\right) \tag{3.39}$$

where:

- $\mathbf{W}$ and $\mathbf{U}$ are the weight matrices for the input and the previous hidden state, respectively.
- $\mathbf{b}$ represents the bias terms for each gate and the candidate cell state.
- $\sigma()$ is the sigmoid activation function
- $\tanh()$ in the hyperbolic tangent activation function

All gates' values are in the range $[0, 1]$, due to use of the sigmoid activation function σ. In contrast, the candidate cell state is in the range $[-1, 1]$, as here the hyperbolic tangent is used. The next step is to calculate the value of the new cell state, and new hidden state. This is done according to Equation 3.40 and 3.41. In this process, the gates function as regulators, determining the amount of new information to integrate into the memory, the extent of forgetting previous information, and the fraction of memory to output to the network. The flow of information in an LSTM cell is summarized in Figure 3.3. During training, the weights are adjusted to retain and output important information, while forgetting irrelevant parts.

$$cs^{(t)} = FG^{(t)} \cdot cs^{(t-1)} + IG^{(t)} \cdot \tilde{cs}^{(t)} \tag{3.40}$$

$$z^{(t)} = OG^{(t)} \cdot \tanh(cs^{(t)}) \tag{3.41}$$

Using these mechanisms, data can be passed on over multiple time steps, if the Forget and Input Gates stay close to 1 and 0, respectively. This helps to counteract the vanishing gradient problem and improves the network's capabilities to learn long-term dependencies.

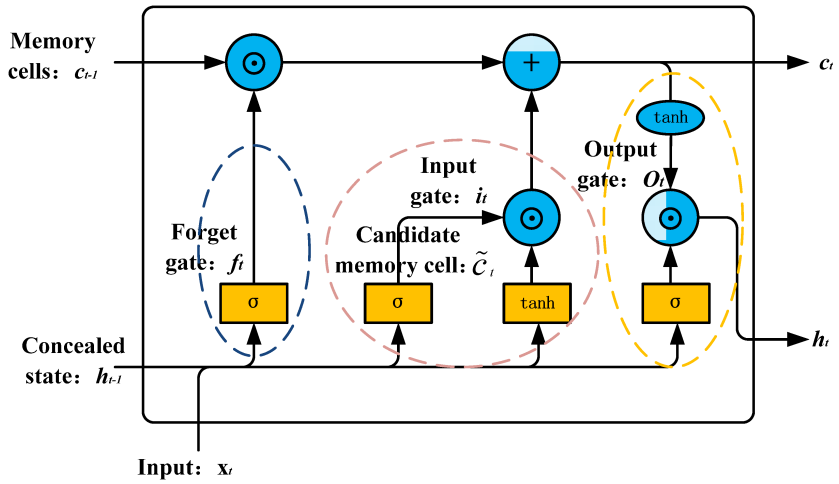


Figure 3.3: Illustration of an LSTM cell with gate mechanisms. Adopted from: *Based on the Improved PSO-TPA-LSTM Model Chaotic Time Series Prediction* by Cai et al. [58]. Licensed under CC BY.

3.4. Generative Adversarial Networks

A Generative Adversarial Network (GAN) is a system of two competing networks, first introduced by Goodfellow et al. in 2014 [59]. It consists of a generator and a discriminator, which work against each other. The generator is trained to produce realistic samples of the target distribution. In contrast, the discriminator is trained to distinguish real from generated samples.

A setup of a GAN is depicted in Figure 3.4. The generator receives a random noise vector $\mathbf{r}$ as input. It outputs a fake sample y_G. This sample is then fed into the discriminator, which judges whether the sample is real or not. The discriminator output y_D is thus a scalar value, that can be interpreted as the probability of y_G being real. This output is used by the generator to update its weights.

3.4.1. Training of GANs

To provide valuable feedback, the discriminator must first learn how to distinguish real from fake data. For that, it is provided with both, real and fake samples. The objective is twofold:

1. To maximize the output for the real sample.
2. To minimize the output for the fake sample.

This objective is represented in the adversarial loss of the discriminator, L_{AD}:

$$L_{AD} = -\mathbb{E}_{\mathbf{t}\sim p_{\text{data}}}[\log D(\mathbf{q}=\mathbf{t})] - \mathbb{E}_{\mathbf{r}\sim p_r}[\log(1 - D(\mathbf{q}=G(\mathbf{r})))] \tag{3.42}$$

where:

- $D(\mathbf{q}=\mathbf{t})$ is the probability assigned by the discriminator that a real data sample is real.
- $D(\mathbf{q}=G(\mathbf{r}))$ is the probability assigned by the discriminator that a generated sample $G(\mathbf{r})$ is real.
- $\mathbf{t} \sim p_{\text{data}}$ denotes samples drawn from the real data distribution p_{data}.
- $\mathbf{r} \sim p_r$ represents samples drawn from the noise distribution p_r, used as input to the generator.
- $\mathbb{E}_{\mathbf{t}\sim p_{\text{data}}}$ and $\mathbb{E}_{\mathbf{r}\sim p_r}$ denote the expectations over real data and noise distributions, respectively.

Expectations in this context are approximated by taking the average over a batch of training data. Next, the training of the generator is illustrated. The process starts by generating a fake sample. This sample

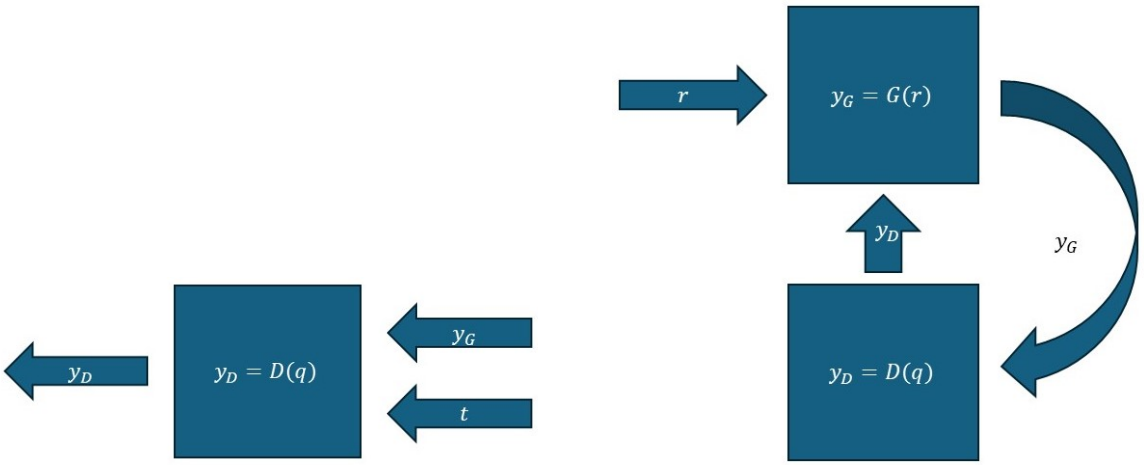
is then evaluated by the discriminator: if the discriminator assigns a low probability, the generator is penalized. The loss function, L_{AG}, is expressed as:

$$L_{AG} = -\mathbb{E}_{\mathbf{r}\sim p_r}[\log D(\mathbf{q} = G(\mathbf{r}))] \tag{3.43}$$

where:

- $G(\mathbf{r})$ is the generated data sample from the generator given noise input $\mathbf{r}$.
- $D(\mathbf{q} = G(\mathbf{r}))$ is the probability assigned by the discriminator that the generated sample $G(\mathbf{r})$ is real.
- $\mathbf{r} \sim p_r$ represents samples drawn from the noise distribution p_r, used as input to the generator.
- $\mathbb{E}_{\mathbf{r}\sim p_r}$ denotes the expectation over the noise distribution.

Again, the expectation is approximated by averaging over a batch of training data. The 2-step process described above is visualized in Figure 3.4. It is repeated for many iterations. When trained properly, the generator has learned to reproduce the distribution of real data, while the discriminator has learned to distinguish real and fake distributions [59]. The Equations 3.42 and 3.43 together form the so-called adversarial loss. By definition, the improvement of one of the networks implies a degradation of its opponent. This effect, amongst others, makes the training process of a GAN highly unstable.



(a) Training the Discriminator **(b)** Training the Generator

Figure 3.4: Classical GAN training: discriminator and generator

Challenges in Training GANs

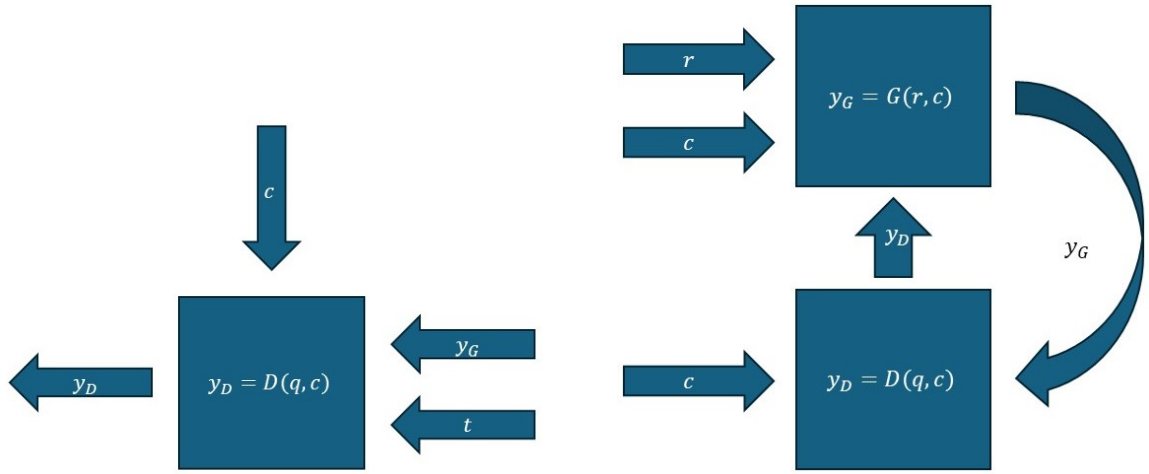
Due to the adversarial nature of the loss function, training is very unstable. Convergence to the global minimum of the loss function is not straightforward when working with GANs. When not properly balanced, one of the two networks can become stronger than the other, leading to vanishing gradients for its opponent. A result of that is stagnation of the training process [60]. Another common issue in GANs is mode collapse. When mode collapse happens, the generator produces a limited number of outputs, instead of exploring the full distribution of the real data. To address mode collapse, a discriminator with strong generalization capabilities is essential [60].

Another well-known challenge in GAN training is to determine when to stop the training process. Contrary to traditional NNs, the value of the loss function can not directly be used to assess the performance of the model. As a result, a low adversarial loss does not automatically imply a high quality of the generated samples. Next to that, the loss is typically not steadily decreasing. Instead, it shows oscillating or cyclic behavior [60]. The question of when to conclude the GAN training is an ongoing area of research. State-of the art methods suggest that the model should be evaluated using additional tools, rather than relying on the loss function [61]. There are two broad categories of model evaluation: quantitative evaluation and qualitative evaluation. The latter typically involves human inspection of the generated data

and assessing the quality. This method is time-consuming and subjective. A quantitative evaluation involves applying some distance metric. Examples of that are the Inception Score [62] and the Fréchet Inception Score [63], which were designed to be applied to image data. An alteration of the inception score that can be applied to non-imagery data is the Confidence and Diversity Score [61]. Nazari et al. [61] provide a detailed overview of the state-of-the-art regarding GAN evaluation and propose a solution for improvement, named AutoGan. In AutoGan, the quality of the generated samples is evaluated after each iteration and the algorithm concludes training when the model is not improving anymore. The user still has to define how to measure an improvement. A similar method is used to stop the training process in this project.

3.4.2. Conditional GANs

Conditional GANs (cGANs) generate diverse samples, taking into account conditional information. They were introduced in 2014 by Mirza et al. [64]. The generator and discriminator are provided with additional information, $\mathbf{c}$, which is included in the generation and discrimination process. Figure 3.5 shows a setup of a cGAN.



(a) Training the conditional Discriminator **(b)** Training the conditional Generator

Figure 3.5: Conditional GAN training: discriminator and generator

Extending the concepts discussed before, the loss function of the discriminator becomes:

$$L_{AD} = -\mathbb{E}_{\mathbf{t}\sim p_{\text{data}}}[\log D(\mathbf{q} = \mathbf{t} \mid \mathbf{c})] - \mathbb{E}_{\mathbf{r}\sim p_r}[\log(1 - D(\mathbf{q} = G(\mathbf{r}, \mathbf{c}) \mid \mathbf{c}))] \tag{3.44}$$

Similarly, for the generator:

$$L_{AG} = -\mathbb{E}_{\mathbf{r}\sim p_r}[\log D(\mathbf{q} = G(\mathbf{r}, \mathbf{c}) \mid \mathbf{c})] \tag{3.45}$$

A properly trained generator can approximate the distribution $p(\mathbf{t}|\mathbf{c})$, aiming to produce samples that match the real data conditioned on $\mathbf{c}$:

$$p(G(\mathbf{r}, \mathbf{c}) \mid \mathbf{c}) \approx p(\mathbf{t} \mid \mathbf{c}) \tag{3.46}$$

In contrast, the discriminator is trained to discover discrepancies in Equation 3.46, allowing it to distinguish between the real and generated conditional distributions.

3.5. Statistical Theorems and Measurement Metrics

In addition to ML techniques, some basic statistical methods and measurement metrics are used in this project. These are necessary to assess the performance of the developed models and to post-process the model output to perform anomaly detection. This section provides background information on hypothesis testing and the Wasserstein distance.

3.5.1. Hypothesis Testing

Hypothesis testing is a method to evaluate a certain statement based on the evidence in the data [65]. For that two hypotheses are defined:

- Null Hypothesis (H_0): The default assumptions, usually suggesting that there is no difference
- Alternative Hypothesis (H_1): The assumption that there is a difference. In other words: what is tried to be proven.

In the context of this project, hypothesis testing is used to evaluate if an observation s is drawn from a known distribution v. The hypotheses can be defined as:

- H_0: The observed sample is drawn from the known distribution v.
- H_1: The observed sample is not related to the distribution v.

The next step is to look for evidence in the data, that allows the null hypothesis to be rejected. For that, a significance level α needs to be defined. This is a measure of the maximum allowable risk taken for falsely rejecting the null hypothesis. A test statistic is computed using the data. Finally, the p-value of the test statistic can be calculated, which represents the probability of observing the test statistic under the assumption that the null hypothesis is true.

Since H_0 typically involves assuming a known distribution, the knowledge about this distribution and the test statistics can be used to determine the p-value. When the p-value falls below the pre-defined significance level α, H_0 is rejected. The risk of falsely rejecting H_0 is then equal to the p-value.

3.5.2. Wasserstein Distance

The Wasserstein distance, also referred to as the Earthmover distance, can be used to measure similarity of two probability distributions. The formal definition is given by:

$$W_1(P,Q) = \inf_{\gamma \in \Pi(P,Q)} \int |x-y| \, \mathrm{d}\gamma(x,y) \tag{3.47}$$

- $W_1(P,Q)$: The L1 Wasserstein distance between distributions $P(x)$ and $Q(y)$.
- $|x-y|$: The L1 distance to transport mass from x to y.
- $\gamma(x,y)$: A coupling that specifies how much mass is transported from point x in P to point y in Q.
- $\Pi(P,Q)$: The set of all possible couplings of P and Q.
- $\inf$: The infimum over all couplings γ of P and Q.

Consider the following illustrative example: There are two probability distributions, P, and Q, which are functions of x and y, respectively. The Wasserstein distance is the lowest possible work to transform Q into P. Work in this context refers to both the amount of mass and the distance over which it is moved. In this specific example, the distance moved is measured with the L1 distance. To find the Wasserstein distance, all possible ways of transforming Q to P are evaluated. These are described by the set of couplings $\Pi(P,Q)$. So each coupling describes a certain way of transforming one distribution into the other one. For all of these couplings, the work to transform Q to P is calculated. Finally, the infimum work of all possible ways of transformation equals the Wasserstein distance. It can be interpreted as the most efficient way of transforming Q into P.

A large value of the Wasserstein distance means a lot of work has to be done for the transformation, i.e. the distributions are different. In contrast, a small value indicates that the considered distributions are similar.

In this project, the Wasserstein distance is calculated between empirical, 1-dimensional distributions. It then simplifies to Equation 3.48:

$$W_1(P,Q) = \frac{1}{m} \sum_{i=1}^{m} \left| x_{(i)} - y_{(i)} \right| \tag{3.48}$$

- $W_1(P,Q)$: The L1 Wasserstein distance between empirical distributions P and Q.

- m: The number of samples in each distribution (assuming equal sample sizes for simplicity).
- $x_{(i)}$ and $y_{(i)}$: The i-th ordered sample from distributions P and Q, respectively.
- $\left|x_{(i)} - y_{(i)}\right|$: The L1-distance of paired samples

In case of unequal sample sizes, the sample size of the smaller set can be increased by interpolation. For more detailed information on the Wasserstein distance, please refer to *Optimal Transport* by Villani [66].

3.6. Agglomerative Hierarchical Clustering

Clustering methods are used to group data points based on similarity. In contrast to the previously described ML methods, clustering is typically unsupervised. This means that the true relationship in the data is unknown to the user. Using clustering algorithms, natural groupings in the data can be uncovered. The algorithms mainly differ by:

- The order to group data points.
- How to measure similarity.

For a detailed discussion of clustering methods, please refer to *Pattern Recognition and Machine Learning* by Bishop [56]. In this project, the agglomerative hierarchical clustering method is to group datapoints in similar groups, with the goal to assess the model performance in different operational regimes. Therefore, this technique is now described in detail.

Initializing the algorithm, each data point starts on its own, forming a single point cluster. Then the two most similar clusters are merged. How to define similarity is up to the user. Next again, the algorithm determines the two most similar clusters and merges them. This iterative process continues until all data points are merged to form a single cluster. The algorithm also keeps track of the distance (based on the defined similarity metric) that needs to be overcome to merge clusters. This information is visualized in a dendrogram, see Figure 3.6. The Figure displays the merging operations on 6 samples: at first, the samples in the middle are merged, followed by the right two samples. Next, the samples on the left are merged. In the fourth iteration, the two left clusters are grouped to form one cluster. Finally, in the last step, all data points are merged.

An advantage of the method is that the number of clusters does not need to be defined beforehand. Using the dendrogram, it is possible to decide on a suitable number. The dendrogram should be cut at a point where a lot of distance needs to be overcome for the next merging operation. A possible choice for that is indicated in Figure 3.6 by the dashed line. Note that this choice is always subjective and depends on the user.

The last tool needed to apply the method is to define the similarity, also called the linkage criterion. Common options include [56]:

- Single Linkage: The distance between the closest points of two clusters.
- Complete Linkage: The distance between the farthest points of two clusters.
- Average Linkage: The average distance between all pairs of points from the two clusters.
- Ward Linkage: The distance is measured by minimizing the increase in total within-cluster variance.

In this project, the Ward linkage method is used. The method aims to reduce the sum of all in-cluster variances. For a single cluster, the in-cluster variance IV is computed as:

$$IV = \sum_{i=1}^{N} \|\mathbf{d}_i - \boldsymbol{\mu}\|^2 \tag{3.49}$$

Where:

- $\mathbf{d}_i$ represents a data point in the cluster.
- $\boldsymbol{\mu}$ represents the centroid (mean) of the cluster.

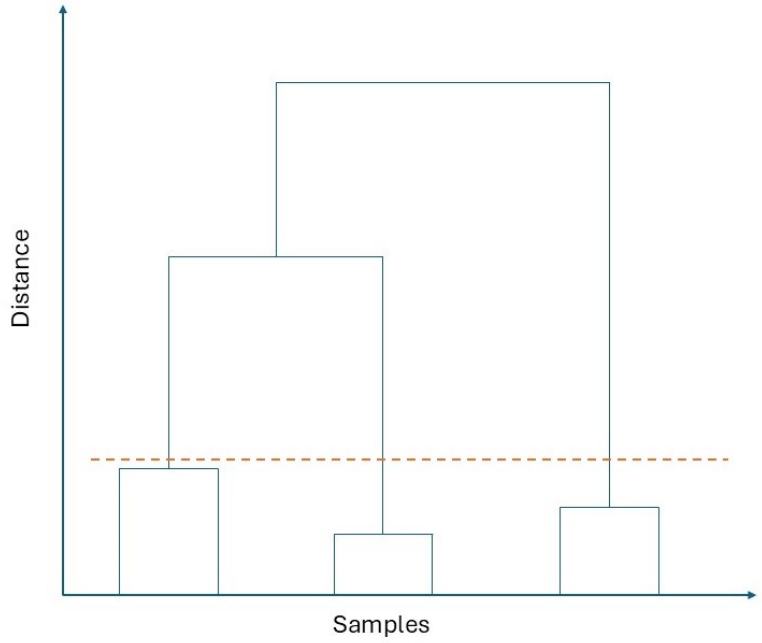


Figure 3.6: Agglomerative Hierarchical Clustering: dendrogram of merging operations. The dashed line indicates a possible choice to cut the dendrogram.

- $\|\mathbf{d}_i - \boldsymbol{\mu}\|^2$ is the squared Euclidean distance between each data point and the cluster centroid.
- N is the amount of data points in a cluster.

At every merging step, the algorithm computes the variance increase for all possible cluster pairs, denoted by d_W. Finally, the clusters are merged which minimizes the increase of in-cluster variance.

$$d_W(A, B) = \Delta IV_{A,B} = IV_{A \cup B} - (IV_A + IV_B) \tag{3.50}$$

Where:

- $IV_{A \cup B}$: The in-cluster variance for the newly merged cluster $A \cup B$.
- IV_A: The in-cluster variance within cluster A.
- IV_B: The in-cluster variance within cluster B.

The method favors compact, spherical clusters with minimum in-cluster variance [56]. The number of computations for each merging step is proportional to the square of the number of clusters at that time. As a result, the total computational complexity is proportional to the cube of the number of initial data points. Therefore, agglomerative hierarchical clustering is only suitable for small to medium-sized datasets.

4 Data

ML models are used to uncover, or approximate relationships that are described by data. It follows that the results are heavily dependent on the dataset, and preprocessing steps taken before using the data in the model. This Chapter summarizes important characteristics of the EDP dataset and highlights the most relevant preprocessing steps applied before using the data in the model. The choice of features and targets is discussed as well as the splitting of the dataset into train, validation, and test sets.

4.1. General Dataset Information

The data used for this project is the open-access SCADA dataset provided by Energias de Portugal (EDP) [24]. It contains data from 5 offshore WTs located in the West African Gulf of Guinea [67]. The data has been logged in the years 2016 and 2017. Next to the SCADA dataset, EDP also provided SCADA alarm log files, failure log files, and meteorological measurements of a closely located mast (metmast). Table 4.1 summarizes the different types of data provided by EDP.

Table 4.1: Description of the content of the EDP dataset.

File	Variables	Size	Sample Frequency
SCADA signals	Rotor RPM, active power, gearbox oil temperature, etc.	521784	10 min
Metmast signals	Air Pressure, humidity, wind speed, etc.	87528	10 min
Alarm logs	Alarms of values exceeding a threshold, operator intervention, settings, etc.	318835	Random
Failure logs	Major faults and errors, failures, corrective maintenance actions	28	Random

Next to those files, EDP also provided technical information about the wind turbines and their power curves. The WT characteristics are summarized in Table 4.2.

4.2. Failures

The dataset is a good source of drivetrain failure information, as it contains 17 failure logs related to generator, generator bearing, and gearbox. The remaining 11 failures are associated with the hydraulic system and transformer. Table 4.3 describes the drivetrain failures. While this project is focused on drivetrain monitoring, failures of the other systems are still relevant for the model setup. Therefore, Table 4.4 summarizes the hydraulic and transformer issues in the dataset. Note that both tables also

Table 4.2: EDP Wind turbine specifications

Rated power (kW)	2 000
Cut-in wind speed (m/s)	4
Rated wind speed (m/s)	12
Cut-out wind speed (m/s)	25
Diameter (m)	90
Number of blades	3
Rotor speed, max (rpm)	14.9
Tip speed (m/s)	70
Gearbox type	Planetary/spur
Stages	3
Generator type	Asynchronous
Speed, max (rpm)	2 016
Hub height	80 m

indicate if the failure is in the validation or test set. This is discussed in more detail in Section 4.4.4, when introducing the train, validation, and test split.

4.3. Features & Targets

The selection of features and targets influences the detection capabilities of the model. While the choice of the targets strongly impacts the types of faults to be detectable, the features need to have a deep relationship with the target for accurate target prediction. Note, that not all failure modes of a certain component can be detected with the same target [68]. Furthermore, the challenge of detection grows as multiple components are involved, each requiring different signals to be monitored. Hence, multiple targets likely increase the detection capabilities to provide a better understanding of the turbine's health. However, as this report is focused on analyzing the impact of probabilistic target variable modeling, the target is chosen to be a single signal: the generator bearing 1 temperature. Based on predicting the healthy generator bearing 1 temperature, it is expected that in particular failures of that specific generator bearing are detectable. Nevertheless, the sensitivity concerning other components is also analyzed. While it is likely that generator bearing 1 is the bearing on the generator drive-end, this is not specified in the information provided by EDP. For the remainder of the report, generator bearing 1 is referred to as generator bearing, unless stated otherwise.

Next, appropriate features for predicting the generator bearing temperature need to be identified. This was done by expert knowledge and by taking into account the physical mechanisms that contribute to a temperature change in the bearing. To avoid mispredictions due to using anomalous input data, only 'primary operational signals' were considered as features. These are signals such as generator speed, ambient temperature, or active power output. In contrast, signals such as the generator bearing - or gearbox oil temperature can be categorized as 'secondary operational signals'. These signals are dependent on the primary operational signals. However, an exception was made for the nacelle temperature. Although typically classified as a secondary operational signal, nacelle temperature was included as a potential feature because, even in the event of a fault, the generated heat in a faulty bearing would have minimal effect on the nacelle temperature. This is due to the large thermal inertia of the air mass and the other components within the nacelle, which buffer the impact.

The generator bearing temperature is a function of generated heat, and lost heat. In this context, the generator speed and active power output are used to model the heat source, as they reflect the rotational speed and load on the bearing, respectively. These factors directly contribute to the heat generated within the bearing. For modeling heat loss, ambient temperature and wind speed are included, as they influence the forced convection cooling of the entire nacelle. Last, also the nacelle temperature is included as a feature because it plays a crucial role in the first stage of heat transfer within the nacelle. The heat generated by the bearing is transferred to the surrounding air mass and other components within the nacelle, a process primarily governed by conduction. As the bearing is directly connected to the generator, one can argue that the generator temperature is a key factor influencing the heat conduction to or from the bearing. This is also concluded by other researchers who model the generator

Table 4.3: Drivetrain failure logs grouped by component and Turbine ID.

Component	Turbine	Date	Remarks	Validation Set	Test Set
Generator	T06	2016-07-11	Generator replaced	x	
	T06	2016-07-24	Generator temperature sensor failure	x	
	T06	2016-09-04	High temperature generator error	x	
	T06	2016-10-02	Refrigeration system and temperature sensors in generator replaced	x	
	T06	2016-10-27	Generator replaced	x	
	T07	2017-08-21	Generator damaged		x
	T11	2016-03-03	Electric circuit error in generator		x
Generator Bearing	T07	2016-04-30	High temperature in Generator bearing (replaced sensor)	x	
	T07	2017-08-20	Generator bearings damaged		x
	T09	2016-06-07	High temperature generator bearing		x
	T09	2016-08-22	High temperature generator bearing		x
	T09	2016-10-17	Generator bearings replaced		x
	T09	2017-01-25	Generator bearings replaced	x	
Gearbox	T01	2016-07-18	Gearbox pump damaged		x
	T06	2017-10-17	Gearbox bearings damaged	x	
	T09	2016-10-11	Gearbox repaired		x
	T09	2017-10-18	Gearbox noise	x	

Table 4.4: Failure logs for Transformer and Hydraulic Group grouped by component and Turbine ID.

Component	Turbine	Date	Remarks	Validation Set	Test Set
Transformer	T01	2017-08-11	Transformer fan damaged		x
	T07	2016-07-10	High temperature transformer	x	
	T07	2016-08-23	Transformer refrigeration repaired	x	
Hydraulic Group	T06	2016-04-04	Error in pitch regulation	x	
	T06	2017-08-19	Oil leakage in Hub	x	
	T07	2017-06-17	Oil leakage in Hub		x
	T07	2017-10-19	Oil leakage in Hub		x
	T09	2017-09-16	Pitch position error related GH	x	
	T11	2016-10-17	Hydraulic group error in the brake circuit		x
	T11	2017-04-26	Hydraulic group error in the brake circuit		x
	T11	2017-09-12	Hydraulic group error in the brake circuit	x	

bearing temperature [69]. However, due to the previous considerations about primary and secondary operational signals, the generator temperature is not used as a feature in this project.

While more sophisticated feature selection procedures are available, using expert knowledge is a common approach in the wind domain. For instance, domain knowledge was also used by Latiffianti et al. [52] and Jankauskas et al. [37]. Both approaches utilize the EDP dataset and rely on a similar set of features to predict component temperatures.

4.4. Data Preprocessing

ML model performance is sensitive to data preprocessing. Not only data quality but also the value range and handling of missing values can have a big influence on the results. Possible measures range from manual inspection of data, missing values, etc up to sophisticated statistical methods. In the case of NBM, it is important to identify erroneous data points as well as abnormal operating conditions. The latter are excluded from training the NBM, whereas the incorrect datapoints are corrected or deleted. Barber et al. [51] have conducted an extensive study comparing different ML methods working with the EDP dataset. Most of those methods apply very limited filtering to the dataset, with the idea of preserving as much information as possible. Minding this approach, the data pre-processing steps are as follows:

1. Filtering the data for non-physical values.
2. Resampling to a lower temporal resolution and identifying remaining temporal gaps in the data.
3. Splitting the data into healthy and faulty instances.

4. Splitting the data into train, validation, and test sets.
5. Normalizing the data.

4.4.1. Filtering for Non-Physical Values

The selected features are checked to fall within a reasonable data range. The upper and lower limits of all variables are summarized in Table 4.5. Data points that fall out of this range are assumed to be caused by sensor errors or data logging errors. Therefore, they are corrected by setting the value to the upper or lower operational limit, depending on the original value being too large or too small.

Table 4.5: Operational limits for features and targets.

Variable	Range
Generator speed	0 - 2016 rpm
Active power out	0 - 2000 kW
Nacelle temperature	0 - 50 °C
Ambient wind speed	0 - 50 m/s
Ambient temperature	-5 - 40 °C
Generator bearing temperature	0 - 200 °C

4.4.2. Resampling & Data Gaps

ML models can be sensitive to noisy data. To reduce noise and highlight longer-term effects, the dataset is resampled to 1h frequency. Values are aggregated using the mean of the considered interval. Another advantage of resampling to a lower temporal resolution is that it reduces the apparent density of missing data points. This improves the continuity of the dataset, which is particularly important for ML models that process sequential data, such as the methods used in this project.

Nevertheless, some gaps in the data remain after resampling. The remaining gaps are handled as follows:

- Gaps smaller than 4 timesteps are ignored: The sequence is considered to be intact and can be used as input.
- Sequences that contain gaps equal to or larger than 4 timesteps are considered discontinuous: These sequences can not be used as input to the model.

4.4.3. Healthy & Faulty Instances

Before splitting the data into train, validation, and test sets, the data must be categorized into healthy and faulty data. The time window around a failure, in which a turbine is considered faulty varies depending on the failure type, as summarized in Table 4.6. A turbine is considered to be faulty for the entire period starting at the time before failure until the time after failure.

The window for drivetrain failures might seem large on first impression. The choice for that originates from Latiffianti et al. [52], who have been able to detect gearbox faults 89 days prior to failure in the EDP dataset. For mechanical components, it is not unusual that degradation happens slowly, over a large time window.

While the time windows defined in Table 4.6 have been applied to most turbines, Turbine 9 requires special treatment: after applying these limits, the maximum value of the generator bearing temperature in healthy conditions for Turbines 1,6,7 and 11 is below 100 °C. However, Turbine 9 exceeds this value frequently in the first half of the dataset, likely due to an ongoing generator bearing issue. Therefore, for Turbine 9, the entire year 2016 has been categorized as unhealthy.

Table 4.6: Time windows used to categorize the data into healthy and faulty instances around various failure types.

Component	Time before failure	Time after failure
Gearbox	95 days	7 days
Generator	95 days	7 days
Generator bearing	95 days	7 days
Hydraulic	30 days	7 days
Transformer	30 days	7 days

4.4.4. Train, Validation & Test Split

Once healthy and faulty data points are categorized, the dataset can be split into train, test, and validation sets using a ratio of 70 %, 15 %, and 15 %, respectively. To ensure a balanced split, the data is split manually, adhering to the following conditions:

- The data is split for each turbine individually. That means, for every turbine the required ratios hold with minimal deviation.
- Validation and train sets contain data from all seasons.
- A period assigned to validation or train set can not exceed 50 days, to ensure diverse contributions from different seasons.
- At least two independent periods are assigned to both the validation and test sets. Turbine 9 is an exception from that, due to the low number of healthy data points.

The split of the dataset in time is visualized in Section 6.3.2 when presenting the results of the investigated methods. A detailed numerical description of validation and test sets is given in Tables 4.7 and 4.8. The train data is the remaining data that has not been assigned to either the validation or test set, consisting of 35697 data points. When referring to healthy validation, or test data, this implies the healthy parts of the data set, as described in the Tables 4.7 and 4.8.

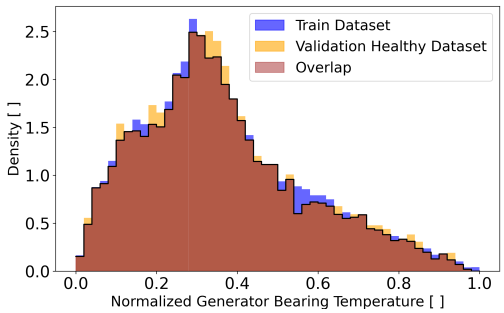
Table 4.7: Description of the content of the validation dataset.

Category	Start Date	End Date	Turbine	Total Dataset Size
Healthy	2016-09-10 2017-05-01 2017-11-23	2016-09-30 2017-05-31 2017-12-31	Turbine 1	7827 datapoints
	2016-01-01 2017-05-01	2016-01-31 2017-05-27	Turbine 6	
	2017-03-01 2016-05-08	2017-03-31 2016-06-09	Turbine 7	
	2017-11-27	2017-12-31	Turbine 9	
	2016-08-01 2017-01-01	2016-09-10 2017-02-10	Turbine 11	
Unhealthy	2016-03-05 2017-07-14	2016-11-03 2017-10-24	Turbine 6	20448 datapoints
	2016-01-26 2016-06-10	2016-05-07 2016-08-30	Turbine 7	
	2016-11-30 2017-07-15	2017-02-01 2017-10-25	Turbine 9	
	2017-08-13	2017-09-19	Turbine 11	

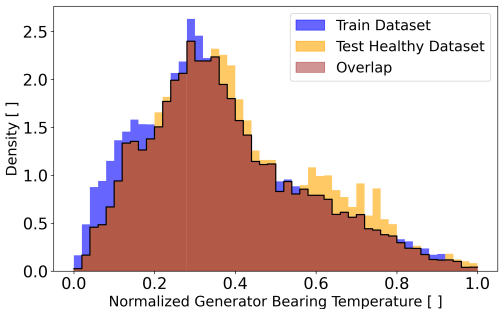
Table 4.8: Description of the content of the test dataset.

Category	Start Date	End Date	Turbine	Total Dataset Size
Healthy	2016-10-01 2017-06-01 2017-10-01	2016-10-09 2017-06-30 2017-11-19	Turbine 1	7754 datapoints
	2016-02-01 2017-04-01	2016-02-28 2017-04-30	Turbine 6	
	2017-04-01 2017-11-30	2017-04-30 2017-12-31	Turbine 7	
	2017-04-01	2017-05-05	Turbine 9	
	2017-11-21 2017-06-21	2017-12-31 2017-07-31	Turbine 11	
Unhealthy	2016-04-14 2017-07-12	2016-07-24 2017-08-18	Turbine 1	14983 datapoints
	2017-05-17 2017-09-19	2017-08-28 2017-10-26	Turbine 7	
	2016-01-01	2016-10-31	Turbine 9	
	2016-01-01 2016-09-17 2017-03-27	2016-03-10 2016-10-24 2017-05-03	Turbine 11	

The split of the healthy data points is assessed by comparing the target distributions of the healthy validation and test sets to the train set. As presented in Figure 4.1, the healthy validation set accurately represents the train set. By evaluating the model on the healthy validation set, the model's generalization performance is assessed. In Chapter 5, it will become clear how the healthy validation set is used to determine the best-performing model during training. In contrast, the distribution of the healthy test set differs more from the train set. Therefore, the model's performance on the test set can be used as an indicator of robustness, which will be assessed in Chapter 6.



(a) Distribution comparison of the healthy validation and train set.



(b) Distribution comparison of the healthy test and train set.

Figure 4.1: Target distribution comparison between train, validation, and test sets.

4.4.5. Normalization

As discussed in Chapter 3, NNs benefit from normalizing the features and targets. In this project, min-max normalization is applied to both features and targets. Each turbine is normalized individually, highlighting changes concerning its own normal operating conditions. The train set is normalized first. Subsequently, validation and test sets are normalized using the same scaling parameters applied to the train set.

5 Method

This chapter describes the methodology used in this project. First, in Section 5.1, the cGAN structure, training, and temperature modeling setup are described. Next, Section 5.2 explains how the cGAN output is processed to derive the Health Indicators, which are used for anomaly detection. It follows a brief description of the base model in Section 5.3. Finally, Section 5.4 illustrates how the Health Indicators are used to perform anomaly detection. Also, the approach to compare the cGAN with the base model is described in this section. An overview of the methodology is provided in Figure 5.6. This overview does not cover all details, however, it is used for general illustration of the concepts.

5.1. Normal Temperature Modeling with a cGAN

The approach of using a cGAN for regression tasks has been demonstrated in the literature, for example by Xiao et al. [70] or Jobson et al. [23]. Inspired by these approaches, the temperature modeling framework of this project has been designed.

5.1.1. cGAN Structure

The exact structure of the cGAN is developed based on trial and error. Many experiments with different structures have been performed to develop a well-working model setup. The starting point for the experiments was the research conducted by Jankauskas et al. [37], who used a network with two LSTM layers consisting of 128 and 100 cells, respectively. The generator and discriminator both share the concept of using two layers of LSTMs in their hidden layers. Apart from that, there are some significant differences to be highlighted in this section.

As mentioned in Chapter 3, the input for the generator is a sequence of feature vectors (the condition $\mathbf{c}$) and a sequence of noise vectors $\mathbf{r}$. The sequence length S was chosen to be 4 timesteps, which is equivalent to 4h of data. This is close to the optimal setting in the experiments performed by Jankauskas et al [37], who use a time window of 3h. However, in this project, performance benefits could be observed by increasing the sequence length slightly to 4h. Both $\mathbf{c}$ and $\mathbf{r}$, have the same dimensionality of 5×4. Hence, the combined input dimensionality of the generator becomes 10×4.

Using this information, the generator generates a target value that corresponds to the last timestep of the input sequence (the fake target). It has thus dimensionality 1. Figure 5.1 displays the generator's structure. In this diagram, inputs are indicated in gray, while outputs are marked in green. Hidden layers and operators are shown in blue and orange, respectively.

The input to the network consists of the condition, $\mathbf{c}$, and noise vectors,$\mathbf{r}$, which are concatenated along the feature dimension. This combined input is passed to the LSTM layers. Each LSTM cell in a hidden layer receives all signals from the previous layer to calculate the cell state and gates as described in Chapter 3. After one timestep is evaluated, the network receives the subsequent timestep for processing. When the last timestep is processed, the network has seen the full sequence. The second LSTM layer's output of the last timestep, $\mathbf{z}^{(2)}_{S=4}$, is transformed by a linear layer to generate the

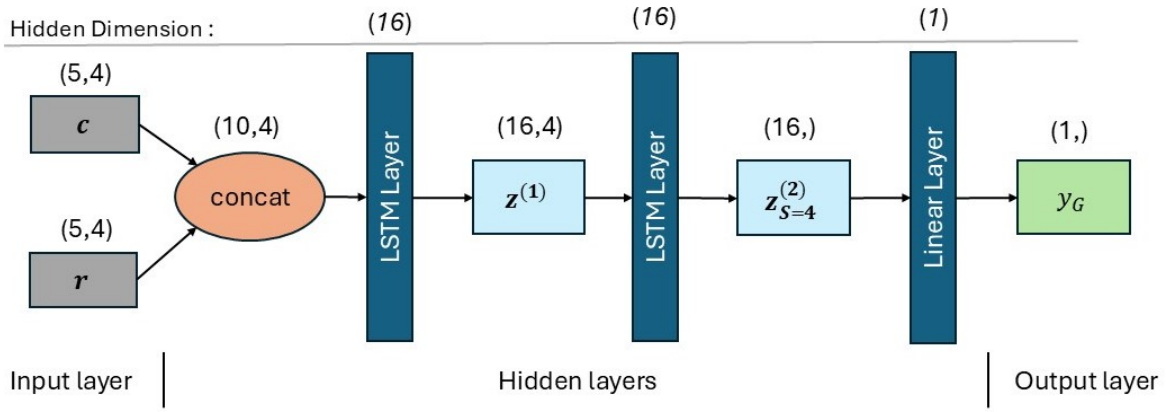


Figure 5.1: The generator's architecture. The non-italicized values in between brackets refer to the dimension of the data flowing through the network. The italicized values in between brackets define the dimension of the hidden layers.

network output, y_G.

The discriminator input consists of the condition $\mathbf{c}$, and a generator bearing temperature value q. Its task is to distinguish whether this temperature fits the condition, or in other words, if it is a real value from the dataset or a fake one generated by the generator. The discriminator's architecture is shown in Figure 5.2. Again, the in- and outputs are depicted in gray and green, respectively, while hidden layers are shown in blue and operators are represented in orange.

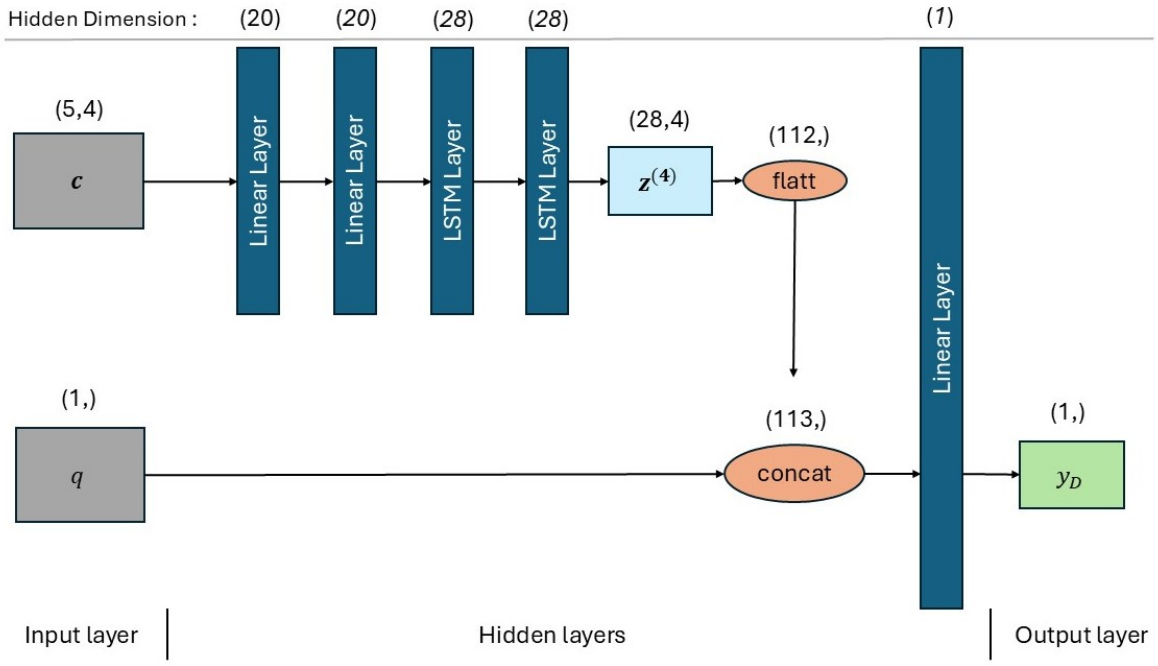


Figure 5.2: The discriminator's architecture. The non-italicized values in between brackets refer to the dimension of the data flowing through the network. The italicized values in between brackets define the dimension of the hidden layers.

The input condition is processed by two linear layers, before it is passed on to the LSTM layers. In the first two linear layers, a leaky ReLu activation function is used, as described in Equation 3.8. The tuning parameter of the leaky ReLu activation function, γ, is set to 0.2, which has been determined by trial and error. These linear layers extract features within each timestep, i.e. there is no transportation of information over timesteps. Next, the extracted features are passed on to the LSTM layers. For generating the final output, the discriminator considers the hidden cell outputs of the second LSTM layer $\mathbf{z}^{(2)}$ over all timesteps. These are stacked in a single 1-dimensional vector of length $H \cdot S$, where

H is the hidden dimension of the LSTM layer, and concatenated with the target value q. This vector is transformed in another linear layer to generate a scalar network output. The linear layer applies a sigmoid activation function. Due to the sigmoid activation function, output values range between 0 and 1, and can be interpreted as the probability that the target value is a real one:

$$D(q|\mathbf{c}) = P(q = real|\mathbf{c}) \tag{5.1}$$

5.1.2. Training the cGAN

After preparing the data and defining the model, the cGAN can be trained. This section discusses the setting of the loss function, regularization and training hyperparameters. It also discusses the methodology used to determine when to stop the training process.

Loss Function

Both the generator and discriminator are trained with the adversarial loss described in Equations 3.44 and 3.45. This loss function penalizes the generator for not fooling the discriminator, which means the discriminator is allocating a low probability for the generated samples. The discriminator is penalized for wrongly classifying the real and fake samples, i.e. for allocating a low probability to real samples, and a high probability to fake samples. This concept is illustrated in row two in Figure 5.6.

Training Hyperparameters

The model training is governed by the setting of several hyperparameters. Finding a stable and effective setting requires many trials and careful tuning of the parameters. GANs are known to be very sensitive to this tuning [71]. Two additional measures are taken to realize a stable training process:

1. The discriminator is trained in multiple steps per iteration, denoted as D_{steps}. This involves generating multiple outputs y_G for the same condition and evaluating the discriminator on these outputs, and the real targets (see Figure 3.5a).
2. The learning rate of the discriminator and generator is decaying exponentially after every epoch. For that, the learning rates are multiplied by a factor δ_D and δ_G, respectively.

Dropout is activated during training on the first hidden layer of both networks. The ADAM optimization algorithm is applied with smoothing parameters β_1 and β_2 for the first and second moment estimates, respectively. The setting of training hyperparameters is summarized in Table 5.1.

Table 5.1: Training hyperparameter settings.

	Symbol	Value
Batch size	B	64
Learning rate (generator)	η_G	0.0015
Decay rate (generator)	δ_G	0.94
Learning rate (discriminator)	η_D	0.0012
Decay rate (discriminator)	δ_D	0.9
First moment decay rate	β_1	0.5
Second moment decay rate	β_2	0.999
Number of epochs	-	100
Discriminator updates per iteration	D_{steps}	3
Dropout rate	δ	0.2

Due to the vast amount of parameters, it is almost impossible to find the optimal solution by manual tuning. For that, more sophisticated methods such as the Hyperopt optimization algorithm [72] can be used. However, the proposed settings yielded the best results that could be achieved by manual tuning. The metric used to assess the quality of the results is introduced in the remainder of this subsection.

Concluding the Training Progress of the cGAN

When is the training completed? As mentioned in Chapter 3, this question is typically not easy to answer for GANs, due to the adversarial nature of the loss function. Inspired by the solution proposed by Nazari et al. [61], the cGAN training progress is evaluated based on two metrics, which are calculated at the end of every epoch:

1. The accuracy of the generator output is measured by comparing the median of its outputs to the real target value. For this comparison, the MSE is used as in Equation 3.11.
2. The diversity of the generator output is measured using the standard deviation of its outputs.

By resampling the generator for a fixed condition while changing the noise vector, a set of outputs $\mathbf{Y}$ can be obtained, as in Equation 5.2. From this set, the median and standard deviation of the generator output are estimated.

$$\mathbf{Y} = \{y_G^{(i)}\}_{i=1}^{N} = \{G(\mathbf{c}, \mathbf{r}_i)\}_{i=1}^{N} \tag{5.2}$$

where:

- $\mathbf{Y}$: Set of generator outputs for a fixed condition $\mathbf{c}$.
- $y_G^{(i)}$: The i-th output of the generator.
- $G(\mathbf{c}, \mathbf{r}_i)$: Generator function, where $\mathbf{c}$ is the fixed condition and $\mathbf{r}_i$ is the i-th noise vector.
- $\mathbf{c}$: Fixed condition input to the generator.
- $\mathbf{r}_i$: Noise vector sampled for the i-th output.
- N: Total number of samples generated by the generator.

After an epoch of training, the generator accuracy and diversity are evaluated on the train and healthy validation dataset. The statistical properties of the generated output are estimated by generating a set $\mathbf{Y}$ of N = 100 samples. According to the Law of Large Numbers (LLN), the accuracy of this estimation is increasing for larger sample sizes. However, the increased benefit in accuracy comes at the cost of computational expenses. A size of 100 samples was chosen to balance these expenses with the gain of increased accuracy. From this set, the median and standard deviation are estimated. In the remainder of this project, the median serves as a point estimate for the target value, similar to classical regression setups. The MSE of point estimate and the real target value, is used as the metric to assess the accuracy of the generator. See row three in Figure 5.6 for a visualization of this method.

Training proceeds over 100 epochs. After every epoch, performance metrics over train and healthy validation data are calculated and the weights are saved. Finally, the model with the lowest total error and sufficient diversity is chosen as the optimal solution. The contributions of the healthy validation set and train set are weighted equally to minimize the risk of overfitting:

$$E = \frac{E_t + E_{v-h}}{2} \tag{5.3}$$

Where:

- E is the total error.
- E_t is the training error.
- E_{v-h} is the healthy validation error.

Similarly, the standard deviations of train and validation data are summarized:

$$\tau = \frac{\tau_t + \tau_{v-h}}{2} \tag{5.4}$$

Where:

- τ is the total standard deviation.

- τ_t is the standard deviation of the training data.
- τ_{v-h} is the standard deviation of the healthy validation data.

Sufficient diversity is defined as the total standard deviation being larger than the error:

$$\tau >= E \tag{5.5}$$

To further reduce the possibility of overfitting, the optimal solution is subjected to an additional constraint that both errors should be of a similar scale:

$$1/2 <= \frac{E_{v-h}}{E_t} <= 2$$

5.1.3. Validating the cGAN output

During the training process, the models' point estimate and diversity is continuously evaluated. When the training is completed, the final model is validated in more detail. For that, consider again the modelling goal of the generator from Equation 3.46:

$$p(G(\mathbf{r}, \mathbf{c}) \mid \mathbf{c}) \approx p(\mathbf{t} \mid \mathbf{c})$$

Ideally, the generator output should be validated by showing that Equation 3.46 results in a small error and the output approximates the real conditional distribution. However, many data points with exactly the same input sequence are necessary to approximate the real distribution accurately. It follows that a simplification of the problem is necessary to compare the generated distribution to the real data distribution.

This problem is addressed by grouping the real data into clusters of similar operating conditions. The clustering and subsequent comparison of the distributions is performed using the healthy validation data. A strong simplification of the problem is to only consider the last timestep of a sequence for the clustering. If more timesteps were considered, it would be much harder to find similar sequences. The simplification is motivated by considering that the last timestep of the sequence contains the most valuable information as it likely has the biggest impact on the target value.

The clusters are determined using Agglomerative Hierarchical Clustering and the Ward linkage method as described in Section 3.6. The dendrogram is presented in Figure 5.3.

Cutting the dendrogram in Figure 5.3 at a distance of 12 results in 5 clusters. The properties of the whole healthy validation dataset and the five clusters are visualized in radar plots in Figure 5.4. The visualization aims to describe the mean and spread of the features across different clusters. A blue cross indicates the mean, while a red dot indicates the standard deviation. The dot size is proportional to the standard deviation of the respective feature. Figure 5.4 a) indicates that Ambient Temperature, Nacelle Temperature, and Wind Speed have relatively low spread in the full healthy validation dataset. In contrast, Active Power and Generator RPM have much higher spread. Cluster 1, depicted in Figure 5.4 b), is characterized by low Active Power, Generator RPM, and Wind Speed. The distributions of Nacelle Temperature and Ambient Temperature are similar to those of the full healthy validation dataset. Cluster 2 consists of datapoints with rather high values in all five features, particularly Generator RPM and Active Power, which is indicated in Figure 5.4 c). Cluster 3, visualized in Figure 5.4 d), is formed by datapoints that share medium Generator RPM and low values in the remaining four features. Cluster 4 is summarized by high Generator RPM and elevated Active Power. As indicated in Figure 5.4 e), the remaining three features have a similar mean as the full dataset, while showing significantly reduced spread. Finally, Cluster 5 is characterized by high Ambient Temperature, combined with low Wind Speed and Active Power. Nacelle Temperature and Generator RPM are typically higher than in the full healthy validation dataset. This is illustrated in Figure 5.4 f). Appendix A provides the numerical description of the clusters. The cGAN is validated by comparing the generated target distribution with the real target distribution for the whole data set, and the individual clusters.

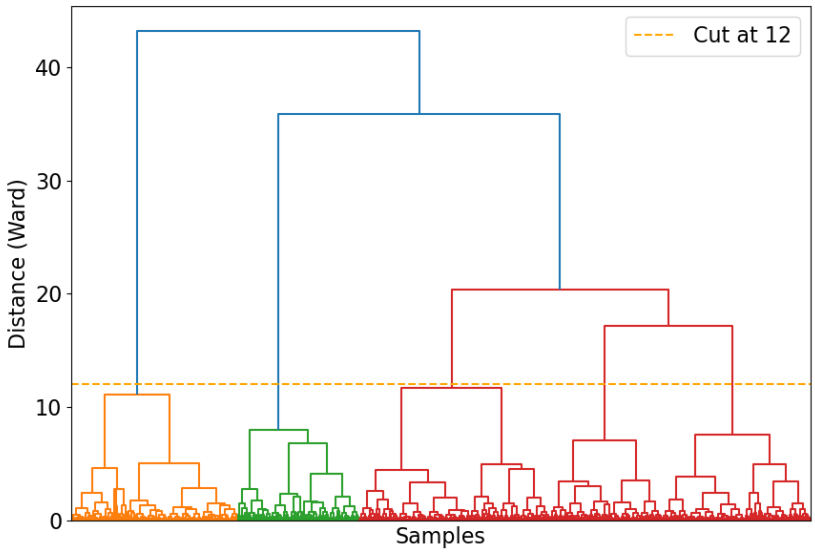


Figure 5.3: The dendrogram obtained through Agglomerative Hierarchical Clustering of the healthy validation data. It is cut at a distance of 12.

5.2. Deriving Health Indicators from the cGAN Output

The goal of this project is to detect faults in the wind turbine that lead to failure in the near future, by probabilistic target variable modeling. This section discusses how the output of the cGAN is used to quantify the anomaly of an observation.

Assuming the generator is properly trained and Equation 3.46 is approximated with a small error, the generator's outputs reflect possible target temperature values given an input condition. It follows that sampling the generator space as in Equation 5.2 is equivalent to sampling the real conditional target distribution $p(t/\mathbf{c})$:

$$\mathbf{Y} = \{y_G^{(i)}\}_{i=1}^{N} = \{G(\mathbf{c}, \mathbf{r}_i)\}_{i=1}^{N} \sim p(t|\mathbf{c}) \tag{5.6}$$

By training the generator in this way, its output can be used to approximate the real conditional target distribution. In the remainder of this section, two possible ways of determining a Health Indicator, HI, from the set $\mathbf{Y}$ are discussed:

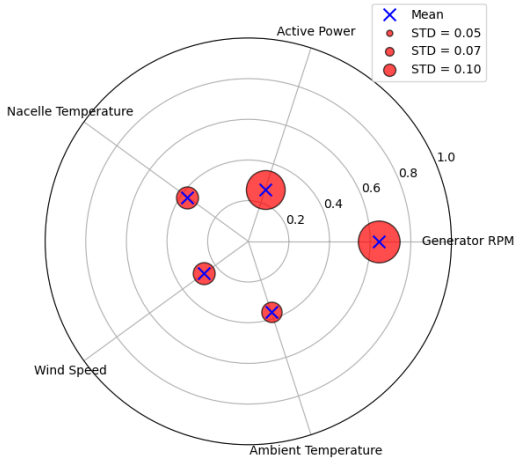
1. By comparing the real temperature to a point estimate, which is derived from the set $\mathbf{Y}$.
2. By taking into account the relative position of the real temperature in the set $\mathbf{Y}$.

5.2.1. Point Estimate Health Indicator

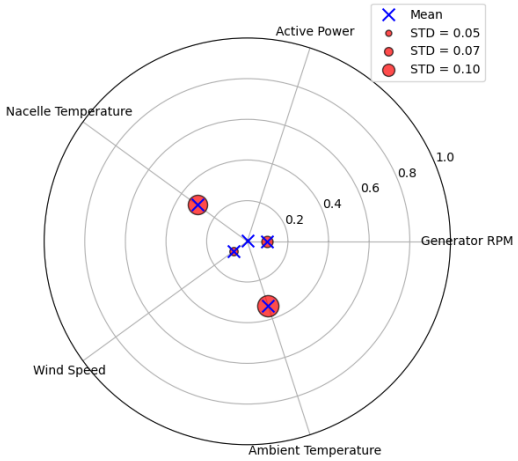
The Point Estimate Health Indicator (PEHI) is derived by reducing the generator output to a point estimate. This reduction essentially reduces the character of probabilistic modeling, as the information provided in the conditional distribution is summarized in a point estimate. However, it still leverages a probabilistic training approach.

First, the sampled set is generated using Equation 5.6, Next, the median from this set is determined to be used as the point estimate, $\hat{t}$, of the target value :

$$\hat{t} = \text{median}(\mathbf{Y}) \tag{5.7}$$

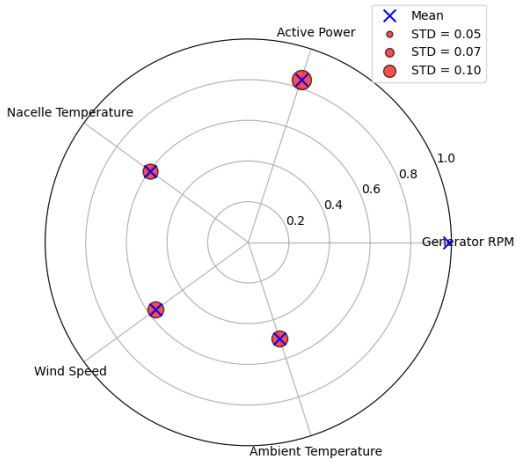


(a) Radar plot for the full healthy validation data set.

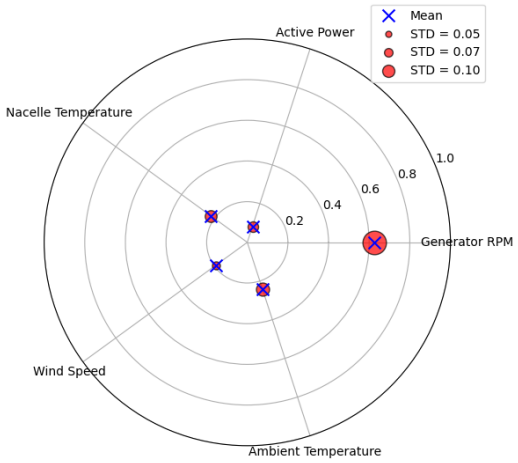


(b) Radar plot for Cluster 1. Dataset size = 2305 samples

Figure 5.4: Radar plots for healthy validation dataset and clusters. The size of the red dots is proportional to the standard deviation of a certain feature.



(c) Radar plot for Cluster 2. Dataset size = 1773 samples



(d) Radar plot for Cluster 3. Dataset size = 3001 samples

Figure 5.4: Radar plots for healthy validation dataset and clusters. The size of the red dots is proportional to the standard deviation of a certain feature.

After determining $\hat{t}$, the residual with the real value t is used to quantify how much the turbine's operation condition deviates from the expected state.

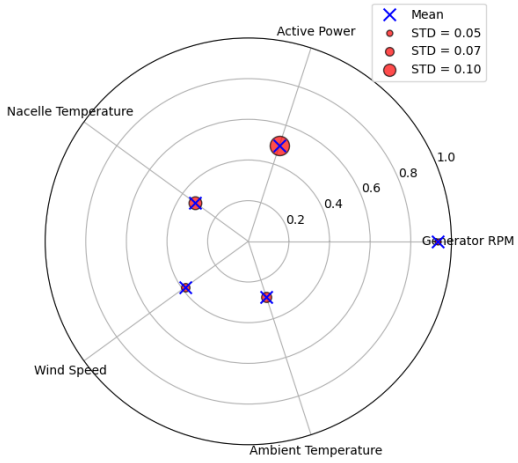
$$R_{PE} = t - \hat{t} \tag{5.8}$$

Since the differences concerning healthy operating conditions are to be highlighted, the residuals are standardized. Each turbine is standardized with its healthy mean and healthy standard deviation.

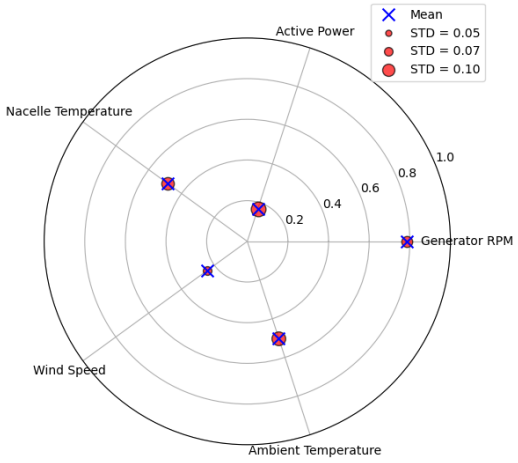
$$HI_{PE} = \frac{R_{PE} - \mu_{R_{PE},H}}{\tau_{R_{PE},H}} \tag{5.9}$$

Where:

- HI_{PE} is the turbine specific PEHI.
- $\mu_{R,H}$ and $\tau_{R,H}$ are turbine-specific healthy mean and standard deviation of the residual, respectively.



(e) Radar plot for Cluster 4. Dataset size = 1706 samples



(f) Radar plot for Cluster 5. Dataset size = 2008 samples

Figure 5.4: Radar plots for healthy validation dataset and clusters. The size of the red dots is proportional to the standard deviation of a certain feature.

The standardized residual is the Point Estimate Health Indicator HI_{PE}. The derivation of the PEHI is illustrated in rows four and five in Figure 5.6.

5.2.2. Distributional Health Indicator

The idea behind a Distributional Health Indicator (DHI) is to better account for uncertainty in the estimated target temperature. Instead of reducing the generator output to a single value, this Health Indicator leverages the full information provided in the conditional distribution of Equation 5.6. Consequently, not only the training process but also the model outputs are of a probabilistic nature.

The starting point is also a sampled set of target temperatures, as described in Equation 5.6. Recalling the assumption that this set is sampled from the true healthy distribution $p(t/\mathbf{c})$, it is possible to compare the real temperature to the distribution of healthy temperatures.

Since $p(t/\mathbf{c})$ is not necessarily normally distributed, this comparison is done using the robust Z-score, which is well suited for problems involving outliers and non-normal data [73].

$$Z_r = \frac{t - \hat{t}}{\text{MAD}(\mathbf{Y})} \tag{5.10}$$

Where:

- Z_r: The robust Z-score of the observation t.
- t: The real target temperature.
- $\hat{t}$: The median of the sampled set.
- MAD$(\mathbf{Y})$: The Median Absolute Deviation of the sampled set.

The Median Absolute Deviation (MAD) of the sampled set $\mathbf{Y}$ is given by:

$$\text{MAD}(\mathbf{Y}) = \text{median}\left(\left|y_G^{(i)} - \hat{t}\right|\right), \quad i = 1, 2, \ldots, N \tag{5.11}$$

Similarly as before, also the robust Z-score of the observation is standardized. Again, each turbine is standardized individually with its own healthy mean $\mu_{Z_r,H}$ and healthy std $\tau_{Z_r,H}$:

$$HI_D = \frac{Z_r - \mu_{Z_r,H}}{\tau_{Z_r,H}} \tag{5.12}$$

Where:

- HI_D is the turbine-specific DHI.
- $\mu_{Z_r,H}$ and $\tau_{Z_r,H}$ are turbine-specific healthy mean and standard deviation of the robust z-score, respectively.

The standardized robust Z-Score is used as the Distributional Health Indicator HI_D. See row four and five in Figure 5.6 for a derivation of the DHI.

5.3. The Base Model

The goal of the report is to compare probabilistic target variable modeling with a deterministic model setup. In the latter, the problem is approached in a classical regression setting: for a certain input $\mathbf{c}$ the model output is a single deterministic value y_B, contrary to the distribution that is modeled with a cGAN, similar to the concepts explained in Section 3.3. The base model's purpose is to examine the impact of a probabilistic modeling setup compared to a traditional deterministic approach. Therefore, it was chosen to approach the problem with a network that is very similar to the generator. Most parts of the generator's architecture, and hyperparameters, are kept to allow for a fair comparison that is focused on the modeling approach. The architecture of the base model is depicted in Figure 5.5. Only the input layer is different compared to the generator's architecture, as the base model does not receive a noise vector as input. Hidden- and output layers are kept the same as in the generator.

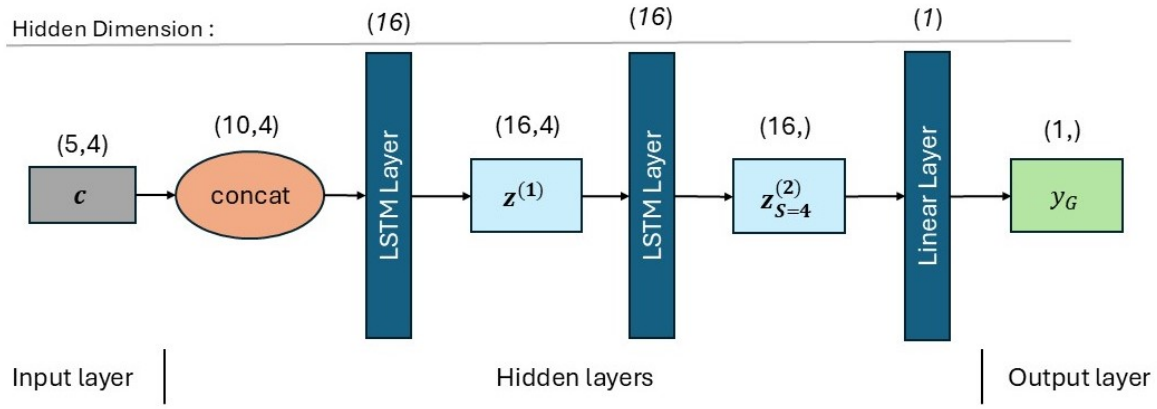


Figure 5.5: The base model's architecture. Data is flowing from left to right.

The second major change compared to the generator setup is the loss function: A standard MSE loss is used to train the base model as described in Equation 3.11:

$$L_{BM} = \text{MSE} \tag{5.13}$$

Note that the discriminator is not used in the base model, as its only purpose is to provide feedback to the generator. Apart from these changes, the base model is trained in the same way as the cGAN.

5.3.1. Base Model Health Indicator

The Base Model Health Indicator (BHI) is derived similarly to the PEHI. Since the base models' output is a single value, the residual can directly be calculated:

$$R_B = t - y_B \tag{5.14}$$

The residual is standadized with its healthy mean $\mu_{R_B,H}$ and standard deviation $\tau_{R_B,H}$ to highlight differences concerning normal operating conditions, resulting in the BHI, HI_B:

$$HI_B = \frac{R_B - \mu_{R_B,H}}{\tau_{R_B,H}} \tag{5.15}$$

5.4. Anomaly Detection & Model Comparison

This section is dedicated to detecting anomalies using the Health Indicators of the cGAN, and the base model. A classical approach to identifying anomalies in any signal is by detecting outliers in the signal. This can be achieved by standadizing the signal and then analyzing it in terms of standard deviations. It is a common method, which has not only been applied to the EDP dataset [51], but is also used in many other domains [74]. This approach typically assumes that the healthy data is normally distributed, which allows for direct use of the standardized signal in the standard normal cumulative distribution function.

While this approach works well for signals that are approximately normally distributed, it may struggle with signals that do not follow a normal distribution. As will be demonstrated in Chapter 6, the Health Indicators derived from the base model and cGAN are not exactly normally distributed. This is particularly true for the DHI.

To address the non-normally distributed Health Indicators, the hypothesis test is formulated without any assumptions about the underlying data distribution. Instead of assuming a normal distribution, the empirical distribution of the healthy data points is used to identify outliers.

Every new observation of HI can be tested, considering the following hypotheses:

- H_0: The new observed health indicator HI belongs to the empirical distribution v_{healthy} of healthy data:
$$H_0 : HI \sim v_{\text{healthy}}$$
- H_1: The new health indicator HI does not belong to the empirical distribution v_{healthy} of healthy data:
$$H_1 : HI \not\sim v_{\text{healthy}}$$

The test statistic used for this is the empirical percentile rank of the observed new Health Indicator HI in the distribution of v_{healthy}. From the significance level, α, and the dataset size of healthy data, a threshold for the rank to reject H_0 can be determined. As this test only considers positive deviations from the empirical distribution $v_{healthy}$, it is referred to as a one-sided p-test.

Setting α is not straightforward, and the choice has strong implications for the sensitivity of the anomaly detection method. Several techniques to determine the appropriate significance level are discussed in the literature, for example, optimizing α to reduce maintenance costs associated with the test result of the anomaly detection method. However, as this project aims to compare probabilistic vs deterministic target variable modeling, α is set to a pre-defined level. From this level, a threshold for the Health Indicators is derived, taking into account the empirical healthy distribution. Additionally, two more parameters are introduced to balance the sensitivity of the algorithm with the need to minimize False Positives (FPs): A positive test is triggered only when the threshold is exceeded for a number of times T_S in a considered time window T_W. Applying this approach reduces the sensitivity of the algorithm for single observations, thereby emphasizing the detection of consistently anomalous conditions. These two parameters, T_S and T_A should also be optimized to maximize the reward associated with true positives (TPs), and minimize the losses associated with FPs and false negatives (FNs). However, this is challenging as the true causal relationship between a test result and the machine condition is not known in detail. Therefore, optimization of these parameters is not performed and static settings are defined:

- α = 0.5 %
- T_S = 6
- T_W = 7

This step, together with the most important steps of the entire method are summarized in Figure 5.6. Note that not all details are included in the Figure. The train and healthy validation sets are abbreviated as T and VH, respectively.

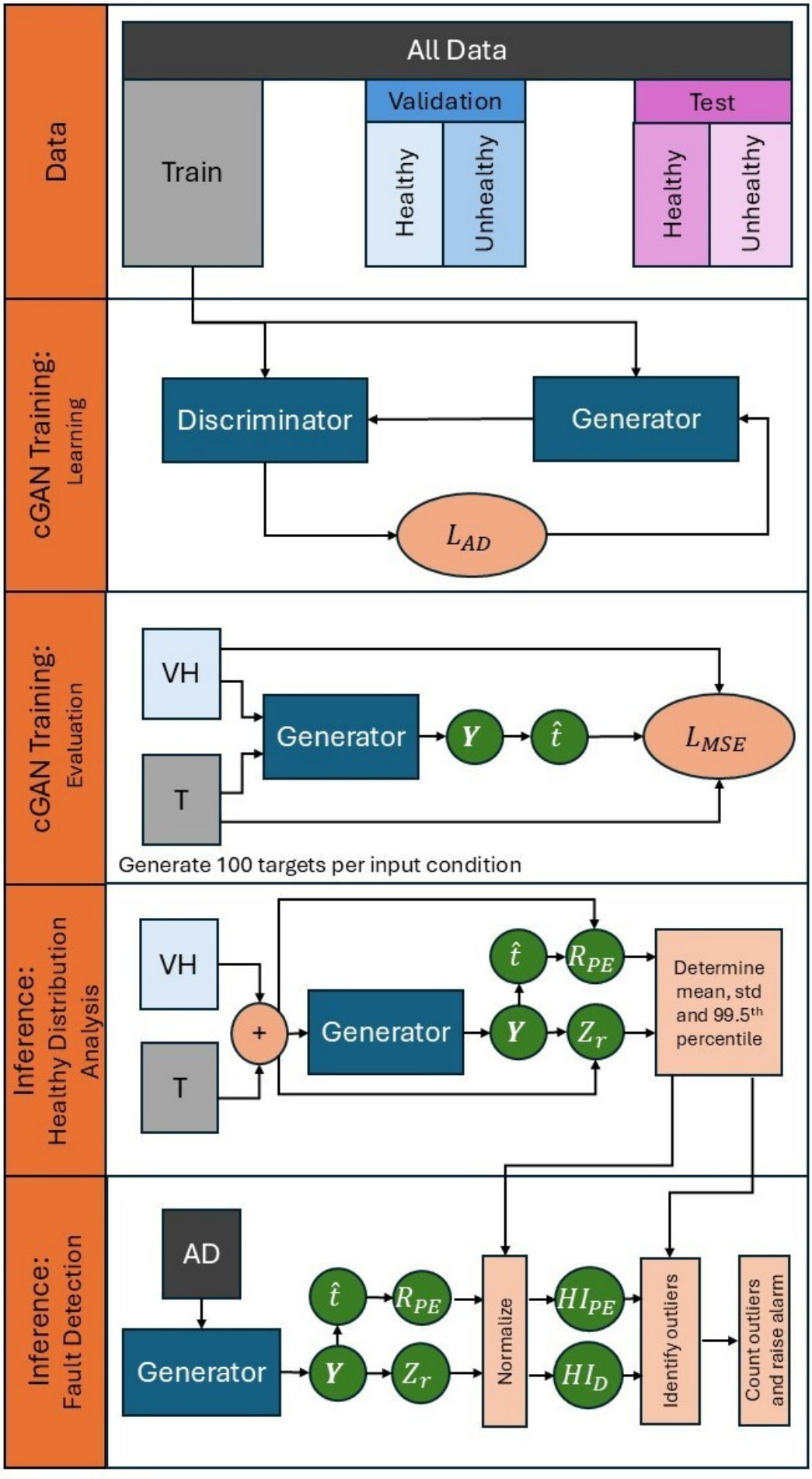


Figure 5.6: Overview of the cGAN modeling approach developed in this project.

6

Results

This chapter presents the results obtained from implementing the probabilistic and deterministic models. Section 6.1 focuses on the results from validating the cGAN, while Section 6.2 briefly shows the results of validating the base model. Section 6.3 compares both models by looking into the fault sensitivity of the three Health Indicators, and applying them to the entire 2 years of data.

6.1. Validation of the cGAN

This section validates the cGAN output using the healthy data from the train and validation sets. As the model is trained to learn the healthy data distribution, the model output, and real healthy data must be closely related.

After 100 epochs of training, the model with the best performance according to the criteria mentioned in Section 5.1.2 is selected. Table 6.1 summarizes the values of the measured performance metrics in the best epoch.

Table 6.1: Model metrics summary for the best performing epoch (cGAN).

	Symbol	Value
Epoch	-	33
Generator		
Train Loss Generator	L_G	0.694
Train Accuracy Generator	E_t	0.0017
Validation Accuracy Generator	E_{v-h}	0.0011
Train Diversity	τ_t	0.025
Validation Diversity	τ_{v-h}	0.035
Total Accuracy	E	0.0014
Total Diversity	τ	0.03
Discriminator		
Train Loss Discriminator	L_D	0.693

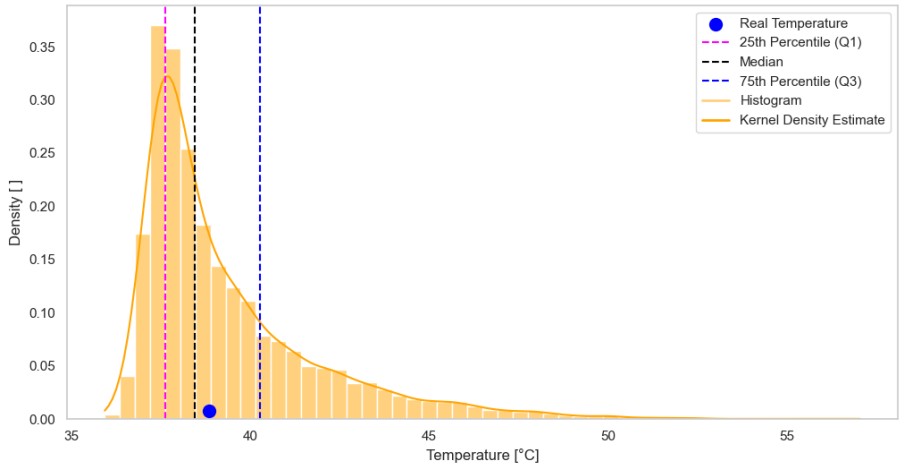
Before diving into the model's behavior on a bigger scale, the generator's output for single data points is examined. In Figure 6.1, the generated set $\mathbf{Y}$ for three different input conditions $\mathbf{c}$ is represented. To illustrate the model behavior in different operational regimes, Figure 6.1 contains a data point from the low, mid, and high target value ranges, which are defined in normalized terms. One point from each temperature region is randomly selected:

- Low-temperature range: 0.3 ± 0.1
- Medium-temperature range: 0.6 ± 0.1
- High-temperature range: 0.9 ± 0.1

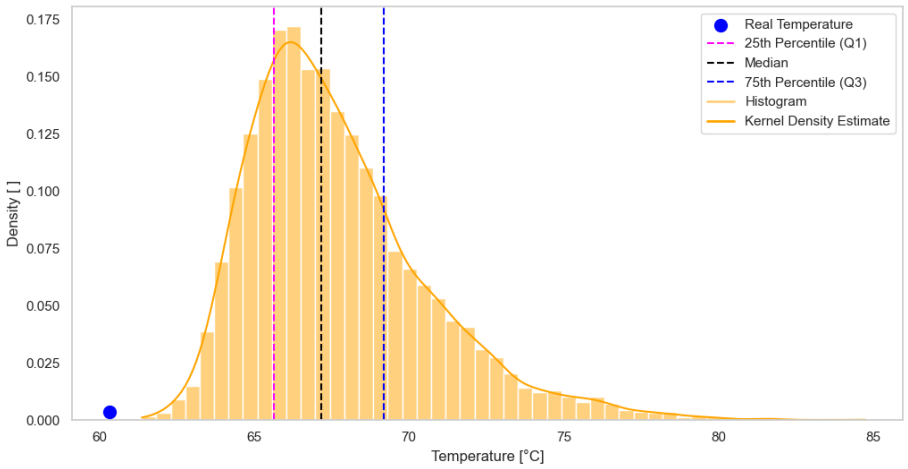
Since each turbine is normalized individually, these can not be translated into a unique range that covers all turbines, as already discussed in Chapter 4. The purpose of including Figure 6.1 is to illustrate the output of the generator. This is done by sampling the generator space, as explained in Equation 5.2. The size of the set $\mathbf{Y}$ is increased to 10000 samples, to get a better understanding of the generated distribution. The set $\mathbf{Y}$ is visualized with a histogram and kernel density estimate, along with the interquartile ranges Q1 and Q3, which represent the 25th and 75th percentile, respectively. Next, also the median of $\mathbf{Y}$ is included, which was already introduced as the point estimate of the target temperature. The real target temperature is also included in the Figure as a blue dot.

Figure 6.1a demonstrates an example where the real target temperature is very close to the median of the set $\mathbf{Y}$. In contrast, there is a larger deviation for Figures 6.1b and 6.1c. It follows that the residual R_{PE} is significantly smaller for Figure 6.1a, compared to Figures 6.1b and 6.1c. Recalling that the MAD relates to the spread of a distribution, this value is smaller for Figure 6.1a, compared to Figures 6.1b and 6.1c. The value of HI_D reflects the uncertainty in the model's predictions, by taking into account information about the spread. Calculating HI_D, the smaller residual in Figure 6.1a is weighted more heavily, while the larger residuals of Figures 6.1b and Figure 6.1c are smoothed.

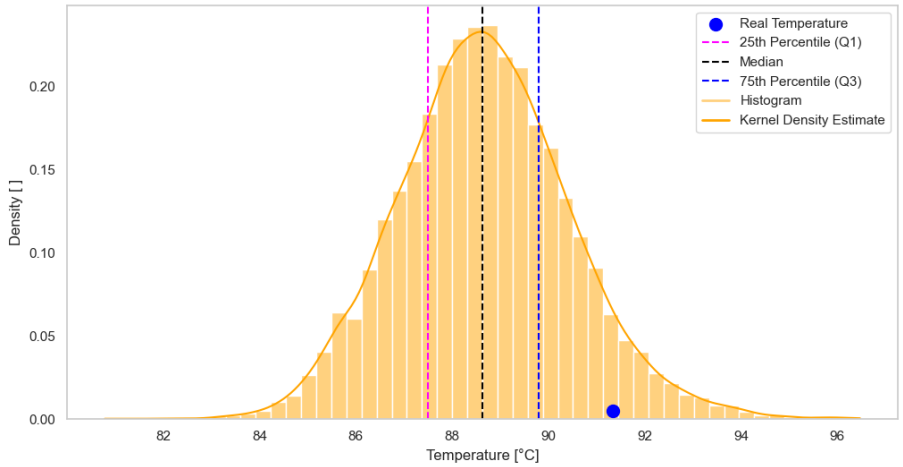
While there seems to be a general trend that the conditional distributions at lower temperatures are narrower, Figure 6.1 does not aim to investigate this. Further, the provided examples are not intended as general trends in the different operational regimes. To draw conclusions about the behavior of the generator in different operational intervals, i.e. low, mid and high target temperature, more datapoints need to be taken into account during the analysis.



(a) Generator output distribution for a data point with low target temperature.



(b) Generator output distribution for a data point with medium target temperature.

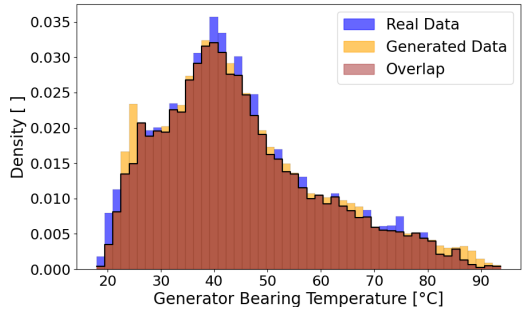


(c) Generator output distribution for a data point with high target temperature.

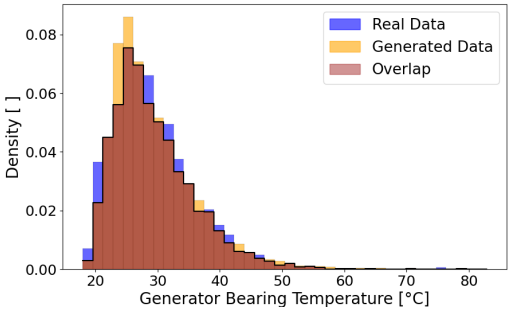
Figure 6.1: Generator output distributions for data points with different target temperatures: low, medium, and high.

6.1.1. cGAN Output for the Clusters

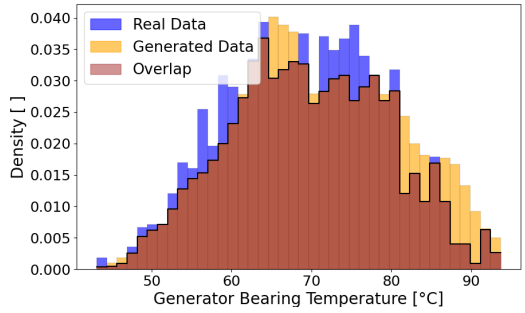
The output of the generator is validated by comparing the generated target temperature distribution to the real target temperature distribution. For that, the healthy validation set is divided into 5 clusters. Characteristics of these clusters are described in Section 5.1.3. The distributions within each cluster are compared. Next to that, also the distributions of the whole healthy validation data are included in the comparison. Figure 6.2 shows both the real and generated distributions for these six cases. The dissimilarity is also quantified by the L1 - Wasserstein distance W_1. The per-cluster results suggest that the model is struggling to represent the conditions in Clusters 2 and 5, while it is performing well in Clusters 1 and 3. The L1 - Wasserstein distance on Cluster 4 is similar to the one obtained with the full healthy validation dataset, indicating that the model performs at comparable performance.



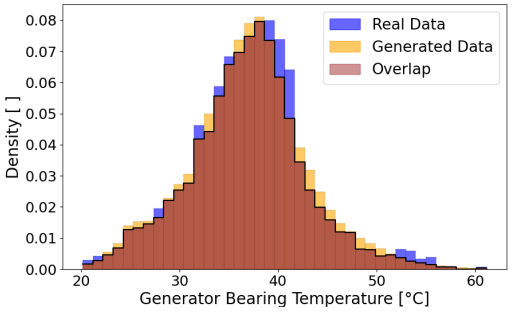
(a) Distribution of real and generated target temperature for the full healthy validation data set. W_1 = 0.56



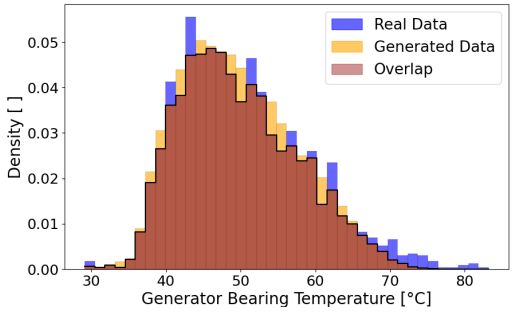
(b) Distribution of real and generated target temperature for Cluster 1. W_1 = 0.38



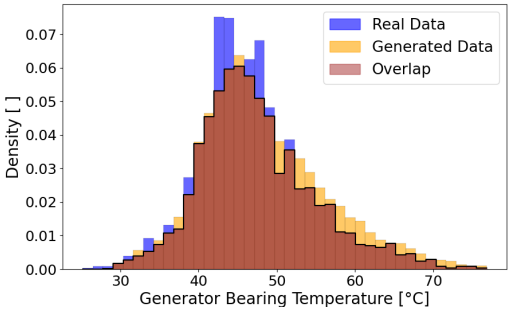
(c) Distribution of real and generated target temperature for Cluster 2. W_1 = 1.87



(d) Distribution of real and generated target temperature for Cluster 3. W_1 = 0.20



(e) Distribution of real and generated target temperature for Cluster 4. W_1 = 0.66



(f) Distribution of real and generated target temperature for Cluster 5. W_1 = 1.38

Figure 6.2: Histograms of real and generated target temperature data for the healthy validation data set and clusters (cGAN). W_1 values represent the Wasserstein distance between real and generated data.

6.1.2. cGAN Output Time-Series

A time-series of the generated output set $\mathbf{Y}$ and real temperature value t is presented in Figure 6.3. The set is described by its median, the median absolute deviation, and the 5th and 95th percentile.

Note that the size of $\mathbf{Y}$ for each point in time is 100 samples, as discussed in Chapter 5. 1 week of data from Turbine 1's healthy validation set is displayed. Appendix C contains a week of data from the remaining turbines.

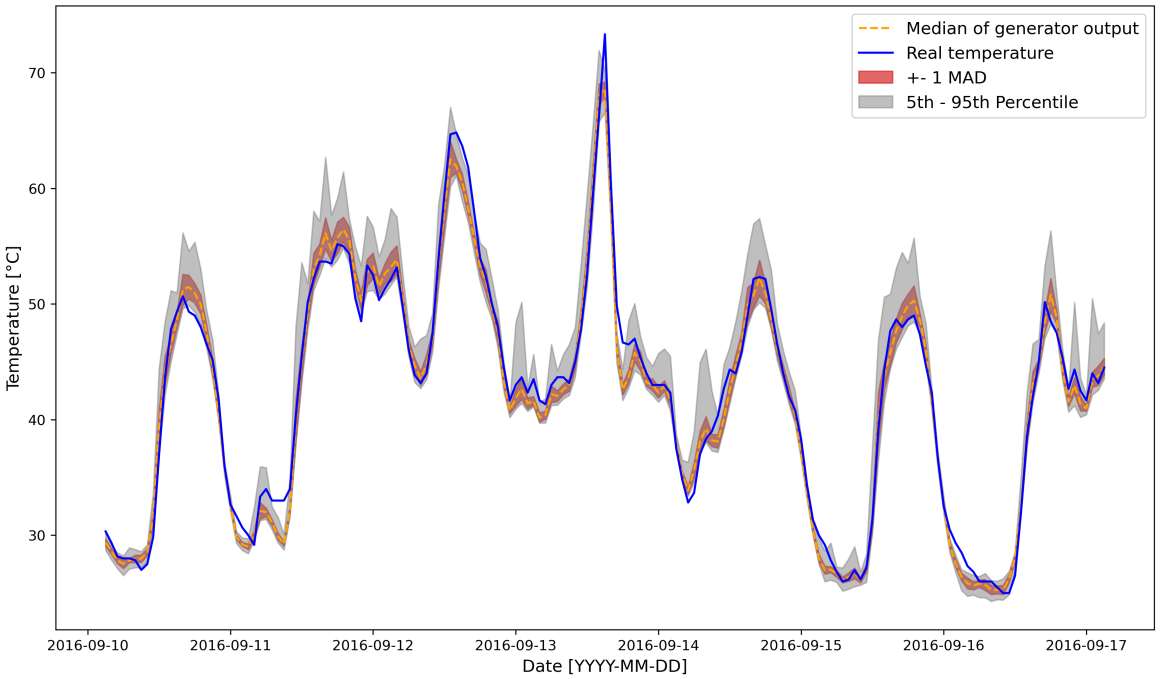


Figure 6.3: Turbine 1: Time series of the generator output set $\mathbf{Y}$, and the real temperature t. The output $\mathbf{Y}$ is described by its median, $\pm$ 1 median absolute deviation, and the 5th and 95th percentiles (first week of validation data).

From Figure 6.3 it can be observed that the generator output is typically less diverse for lower temperature values. This is suggested as low temperature values typically show a smaller interval between the 5th and 95th percentile, and a smaller MAD. Further, the generated distributions are typically right-skewed, which is also indicated by Figure 6.1a and 6.1b.

6.2. Validation of the Base Model

The base model is validated by analyzing the model output for train, validation, and test sets. As the base model uses a classical regression setup, the validation loss can directly be used for evaluating the model's performance. The model metrics of the best-performing Epoch are summarized in Table 6.2.

Table 6.2: Model metrics summary for the best performing epoch (base model).

	Symbol	Value
Epoch	-	22
Base Model		
Train Loss	L_{BM}	0.0031
Validation Loss	L_{BM}	0.0027

Figure 6.4 shows a comparison of the real and generated data from the healthy validation set. The figure suggests that the base model properly represents the data in the healthy validation dataset, which is also suggested by the L1 - Wasserstein distance of 0.49. As the performance on the healthy validation set is used for identifying the best-performing model during training, the figure is included here. A full comparison of the model output on the train, validation, and test sets is included in Section 6.3.

The comparison per cluster is included in Appendix B, although this is not analyzed in detail.

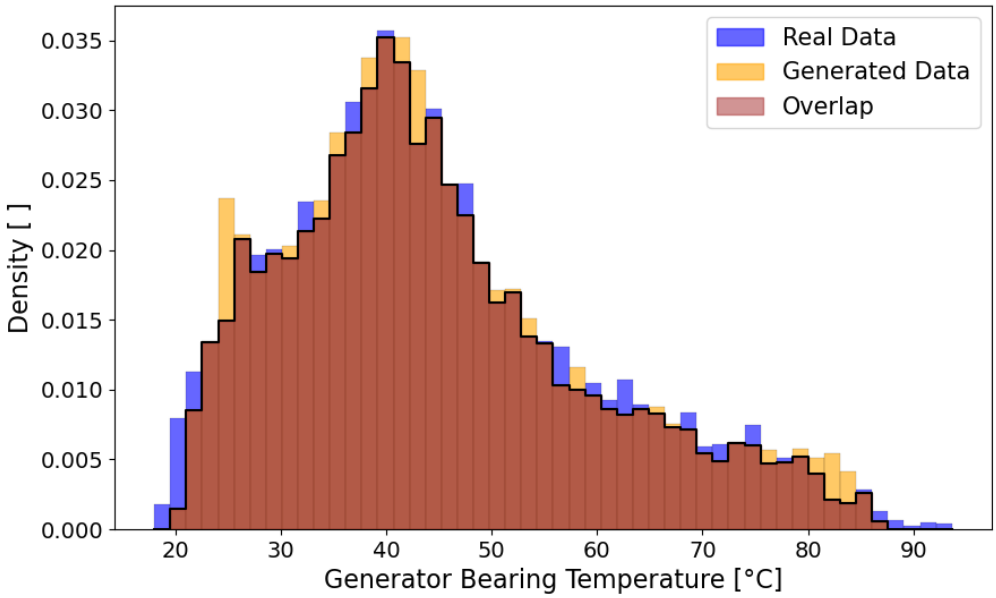
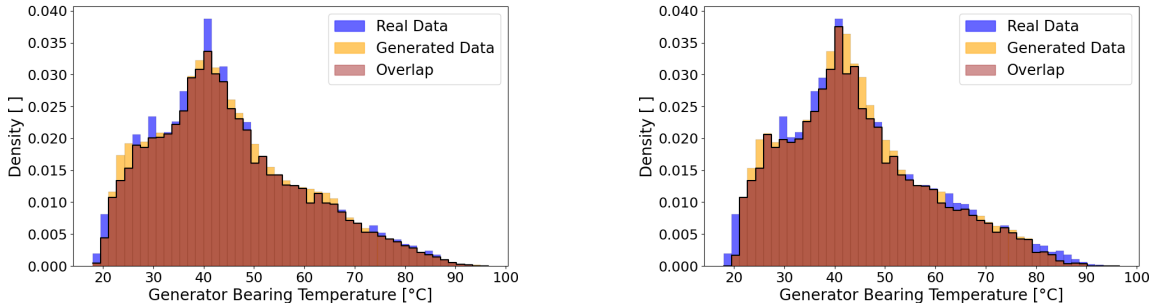


Figure 6.4: Histogram of real and generated target temperature for the full healthy validation data set. W_1 = 0.49 (base model).

6.3. Model Comparison

Before further using the models for fault detection, the real and generated distributions on the train, validation, and test sets are compared. Figure 6.5 presents the real and generated distributions of both models, for all healthy data. Figures 6.5a,c, and e show the results for the cGAN, while Figures 6.5b,d, and f present the results for the base model. A comparison of Figures 6.5a and b suggests that the cGAN better captures the distribution of the train data. In contrast, the base model has a slight performance advantage on the validation data, as a comparison of Figures 6.5c and d reveals. Comparing the results presented in Figures 6.5e and f, the cGAN has superior performance on the test data. The implications of that are discussed in Chapter 7.



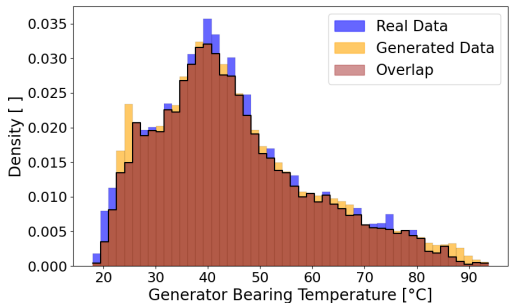
(a) cGAN: Distribution of real vs generated data for all train data. W_1 = 0.30

(b) Base model: Distribution of real vs generated data for all train data. W_1 = 0.52

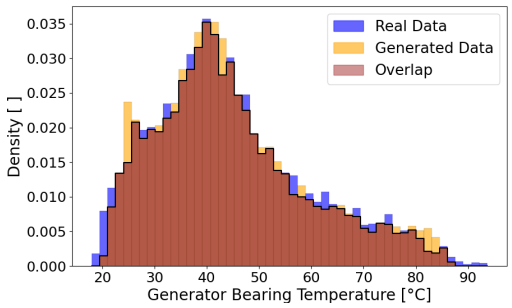
Figure 6.5: Distribution comparison of real and generated data for all healthy data.

6.3.1. Fault Sensitivity Assessment of Health Indicators

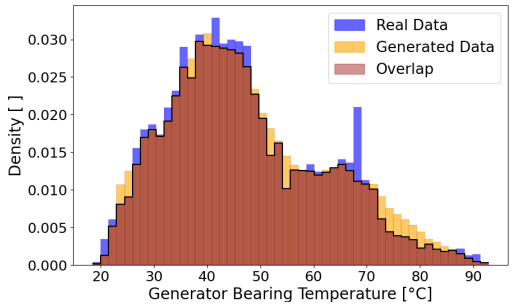
After validating and comparing the generator, and base model output, this section investigates the Health Indicators, which describe the real target temperature t in relation to the model outputs. From the sets $\mathbf{Y}$ and the real target temperatures, the residuals R_{PE} and robust z-scores Z_r of the cGAN



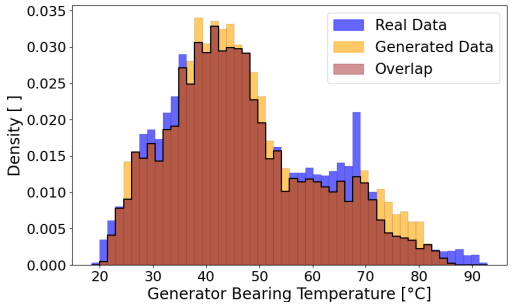
(c) cGAN: Distribution of real vs generated data for all healthy validation data. W_1 = 0.56



(d) Base model: Distribution of real vs generated data for all healthy validation data. W_1 = 0.49



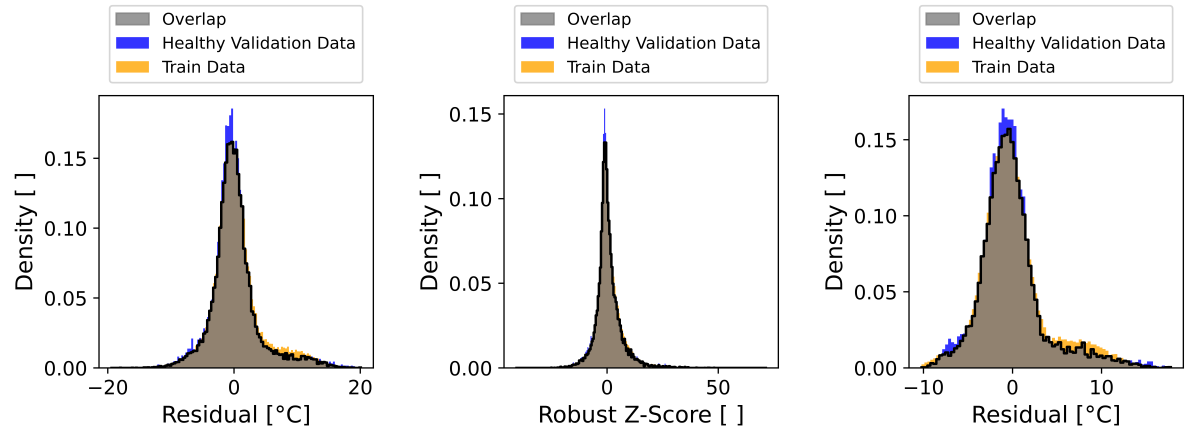
(e) cGAN: Distribution of real vs generated data for all healthy test data. W_1 = 0.59



(f) Base model: Distribution of real vs generated data for all healthy test data. W_1 = 0.81

Figure 6.5: Distribution comparison of real and generated data for all healthy data.

can be calculated. The residuals R_B of the base model are calculated directly with the base model outputs and the real temperatures. The distribution of residuals and robust z-scores of the train and healthy validation dataset is compared in Figure 6.6 for both cGAN and base model. The W_1 distance in the three cases is low, which is an indicator that the models are performing similarly on the train and validation set, i.e. they are generalizing well. Therefore, for the remainder of this report, the healthy mean and healthy standard deviation of the residual and robust z-score are estimated using both the train and healthy validation data. These parameters are used for standardization to calculate the cGAN's and base model's Health Indicators, according to Equations 5.9, 5.12 and 5.15.



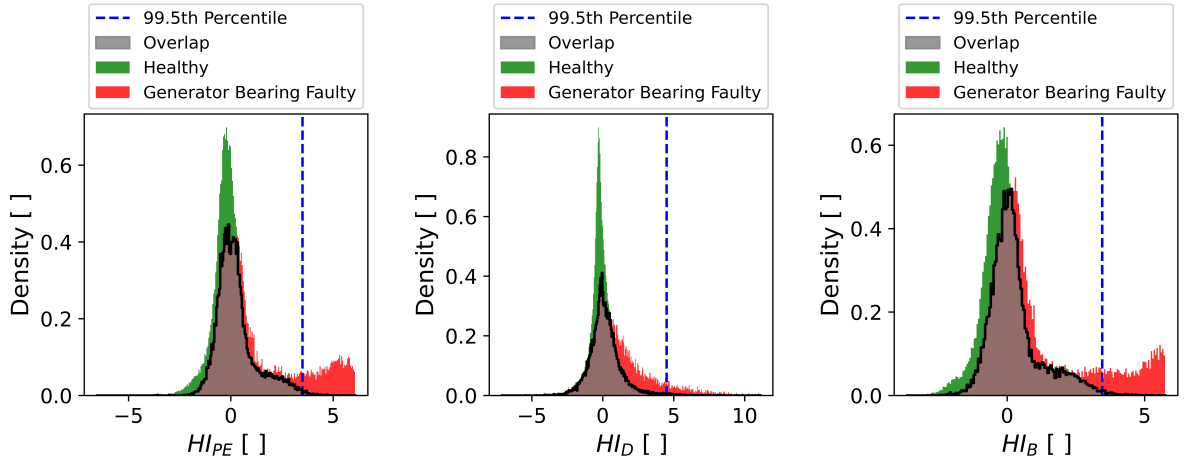
(a) Distributions of R_{PE} for train and healthy validation data. W_1 = 0.43 (cGAN)

(b) Distributions of Z_r for train and healthy validation data. W_1 = 0.59 (cGAN)

(c) Distributions of R_B for train and healthy validation data. W_1 = 0.35 (base model)

Figure 6.6: Distribution comparison of R and Z_r for train and healthy validation data.

As stated at the beginning of the report, the project's goal is to analyze probabilistic modeling of the generator bearing temperature in the context of drivetrain monitoring. Therefore, the sensitivity of the Health Indicators concerning the different failure types must be analyzed, before using the Health Indicators for fault detection. For that, the distributions of the Health Indicators in healthy conditions are compared with the fault-affected counterparts. Also, from the distribution of healthy data, the thresholds to consider an observation anomalous, are derived. The time window of a faulty component condition varies depending on the failure type, which was discussed in Section 4.4.4. For each failure type, only the corresponding time window around the failure is included to generate the fault-affected distributions. First, consider the comparison of healthy data and generator bearing fault-affected data, displayed in Figure 6.7.



(a) Distribution comparison of HI_{PE}. W_1 = 1.83

(b) Distribution comparison of HI_D. W_1 = 1.88

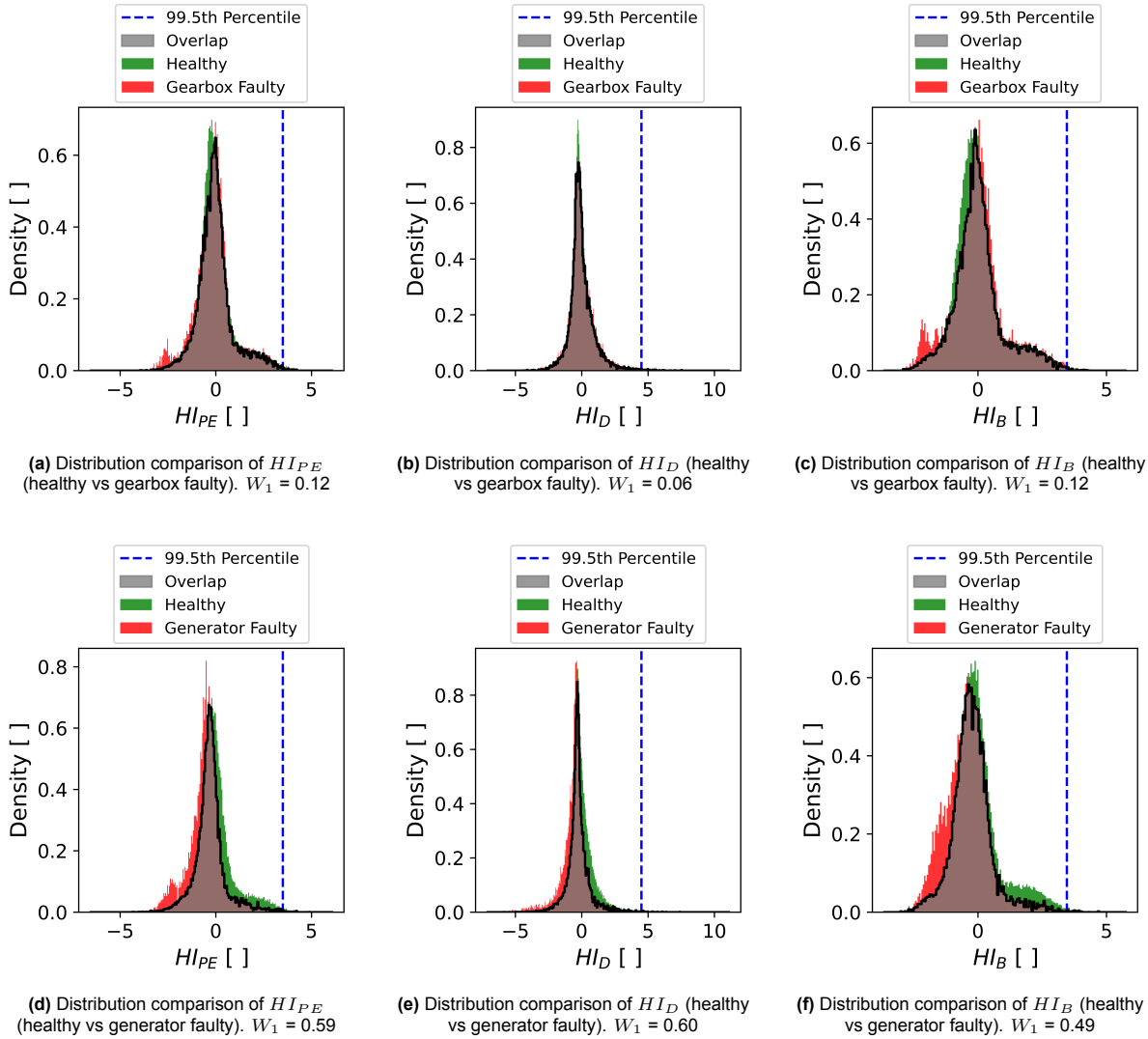
(c) Distributions of HI_B for train and healthy validation data. W_1 = 1.94

Figure 6.7: Comparative distributions of HI_{PE}, HI_D, and HI_B for healthy data vs generator bearing fault-affected data.

Figure 6.7 indicates that the chosen signal can be used to detect generator bearing failures, as a significant part of the faulty data falls out of the healthy data distribution, i.e. to the right of the 99.5th percentile indicated as a blue line. The lines correspond to the thresholds used for the outlier detection, which are derived from a significance level of α = 0.5 %. The thresholds are summarized as follows:

- HI_{PE} = 3.52
- HI_D = 4.51
- HI_B = 3.47

Next, the sensitivity concerning other drivetrain failure types is analyzed. Consider Figure 6.8, which compares the healthy distribution with fault-affected data from gearbox and generator.



(a) Distribution comparison of HI_{PE} (healthy vs gearbox faulty). W_1 = 0.12

(b) Distribution comparison of HI_D (healthy vs gearbox faulty). W_1 = 0.06

(c) Distribution comparison of HI_B (healthy vs gearbox faulty). W_1 = 0.12

(d) Distribution comparison of HI_{PE} (healthy vs generator faulty). W_1 = 0.59

(e) Distribution comparison of HI_D (healthy vs generator faulty). W_1 = 0.60

(f) Distribution comparison of HI_B (healthy vs generator faulty). W_1 = 0.49

Figure 6.8: Comparative distributions of HI_{PE}, HI_D, and HI_B for healthy vs fault-affected data across gearbox, and generator failures.

Figure 6.8 suggest that gearbox and generator failures are not detectable with the chosen target signal and one-sided hypothesis test as described in Chapter 5. The W_1 distances between fault-affected and healthy distributions are smaller compared to generator bearing fault-affected data. Further, a negative shift from the healthy data as visible in Figure 6.8d and 6.8e is not detectable by a one-sided p-test targeted at detecting positive deviations.

6.3.2. Fault Detection

After output and failure sensitivity have been analyzed, the model's outputs are used for fault detection.

The results are presented individually for each turbine. Figures 6.10 - 6.13 compare the Health Indicators of the cGAN, and the base model. In this Section, the results are presented and described. An interpretation of the results can be found in Chapter 7. Recalling the definition that a positive test result of the algorithm is triggered only when the Health Indicators exceed their thresholds for a number of times T_S, in a window T_W, the term 'positive test result' refers to this event happening. In contrast, 'increased reactivity', 'activity', or 'response' refers to the Health Indicators exceeding their threshold (defined by α and the healthy data distribution) at a higher rate without necessarily causing a positive test result. A 'negative response' refers to a Health Indicator signal shifting towards negative values.

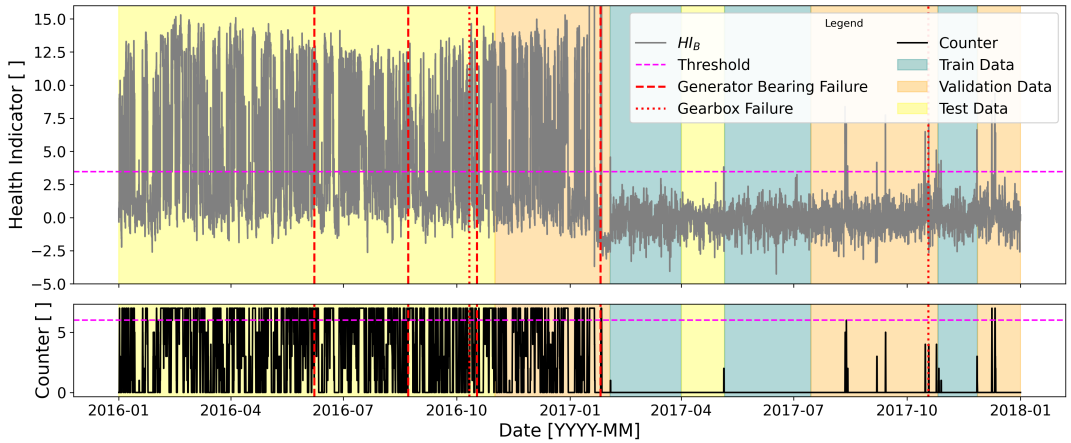
The figures are organized as follows: on top, the time series of the Health Indicator is displayed. The thresholds of the Health Indicators are included in the Figures in magenta. This threshold refers to the significance level α, and the healthy data distribution of the Health Indicators. In a second graph below, the figures include the rolling sum of events that exceed the threshold, represented by the Counter. Note that it is not the Health Indicator itself being summed, but rather a binary value indicating whether the threshold is exceeded or not. The window size of this rolling sum is equal to T_W = 7. Whenever the rolling sum exceeds a value of T_S = 6 events, the alarm is triggered, i.e. a positive test result is obtained. This is indicated by a second threshold line in magenta. Drivetrain failures are included in the Figures:

- Generator bearing failures, which are indicated by a red, dashed, vertical line.
- Gearbox failures, which are indicated by a red, dotted, vertical line.
- Generator failures, which are indicated by a blue, dotted, vertical line.

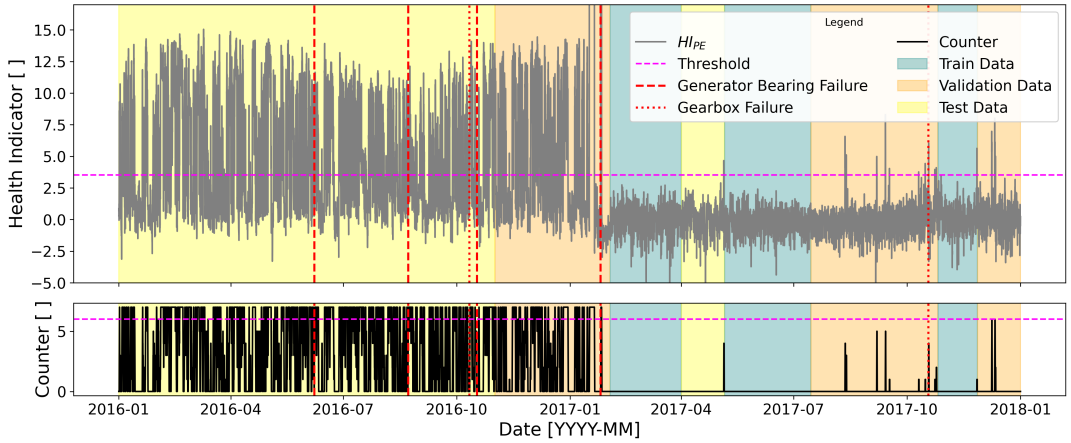
The background color of a time interval indicates to which dataset the data belongs: Train, validation, or test set.

The time series of Turbine 9's Health Indicators are presented in Figure 6.9. Turbine 9 experienced multiple generator bearing interventions in 2016 until the issue was finally resolved by replacing both generator bearings in 2017-01. All Health Indicators show high sensitivity to the problem. In the period following the last generator bearing intervention the response of all Health Indicators is drastically reduced. This is an expected result and indicates that the Health Indicators are sensitive to the generator bearing problems that Turbine 9 experienced from 2016-06 until 2017-02. In Chapter 7, these results are discussed in more detail.

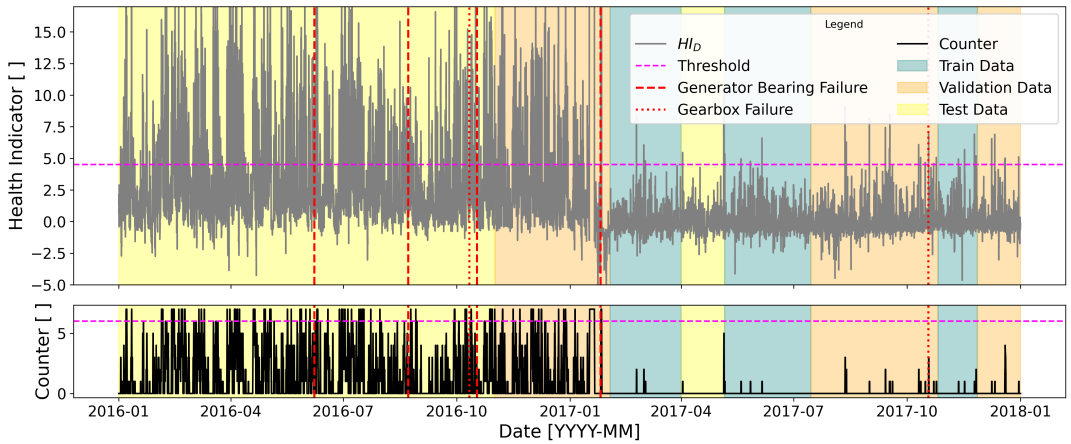
In 2017-10, the turbine's gearbox was found to be noisy. While the BHI and PEHI show elevated activity before the failure (from 2017-08 until end of 2017-10), this is not as clearly visible for the DHI. The BHI tests positive about 2 months before the gearbox noise was diagnosed. Both BHI and PEHI test positive two times in the last block of validation data, while the DHI does not show as much activity.



(a) Turbine 09: Time series of HI_B and Exceedance Counter (base model).



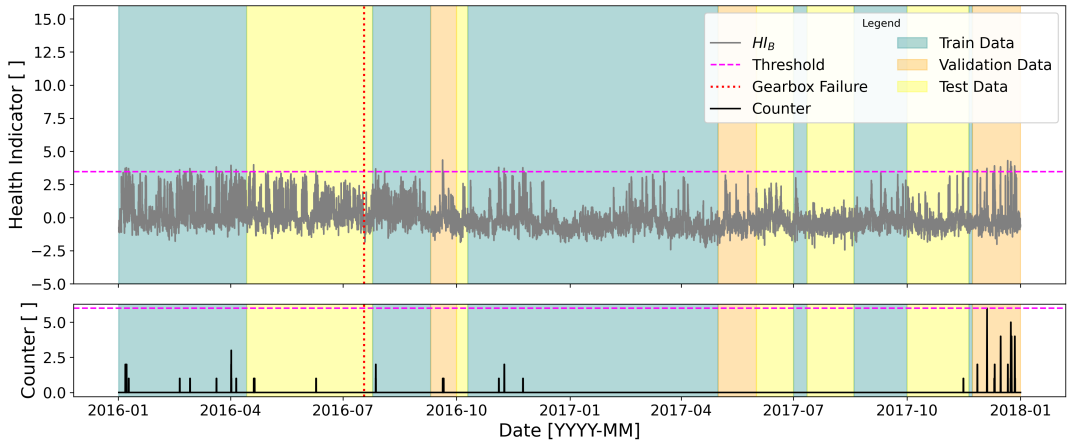
(b) Turbine 09: Time series of HI_{PE} and Exceedance Counter (cGAN).



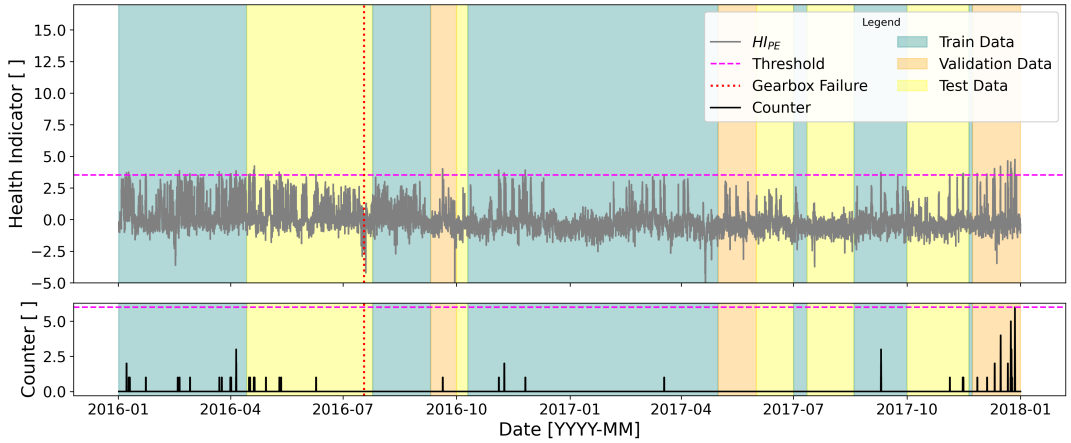
(c) Turbine 09: Time series of HI_D and Exceedance Counter (cGAN).

Figure 6.9: Time series of the three Health Indicators and Exceedance Counters (Turbine 09).

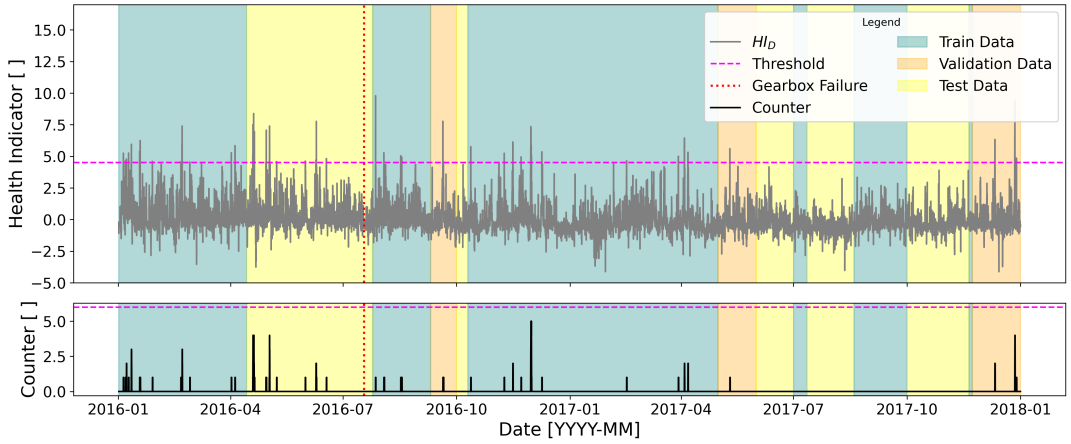
In Figure 6.10, the time-series of Health Indicators and Exceedance Counter for Turbine 1 is displayed. In 2016-07 the turbine's gearbox pump was damaged. In the period before the problem (from 2016-01), all Health Indicators, but particularly HI_{PE} and HI_D show slightly elevated responses, compared to the rest of this Turbine's data. In the period from 2017-11 until 2018-01, both HI_B and HI_{PE} exhibit increased reactivity, which eventually causes the Exceedance Counter to reach the threshold, i.e. a positive test result.



(a) Turbine 01: Time series of HI_B and Exceedance Counter (base model).



(b) Turbine 01: Time series of HI_{PE} and Exceedance Counter (cGAN).

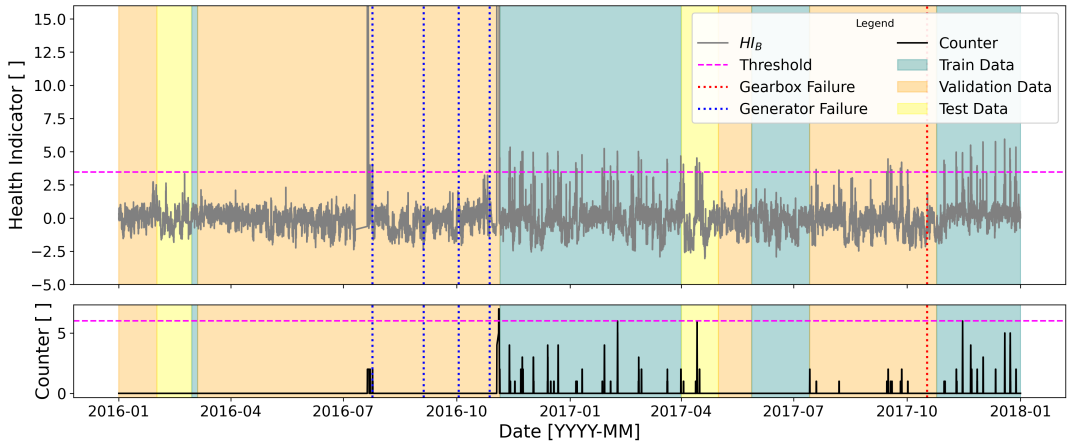


(c) Turbine 01: Time series of HI_D and Exceedance Counter (cGAN).

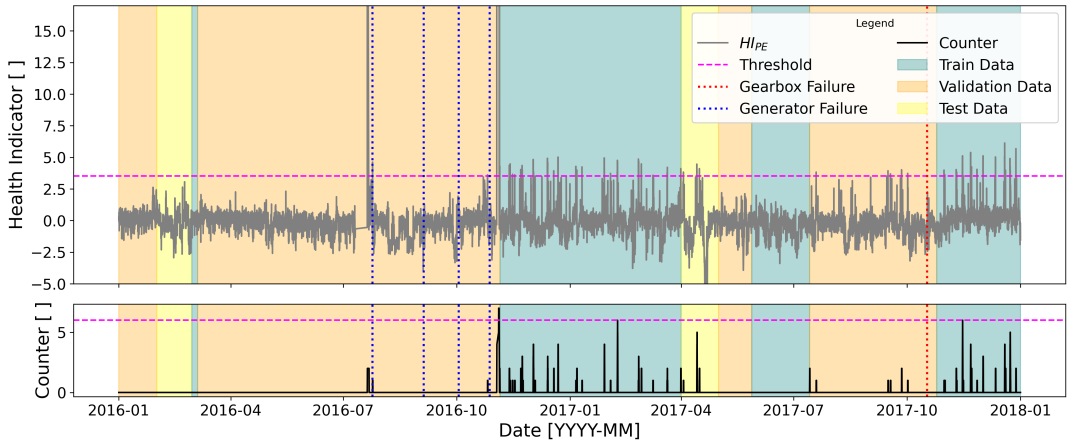
Figure 6.10: Time series of the three Health Indicators and Exceedance Counters (Turbine 01).

Next, the time-series for Turbine 6 are presented in Figure 6.11. This Turbine experienced a gearbox bearing issue in 2017-10, which appears to be not detectable by any of the three Health Indicators, i.e. there is no increased activity prior to the failure. Again, HI_B and HI_{PE} show comparable behavior, while HI_D differs. Especially in the block of training data following the generator issues, HI_D shows less sensitivity because it exceeds the threshold less frequently, but typically by larger values. These generator issues were finally addressed by replacing the complete generator in 2016-10. Figure 6.11 suggests that the replacement has an impact on the generator bearing temperature in the period fol-

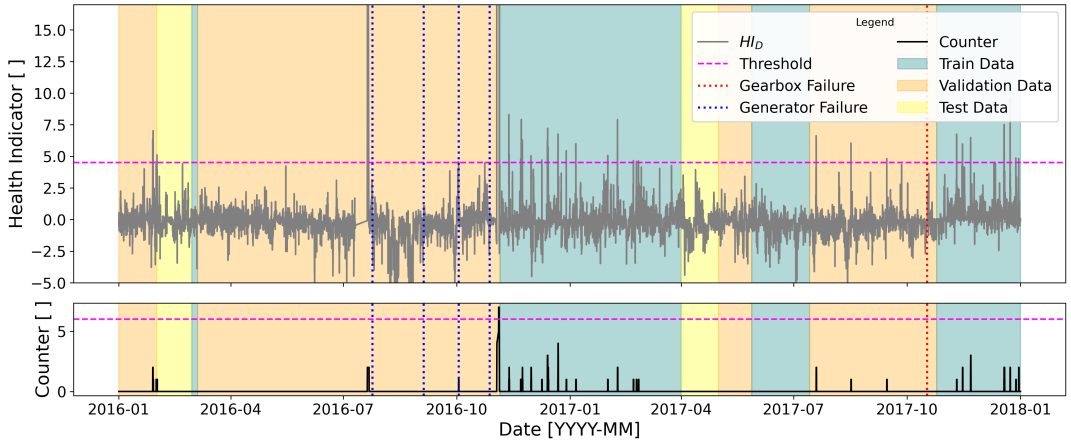
lowing the intervention, where Figure 6.11a and 6.11b show higher sensitivity. When the Exceedance Counter reaches its threshold in 2016-11 for all turbines, the reason is likely an inactive turbine, as a result of the maintenance actions. Therefore, it is not considered further in the analysis. Apart from this result, the BHI Exceedance Counter reaches the threshold 3 times, while this happens only two times for the PEHI. The DHI does not test positive again.



(a) Turbine 06: Time series of HI_B and Exceedance Counter (base model).



(b) Turbine 06: Time series of HI_{PE} and Exceedance Counter (cGAN).

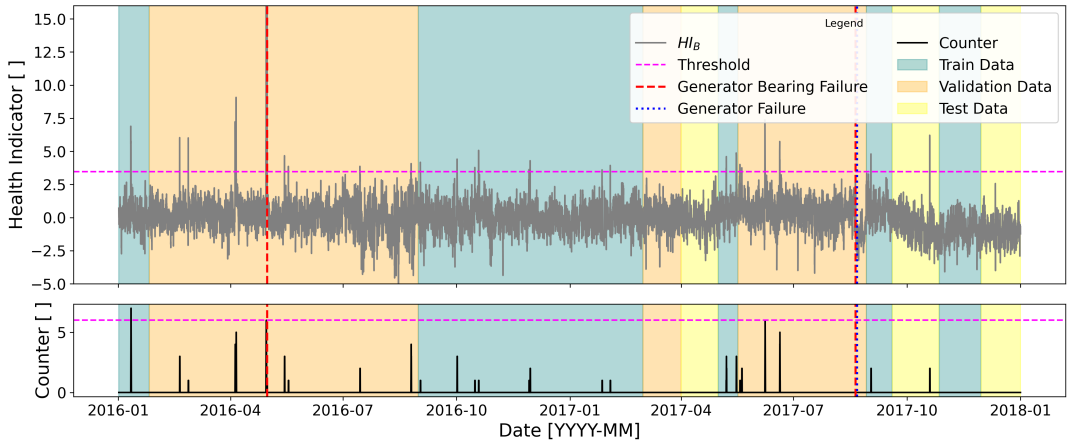


(c) Turbine 06: Time series of HI_D and Exceedance Counter (cGAN).

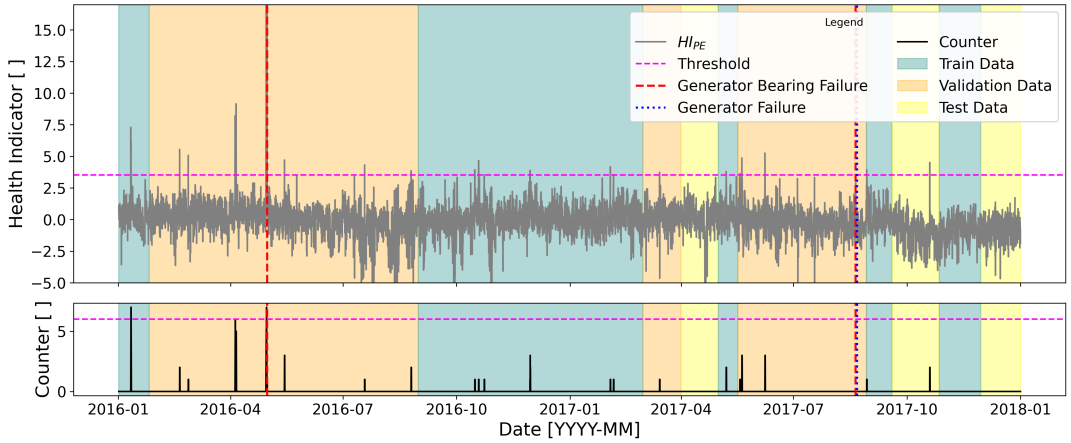
Figure 6.11: Time series of the three Health Indicators and Exceedance Counters (Turbine 06).

Consider Figure 6.12 for the results obtained from Turbine 7. The generator bearing temperature sensor

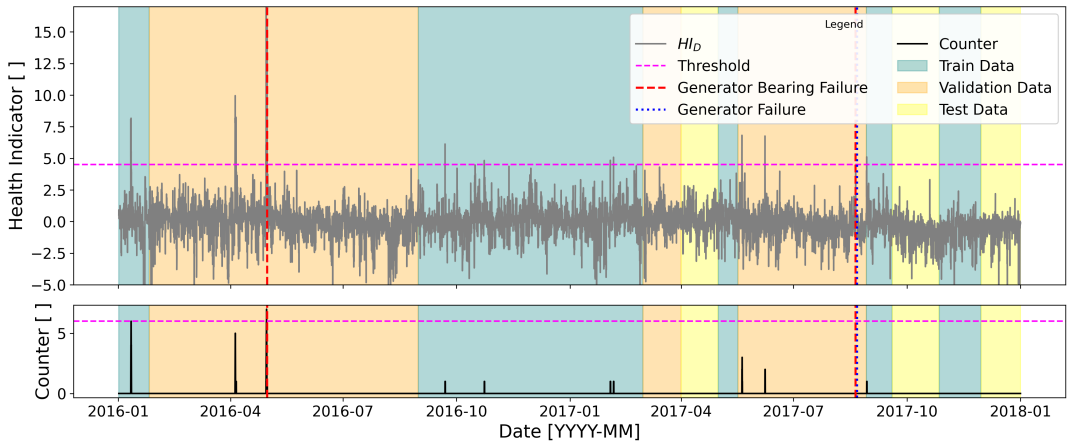
was replaced on 2016-04, due to readings of too high temperature values. While PEHI Exceedance Counter reaches the threshold around a month before the temperature sensor was changed, the activity of the PEHI is generally low around the failure. The DHI and BHI Exceedance Counter do not reach the treshold in advance, but both Health Indicators also show a single peak in activity at the same time where the PEHI Exceedance Counter reaches its threshols. All three Health Indicators' Exceedance Counters reach the threshold right before the failure, and also in 2016-01. In 2017-08, the generator bearing, and generator were damaged. While the BHI Exceedance Counter reaches its threshold about 2 months before the failure, none of the Health Indicators seems to detect this issue effectively.



(a) Turbine 07: Time series of HI_B and Exceedance Counter (base model).



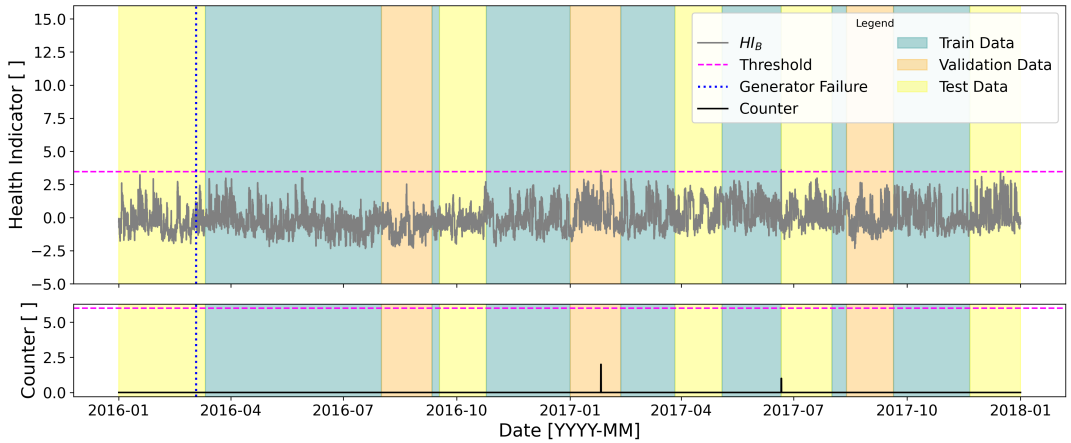
(b) Turbine 07: Time series of HI_{PE} and Exceedance Counter (cGAN).



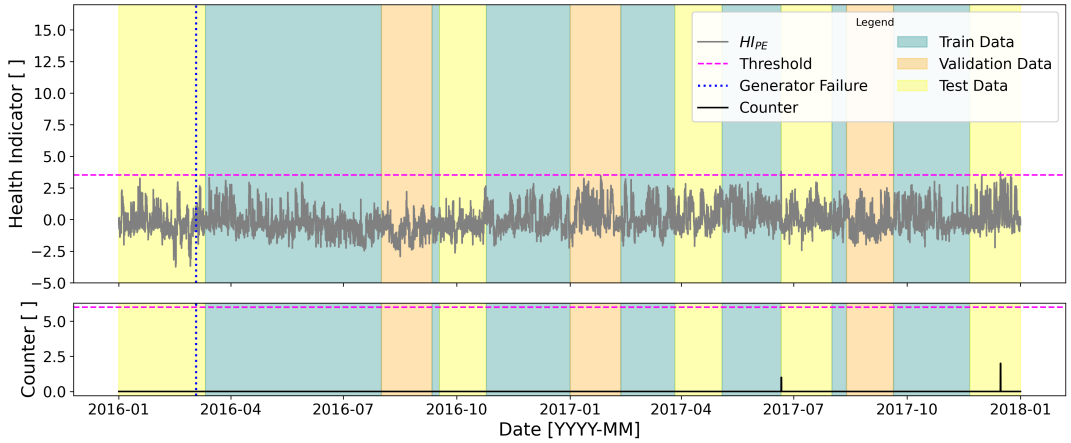
(c) Turbine 07: Time series of HI_D and Exceedance Counter (cGAN).

Figure 6.12: Time series of the three Health Indicators and Exceedance Counters (Turbine 07).

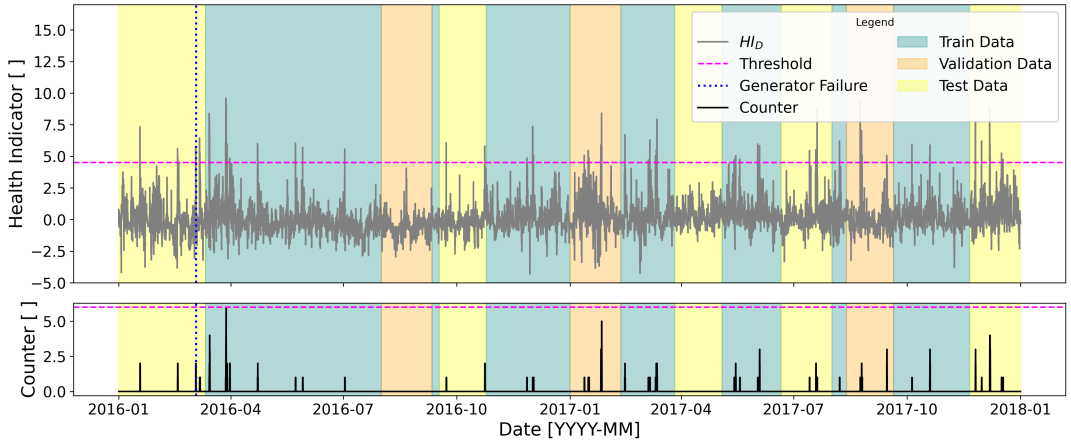
Figure 6.13 shows the results for Turbine 11. Turbine 11 experienced a generator electrical fault in 2016-03. After that, the turbine's drive train remains failure-free for the remainder of the data available. Over the entire 2 years period, HI_B and HI_{PE} show similar behavior: both Health Indicators stay below the threshold for almost the entire 2 years. In contrast, HI_D is more noisy and contains more outliers. About a month after the generator problem in 2016-03, the DHI Exceedance Counter reaches its threshold.



(a) Turbine 11: Time series of HI_B and Exceedance Counter (base model).



(b) Turbine 11: Time series of HI_{PE} and Exceedance Counter (cGAN).



(c) Turbine 11: Time series of HI_D and Exceedance Counter (cGAN).

Figure 6.13: Time series of the three Health Indicators and Exceedance Counters (Turbine 11).

7

Discussion

Interpreting the findings from applying the Health Indicators is challenging, as the meaning of positive or negative test results, or responses, heavily depends on whether an actual physical relationship between the target variable and turbine condition is present. As the failure information in the dataset is sparse, the maintenance and repair actions taken are not known in detail. To provide a thorough interpretation of the results, the most relevant steps in the modeling process, and the results themselves, are discussed. First, the architecture, data pre-processing, and model setup are discussed in Section 7.1. Next, the training and validation results are examined critically in Section 7.2. Section 7.3 first discusses the results of the fault sensitivity analysis. Next, all prior considerations are used to interpret the results from applying the model to the entire 2 years of data. From this, the specific test results of the Health Indicators are interpreted as true or false responses. Finally, in Section 7.4, the research questions stated at the beginning of the report are addressed.

7.1. Model Development

This section discusses data preprocessing, model setup & architecture.

Data Preprocessing

As already mentioned in Chapter 4, the results obtained from using ML models heavily depend on the data used, and preprocessing steps. In this project, a critical decision is the split of the data into healthy and faulty data points. The split into healthy and faulty data is done based on a static time window. This is not an ideal solution, as there is a high possibility that either:

1. Faulty data points end up in the healthy dataset.
2. Healthy data points end up in the faulty dataset.

The first scenario of these two possibilities is a major concern, as it compromises the model's capability to learn the relationships associated with normal operating conditions. During modeling, the static time window for mechanical failures had to be increased for Turbine 9, as the target temperature exceeded the healthy temperature value of about 100 °C regularly in the entire year 2016. This clearly shows the vulnerability of this approach. The second scenario does not directly lead to the model learning unwanted relationships. However, healthy data is excluded from training, which reduces the model's source of information, thereby potentially limiting performance. Hydraulic, and transformer fault-affected data was excluded from training. While this is a safe way of ensuring that the model does not learn faulty patterns, it is questionable if hydraulic and transformer faults interact with the generator bearing temperature. Therefore, significant amounts of information are potentially lost in this step. In general, more sophisticated methods should be applied to split the data into healthy and faulty points, and, the split should be verified with a sensitivity analysis.

Model Setup & Architecture

Comparing the architecture and model setup with literature is challenging, as the exact setup strongly

depends on the data used, and the task to be achieved. The study conducted by Jankauskas et al. [37] served as an initial example for the setup, since the researchers investigated the use of RNNs to predict the gearbox bearing temperature, demonstrating their setup with the EDP dataset. While the sequence length used in this project (4h) is very close to their identified optimum of 3h, the complexity of this model is much lower: Jankauskas et al. identified a bidirectional LSTM with 2 layers of 128 and 100 cells as the best-performing setup. In contrast, the models investigated in this project solely rely on 2 x 16, non-bidirectional LSTM cells for target temperature prediction. The significantly smaller complexity could be due to using a smaller set of features for prediction. Jankauskas et al. use 18 features, compared to the five features used in this project. This simplifies the relationship to be modeled, reducing the need for model complexity. Next to that, the target is different, which could also impact the complexity of the relationship to be modeled. From the five features used in this model, only the nacelle temperature is not used by Jankauskas et al. As discussed in Chapter 4, the nacelle temperature is not a primary operational signal, which increases the risk of mispredictions due to faults anywhere in the turbine. A different study was conducted by Bindingsbø et al. [69]. They predicted the generator bearing temperature based on five features, including the nacelle temperature and the generator temperature itself. In comparison, relying solely on the nacelle temperature as a secondary operational signal could reduce dependency on generator temperature and minimize the risk of mispredictions during generator faults. Based on these considerations, it seems reasonable to include the nacelle temperature in the set of features. However, more research is necessary to confirm the validity of these assumptions.

7.2. Model Training & Validation

The training progress of the cGAN is evaluated by estimating its accuracy, which is done by reducing the generator output to a point estimate and calculating the MSE with the real target. This metric does not actually align with the modeling goal of the generator, which aims at matching the generated and real conditional distributions. The real modeling goal is difficult to measure, as it is problematic to define the real conditional distribution. For these reasons, the generator is evaluated using the median of its outputs. However, using this metric for evaluation appears to have implications for the model, as will be discussed below.

The cGAN's accuracy (measured with MSE of median and real target) is higher than the accuracy of the base model which directly uses the MSE during training. This result seems counterintuitive in the first place. However, it is in accordance with literature. Jobson et al. [23] compared the mean absolute errors of a traditional regression model with a cGAN, and concluded that the cGAN's accuracy is higher compared to the deterministic model. As this is essentially the metric defined for evaluation, it is not surprising that the cGAN performs well at this task. After the training is completed, the model is further validated by comparing the generated distributions with the real distributions. In this step, the whole generator output is included, instead of the point estimate used before. This is done for individual clusters and the full healthy validation set. The same approach is applied to the base model. The result is rather surprising: while the cGAN has a lower MSE on the healthy validation dataset, the cGAN's W_1 distance between real and generated distributions is higher the base model's. It seems that, while the cGAN's central tendency is more accurate than the base model's, it is still not capturing the full data distribution better than the base model. This is possibly a result of evaluating the model on its point estimate, rather than the full distribution. As a result, training is stopped when the point estimate aligns well with the real data distribution, rather than the full generator output.

While the comparison of real and generated distributions on the healthy validation set suggests a slight advantage for the base model, the results differ when examining the train and test sets. Comparing the W_1 distances of real and generated data obtained from cGAN and base model suggests that the cGAN better represents the train data distribution. Further, and even more significant, is the comparison of the representation of the healthy test sets. The cGAN seems to better represent the test set's data, with only a minor increase in W_1 distance, compared to its performance on the healthy validation set. Recapping that the healthy test set differs more from the train set than the healthy validation set, the base model seems to struggle more with this difference. This indicates that the cGAN has improved robustness compared to the base model.

Splitting up the data into clusters was intended to address the challenge of defining the true conditional distribution. By grouping the data into similar features, the target space per group could potentially ap-

proximate the true conditional distribution. However, this problem becomes challenging with increasing sequence length, as not only individual data points, but sequences must be similar. By focusing only on the last timestep for grouping the data, the original goal could not be achieved. However, the analysis still reveals conditions that the cGAN struggles to represent. It appears that the data in Cluster 2 and 5 poses a challenge for the model. The exact origin of that is unclear and requires further investigation. In future works, model performance could potentially be improved by analyzing the origin of this performance drop and taking appropriate measures.

To summarize the findings regarding the training and validation of the models:

- While the cGAN's generalization performance using its median does outperform the base model, this does not hold when considering the full generated distribution.
- Evaluating the cGAN based on a point estimate appears to have implications for the results obtained. If the full generator output distribution is used in the further process, this evaluation method is likely not ideal.
- The results on the test set suggest that the cGAN has improved robustness compared to the base model.
- The results on the train set suggest that the cGAN better represents the train distribution.

7.3. Model Outputs

This section first discusses the fault sensitivity of the three Health Indicators. Next, taking all previous considerations into account, the time series obtained from applying the Health Indicators are interpreted.

Fault Sensitivity

For a fault to be detectable by an NBM of the generator bearing temperature, the fault needs to cause a change of that temperature. When considering the comparison of healthy data with generator bearing fault-affected data, the fault-affected data distribution has a right tail, no matter which Health Indicator is used. This indicates that faulty generator bearings can cause the temperature to be higher than under normal operation. Thus, the target variable is sensitive to some of the failure types of generator bearings. This is in accordance with the results of other researchers that used the generator bearing temperature to detect generator bearing faults [69]. While the BHI's and PEHI's healthy and fault-affected distributions are of similar shape, the DHI's distribution looks different. The key difference in the computation of the DHI is the scaling of the residual by the mean absolute deviation of the generated conditional distribution. This affects the healthy distribution. Not only the density around 0 is increased, but the distribution also has a wider range. As a consequence, the threshold associated with a 0.5 % significance level is higher. The W_1 distance between healthy - and generator bearing fault-affected data is comparable for all three Health Indicators. However, the PEHI and BHI appear to more effectively identify a faulty generator bearing, as a larger proportion of the fault-affected data exceeds the threshold. The deficit of the DHI could potentially be mitigated by applying a different strategy to identity outliers, instead of setting a threshold based on 0.5 % significance level as applied in this project.

Analyzing the fault sensitivity concerning the other components, it appears that gearbox faults do not impact the distribution of generator bearing temperature significantly. There are two mechanisms how a faulty gearbox could possibly affect the generator bearing temperature: Firstly, a faulty gearbox could generate heat, that affects the generator bearing by conduction through the shaft. As the generator bearing is mounted outside of the gearbox, the faulty gearbox would need to generate a large amount of heat to affect the generator bearing temperature through heat transfer. Secondly, gearbox issues that cause imperfections in the alignment of the drive shaft, or excessive vibrations, could affect the heat generation in the generator bearing directly. The fault sensitivity analysis suggests that none of these two mechanisms plays a role in the considered data. PEHI and BHI yield very similar results on gearbox fault-affected data. Given that gearbox faults do indeed not affect the generator bearing temperature, the DHI appears to slightly better reflect the healthy state of the turbines, as the W_1 distance of healthy and fault-affected data is smaller. This could be due to the improved robustness of the cGAN, as discussed earlier. However, more research is necessary to confirm this hypothesis. Regardless of this

statement, all Health Indicators seem not to be sensitive to the gearbox faults present in this dataset.

Considering the generator fault-affecetd data, some surprising observations can be made. The distribution of generator fault-affected data is left shifted from the healthy data distribution. Hence, both cGAN and base model suggest that the generator bearing temperature in the generator fault-affected data is lower than under normal operation. The left shift is slightly more significant for the cGAN. To interpret the observed behavior, some considerations are made: Turbine 6 is the main source of generator fault-affected data. A series of generator related repairs took place from 2016-07 until 2016-10, where the first and last interventions are the replacement of the generator. From this, it appears that the first intervention did not solve the original issue, and the turbine operated in faulty condition until the generator was replaced again, in 2016-10. There are 2 main hypotheses for explaining the observed behavior of the Health Indicators: The distribution shift is a modeling issue, i.e. the model is not capturing the conditions encountered in the data correctly. Another explanation is, that there is a true physical relationship that causes the generator bearing temperature to drop relative to normal operation during the period of interventions. Regarding the first possibility, it was investigated if Turbine 6 had increased downtime in the considered period, which could lead to a continuous overprediction of the generator bearing temperature. Investigating the power curve of the generator bearing fault-affected data revealed that Turbine 6 is slightly more frequently non-operational, compared to healthy operation. Non-operational points are identified by filtering for data points that share wind speed values > 4 m/s and power output < 50 kW. While in the healthy data set, there are about 3.2 % of data points non-operational, the generator bearing fault affected data of Turbine 6 records 5.5 % of non-operational points. Comparing this to the 4.7 % of non-operational points in the gearbox fault-affected data, where no left distribution shift is observed, it is unlikely that the increase in downtime at this scale affects the model predictions. While this analysis does not fully exclude the possibility of a modeling issue, no direct evidence to support such a statement could be found. Therefore, a physical reason to explain the distribution's left shift is likely. Particularly, assembly-related problems that either increase heat transfer from the bearing or decrease heat generation in the bearing are potential candidates. When observing the DHI time series applied to Turbine 6, a sudden decrease is noted after the first generator intervention. This behavior aligns with the hypothesis of an assembly issue. However, more investigations are necessary to confirm this theory. While the Health Indicators appear to have some sensitivity to the generator fault-affected data, the cause of that is likely still related to issues with the generator bearing. Regardless, in this project only positive deviations from normal operation are identified with the one-sided hypothesis test, which leaves these issues undetected in the final application.

Interpretation of Full Time Series

Each turbine's time series, depicted in Figures 6.9 - 6.13, is interpreted separately. In general, it appears that PEHI and BHI behave similarly. However, typically the PEHI shows less response than it's purely deterministic counterpart, especially in blocks of validation data. This aligns with the findings above, stating that the cGAN output, represented by its median has better generalization capabilities than the base model.

Turbine 9
The generator bearing problems in Turbine 9 are detected by all three Health Indicators. While the BHI and PEHI show very high sensitivity, the DHI's response is less extreme. This aligns with the results of the fault sensitivity analysis, which indicated that BHI and PEHI are more sensitive to generator bearing faults. Ahead of the gearbox noise that was diagnosed in 2017-10, the BHI and PEHI show increased responses compared to the DHI. A noisy gearbox could imply higher friction and elevated temperatures. Considering that the monitored generator bearing is probably located close to the gearbox, it is therefore not possible to outrule a true physical relationship between a noisy gearbox and the increased response. However, the fault sensitivity analysis (Figure 6.8) emphasizes that this relationship is not captured by the chosen modeling approach. Therefore the observed behavior is interpreted as a false response.

Turbine 1
The damaged gearbox pump in Turbine 1 appears to have no impact on any of the Health Indicators. This is a reasonable result, as there is no direct link between the gearbox pump and the generator bearing temperature. Although the DHI has in general a higher noise level, it seems to be more robust to the conditions encountered around 2017-12. While the BHI and PEHI respond strongly, the DHI is little affected.

Turbine 6
After the generator replacement in 2016-10, all three Health Indicators show increased responses. The details of the work performed on the turbine are not specified in the dataset. While it is unclear whether the generator replacement involved a bearing replacement or the assembly and disassembly of the bearing, this scenario is considered likely. In principle, the response's origin could be due to both, a physical or modeling reason. Therefore, both possibilities are discussed: In case the bearing was replaced during the maintenance action, the increased response of the Health Indicators could be due to a running-in phase of the bearing, or other consequences of the maintenance that increase friction in the bearing. Increased friction could cause elevated bearing temperatures. In this case, the BHI and PEHI would have significant performance advantages. However, increased Health Indicator responses are not observed after the well-known generator bearing replacement in Turbine 9. Also, the diagnosis 'generator damaged' of Turbine 7 in 2017-08 suggests an intervention, which does not seem to affect the Health Indicators in the period after the maintenance action. Given these reasons, the responses after the generator replacement of Turbine 6 are interpreted as non-physically related.

A similar phenomenon is observed in the last quarter of 2017: After an issue (gearbox bearings damaged), both the PEHI and BHI have large responses, while the DHI shows only slightly elevated responses. Again, the exact content of the maintenance actions is unknown. While the entry 'gearbox bearings damaged' does not suggest any interventions, a replacement of the gearbox bearings is likely. If the gearbox was opened, or disassembled, subsequent alignment issues on the generator bearing could not be outruled, as the generator bearing is probably mounted on the gearbox side. This could cause a temperature increase in the generator bearing after the intervention. The observation that the increased response does not happen immediately after the intervention makes a relationship between intervention and a positive test unlikely. Therefore, the positive tests of PEHI and BHI are interpreted as false responses. However, this can not be concluded with certainty.

The negative response of the DHI after the first generator replacement in 2016-07 has already been discussed. Given the considerations, it is possible that this reflects an actual physical anomaly. However, this must be researched further before drawing final conclusions.

Turbine 7
All three Health Indicators appear to be sensitive to the generator bearing temperature sensor issue in 2016-04. Since this is a sensor issue, it is highly possible that the malfunction takes place only occasionally, as suggested by all three Health Indicators. As it is not specified in the dataset which of the two generator bearing temperature sensors experienced the failure, this could also be a false response. While the true nature of this behavior is uncertain, the observed sensitivity of the Health Indicators is likely related to an actual fault.

Although the BHI tests positive about 2 months before a damaged generator and generator bearing were diagnosed in 2017-08, it is questionable if this positive test result is actually related to the failure. More outliers close to the failure would be expected. As there is only a response far from the failure, it is likely that the failure is actually not detectable with the chosen modeling approach. Therefore, the response is interpreted as a false response.

Turbine 11
In Turbine 11, the responses of BHI and PEHI are low over the entire 2 years of data. The DHI is in general more noisy on this turbine's data. Its positive test result after the generator electrical fault in 2016-03 is likely a false response since there is no evidence that the problem or maintenance actions have an impact on the generator bearing temperature.

To summarize the key aspects of discussing the model outputs:

- All three Health Indicators are sensitive to detect generator bearing issues, which is demonstrated in turbines 9 and 7.
- The DHI as it is used in this setup is less sensitive to generator bearing issues than PEHI and BHI.
- The results suggest that the Health Indicators, particularly the DHI, are sensitive to the generator issue in Turbine 6. Nevertheless, the exact physical mechanism of that is unknown and needs further investigation.

- PEHI and BHI behave similarly. However, it appears that the PEHI benefits from the improved generalization of the cGAN.
- The DHI appears to yield fewer false responses than BHI and PEHI. This could be due to the improved robustness of the cGAN compared to the base model. As this statement relies heavily on the definition of false responses and associated assumptions, more research is necessary to investigate the full potential of the DHI.

7.4. Research Questions Revisited

The main research question addressed in this report is:

> **What impact does probabilistic target variable modeling of the generator bearing temperature have on drivetrain monitoring and fault detection?**

To answer this, the following sub-questions are explored:

1. *How does probabilistic modeling compare to a deterministic approach in representing the generator bearing temperature? What implications does this have concerning the goal of fault detection?*

   With probabilistic modeling, the generator bearing temperature is represented by a distribution, instead of a single point. This allows to account for uncertainty in the model's predictions. The results show that the median of this distribution better reflects the real temperature, compared to the base model. Additionally, using all information provided in the distribution, false responses can be reduced significantly.

2. *What is the impact of probabilistic modeling on the sensitivity of the detection method?* The sensitivity of fault detection heavily depends on how the predicted temperature distribution is post-processed. The probabilistic setups investigated in this project show equal or decreased sensitivity to generator bearing faults. In contrast, a slight sensitivity increase concerning generator fault-affected data was notable. As the physical relationship of this is unknown, further research is necessary to understand if a causal relationship is present.

3. *How does probabilistic modeling influence the robustness of fault detection in varying operational conditions?* The results indicate that probabilistic modeling improves robustness. While this is likely a contributing factor to the model's improved performance, establishing a direct causal relationship remains challenging and would require further investigation.

4. *What are the main advantages and disadvantages of deterministic vs. probabilistic approaches in this context?* Performance-wise, probabilistic modeling appears to benefit drivetrain monitoring by improving robustness and decreasing false responses. Additionally, there is potential for improved sensitivity to other failure types. The main drawbacks of probabilistic modeling are the significant effort necessary for tuning the model to yield satisfactory results, and the increased computational time. Although computational expense was not explicitly measured in this study, it is a well-known drawback of probabilistic modeling approaches that needs consideration when applying such approaches for wind turbine fault detection on a larger scale.

8

Conclusion

This study aimed to investigate the potential of probabilistic target variable modeling in the context of monitoring wind turbine drivetrains. For that, a cGAN and a deterministic base model were developed to model the generator bearing temperature. To focus on the implications of probabilistic modeling of the generator bearing temperature in contrast to deterministic modeling, cGAN and base model were kept as similar as possible.

Comparison of both models indicates that probabilistic target modeling benefits from improved robustness compared to the base model. Next to that, probabilistic modeling has the potential for better generalization. This however, depends on how the probabilistic model is evaluated during training. Fault sensitivity comparison of probabilistic and deterministic models does not suggest a significant advantage for either of the models.

Regarding the findings from applying both models for fault detection, interpretation of the results is challenging, as the meaning of positive or negative responses heavily depends on the existence of a true physical relationship between the target variable and turbine condition. From the information provided in the dataset, it is not always clear if this relationship is present. Minding these limitations, the findings suggest that probabilistic modeling reduces the amount of false responses of the model. The improved robustness likely plays a key role in this performance advantage. While this study provides evidence that probabilistic modeling improves robustness, the limited clarity on the physical relationship between the target variable and turbine condition prevents a definitive assessment of its overall impact on fault detection.

Nevertheless, the findings highlight the potential of probabilistic modeling to improve fault detection in wind turbine drivetrains, especially under varying operational conditions. By that, probabilistic modeling could contribute to making wind energy more competitive, ultimately accelerating the energy transition and supporting global efforts to combat climate change.

Limitations and Further Research

To fully assess the potential of probabilistic target modeling in wind turbine drivetrains, the improved robustness demonstrated in this study must be verified on more diverse data. Next to that, further research should investigate the extent to which improved robustness contributes to performance benefits. While the interpretation of the results obtained from applying the probabilistic model to the EDP dataset suggest that false responses are significantly reduced, this must be verified on a dataset that provides more detailed information about maintenance interventions. The fault sensitivity of the model strongly depends on post-processing of the model outputs, and identification of outliers. Therefore, further research is needed to explore alternative methods for post-processing the generated distributions and identifying outliers. During the project, many decisions have been made based on tuning model parameters to achieve good model performance. This results in non-optimal choices for a variety of decisions. This is a general problem associated with ML techniques. However, critical model parameters should be optimized with more sophisticated methods. The necessity for model explainability is not addressed in this project and needs further investigation. While this research project suggests that probabilistic

modeling improves model robustness, which enhances trustworthiness, the internal workings of the model remain a black box, posing a key obstacle for deployment in critical applications.

References

[1] Global Wind Energy Council, "Global wind report 2023," 2023, Accessed: May 18, 2024. [Online]. Available: https://gwec.net/globalwindreport2023/.

[2] J. Maldonado-Correa, S. Martín-Martínez, E. Artigao, and E. Gómez-Lázaro, "Using scada data for wind turbine condition monitoring: A systematic literature review," *Energies*, vol. 13, no. 12, Art. no. 3132, Jun. 2020. DOI: 10.3390/en13123132.

[3] E. Artigao, S. Martín-Martínez, A. Honrubia-Escribano, and E. Gómez-Lázaro, "Wind turbine reliability: A comprehensive review towards effective condition monitoring development," *Appl. Energy*, vol. 228, pp. 1569–1583, Oct. 2018. DOI: 10.1016/j.apenergy.2018.07.037.

[4] J. Carroll, A. McDonald, and D. McMillan, "Failure rate, repair time and unscheduled O&M cost analysis of offshore wind turbines," *Wind Energy*, vol. 19, no. 6, pp. 1107–1119, Jun. 2016. DOI: 10.1002/we.1887.

[5] J. Carroll, A. McDonald, I. Dinwoodie, D. McMillan, M. Revie, and I. Lazakis, "Availability, operation and maintenance costs of offshore wind turbines with different drive train configurations," *Wind Energy*, vol. 20, no. 2, pp. 361–378, Feb. 2017. DOI: 10.1002/we.2011.

[6] V. Yildirir, E. Rusu, and F. Onea, "Wind energy assessments in the northern Romanian coastal environment based on 20 years of data coming from different sources," *Sustainability*, vol. 14, no. 7, Apr. 2022. DOI: 10.3390/su14074249.

[7] J. Carroll, A. McDonald, and D. McMillan, "Reliability comparison of wind turbines with DFIG and PMG drive trains," *IEEE Trans. Energy Convers.*, vol. 30, no. 2, pp. 663–670, Jun. 2015. DOI: 10.1109/TEC.2014.2367243.

[8] D. Zappalá and P. J. Tavner, "Wind turbine reliability - maintenance strategies," in *Comprehensive Renewable Energy, Second Edition*, T. M. Letcher, Ed., Amsterdam, Netherlands: Elsevier, Jan. 2022, ch. 2, sec. 11, pp. 353–370. DOI: 10.1016/B978-0-12-819727-1.00154-0.

[9] A. Turnbull and J. Carroll, "Cost benefit of implementing advanced monitoring and predictive maintenance strategies for offshore wind farms," *Energies*, vol. 14, no. 16, Aug. 2021. DOI: 10.3390/en14164922.

[10] D. Astolfi, "Wind turbine drivetrain condition monitoring through SCADA-collected temperature data: discussion of selected recent papers," *Energies*, vol. 16, no. 9, May 2023. DOI: 10.3390/en16093614.

[11] W. Teng, X. Ding, S. Tang, J. Xu, B. Shi, and Y. Liu, "Vibration analysis for fault detection of wind turbine drivetrains—a comprehensive investigation," *Sensors*, vol. 21, no. 5, pp. 1–33, Mar. 2021. DOI: 10.3390/s21051686.

[12] Y. Zhang, W. Lu, and F. Chu, "Planet gear fault localization for wind turbine gearbox using acoustic emission signals," *Renew. Energy*, vol. 109, pp. 449–460, 2017. DOI: 10.1016/j.renene.2017.03.035.

[13] W. Yang, P. J. Tavner, and M. R. Wilkinson, "Condition monitoring and fault diagnosis of a wind turbine synchronous generator drive train," *IET Renew. Power Gener.*, vol. 3, no. 1, pp. 1–11, 2009. DOI: 10.1049/iet-rpg:20080006.

[14] J. Zhu, J. Yoon, D. He, B. Qiu, and E. Bechhoefer, "Online Condition monitoring and remaining useful life prediction of particle contaminated lubrication Oil," in *Proc. IEEE Int. Conf. Progn. Health Manage. (PHM)*, Gaithersburg, MD, USA, 2013, pp. 1–14. DOI: 10.1109/ICPHM.2013.6621415.

[15] W. Mu, Y. Gao, Y. Wang, G. Liu, and H. Hu, "Modeling and analysis of acoustic emission generated by fatigue cracking," *Sensors*, vol. 22, no. 3, Feb. 2022. DOI: 10.3390/s22031208.

[16] S. Soua, P. Van Lieshout, A. Perera, T. H. Gan, and B. Bridge, "Determination of the combined vibrational and acoustic emission signature of a wind turbine gearbox and generator shaft in service as a pre-requisite for effective condition monitoring," *Renew. Energy*, vol. 51, pp. 175–181, Mar. 2013. DOI: 10.1016/j.renene.2012.07.004.

[17] A. K. Biswas, A. K. Datta, P. Topdar, and S. Sengupta, "Acoustic emission-based damage detection and classification in steel frame structure using wavelet transform and random forest," *Period. Polytech.: Civ. Eng.*, vol. 66, no. 4, pp. 1183–1198, Sep. 2022. DOI: 10.3311/PPci.20447.

[18] C. P. Salomon, C. Ferreira, W. C. Sant'Ana, *et al.*, "A study of fault diagnosis based on electrical signature analysis for synchronous generators predictive maintenance in bulk electric systems," *Energies*, vol. 12, no. 8, Apr. 2019. DOI: 10.3390/en12081506.

[19] M. R. Shahriar, P. Borghesani, G. Ledwich, and A. C. Tan, "Performance analysis of electrical signature analysis-based diagnostics using an electromechanical model of wind turbine," *Renew. Energy*, vol. 116, pp. 15–41, Feb. 2018. DOI: 10.1016/j.renene.2017.04.006.

[20] X. Xu, X. Huang, H. Bian, J. Wu, C. Liang, and F. Cong, "Total process of fault diagnosis for wind turbine gearbox, from the perspective of combination with feature extraction and machine learning: A review," *Energy AI*, vol. 15, Jan. 2024. DOI: 10.1016/j.egyai.2023.100318.

[21] E. Gonzalez, B. Stephen, D. Infield, and J. J. Melero, "Using high-frequency SCADA data for wind turbine performance monitoring: A sensitivity study," *Renew. Energy*, vol. 131, pp. 841–853, Feb. 2019. DOI: 10.1016/j.renene.2018.07.068.

[22] J. Xu, X. Jiang, S. Liao, *et al.*, "Probabilistic prognosis of wind turbine faults with feature selection and confidence calibration," *IEEE Trans. Sustain. Energy*, vol. 15, no. 1, pp. 52–67, Jan. 2024. DOI: 10.1109/TSTE.2023.3272317.

[23] D. Jobson and E. Hudson, "Generalized regression with conditional GANs," *arXiv preprint*, 2024, arXiv:2404.13500. [Online]. Available: https://arxiv.org/abs/2404.13500.

[24] Energias de Portugal, *EDP Open Data*, [Online]. https://www.edp.com/en/innovation/open-data/data, Accessed: 2024-03-16.

[25] A. Stetco, F. Dinmohammadi, X. Zhao, *et al.*, "Machine learning methods for wind turbine condition monitoring: A review," *Renew. Energy*, vol. 133, pp. 620–635, Apr. 2019. DOI: 10.1016/j.renene.2018.10.047.

[26] G. Helbing and M. Ritter, "Deep Learning for fault detection in wind turbines," *Renew. Sustain. Energy Rev.*, vol. 98, pp. 189–198, Dec. 2018. DOI: 10.1016/j.rser.2018.09.012.

[27] R. Morrison, X. Liu, and Z. Lin, "Anomaly detection in wind turbine SCADA data for power curve cleaning," *Renew. Energy*, vol. 184, pp. 473–486, Jan. 2022. DOI: 10.1016/j.renene.2021.11.118.

[28] D. Astolfi, F. De Caro, and A. Vaccaro, "Condition monitoring of wind turbine systems by explainable artificial intelligence techniques," *Sensors*, vol. 23, no. 12, Art. no. 5376, Jun. 2023. DOI: 10.3390/s23125376.

[29] X. Xia, X. Pan, N. Li, *et al.*, "GAN-based anomaly detection: A review," *Neurocomputing*, vol. 493, pp. 497–535, Jul. 2022. DOI: 10.1016/j.neucom.2021.12.093.

[30] C. O. Retzlaff, A. Angerschmid, A. Saranti, *et al.*, "Post-hoc vs ante-hoc explanations: xAI design guidelines for data scientists," *Cogn. Syst. Res.*, vol. 86, Art. no. 101243, Aug. 2024. DOI: 10.1016/j.cogsys.2024.101243.

[31] A. Saranya and R. Subhashini, "A systematic review of Explainable Artificial Intelligence models and applications: Recent developments and future trends," *Decis. Anal.*, vol. 7, Art. no. 100230, Jun. 2023. DOI: 10.1016/J.DAJOUR.2023.100230.

[32] D. Astolfi, F. De Caro, and A. Vaccaro, "Recent advances in the use of eXplainable Artificial intelligence techniques for wind turbine systems condition monitoring," *Electronics*, vol. 12, no. 16, Art. no. 3509, Aug. 2023. DOI: 10.3390/electronics12163509.

[33] R. Marcinkevičs and J. E. Vogt, "Interpretable and explainable machine learning: A methods-centric overview with concrete examples," *WIREs Data. Mining. Knowl. Discov.*, vol. 13, no. 3, Art. no. e1493, Feb. 2023. DOI: 10.1002/widm.1493.

[34] G. Li, L. Chen, C. Fan, T. Li, C. Xu, and X. Fang, “Interpretation and explanation of convolutional neural network-based fault diagnosis model at the feature-level for building energy systems,” *Energy Build.*, vol. 295, Art. no. 113326, Sep. 2023. DOI: 10.1016/j.enbuild.2023.113326.

[35] X. Xiao, J. Liu, D. Liu, Y. Tang, and F. Zhang, “Condition monitoring of wind turbine main bearing based on multivariate time series forecasting,” *Energies*, vol. 15, no. 5, Art. no. 1951, Mar. 2022. DOI: 10.3390/en15051951.

[36] J. Liu, G. Yang, X. Li, Q. Wang, Y. He, and X. Yang, “Wind turbine anomaly detection based on SCADA: A deep autoencoder enhanced by fault instances,” *ISA Trans.*, vol. 139, pp. 586–605, Aug. 2023. DOI: 10.1016/j.isatra.2023.03.045.

[37] M. Jankauskas, A. Serackis, M. Šapurov, *et al.*, “Exploring the limits of early predictive maintenance in wind turbines applying an anomaly detection technique,” *Sensors*, vol. 23, no. 12, Art. no. 5695, Jun. 2023. DOI: 10.3390/s23125695.

[38] Z. Fu, Z. Zhou, and Y. Yuan, “Fault diagnosis of wind turbine main bearing in the condition of noise based on Generative Adversarial Network,” *Processes*, vol. 10, no. 10, Art. no. 2006, Oct. 2022. DOI: 10.3390/pr10102006.

[39] Z. Fan, Y. Wang, L. Meng, G. Zhang, Y. Qin, and B. Tang, “Unsupervised anomaly detection method for bearing based on VAE-GAN and time-series data correlation enhancement,” *IEEE Sens. J.*, vol. 23, no. 23, pp. 29 345–29 356, Dec. 2023. DOI: 10.1109/JSEN.2023.3326335.

[40] C. Zhang and T. Yang, “Anomaly detection for wind turbines using long short-term memory-based variational autoencoder wasserstein generation adversarial network under semi-supervised training,” *Energies*, vol. 16, no. 19, p. 7008, Oct. 2023. DOI: 10.3390/en16197008.

[41] M. Hoh, A. Schöttl, H. Schaub, and F. Wenninger, “A generative model for anomaly detection in time series data,” *Procedia Comput. Sci.*, vol. 200, pp. 629–637, 2022. DOI: 10.1016/j.procs.2022.01.261.

[42] O. M. Ezeme, Q. H. Mahmoud, and A. Azim, “Design and development of AD-CGAN: Conditional generative adversarial networks for anomaly detection,” *IEEE Access*, vol. 8, pp. 177 667–177 681, 2020. DOI: 10.1109/ACCESS.2020.3025530.

[43] Y. Qiu, T. Misu, and C. Busso, “Driving anomaly detection using conditional generative adversarial network,” *arXiv preprint*, arXiv:2203.08289, Mar. 2022. [Online]. Available: http://arxiv.org/abs/2203.08289.

[44] A. Wang, Y. Pei, Y. Zhu, and Z. Qian, “Wind turbine fault detection and identification through self-attention-based mechanism embedded with a multivariable query pattern,” *Renew. Energy*, vol. 211, pp. 918–937, Jul. 2023. DOI: 10.1016/j.renene.2023.05.003.

[45] A. Oliveira-Filho, R. Zemouri, P. Cambron, and A. Tahan, “Early detection and diagnosis of wind turbine abnormal conditions using an interpretable supervised variational autoencoder model,” *Energies*, vol. 16, no. 12, Art. no. 4544, Jun. 2023. DOI: 10.3390/en16124544.

[46] C. M. Roelofs, M. A. Lutz, S. Faulstich, and S. Vogt, “Autoencoder-based anomaly root cause analysis for wind turbines,” *Energy AI*, vol. 4, Art. no. 100065, Jun. 2021. DOI: 10.1016/j.egyai.2021.100065.

[47] J. H. Park, H. S. Jo, S. H. Lee, S. W. Oh, and M. G. Na, “A reliable intelligent diagnostic assistant for nuclear power plants using explainable artificial intelligence of GRU-AE, LightGBM and SHAP,” *Nucl. Eng. Technol.*, vol. 54, no. 4, pp. 1271–1287, Apr. 2022. DOI: 10.1016/j.net.2021.10.024.

[48] K. Roshan and A. Zafar, “Utilizing XAI technique to improve autoencoder based model for computer network anomaly detection with shapley additive explanation (SHAP),” *Int. J. Comput. Netw. Commun*, vol. 13, no. 6, pp. 109–128, 2021. DOI: 10.5121/ijcnc.2021.13607.

[49] H. A. Gamal Al-Kaf and K. B. Lee, “Explainable machine learning method for open fault detection of NPC inverter using SHAP and LIME,” in *Proc. IEEE Conf. Energy Convers. (CENCON)*, 2023, pp. 14–19. DOI: 10.1109/CENCON58932.2023.10368888.

[50] J. Lee, I. Noh, J. Lee, and S. W. Lee, “Development of an explainable fault diagnosis framework based on sensor data imagification: A case study of the robotic spot-welding process,” *IEEE Trans. Ind. Inform.*, vol. 18, no. 10, pp. 6895–6904, Oct. 2022. DOI: 10.1109/TII.2021.3134250.

[51] S. Barber, L. A. M. Lima, Y. Sakagami, *et al.*, “Enabling co-innovation for a successful digital transformation in wind energy using a new digital ecosystem and a fault detection case study,” *Energies*, vol. 15, no. 15, Art. no. 5638, Aug. 2022. DOI: 10.3390/en15155638.

[52] E. Latiffianti, S. Sheng, and Y. Ding, “Wind turbine gearbox failure detection through cumulative sum of multivariate time series data,” *Front. Energy Res.*, vol. 10, Art. no. 904622, May 2022. DOI: 10.3389/fenrg.2022.904622.

[53] World Wide Fund for Nature (WWF) and Boston Consulting Group (BCG), “Building a nature-positive energy transformation,” 2021, [Online]. Available: https://wwfint.awsassets.panda.org/downloads/wwf-bcg-building-a-nature-positive-energy-transformation.pdf. Accessed: Nov. 11, 2024.

[54] Global Wind Energy Council, “Wind can power 3.3 million new jobs worldwide over next five years,” 2021, [Online]. Available: https://gwec.net/wind-can-power-3-3-million-new-jobs-worldwide-over-next-five-years/. Accessed: Nov. 11, 2024.

[55] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016, [Online]. Available: http://www.deeplearningbook.org.

[56] C. M. Bishop, *Pattern Recognition and Machine Learning*. Berlin, Germany: Springer, 2006.

[57] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Comput*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997. DOI: 10.1162/neco.1997.9.8.1735.

[58] Z. Cai, G. Feng, and Q. Wang, “Based on the improved pso-tpa-lstm model chaotic time series prediction,” *Atmosphere*, vol. 14, no. 11, 2023. DOI: 10.3390/atmos14111696.

[59] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, *et al.*, “Generative adversarial nets,” in *Proc. Int. Conf. Neural Inf. Process. Syst.*, vol. 2, Montreal, Canada, 2014, pp. 2672–2680.

[60] M. Wiatrak, S. V. Albrecht, and A. Nystrom, “Stabilizing generative adversarial networks: A survey,” *arXiv preprint*, arXiv:1910.00927, 2019. [Online]. Available: https://arxiv.org/abs/1910.00927.

[61] E. Nazari, P. Branco, and G. V. Jourdan, “AutoGAN: An automated human-out-of-the-loop approach for training generative adversarial networks,” *Mathematics*, vol. 11, no. 4, Art. no. 977, Feb. 2023. DOI: 10.3390/math11040977.

[62] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, “Improved techniques for training GANs,” in *Proc. Int. Conf. Neural Inf. Process. Syst.*, Barcelona, Spain, Dec. 2016, pp. 2234–2242.

[63] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, “GANs trained by a two time-scale update rule converge to a local nash equilibrium,” in *Adv. Neural Inf. Process. Syst.*, Long Beach, CA, USA, Dec. 2017, pp. 6626–6637.

[64] M. Mirza and S. Osindero, “Conditional Generative Adversarial Nets,” *arXiv preprint*, 2014. [Online]. Available: https://arxiv.org/abs/1411.1784.

[65] M. H. DeGroot and M. J. Schervish, *Probability and Statistics*. London, UK: Pearson Education, 2012.

[66] C. Villani, *Optimal Transport: Old and New* (Fundamental Principles of Mathematical Sciences). Berlin, Germany: Springer, 2009, vol. 338.

[67] D. Menezes, M. Mendes, J. A. Almeida, and T. Farinha, “Wind farm and resource datasets: A comprehensive survey and overview,” *Energies*, vol. 13, no. 18, Art. no. 4702, Sep. 2020. DOI: 10.3390/en13184702.

[68] P. Teimourzadeh Baboli, D. Babazadeh, A. Raeiszadeh, S. Horodyvskyy, and I. Koprek, “Optimal temperature-based condition monitoring system for wind turbines,” *Infrastructures*, vol. 6, no. 4, Art. no. 50, 2021. DOI: 10.3390/infrastructures6040050.

[69] O. T. Bindingsbø, M. Singh, K. Øvsthus, and A. Keprate, “Fault detection of a wind turbine generator bearing using interpretable machine learning,” *Front. in Energy Res.*, vol. 11, Art. no. 1284676, 2023. DOI: 10.3389/fenrg.2023.1284676.

[70] B. Xiao, H. Ni, and W. Yang, “MCGAN: Enhancing GAN training with regression-based generator loss,” *arXiv preprint*, arXiv:2405.17191, 2024. [Online]. Available: https://arxiv.org/abs/2405.17191.

[71] A. Brock, J. Donahue, and K. Simonyan, “Large scale GAN training for high fidelity natural image synthesis,” *arXiv preprint*, arXiv:1809.11096, 2018. [Online]. Available: https://doi.org/10.48550/arXiv.1809.11096.

[72] J. Bergstra, B. Komer, C. Eliasmith, D. Yamins, and D. D. Cox, “Hyperopt: A python library for model selection and hyperparameter optimization,” *Comput. Sci. Discov*, vol. 8, Art. no. 014008, 2015. DOI: 10.1088/1749-4699/8/1/014008.

[73] R. A. Maronna, R. D. Martin, and V. J. Yohai, *Robust Statistics: Theory and Methods*. Hoboken, NJ, USA: John Wiley & Sons, 2006.

[74] P. Venkataanusha, C. Anuradha, D. Murty, and D. Chebrolu, “Detecting outliers in high dimensional data sets using z-score methodology,” *Int. J. Innov. Technol. Explor. Eng.*, vol. 9, pp. 48–53, Nov. 2019. DOI: 10.35940/ijitee.A3910.119119.

A

Numerical description of Clusters used for Validation

The Clusters used for validation are previously described using radar plots. The following table provides the numerical description:

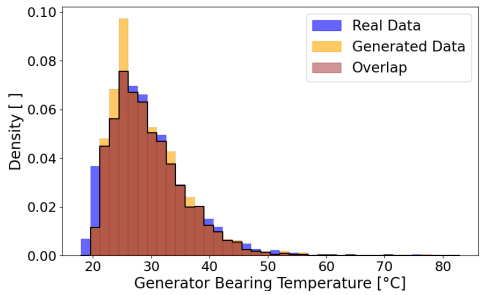
Feature	Mean	Std Dev
Full Dataset		
Gen_RPM_Avg	0.644262	0.341930
Prod_LatestAvg_ActPwrGen1	0.268413	0.317840
Nac_Temp_Avg	0.370465	0.179974
Amb_WindSpeed_Avg	0.269016	0.179300
Amb_Temp_Avg	0.363509	0.166335
Cluster 1		
Gen_RPM_Avg	0.097520	0.092314
Prod_LatestAvg_ActPwrGen1	0.001811	0.018910
Nac_Temp_Avg	0.305230	0.158680
Amb_WindSpeed_Avg	0.084872	0.068414
Amb_Temp_Avg	0.334314	0.174258
Cluster 2		
Gen_RPM_Avg	0.986554	0.015918
Prod_LatestAvg_ActPwrGen1	0.839587	0.157368
Nac_Temp_Avg	0.596599	0.123148
Amb_WindSpeed_Avg	0.562760	0.130841
Amb_Temp_Avg	0.497709	0.130167
Cluster 3		
Gen_RPM_Avg	0.626091	0.192293
Prod_LatestAvg_ActPwrGen1	0.083321	0.086207
Nac_Temp_Avg	0.222286	0.096661
Amb_WindSpeed_Avg	0.192661	0.065168
Amb_Temp_Avg	0.241812	0.107516
Cluster 4		
Gen_RPM_Avg	0.933217	0.049821
Prod_LatestAvg_ActPwrGen1	0.494221	0.158385
Nac_Temp_Avg	0.322776	0.104679
Amb_WindSpeed_Avg	0.384412	0.069117
Amb_Temp_Avg	0.289523	0.082579
Cluster 5		
Gen_RPM_Avg	0.784519	0.089421
Prod_LatestAvg_ActPwrGen1	0.167899	0.120330
Nac_Temp_Avg	0.485750	0.102667
Amb_WindSpeed_Avg	0.246642	0.070749
Amb_Temp_Avg	0.499383	0.112140

Table A.1: Summary of mean and standard deviation for full dataset and clusters.

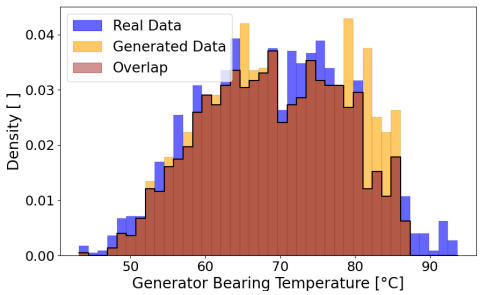
B

Comparison of Base Model Output for the Clusters

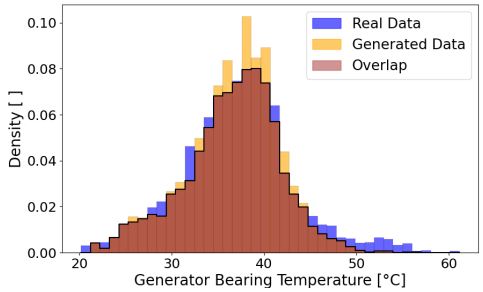
This Appendix contains the comparison of the base model output, and real data, for each of the clusters:



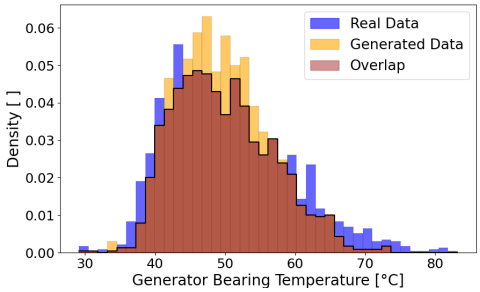
(a) Distribution of real and generated target temperature for Cluster 1. W_1 = 0.39 (base model).



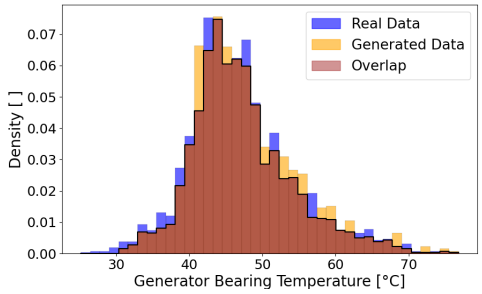
(b) Distribution of real and generated target temperature for Cluster 2. W_1 = 1.4293 (base model).



(c) Distribution of real and generated target temperature for Cluster 3. W_1 = 0.6239 (base model).



(d) Distribution of real and generated target temperature for Cluster 4. W_1 = 1.3527 (base model).

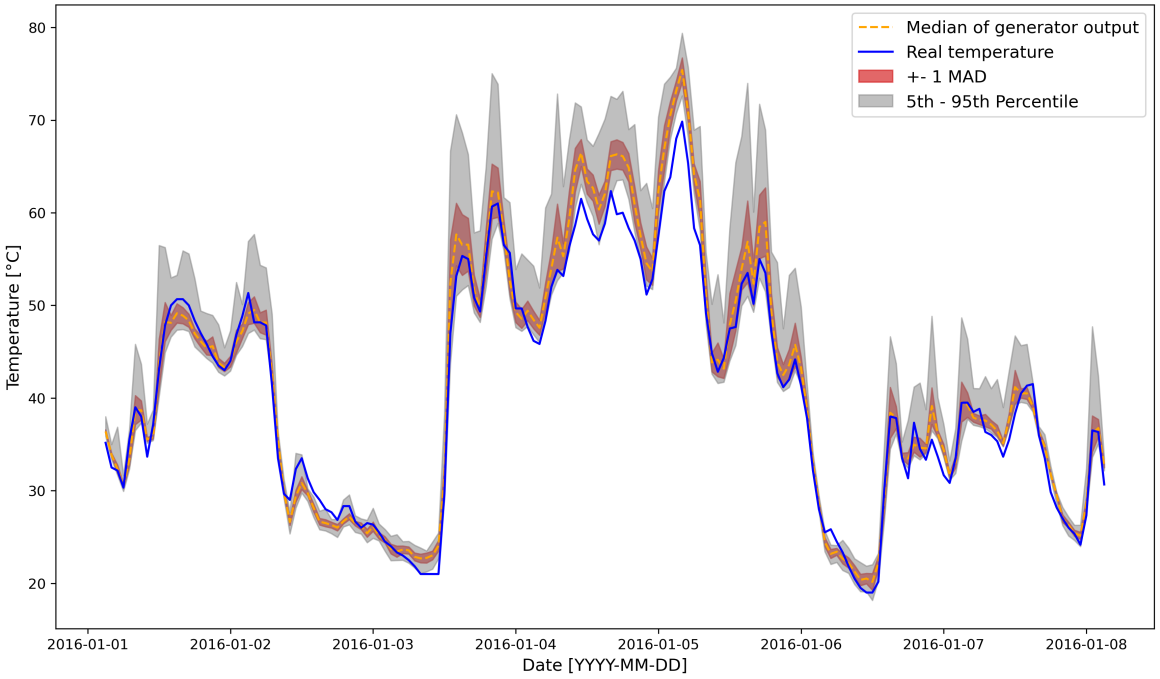


(e) Distribution of real and generated target temperature for Cluster 5. W_1 = 0.5360 (base model).

Figure B.1: Distribution comparison of real and generated target temperature (base model, healthy validation set).

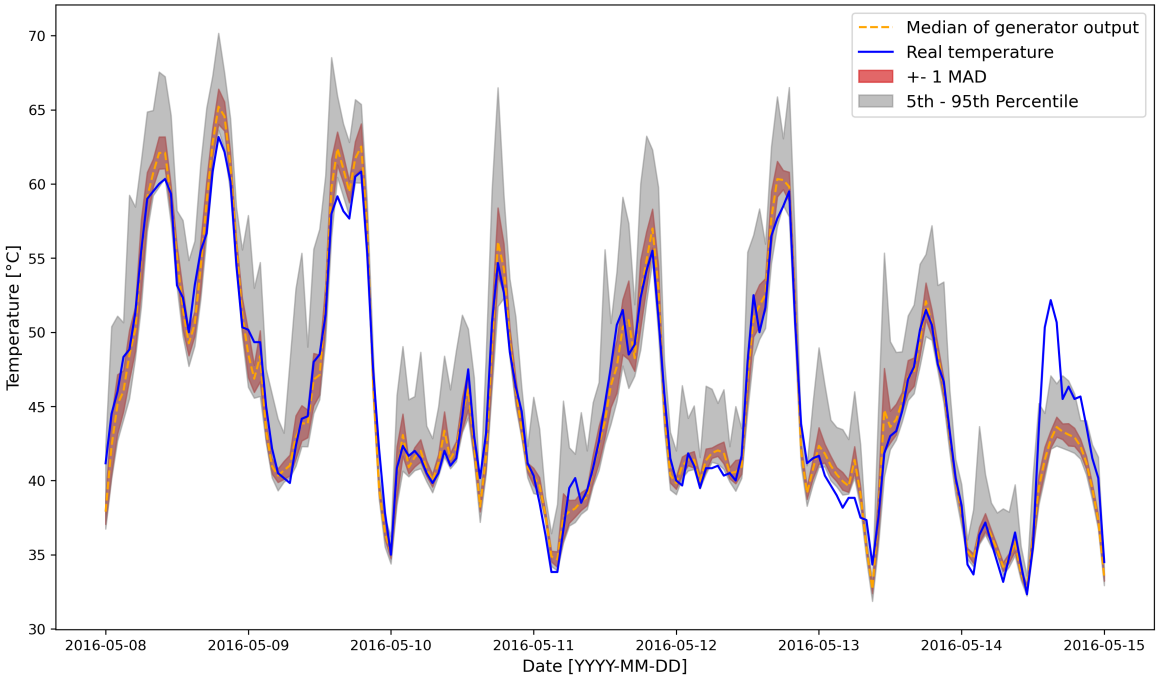
C

Additional Time Series of Generator Output



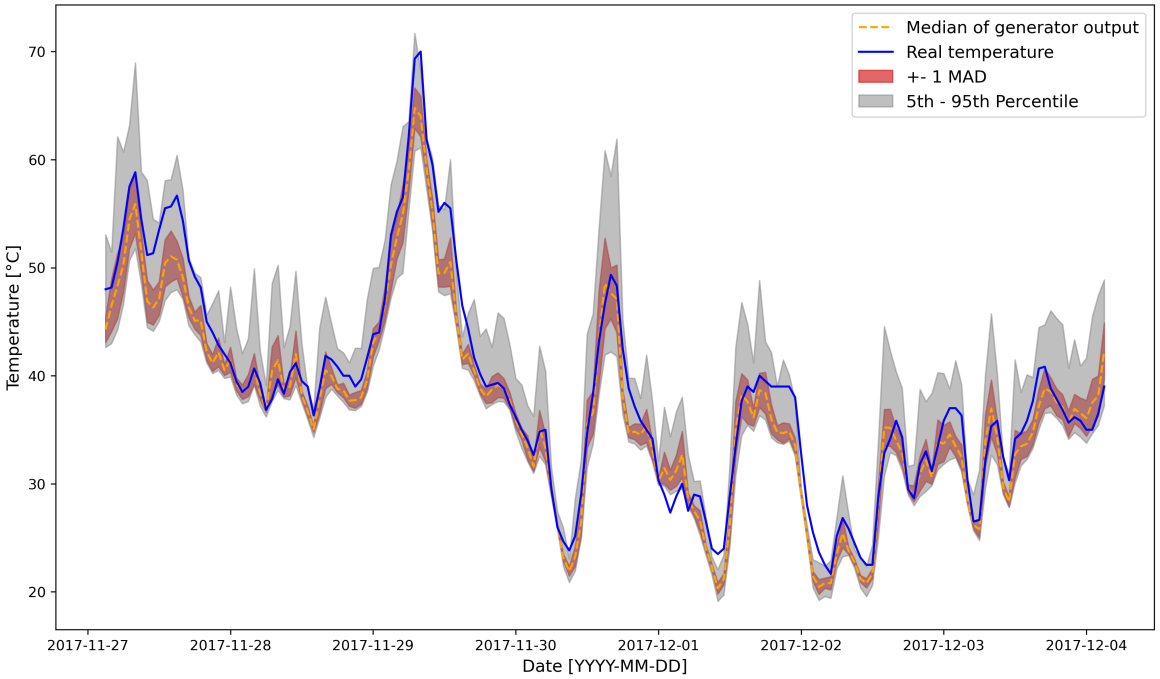
(a) Turbine 6: Time series of the first week in the validation set

Figure C.0: Time series of the generator output set $\mathbf{Y}$, and the real temperature t. The output $\mathbf{Y}$ is described by its median, $\pm$ 1 median absolute deviation, and the 5th and 95th percentiles.



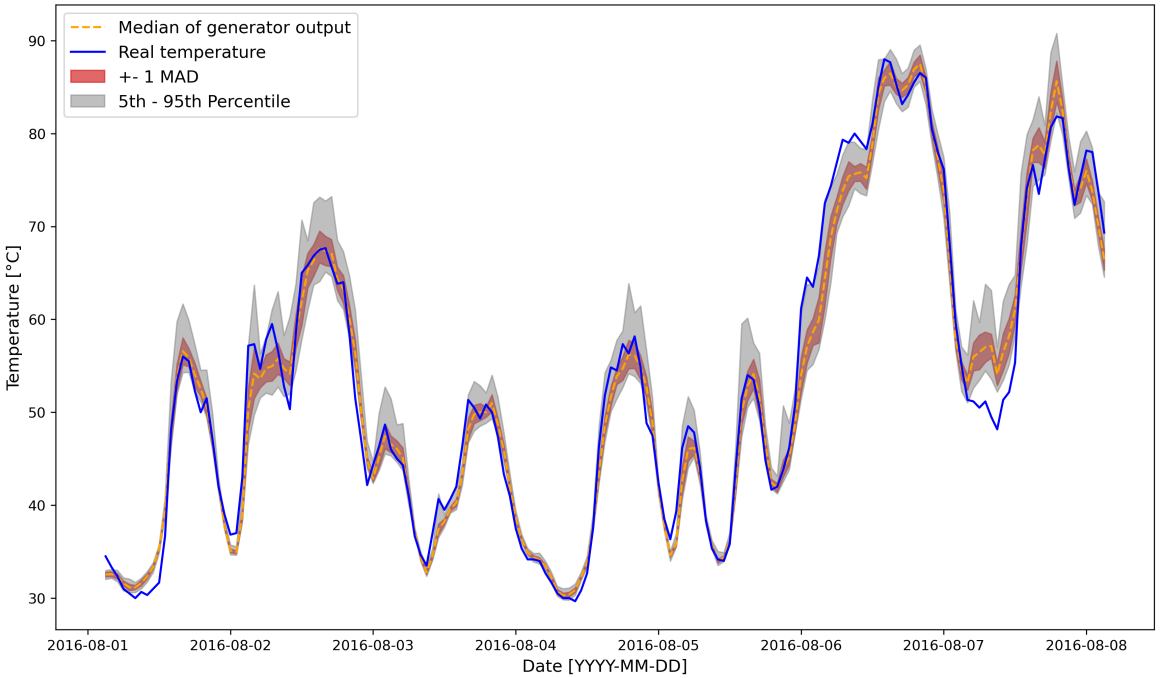
(b) Turbine 7: Time series of the first week in the validation set

Figure C.0: Time series of the generator output set $\mathbf{Y}$, and the real temperature t. The output $\mathbf{Y}$ is described by its median, $\pm$ 1 median absolute deviation, and the 5th and 95th percentiles.



(c) Turbine 9: Time series of the first week in the validation set

Figure C.0: Time series of the generator output set $\mathbf{Y}$, and the real temperature t. The output $\mathbf{Y}$ is described by its median, $\pm$ 1 median absolute deviation, and the 5th and 95th percentiles.



(d) Turbine 11: Time series of the first week in the validation set

Figure C.0: Time series of the generator output set $\mathbf{Y}$, and the real temperature t. The output $\mathbf{Y}$ is described by its median, $\pm$ 1 median absolute deviation, and the 5th and 95th percentiles.